LVS-HOWTO

Joseph Mack (C) 1999-2002, released under GPL jmack@wm7d.net v2002.04 Apr 2002

Install, testing and running of a Linux Virtual Server with 2.2.x and 2.4.x kernels

Contents

1 Introduction

1.1 ChangeLog

LVS-HOWTO

- v0.1 12 Jun 99, sent to Wensong for checking
- v0.2 18 Jul 99, for kernel 2.0.36
- v0.3 6 Nov 99, for kernel 2.2.12, LVS v0.9.2, ipvsadm v1.5 (17Oct99 release)
- v0.4 7 Nov 99, for kernel 2.2.13, LVS v0.9.3, ipvsadm v1.5 (7Nov99 release)
- v0.5 14 Nov 99, for kernel 2.2.13, LVS v0.9.4, ipvsadm v1.5 (11Nov99 release)
- v0.6 17 Nov 99, updates off the mailing list
- v0.7 1 Dec 99, fixes to "arp problem"
- v1.0 Mar 2001, Transparent proxy for LVS-DR (Horms' method) and for LVS-Tun (from Julian) corrections to LVS-NAT instructions for LVS-DR for non-Linux realservers (from ratz). Change to sgml format. Lots of updates
- v1.1 Mar 2001, transparent proxy for 2.4.x
- v1.2 Apr 2001, fwmark
- v1.3 May 2001, kernels 0.2.12-2.4.4, 1.0.7-2.2.19. e-commerce comments from Ratz, ftp is not secure, netscape persistence, dynamic web pages.
- v1.4 May 2001, kernels 0.8.0-2.4.4, 1.0.7-2.2.19
- v1.5 Jun 2001, dead links fixed (thanks to Dave Byrne `dave@glynjones.com`), other random stuff from the mailing list.
- v1.6 Jul 2001, more stuff from the mailing list.
- v1.7 Aug 2001, more stuff from the mailing list, LVSGSP,
- v1.8 Sep 2001, failover for nfs, server state synchronisation tables (mentioned), synchronising server content, dual NIC failover, shutdown of LVS.
- v1.9 Sep 2001, some humor in poor taste removed.
- v1.10 Oct 2001, ingress for limiting number of connections, Peter Mueller's writeup on using Linux-HA for failover of directors.
- v1.11 Nov 2001, some well needed reformatting.
- v1.12 Dec 2001, vrrpd, NIC updates.
- v2002.04 Apr 2002, new numbering scheme based on date (there isn't anything here that has versions in the sense of backward compatibility), vrrpd, lots of little fixes.

posted to the LVS site,*LVS website http://www.linuxvirtualserver.org* `<http://www.linuxvirtualserver.org/>`

1.2 Purpose of this HOWTO

To enable you to understand how a Linux Virtual Server (LVS) works. Another document, the LVS-mini-HOWTO, tells you how to setup and install an LVS without understanding how the LVS works. The mini-HOWTO was extracted from this HOWTO. Eventually the redundant material will be dropped from this HOWTO.

The material here covers directors and realservers with 2.2 and 2.4 kernels. The code for 2.0.x kernels still works fine and was used on production systems when 2.0.x kernels were current, but is not being developed further. For 2.2 kernels, the networking code was rewritten, producing the 3 (arp problem). This changes the installation of LVS from a simple process that can be done by almost anyone, to thought provoking, head scratching exercise, which requires detailed understanding of the workings of LVS. For 2.0 and 2.2 LVS is stand alone code based on ip_masquerading and doesn't integrate well with other uses of ip_masquerading. For 2.4 kernels, LVS was rewritten as a netfilter module to allow it to fit into and be visible to other netfilter modules. Unfortunately the fit isn't perfect but cooperation with netfilter does work in most cases. Being a netfilter module, the latency and throughput are slightly worse for 2.4 LVS than for the 2.2 code. However with modern CPUs being running at 800MHz, the bottleneck now is network throughput rather than LVS throughput (you only need a small number of realservers to saturate 100Mbps ethernet).

In general ipvsadm commands and services have not changed between kernels.

1.3 Nomenclature/Abbreviations

If you use these terms when you mail us, we'll know what you're talking about.

1.3.1 Preferred names

- *IPVS,ipvs,ip_vs* the code that patches the linux kernel on the *director*.
- *LVS, linux virtual server* This is the *director* + *realservers*. Together these machines are the *virtual server*, which appears as one machine to the *clients*.
- *director*: the node that runs the *ipvs* code. *Clients connect* to the *director*. The *director forwards* packets to the realservers. The *director* is nothing but an IP router with special rules that make the *LVS* work.
- *realservers*: the hosts that have the *services*. The *realservers* handle the requests from the clients.
- *client* the host or user level process that connects to the *VIP* on the *director*

1.3.2 synonyms

Please use the first term in these lines. The other words are valid but less precise (or are redundant).

- *director*: load balancer, dispatcher, redirector.
- *realserver*: servers, real servers, real-servers.
- *LVS*: the whole cluster, the (linux) virtual server (LVS)

1.3.3 backend (multi-tier) servers

The *realservers* sometimes are frontends to other *backend* servers. The *client* does not connect to these *backend* servers and they are not in the ipvsadm table.

e.g.

- a *realserver* may run a web application. The web application in turn connects to a database on another *backend* server.
- a webcaching *realserver* (*e.g.* a squid). The squid connects to *backend* webserver(s).

These *backend* servers are setup separately from the LVS.

1.3.4 the term "the server" is ambiguous

People sometimes call the *director* or the *realservers*, "the server". Since the LVS appears as a server to the *client* and since the *realservers* are also serving services, the term "server" is ambiguous. Do not use the term "the server" when talking about LVS. Most often you are referring to the "director" or the "realservers". Sometimes (*e.g.* when talking about throughput) you are talking about the LVS.

I use "realserver" as I despair of finding a reference to a "real server" in a webpage using the search keys "real" and "server". Horms and I (for reasons that neither of us can remember) have been pushing the term "real-server" for about a year, on the mailing list, and no-one has adopted it. We're going back to "realserver".

1.3.5 names of IPs in an LVS

```
client IP     = CIP
virtual IP    = VIP (the IP on the director that the client connects to)
director IP   = DIP (the IP on the director in the realserver's network)
realserver IP = RIP (and RIP1, RIP2...) (the IP on the realserver)
director GW   = DGW (or director gw)
realserver GW = RGW, SGW (or server gw)
```

1.4 What is an LVS?

A Linux Virtual Server (LVS) is a cluster of servers which appears to be one server to an outside client. This apparent single server is called here a "virtual server". The individual servers (realservers) are under the control of a director (or load balancer), which runs a Linux kernel patched to include the *ipvs* code. The ipvs code running on the director is the essential feature of the LVS (although other user level code can/is used to manage the LVS). The director is a router with modified routing rules.

When a new connection is requested from a client to a service provided by the LVS (eg httpd), the director will choose a realserver for the client. From then, all packets from the client will go through the director to that particular realserver. The association between the client and the realserver will last for only the life of the tcp connection (or udp exchange). For the next tcp connection, the director will choose a new realserver (which may or may not be the same as the first realserver). Thus a web browser connecting to an LVS serving a webpage consisting of several hits (images, html page), may get each hit from a separate realserver.

1.4.1 What is a VIP?

The director presents an IP called the Virtual IP (VIP) to clients. (When using 9 (fwmarks), VIPs are agregated into groups of IPs, but the same principles apply as for a single IP). When a client connects to the VIP, the director forwards the client's packets to one particular realserver for the duration of the client's connection to the LVS. This connection is chosen and managed by the director. The realservers serve services (eg ftp, http, dns, telnet, nntp, smtp) such as are found in /etc/services or inetd.conf. The LVS presents one IP on the director (the virtual IP, VIP) to clients.

Peter Martin `p.martin@ies.uk.com` and John Cronin `jsc3@havoc.gtf.org` 05 Jul 2001

> The VIP is the address which you want to load balance i.e. the address of your website. The VIP is usually an alias (*e.g.* eth0:1) so that the VIP can be swapped between two directors if a fault is detected on one. The VIP is the IP address of the "service", not the IP address of any of the particular systems used in providing the service (ie the director and the realservers).
>
> The VIP be moved from one director to another backup director if a fault is directed (typically this is done by using mon and heartbeat, or something similar). The director can have 25.1 (multiple VIPs). Each VIP can have one or more services associated with it *e.g.* you could have HTTP/HTTPS balanced using one VIP, and FTP service (or whatever) balanced using another VIP, and calls to these VIPs can be answered by the same or different realservers.
>
> Groups of VIPs and/or ports can be setup with 9 (fwmark).
>
> The realservers have to be configured to work with the VIPs on the director (this includes handling the 3 (arp problem)).
>
> There can be 8 (persistence issues), if you are using cookies or https, or anything else that expects the realserver fulfilling the requests to have some connection state information. This is also addressed on the *LVS persistence page* `<http://www.linuxvirtualserver.org/docs/persistence.html>`

1.4.2 Where do you use an LVS?

- for higher throughput. The cost of increasing throughput by adding realservers in an LVS increases linearly, whereas the cost of increased throughput by buying a larger single machine increases faster than linearly
- for redundancy. Individual machines can be switched out of the LVS, upgraded and brought back on line without interuption of service to the clients. Machines can move to a new site and brought on line one at a time while machines are removed from the old site, without interruption of service to the clients.
- for adaptability. If the throughput is expected to change gradually (as a business builds up), or quickly (for an event), the number of servers can be increased (and then decreased) transparently to the clients.

1.4.3 Client/Server relationship is preserved in an LVS

- Client sees only one IP address and believes it is connecting to a single machine. IPs of all servers is mapped to one IP (the VIP).

  While the client is connected to only one machine at a time, however subsequent connections will be assigned to a new and likely different machine.
- servers at different IP addresses believe they are contacted directly by the client.

1.4.4 Basic LVS topology

```
                        --------
                       |        |
                       | client | (local or on internet)
                       |________|
                           |
                        (router)
                           |
--                         |
L                      Virtual IP
i                      ____|_____
n                     |          | (director can have 1 or 2 NICs)
u                     | director |
x                     |__________|
                           |
V                          |
i                          |
r         ---------------------------------
t         |                |              |
u         |                |              |
a         |                |              |
l    -------------   -------------   -------------
    |             | |             | |             |
S   | realserver1 | | realserver2 | | realserver3 |
e   |_____________| |_____________| |_____________|
r
v
e
r
---
```

1.4.5 LVS director is an L4 switch

In the computer beastiary, the director is a layer 4 (L4) switch. The director makes decisions at the IP layer and just sees a stream of packets going between the client and the realservers. In particular an L4 switch makes decisions based on the IP information in the headers of the packets.

Here's a description of an L4 switch from *Super Sparrow Global Load Balancer documentation* `<http://www.supersparrow.org/ss_paper/>`

" Layer 4 Switching: Determining the path of packets based on information available at layer 4 of the OSI 7 layer protocol stack. In the context of the Internet, this implies that the IP address and port are available as is the underlying protocol, TCP/IP or UCP/IP. This is used to effect load balancing by keeping an affinity for a client to a particular server for the duration of a connection. "

This is all fine except

> Nevo Hed `nevo@aviancommunications.com` 13 Jun 2001 The IP layer is L3.

Alright, I lied. TCPIP is a 4 layer protocol and these layers do not map well onto the 7 layers of the OSI model. (As far as I can tell the 7 layer OSI model is only used to torture students in classes.) It seems that everyone has agreed to pretend that tcpip uses the OSI model and that tcpip devices like the LVS director should therefore be named according to the OSI model. Because of this, the name "L4 switch" really isn't correct, but we all use it anyhow.

The director does not inspect the content of the packets and cannot make decisions based on the content of the packets (e.g. if the packet contains a 11.25 (cookie), the director doesn't know about it and doesn't care). The director doesn't know anything about the application generating the packets or what the application is doing. Because the director does not inspect the content of the packets (layer 7, L7) it is not capable of session management or providing service based on packet content. L7 capability would be a useful feature for LVS and perhaps this will be developed in the future (preliminary code is out - May 2001 - 26 (ktcpvs)).

The director is basically a router, with routing tables set up for the LVS function. These tables allow the director to forward packets to realservers for services that are being LVS'ed. If http (port 80) is a service that is being LVS'ed then the director will forward those packets. The director does not have a socket listener on VIP:80 (i.e. netstat won't see a listener).

John Cronin `jsc3@havoc.gtf.org` (19 Oct 2000) calls these types of servers (i.e. lots of little boxes appearing to be one machine) "RAILS" (Redundant Arrays of Inexpensive Linux|Little|Lightweight|L* Servers). Lorn Kay `lorn_kay@hotmail.com` calls them RAICs (C=computer), pronounced "rake".

1.4.6 LVS forwards packets to realservers

The director uses 3 different methods of 5 (forwarding)

- LVS-NAT based on network address translation (NAT)
- LVS-DR (direct routing) where the MAC addresses on the packet are changed and the packet forwarded to the realserver
- LVS-Tun (tunnelling) where the packet is IPIP encapsulated and forwarded to the realserver.

Some modification of the realserver's ifconfig and routing tables will be needed for LVS-DR and LVS-Tun forwarding. For LVS-NAT the realservers only need a functioning tcpip stack (*i.e.* the realserver can be a networked printer).

LVS works with all services tested so far (single and 2 port services) except that LVS-DR and LVS-Tun cannot work with services that initiate connects from the realservers (so far; identd and rsh).

The realservers can be indentical, presenting the same service (eg http, ftp) working off file systems which are kept in sync for content. This type of LVS increases the number of clients able to be served. Or the realservers can be different, presenting a range of services from machines with different services or operating systems, enabling the virtual server to present a total set of services not available on any one server. The realservers can be local/remote, running Linux (any kernel) or other OS's. Some methods for setting up an LVS have fast packet handling (eg LVS-DR which is good for http and ftp) while others are easier to setup (eg transparent proxy) but have slower packet throughput. In the latter case, if the service is CPU or I/O bound, the slower packet throughput may not be a problem.

For any one service (eg httpd at port 80) all the realservers must present identical content since the client could be connected to any one of them and over many connections/reconnections, will cycle through the realservers. Thus if the LVS is providing access to a farm of web, database, file or mail servers, all realservers must have identical files/content. You cannot split up a database amongst the realservers and access pieces of it with LVS.

The simplest LVS to setup involved clients doing read-only fetches (*e.g.* a webfarm). If the client is allowed to write to the LVS (*e.g.* database, mail farm), then some method is required so that data written on one realserver is transferred to other realservers before the client disconnects and reconnects again. This need not be all that fast (you can tell them that their mail won't be updated for 10mins), but the simplest (and most expensive) is for the mail farm to have a common file system for all servers. For a database, the realservers can be running database clients which connect to a single backend database, or else the realservers can be running independant database daemons which replicate their data.

1.4.7 LVS runs on any Linux platform

An LVS requires a Linux director (Intel and 2.1 (Alpha) versions known to work. The LVS code doesn't have any Intel specific instructions and is expected to work on any machine that Linux runs on.

1.4.8 Code for LVS is different for 2.4.x and 2.2.x kernels

There are differences in the coding for LVS for the 2.0.x, 2.2.x and 2.4.x kernels. Development of LVS on 2.0.36 kernels has stopped (May 99). Code for 2.2.x kernels is production level and this HOWTO is up to date for 2.2.19 kernels. Code for 2.4.x kernels is relatively new and the HOWTO is less complete for the 2.4.x material (check on the mailing list). (Jun 2001, we're pretty much upto date now.)

The 2.0.x and 2.2.x code is based on the masquerading code. Even if you don't explicitely use ipchains (eg with LVS-DR or LVS-Tun), you will see masquerading entries with 'ipchains -M -L' (or 'netstat -M').

Code for 2.4.x kernels was rewritten to be compatible with the netfilter code (i.e. its entries will show up in netfilter tables). It is now production level code. Because of incompatibilities with LVS-NAT for 2.4.x LVS was in development mode (till about Jan 2001) for LVS-NAT.

1.4.9 kernels from 2.4.x series are SMP for kernel code

2.4.x kernels are SMP for kernel code as well as user space code, while 2.2.x kernels are only SMP for user space code. LVS is all kernel code. A dual CPU director running a 2.4.x kernel should be able to push packets at twice the rate of the same machine running a 2.2 kernel (if other resources on the director don't become limiting).

1.4.10 OS for realservers

You can have almost any OS on the realservers (all are expected to work, but we haven't tried them all yet). The realservers only need a tcpip stack - a networked printer can be a realserver.

1.4.11 LVS works on ethernet

LVS works on ethernet. There are some limitations on using 3.14 (ATM).

1.4.12 LVS is continually being developed

LVS is continually being developed and usually only the more recent kernel and kernel patches are supported. Usually development is incremental, but with the 2.2.14 kernels the entries in the /proc file system changed and all subsequent 2.2.x versions were incompatible with previous versions.

1.4.13 Other documentation

For more documentation, look at the LVS web site (eg a talk I gave on how LVS works on *2.0.36 kernel directors* `<http://www.linuxvirtualserver.org/Joseph.Mack>`)

Julian has written *Netparse* `<http://www.linuxvirtualserver.org/~julian/>` for which we don't have a lot of documentation yet.

For those who want more understanding of netfilter/iptables etc, here are some starting places. These topics are also covered in many other places.

- *Harald Welte (of the netfilter team) description of what happens to a packet under 2.4* `<http://www.sunbeam.franken.de/projects/packetjourney>`
- *Harald Welte (of the netfilter team) conntrack HOWTO* `<http://www.sunbeam.franken.de/projects/conntrack+nat-HOWTO/>`.

  Conntrack is used in filter rules as a way of accepting "related" packets, *e.g.* the data packets associated with an established ftp connection. Regular filter rules written for these data packets would accept ftp data packets (port 20) even if there were not in response to a PORT call from an already established ftp connection on port 21. In this case the filter rules would accept packets that are part of a DoS attack.

  Conntrack is CPU intensive and lowers throughput (see 25.37 (effect of conntrack on throughput)).
- *the docs/FAQs/HOWTOs on the netfilter site* `<http://netfilter.samba.org>`
- *Linux Network Administrators Guide* `<http://www.linuxdoc.org/LDP/nag2/>`

1.5 LVS Failure

The LVS itself does not provide high availability. Other software is used in conjunction with LVS to provide high availability and is discussed in the 20 (High Availability LVS) section.

1.6 Thanks

Contributions to this HOWTO came from the mailing list and are attibuted to the poster (with e-mail address). Postings may have been edited to fit them into the flow of the HOWTO.

The LVS logo (Tux with 3 lighter shaded penguins behind him representing a director and 3 realservers) is by Mike Douglas spike@bayside.net

LVS homepage `<http://www.linuxvirtualserver.org>` is running on a machine donated by Voxel.

LVS mailing list `<http://www.linuxvirtualserver.org>` is hosted by Lars in Germany`lmb@suse.de`

1.7 HOWTO source is sgml

The HOWTO is written in sgml. The char '&' found in C source code has to be written as &amp; in sgml. If you swipe patches from the sgml rather than say the html rendering of it, you will get code which needs to be edited to fix the &.

1.8 Mailing list, subscribing, unsubscribing and searchable archives

mailing list details `<http://www.linuxvirtualserver.org/mailing.html>`

Thanks to Hank Leininger for the mailing list archive which is searchable not only by subject and author, but by strings in the body. Hank's resource has been of great help in assembling this HOWTO.

1.9 Minimal knowledge required, getting technical help

Ratz `ratz@tac.ch` To be able to setup/maintain an LVS, you need to be able to

- know how to patch and compile a kernel
- the basics of shell-scripting

- have intermediate knowledge of TCP/IP
- have read the man-page, the online-documentation and LVS-HOWTO (this document) (and the LVS-mini-HOWTO)
- know basic system administration tools (e.g. ipchains, syslog)

The mailing list and HOWTOs cover information specific to LVS. The rest you have to handle yourself. All of us knew nothing about computers when we first started, we learnt it and you can too. If you can't setup a simple LVS from the mini-HOWTO, without getting into a major sweat (or being able to tell us what's wrong with the instructions), then you need to do some more homework.

It's hard to believe but we do get postings like

> recompiling the kernel is hard (or I don't read HOWTOs), can't you guys cut me some slack and just tell me what to do?

The answer is: NO WE WON'T

The people on the mailing list answer questions for free, and have important things to do, like keeping up with /. and checking our e-mail. When we're at home, there is beer to drink and Gilligan's Island re-runs to watch. Reading to you does not rate. I expect the people who post these statements don't read the HOWTO, so I may be wasting my time here. Still there's people who think that their time is more important than ours.

> can anybody tell me how to setup a windows realserver? thank you very much! I'm in a hurry.
>
> > robert.gehr@web2cad.de I can't think of anyone who has set up lvs in a hurry :-)

To get technical help:

- Read the docs on the website, the HOWTOs, and search the mailing list archives.
- The LVS-mini-HOWTO shows you how to setup a simple 3 node (client, director, realserver) LVS without you needing to understand how an LVS works. Presumably you will have read the mini-HOWTO before you read this HOWTO :-).
- after you've done a search of the docs, then post to the mailing list. Please don't e-mail me privately with general questions. The mailing list will archive your question and the answer(s) which can be retrieved later. Other people may have more interesting, relevant or useful comments than I will.

  If are writing to me to avoid the public humiliation of showing your ignorance on the mailing list, it's not going to happen. We've had too many good ideas from people who were "ignorant" to let this happen. If your question has been answered many times before and it's in the HOWTO and the archives, you'll just be told to read the HOWTO, that's all.
- updates/problems/bugs - post to the mailing list and cc: mailto:jmack@wm7d.net

1.10 Posting problems/questions to the mailing list

There's always new ideas and questions being posted on the mailing list. We don't expect this to stop.

If you have a problem with your LVS not working, before you come up on the mailing list, please -

- Read the LVS-mini-HOWTO

- Set up a simple LVS (3 nodes: client, director, realserver) with LVS-DR or LVS-NAT forwarding, with the service telnet using the instructions in the LVS-mini-HOWTO. You should be able to do this starting from a freshly downloaded kernel and ipvs-patch pair. Don't setup first with http, with filter rules, with firewalls, with complicated file systems (*e.g.* coda, nfs) or network accelators - debug all these nifty things after you have LVS working with telnet and with no filter rules.
- LVS is distribution neutral. It only needs the kernel (from ftp.kernel.org) and the ipvs patch (from www.linuxvirtualserver.org). If you can get an LVS to work with these files but not with the kernel on your distro, then you have a problem with your distro. If your problem is distribution specific (eg Redhat ships with LVS patches applied, with a version which will be old by the time you get to talk to us; SuSE is planning on doing the same thing) then we may know what the problem is, but it would be better if you contacted your distro - they need the feedback, not us.
- If you are using one of the packages that can be used with LVS (eg heartbeat from the Linux HA project http://www.henge.com/ alanr/ha, or piranha from Redhat), again we may know what the problem is, but they need the feedback that you can't get it to work, not us. Many of us are on each others' mailing lists and we try to help when we can but the best people to handle the problem are the developers for each package.
- Read the LVS-HOWTO
- Consult the *LVS mailing list archives* `<http://marc.theaimsgroup.com/?l=linux-virtual-server&r=1&w=2>`
- Please use our jargon as best you can. The machine names will be client, director, realserver1, realserver2... IPs are CIP, VIP, RIP, DIP. If you do this, we won't have to translate "susanne" and "annie" to their functional names as we scan your posting.
- When you come up on the mailing list we need to know your kernel (eg 2.2.14) and the patch that was applied to it (eg 0.9.11), whether you are using LVS-DR, LVS-NAT or LVS-Tun. Tell us what you did, what you expected and what you got and why that's a problem.
- We didn't know everything when we first started and we don't expect you to be any better at LVS than we were at first, but we'll feel much better about helping you if you've done your homework first. We can't help you if you just tell us that you've read the documents and your LVS doesn't work.

To you, all problems look the same ("it doesn't work"). To help you we need more information. If you don't understand your problem well, here's a suggested submission format from Roberto Nibali `ratz@tac.ch`

1. System information, such as kernel, tools and their versions.

   Example:

   ```
   hog:~ # uname -a
   Linux hog 2.2.18 #2 Sun Dec 24 15:27:49 CET 2000 i686 unknown

   hog:~ # ipvsadm -L -n | head -1
   IP Virtual Server version 1.0.2 (size=4096)

   hog:~ # ipvsadm -h | head -1
   ipvsadm v1.13 2000/12/17 (compiled with popt and IPVS v1.0.2)
   hog:~ #
   ```

2. Short description and maybe sketch of what you intended to setup.

   Example (VS-DR):

```
     o Using LVS-DR, gatewaying method.
     o Load balancing port 80 (http) non-persistent.
     o Network Setup:

                     --------
                    |        |
                    | client |
                    |________|
                        | CIP
                        |
                        |
                        |
                     (router)
                        |
                        |
                        |
                        | GEP
              (packetfilter, firewall)
                        | GIP
                        |
                        |       __________
                        |  DIP |          |
                        +------+ director |
                        |  VIP |__________|
                        |
                        |
                        |
      +-----------------+----------------+
      |                 |                |
      |                 |                |
  RIP1, VIP         RIP2, VIP        RIP3, VIP
 ------------      ------------    ------------
|            |    |            |  |            |
|realserver1 |    |realserver2 |  |realserver3 |
|____________|    |____________|  |____________|


     CIP  = 212.23.34.83
     GEP  = 81.23.10.2       (external gateway, eth0)
     GIP  = 192.168.1.1      (internal gateway, eth1, masq or NAT)
     DIP  = 192.168.1.2      (eth0:1, or eth1:1)
     VIP1 = 192.168.1.110    (director: eth0:110, realserver: lo0:110)
     RIP1 = 192.168.1.11
     RIP2 = 192.168.1.12
     RIP3 = 192.168.1.13
     DGW  = 192.168.1.1      (GIP for all realserver)

     o ipvsadm -L -n
```

```
hog:~ # ipvsadm -L -n
IP Virtual Server version 1.0.2 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.10:80 wlc
  -> 192.168.1.13:80             Route   0      0          0
  -> 192.168.1.12:80             Route   0      0          0
  -> 192.168.1.11:80             Route   0      0          0
hog:~ #

The output from ifconfig from all machines (abbreviated, just need the
IP, netmask etc), and the output from netstat -rn.
```

3. What doesn't work. Show some output like tcpdump, ipchains, ipvsadm and kernlog. Later we may ask you for a more detailed configuration like routing table, OS-version or interface setup on some machines used in your setup. Tell us what you expected. Example:

```
        o ipchains -L -M -n (2.2.x) or cat /proc/net/ip_conntrack (2.4.x)
        o echo 9 > /proc/sys/net/ipv4/vs/debug_level && tail -f /var/log/kernlog
        o tcpdump -n -e -i eth0 tcp port 80
        o route -n
        o netstat -an
        o ifconfig -a
```

1.11 ToDo List

1.11.1 for the HOWTO

Combining HA and LVS (*e.g.* Ultramonkey).

I realise that information in here isn't all that easy to locate yet (there's no index and you'll have to search with your editor) and that the ordering of sections could be improved.

I'll work on it as I have time.

1.12 Other load balancing solutions

from lvs@spiderhosting.com *a list of load balancers* <http://dmoz.org/Computers/Software/Internet/Site_Management/Load_Balancing>

1.13 Software/Information useful/related to LVS

Ultra Monkey <http://ultramonkey.sourceforge.net/> is LVS and HA combined.

from lvs@spiderhosting.com *Super Sparrow Global Load Balancing* <http://supersparrow.sourceforge.net/> using BGP routing information.

From ratz, there's a write up on load imbalance with persistence and sticky bits at our friends at *M$* <http://www.microsoft.com/TechNet/win2000/nlbovw.asp?a=printable>.

From ratz, Zero copy patches to the kernel to speed up network throughput, *Dave Miller's patches* <ftp://ftp.kernel.org/pub/linux/kernel/people/davem>, *Rik van Riel's vm-patches* <http://www.surriel.com/patches> and *more of Rick van Riel's patches* <http://www.linux-mm.org/>. The Zero copy patches may not work with LVS and may not work with netfilter either (from Katejohn@antefacto.com).

From Michael Brown michael_e_brown@dell.com, the *TUX kernel level webserver* <http://www.redhat.com/products/software/webservers/tux>.

From Lars lmb@suse.de *mod_ backhand* <http://www.backhand.org/>, a method of balancing apache httpd servers that looks like ICP for web caches.

A lightweight and simple webbased cluster monitoring tool designed for beowulfs *procstatd* <http://www.phy.duke.edu/brahma/>, the latest version was 1.3.4 (you'll have to look around on this page).

From Putchong Uthayopas pu@ku.ac.th a heavyweight (lots of bells and whistles) cluster monitoring tool *KCAP* <http://smile.cpe.ku.ac.th/software/>

2 Getting Files

Go to the software link on the *LVS homepage* <http://www.linuxvirtualserver.org>. Get the latest patch tarball for the director (available in 2.2.x and 2.4.x flavors). This will contain the kernel patch and source code for various programs and utilities.

2.1 Director Code

You have to patch the standard kernel with the matching ip_vs patch.

The director operates in one (or several) forwarding methods; LVS-NAT (network address translation); LVS-DR (direct routing) and LVS-Tun (tunneling). The mode is chosen individually for each realserver/service by running the user level program ipvsadm.

The director code is well tested on Intel. As well several people are running on Alpha hardware (Flavio Pescuma edtfopa@malmen.ericsson.se on a mailserver, and Philip Copeland copeland@redhat.com who is using it alongside an Intel director to test director failover).

You have to edit the code to run it on Alpha: from Philip (this has been fixed in recent code).

> remember use socklen_t NOT size_t for network calls (See page 27 of R.Stevens UNIX Net. prog. vol1 ed 2) and everything suddenly falls into place. Certainly socklen_t is the correct type to use for portability between 32/64 machines, you'll need to update any other code to use socklen_t thats 64 bit eg ia64 sparc64 etc

2.2 RealServer Code

The realservers must be configured appropriately for the LVS forwarding method. You must

- handle the 3 (arp problem) (VS-DR/VS-Tun; Linux 2.2.x/2.4.x kernels)
- set the default gw

Here's this list of OS's that have been tested with the forwarding methods. (we expect all OS's can be made to work one way or another.)

- Realservers by Mode -

  - LVS-NAT - any OS
  - LVS-DR
    * (from Ratz `ratz@tac.ch`)
    * most OS's (all expected to succumb eventually)
    * Linux 2.0.36, 2.2.x
    * Solaris 2.5.1, 2.6, 2.7;
    * FreeBSD 3.1, 3.2, 3.3
    * NT (although Webserver would crash): 4.0 no SP
    * IRIX 5.3-6.5
    * HPUX 11
  - LVS-Tun - only linux

- realservers by OS -
  - Other OS
    * LVS-NAT - no modifications required to realserver
    * LVS-DR - works for most OS's so far, although some trickery has been required to find the commands neccessary to configure the VIP on the realservers for some OS's. If running Linux-2.2.x, or Linux-2.4.x, you have to handle the 3 (arp problem)
    * LVS-Tun - only linux (realserver must be able to decapsulate IPIP packets). If running Linux-2.2.x, or Linux-2.4.x, you have to handle the 3 (arp problem)
  - Linux 2.0.36: no modifications or patches required. Works with all LVS modes LVS-NAT, LVS-Tun and LVS-DR.
  - Linux 2.2.x,2.4.x:
    * LVS-NAT - no modification to realserver.
    * LVS-DR and LVS-Tun -
      (drum roll - this is the difficult bit, otherwise known as the 3 (arp problem) - see also the section on the 25 (Wisdom of the mailing list)).
      The 3 (arp theory) can be skipped on a first read. You'll need to understand this to set up a working LVS. Here's the summary:

```
IF   (
          (you are using LVS-DR or LVS-Tun on the director)
          AND
          (you are running a Linux 2.2.x, 2.4.x kernel on a realserver)
          AND
          (
                  the VIP on the realserver is on an ethernet device eg lo:0, tunl0:0
                  i.e. is not being accepted by transparent proxy
          )
          AND
          (
                  the realservers can answer arp requests from
                  the client/router (the realservers are on the same
                  piece of wire|network as the director)
          )
     )
THEN
     {
```

```
                YOU MUST HANDLE THE ARP PROBLEM
                }
        FI
```

In general you'll have to handle the 3 (arp problem).

The most common way to handle the arp problem is to hide the VIP from arp requests

- 3.2.1 (2.2.x: use the prepatched 2.2.x kernel in the standard distribution for the realserver)
- 3.2.3 (2.4.x: patch the standard 2.4.x kernel for the realserver).

Although not used very much (I don't know why), another simple method is to put an extra NIC into the realservers for the VIP and not connect it to the network. The NIC doesn't handle any packets, it's just a way of putting the VIP onto the realserver. The NIC can be an old 10Mbps ISA card. The cost of some 100Mbps PCI tulip cards now is less than the salary you'd pay for the time to recompile a 2.4.x kernel with the hidden patch.

All methods of handling the arp problem work, have about the same performance (throughput, latency) and are about equally easy/difficult to setup.

The method of hiding the devices from arp requests is the closest to the standard NOARP unix behaviour and is the method most commonly used on Linux realservers.

2.3 Configure tools(s)

You can set up the LVS by hand with ipvsadm. This is somewhat tedious and error prone. While do-able for a single LVS configuration, this is not the way to go for setting up lots of different LVS configurations.

A 10.1 (configure script) is available. This only sets up a single director and cannot handle director failover. For production systems you will probably want to use the tools that handle 20 (director failover) as well.

2.4 Go setup an LVS with the mini-HOWTO

If you're just reading through the HOWTO and have got to here, you should now setup an LVS using the LVS-mini-HOWTO.

2.5 Upgrading LVS

The only way is to compile a new kernel, reboot, then compile/install the matching ipvsadm, as if you were doing an install from scratch. You can't upgrade in place and you have to bring the director down.

I use the rc.system_map script (comes with the 10.1 (configure_script)) to make sure that I have the correct versions of ipvsadm, kernel and System.map files working together, when I'm testing multiple versions of LVS (*e.g.* a 2.2 and 2.4 version).

3 The arp Problem

3.1 The problem

If you follow the instructions and setup the examples in the LVS-mini-HOWTO, then you don't need to know about the arp problem. If you're going to setup grander LVS's, then you'll need to understand the arp problem.

Although this section comes early in the HOWTO, it has lots of pitfalls. You shouldn't be reading this unless you've at least setup a working VS-NAT and LVS-DR LVS using the canned instructions in the mini-HOWTO.

The LVS allows several machines to function as one machine. For LVS-DR and LVS-Tun some trickery was needed to split the various handshakes etc involved in establishing and maintaining a tcpip connection so that some parts of it came from one machine and other parts from another machine. Most of these problems are handled, and some problems only occur for certain services (eg 17 (identd)) and we've learned to live with them. The worst problem, which ironically only happens with realservers running Linux 2.2.x and 2.4.x kernels, is the "arp problem" (it's just as well we have the source code).

With LVS-DR and LVS-Tun, all the machines (director, realservers) in the LVS have an extra IP, the VIP. Here's a LVS-DR in a test setup where all machines and IPs are on the same physical network (*i.e.* are using the same link layer and can hear each other's broadcasts).

```
                        --------
                       |        |
                       | client |
                       |________|
                            |
                            |
                        (router)
                            |
                            |
                            |       ----------
                            |  DIP |          |
                            |------| director |
                            |  VIP |__________|
                            |
                            |
                            |
        ----------------------------------------
        |                   |                  |
        |                   |                  |
   RIP1, VIP           RIP2, VIP          RIP3, VIP
 ---------------     ---------------     ---------------
|               |   |               |   |               |
| realserver1   |   | realserver2   |   | realserver3   |
|_______________|   |_______________|   |_______________|
```

When the client requests a connection to the VIP, it must connect to the VIP on the director and not to the VIP on the realservers.

The director box acts as an IP router, accepting packets destined for the VIP and then sending them on to a realserver (where the real work is done and a reply is generated). When the client (or router) puts out the arp request "who has VIP, tell client", the client/router must receive the MAC address of the director for the LVS to work. After receiving the arp reply, the client will send the connect request to the director. (The director will update its ipvsadm tables to keep track of the connections that it's in charge of and then forward the connect request packet to the chosen realserver).

If instead, the client gets the MAC address of one of the realservers, then the packets will be sent directly to that realserver, bypassing the LVS action of the director. If the client's packets are consistently sent to the same realserver, then the client will have a normal session connected to that realserver. You can't count on this happening, the MAC address the client gets might change in the middle of a session and a new realserver will start getting packets for connections it knows nothing about (the realserver will send tcp resets). If nothing is done to direct arp requests for the VIP specifically to the director (3.5 (ref)), then in some setups, one particular realserver's MAC address will be in the client/router's arp table for the VIP and the client will only see one realserver. (In my setup, the machine with the fastest CPU is in the client's arp table, suggesting that it's the first machine to reply that gets in. Horms and Steven WIlliams have written that they think it's the last machine to reply whose entry in in the client's arp table.) In other setups where the realservers are identical, the client will connect to different realservers each time the arp cache times out (see comment by Steven WIlliams elsewhere). There the client's connection will hang as the new realserver will be presented with packets from an established connection that it knows nothing about. If the director always gets its MAC address in the router arp table, then the LVS will work without any changes to the realservers (as happened in my case with a director with the fastest CPU in the LVS), although this may not be a reliable solution for production.

Getting the MAC address of the director (instead of the realservers) to the client when the client/router does an arp request is the key to solving the "arp problem".

The arp problem is handled in 2.0.x kernels as serveral devices which don't reply to arp requests (eg dummy0, tunl0, lo:0) were available for the the VIP. For other OS's, the NOARP flag for ifconfig would stop the VIP on the realservers from replying to arp requests.

However with 2.2.x (and now 2.4.x) kernels, the devices which didn't reply to arp requests in 2.0.x, now reply to arp requests. There is a "-arp" (NOARP) option for ifconfig which (according to the man pages) turns off replies to arp requests for that device, and an "arp" option which turns them back on again. Linux does not always honour this flag. You couldn't turn on replies to arp requests for the dummy0 devices in 2.0.36 kernels and you can't turn it off for tunl0 in 2.2.x kernels. eth0 behaves properly in 2.0.36 but in 2.2.x kernels it arps even when you tell it not to arp. This behaviour of not honouring the NOARP flag in the Linux 2.2.x kernels 3.3 (is not regarded as a "problem") by those writing the Linux TCPIP code and is not going to be "fixed".

> Julian 22 May 2001 The flag is used to allow arp requests for the specified device. Although "lo" doesn't reply to arp requests, the requests for the VIP go through eth*, and so the NOARP flag is of no help to us. We can't drop the flag for eth.

Another wrinkle is that in 2.0.36 kernels, aliased devices (eg eth0:1) could be setup independantly of the options on the primary (eth0) device. Thus eth0:1 behaved as if it were on a separate NIC and it's arp'ing behaviour could be set independantly of the primary interface. The settings of an aliased device belonged to the IP. With the 2.2.x kernels, the aliased devices are now just alternate names for each other: you change an option (eg -arp) or up/down of one alias (or primary) the other aliases follow. With 2.2.x kernels, the settings of the aliased device belong to the primary device (there is only one device with several IPs).

When LVS was running on 2.0.36 machines, the VIP was usually configured as an alias (eg lo:0, tunl0) on the main ethernet device (eth0), allowing the nodes in an LVS to have only one NIC.

With 2.2.x kernels care is needed when only one NIC is used on the realserver (the usual case). On a realserver with eth0 carrying the RIP, and the realserver having only one NIC, eth0 must reply to arp requests (to receive packets), then eth0:1 carrying the VIP will reply to arp requests too, even if you ifconfig it with -noarp. Thus if a realserver is running a 2.2.x kernel and has the VIP on an ip_alias, then the VIP on the realserver will reply to arp requests received from the router.

With the 2.4 kernels, the use of ip_aliases is still allowed. However the new tools that come with 2.4 (iproute2 and ip_tables) do not recognise aliases, which have been replaced by secondary IPs.

3.2 The cure(s)

Several cures have been produced in an attempt to solve the arp problem. They involve either

- stopping the realservers from replying to arp requests for the VIP.
- hiding the VIP on the realservers so that they don't see the arp requests.
- priming the client/router in front of the director with the correct MAC address for the VIP.
- allowing the realserver to accept a packet with dst=VIP even though the realserver does not have a device with this IP (*i.e.* the host has nothing to reply to an arp request). This is implemented by transparent proxy or fwmark.
- stopping arp requests for the VIP getting to the realservers.

Pick one -

Note: Some of these cures involve applying a patch to the kernel on the Linux 2.2.x or 2.4.x realserver. The patch (*e.g.* the "hidden" patch), which you apply to the realserver, is different to the patch which you apply to the director (the "ipvs" patch). For the full scoop on the hidden patch see *julian's page* `<http://www.linuxvirtualserver.org/~julian/hidden.txt>`.

3.2.1 2.2.x kernels

The "hidden" patches for kernel >=2.2.14 are now in the standard linux distribution (*i.e.* you can use the "hidden" feature with a standard kernel and don't have to patch the kernel on the realserver anymore). The arp patches allow you to hide a device from arp requests, returning to the no_arp behaviour of the 2.0.x kernels.

To hide devices from arp calls, on the realservers do

```
        #to activate the hidden feature
        echo 1 > /proc/sys/net/ipv4/conf/all/hidden
        #to make lo:0 -arp, put lo here
        echo 1 > /proc/sys/net/ipv4/conf/<interface_name>/hidden
```

To test that the network device (here lo:0) is hidden from arp requests -

- before you hide the lo:0, ping the VIP from another machine, then run 'arp -a' and see that the MAC address for the VIP matches that for eth0 on the realserver
- Clear the entry for the VIP with "arp -d VIP", and show that the arp entry is gone for the VIP (with arp -a)
- ping the VIP and look for the reappearance of the arp entry for the VIP.
- Then hide the lo interface and ping the VIP again from the outside machine. The VIP will most likely reply to the ping since the entry for the VIP is still in the arp table of the outside machine.
- Clear the arp entry (arp -d VIP) and ping the VIP again - this time you'll get no reply.

There is a possible race condition in hiding the VIP -

Kyle Sparger, 15 Feb 2001 I've found an interesting, but not totally unexpected race condition under DR in 2.2.x that I've managed to create when installing VIP's on a machine in DR mode. Basically, the cause is this:

```
ifconfig dummy0 10.0.1.15
echo 1 > /proc/sys/net/ipv4/conf/dummy0/hidden
```

You'll notice that there's going to be a small gap between the two which allows an ARP request to come in, and for the server to reply. And yes, it is big enough to be bitten by – I've been bitten twice by it so far :)

Julian

On boot:

```
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
# For each hidden interface (dummy, lo, tunl):
modprobe dummy0
ifconfig dummy0 0.0.0.0 up
echo 1 > /proc/sys/net/ipv4/conf/dummy0/hidden
# Now set any other IP address
```

Kyle's suggestion

```
echo 1 > /proc/sys/net/ipv4/conf/default/hidden
ifconfig dummy0 10.0.1.15
echo 0 > /proc/sys/net/ipv4/conf/default/hidden
```

The echo 0 command is incase I want to configure other interfaces later that I _do_ want responding to ARP requests. Technically, it's not necessary, I just find it useful in my particular setup.

3.2.2 Older 2.2 kernels (<2.2.12)

These are getting old now and it would better to upgrade. However if you have them, you apply the arp patches to the kernel code of the 2.2.x realservers. These patches are separate from the ipvs patch applied to the kernel on the director.

For kernels <2.2.12, Julian's patch is on the lvs website.

http://www.linuxvirtualserver.org/arp_invisible-2213-2.diff

The patch by Stephen WIllIams is at

http://www.linuxvirtualserver.org/sdw_fullarpfix.patch

This patch is against a 2.2.5 kernel but can be applied to later kernels (tested to 2.2.13). The file appears to have DOS carriage control. Depending what you get on your disk, you may have to convert the file to unix carriage control (with 'tr -d '\015'') (the unix line extension of '\' doesn't work in combination with DOS carriage control).

The whitespace may not match your file so do

```
$ cd /usr/src/linux
$ patch -p1 -l < sdw_fullarpfix.patch
```

(If you are running one of these old kernels, you could upgrade to your kernel.)

If you are using 13.6 (Julian's martian modification) you will need the forward_shared-hidden patch (applied to both director and realservers).

3.2.3 2.4.x kernels

Julian's hidden patch to the standard 2.2.x kernel is not being included in the 2.4.x kernels.

For early 2.4.x kernels (eg x=0), the patch is available at http://www.linuxvirtualserver.org/hidden-2.3.41-1.diff. (This patches a part of the kernel that isn't being actively fiddled with, so hopefully the patch will work against later 2.4.x kernels too.)

The 2.4.x "hidden" patch in now being actively maintained and is included in ipvs-x.x.x/contrib/patches/hidden-x.x.x.diff

Assuming you are patching 2.4.2 with the ipvs-0.2.5 files

```
cd /usr/src/linux
patch -p1 <../ipvs-0.2.5/contrib/patches/hidden-2.4.2-1.diff
```

Then build the kernel (can use same options as for the 2.4 director kernel build).

You activate the hidden feature as for 2.2 (see 3.2.1 (hidden)).

As to why the hidden patch is in the 2.2 kernels but not the 2.4 kernels see the *the mailing list archives* `<http://marc.theaimsgroup.com/?l=linux-kernel&m=98032243112274&w=2>` or for *the thread* `<http://marc.theaimsgroup.com/?t=98019795800013&w=2&r=1>`

3.2.4 Put an extra NIC on the realserver to carry the VIP (on eth1)

Possible cards would be a discarded ISA card (WD80x3), or a cheap 100Mbit PCI card (eg Netgear FA310TX, $16 in USA in Nov 99) There is no traffic going through this NIC and it doesn't matter that it's an old slow card. The extra card is only required so that the realserver can have the VIP on the machine. With 2.2.x kernels you can't stop this device (eth1) from replying to arp requests, but if you don't connect the cable to it or don't put a route to it in the realserver's routing table, then the client won't be able to send it an arp request.

To set this up with the 10.1 (configure script), enter eth1 as the device for the VIP on the realserver.

3.2.5 Put the realservers on a different network to the VIP

Setup routing tables so that the client cannot route to the realserver network 13.2.2 ((Lars' method)). This method requires the director to not forward packets for the VIP (easy to implement if 2 NICS on the director).

3.2.6 On the client(router), set the routing to the VIP to go only to the director

You can hardwire the MAC address of the director as the MAC address of the VIP. You can do this with

```
#arp -s lvs.mack.net 00:80:C8:CA:A7:E4
```

```
or
```

```
arp -f /etc/ethers.
```

Here is my /etc/ethers file (on the client)

```
lvs.mack.net 00:80:C8:CA:A7:E4
```

This requires no extra NICs or patching of realservers. However in a production environment, redundant directors with heartbeat/failover may be required and some method (eg running send-arp) will be needed to change the static arp entry as the failover occurs. If multiple NICs are involved, it is possible that the above instruction will result in a route through the wrong NIC. In this case bring up the NIC of interest first and then run the above command.

Alternately if the router has several NICs, use one for the director and another for the realservers. Route the VIP to the director.

3.2.7 Use transparent proxy allow the incoming packet to be accepted locally - Horms' method.

see LVS-DR and LVS-Tun for details. The 10.1 (configure script) will set this up for you.

3.3 The ARP problem, the first inklings

History: ARP behaviour changed with 2.2.x kernels. Here's the original posting by Wensong and a reply from Alexy Kuznet (2.2 tcpip author)

Wensong Zhang wensong@iinchina.net 24 Mar 1999

Today I upgraded the kernel to 2.2.3 with tunneling support on one of a real server, and found a problem that the Linux 2.2.3 tunnel device answers ARP requests. Even if I used the NOARP options as follows:

```
realserver:# ifconfig tunl0 172.26.20.110 -arp netmask 255.255.255.255 broadcast 172.26.20.110
```

It still answers the ARP requests. This will greatly affect the virtual server via tunneling work properly. In fact, the tunnel device shouldn't answer the ARP requests from the ethernet. I think it is a bug of linux/net/ipv4/ipip.c, which is now a clone of ip_gre.c not the original tunneling code.

If you are interested, you can test yourself on kernel 2.2.3, choose a free IP address of your ethernet and configure it on the tunl0 device, then telnet to that IP address from other host, I guess you can. Finally, have a look at the ipip.c, maybe you can debug it. :-) –

But, what is the IFF_NOARP flag of the tunnel device for?

> kuznet@ms2.inr.ac.ru IFF_NOARP means that ARP is not used by THIS device. On normal IPIP tunnels it does not make much of sense, but may be used f.e. to turn on/off endpoint reachability detection.
>
> I do not see any reasons to disable answering ARP in such curcumstances. Isolation of VPNs on adjacent segments is impossible at routing/arp level, it is just not well-defined behaviour.
>
> If the isolation is made with firewall policy rules, then it is clear that arp policy must be handled at this level too.

In kernel 2.0.x, the tunnel device doesn't answer ARP requests.

> Yes.

Yeah, we can have link-local addresses that doesn't answer ARP requests in kernel 2.2.x. For example, we can configure all the hosts in a network with the following command:

```
ifconfig lo:0 192.168.0.10 up
```

There will no collision. The lookback alias interfaces don't answer ARP requests.

> Are you sure? I am not. Please, test. BTW you risk adding non-loopback addresses on loopback device. They have the HIGHEST preference to be used as router identifier. so that VPN addresses cannot be added to loopback at all.

No, it doesn't fail. I tested it with kernel 2.0.36, it worked.

> It does not work under 2.2. To be honest, I am about to stop to understand you. You talk about 2.2, but all your tests are made for 2.0. 8)

3.4 A posting to the mailinglist by Peter Kese explaining the "arp problem"

(saved for posterity by Ted Pavlic, minor editing by Joe)

`peter.kese@ijs.si`

Before we start, let's assume we have following network configuration for an LVS running LVS-DR.

```
client          10.10.10.10

gw              192.168.1.1

director        192.168.1.10     IP for admin (director IP)
                192.168.1.110    VIP (responds to arp requests)

real server     192.168.1.11     IP to which each service is listening (realserver IP)
                192.168.1.110    VIP (DOES NOT respond to arp requests)
```

The virtualserver is the combination of the director and the realserver running LVS.

Or goal is:

1. Virtual server should respond to arp requests for both the VIP and the director IP.
2. The realserver should respond to arp requests for the realserver IP but NOT the VIP.
3. Gateway sends packets for the VIP to the director IP load balancer no matter what.

Problem 1: Interface aliases

Realserver and director need to have an interface with the VIP in order to respond to packets for virtual server. A real interface is not needed, an IP alias will do just fine and this interface alias could be either eth0:0 or lo:0.

On the 2.0 kernels, the ARP responding ability of an interface alias (eg eth0:0) could either be enabled or disabled independantly of the main (eth0) interface. If you wanted eth0:0 not to respond to ARP requests, you could simply say:

ifconfig eth0:0 192.168.1.2 -arp up

Thus in the 2.0 kernels it is possible, on a realserver, to have the realserver IP (on eth0) respond to arp requests and for the VIP (on eth0:0) to not respond.

In the 2.2 kernels this doesn't work any more. Whether the an interface alias responds to ARP requests or not, depends only on the way the real interface is configured. So if eth0 responds to ARP requests (which it normally will), eth0:0 carrying the VIP will also respond to ARP requests no matter what.

This means an ethernet alias (eth0:0) is not permitted on real servers, because real servers should not respond ARP requests.

On the other hand, loopback aliases never respond ARP requests, which means that the loopback alias (lo:0) must not be used on the director for the VIP.

Problem 2: Loopback aliases

I haven't done much checking on loopback interface problem, but it seems that if an alias is used on a loopback interface (as is required for LVS-DR) on a real server running kernel 2.2.x, the whole ARP gets screwed.

It appears that loopback interfaces get special ARP treatment in the kernel, so I suggest avoiding the loopback aliases as whole.

The question now is: What kind of an interface can I use on real servers?

As I already noted, eth0:0 alias can not be used, because such aliases respond to ARP requests. lo:0 aliases can not be used, because they make ARP problems too.

In case of tunneling VS configuration, the answer is trivial: tunl0. But to be honest, tunl0 interface can also be used for direct routing.

(from Joe, the dummy device is OK too)

With direct routing, the only thing we need an interface for is to let kernel know we posses an additional IP address. This means, we can set up any kind of an interface, as long as it doesn't respond ARP requests. Instead of tunl0, you could also set up a ppp0, slip0, eth1 or whatever. I suggest setting up a tunl0:

ifconfig tunl0 192.168.1.2 -arp up

Problem 3: Real server ARP requests.

Suppose we have set up a virtual server as described at the beginning. All computers are running, but no requests have been made.

Then the client sends a request to the VIP.

When the packet arrives to gateway, the gateway makes an ARP query for the VIP and the director responds. Gateway remembers the director's MAC address and sends the packet to the director. Director receives the packet, looks up its ipvsadm/LVS tables and chooses the real server and forwards the packet to the real server by direct routing or tunneling method.

Real server receives the packet and generates a response packet with destination=client, source=VIP.

(until now everything works correctly)

When real server wants to send the response packet to the gateway, it finds out, that it does not know the gateway's MAC address.

It sends an ARP request to the local network and asks for the gateway MAC address. This should look like:

ARP, who has 192.168.1.1 (gw), tell 192.168.1.11 (realserver IP)

But in reality, real server asks something like:

ARP, who has 192.168.1.1 (gw), tell 192.168.1.110 (VIP), because it takes the source address from the packet it wants to send.

Here the problems come in.

Gateway receives the packet and responds to it, which is correct. But at the same time, gatweay does a little optimization. It finds out, that the realserver's MAC address is not listed in its ARP tables and adds the entry into the table, just in case it might need that address in the near future.

The ARP request contained the VIP address and the realserver's MAC address, so from now on, the gateway will send all packets destined for the VIP to the real server instead (due to MAC address). This means all packets that follow will avoid the virtual server as whole and get responded by the realserver.

If the real server's ARP request would be:

ARP, who has 192.168.1.1 (gw), tell 192.168.1.11 (realserver IP)

all this would not have happened. Therefore I have patched the 2.2 VS kernel in such a way, that it composes ARP requests based on the address of the interface selected by the routing tables instead of the address taken from the packet itself.

In order for virtual server to work correctly, the real servers should have patched kernels as well, or at least copy the patched /usr/src/linux/net/ipv4/arp.c file to the real servers before compiling the kernels.

Conclusion

Those were my experience with ARP problems, and the 2.2 kernel virtual server.

I think it would be wise to add this letter to the web site and notify the network developers about our findings at some point in time.

Here are some golden rules I stick to, when I do virtual server configuration:

```
Rule 1:
        Do not use lo:0 alias on the director.
        Use eth0:0 alias instead.

Rule 2:
        Avoid using lo:0 alias, not even on realservers.
        Use tunl0 or some other simulated interface
        on real servers instead. (Joe: use dummy0)


Rule 3:
        Apply the VS patch to kernels on real servers.
```

3.5 random mailings on the arp problem

symptoms of realservers arp'ing:

> Stephen WIlliams `sdw@lig.net`, Stephen wrote one of the patches that stop devices in 2.2.x kernels from replying to arp requests If you don't use the patch you'll find that the 'active' box will bounce from machine to machine as each one sends an ARP reply that is heard last. Additionally you will get TCP Reset's as connections that were on one box suddenly start going to others. Very nasty and unusable.

This is called 13.2.2 (Lars' method)

> Lars I have thought about how the ARP problem can occur at all with direct routing, because I never noticed it. Then it occured to me that your VIP comes from the same subnet as the RIP of the LVS and also all the real servers share this media.
>
> To avoid the "ARP problem" in this case without adding a kernel patch or anything else, you can just add a direct route for the VIP using the RIP of the LVS as a gateway address on the router in front of the LVS. ("ip route VIP 255.255.255.255 real_ip" on a Cisco, or "route add -host VIP gw RIP" on Linux)
>
> Since I just used 2 ethernet cards and had the LVS act as gateway/firewall anyway, I never noticed the ARP problem. (We have 2 LVS in a standby configuration to eliminate the SPOF)

The arp problem is handled if the router in front of the director has a static route for the VIP to the director (*i.e.* packets for the VIP from the outside world are sent to the director and cannot get to the realservers.

> Wensong For the clients who reach the virtual server through the router, there is no problem if a static route for VIP is added.
>
> However, for the clients who are in the network of virtual server, the "ARP problem" will arise. There is fight in ARP response, and the clients don't know send the packets to the load balancer or the real server.
>
> In my point of view, the VIP address is shared by the director and realservers in LVS-Tun or LVS-DR, only the director does ARP response for VIP to accept request packets, and the realservers has the VIP but don't, so that they can process packets destined for VIP.

Joe, 21 May 2001

Was looking at the ip notes and it says

```
ip arp on|off

--change NOARP flad on the device

1cm NB. This operation is not allowed if the device is in state UP.
Though neither ip utility nor kernel check for this condition, you can
get unpredictable results changing the flag while the device is running.
```

> Julian Anastasov `ja@ssi.bg` 21 May 2001 This is the device ARP flag, same as ifconfig [-]arp. The flag is used to allow ARP packets for the specified device. It is correct that "lo" does not talk ARP, but you connect to the VIPs on "lo" through eth*, so the flag is of no help for LVS. We can't drop the flag for eth device.

Andreas J. Koenig, 02 Jun 2001

kernel 2.4.5 has arp_filter

> Julian Anastasov `ja@ssi.bg` arp_filter does not solve the ARP problem for LVS
>
> This is a new proposal to control the ARP probes and replies based on route flag "noarp". It will be discussed on the netdev mailing list and may be something like this is going to be included in 2.4, may be in 2.2 too, not sure. All you know that the hidden feature is not considered to 2.4. The net developers have the final word. I'll try to maintain the hidden flag in all next kernels while this flag is more usable than the new feature and because the hidden flag has other semantic. And because may be there are some user space tools that rely on this.

3.6 Is the arp behaviour of 2.2.x kernel a bug?

Julian Anastasov is replying to correct an error in a previous version of the HOWTO where I state that the dummy0 device in 2.2.x kernels does not arp. Julian wrote one of the realserver patches which fix the "arp problem".

> In fact, the documentation is incorrect. There is no difference, all devices are reported in the ARP replies: lo, tunl and dummy. So, only the ARP patch can solve the problem. This can be tested using this configuration with any device (before the patch applied):
>
> ```
> Host A:
>         eth:x 192.168.0.1
>
> Host B:
>         eth:x 192.168.0.2
>         lo, dummy, tunl: 192.168.0.3
> ```
>
> On host A try: ping 192.168.0.3
> Host B replies for 192.168.0.3 through 192.168.0.2 device
> So, the ARP problem means: "All local interfaces are reported" until the ARP patch is used. In fact, all ARP patches which use IFF_NOARP to hide the interface are incorrect. I don't expect them in the kernel.

Stephen WIlliams, who wrote another of the patches to fix the arp problem.

Of course the ARP code in the kernel needs to be fixed so my filter code isn't needed. Still, I'm confused by this statement. The IFF_NOARP flag determines whether a device arp replies or not. What's wrong with honoring that?

If you mean that arp replies should never be sent on another interface, that is what I currently believe to be correct.

> Julian My understanding is that 2.2.x ARP code is not buggy and there is no need to be "fixed". I must say that your patch is working for the LVS folks but not for all linux users.
> IFF_NOARP means "Don't talk ARP on this device", from the 'man ifconfig':
> [-]arp Enable or disable the use of the ARP protocol on this interface.
> So, where is the bug ? The ARP code never talks through lo, dummy and tunl devices when they are set NOARP. It uses eth (ARP) device. If You hide all NOARP interfaces from the ARP protocol this is a bug. One example:
>
> ```
>  +--------+ppp0                          +------+
>  | Host A |------------ppp link----------|ROUTER|------ The World
>  +--------+A.B.C.1 (www.domain.com)      +------+
>    |eth0
>    |A.B.C.2
>    |
>    |A.B.C.3
>  +--------+
>  | Host B |
>  +--------+
> ```
>
> Is it possible after your patch Host B to access www.domain.com ? How ? Host A doesn't send replies for A.B.C.1 through eth0 after your patch. OK, may be this is not fatal. Tell it to

all kernel users. You hide all their NOARP interfaces. May be there are other examples where this is a problem too. Or may be there is something wrong in this configuration?

I want to say that this patch hurts all users if present in the kernel. On Nov 6 I posted one patch proposal to the linux-kernel list which adds the ability to hide interfaces from the ARP queries and replies. But the difference is that only specified interfaces are not replied, not all NOARP interfaces. Its arp_invisible sysctl can be used by LVS folks to hide lo, tunl or dummy interfaces but this feature doesn't hurt all kernel users. I think, this patch is more acceptable and can be included in the 2.2 kernel, may be after some tunning. And I'm still expecting comments from the net folks and from all LVS users.

3.7 How to tell if an interface is replying to arp requests

on the machine with that IP (usually the VIP)

$ ping VIP

look in /proc/net/arp for MAC address

on a machine on a network (eg 192.168.1.0/24) to see which addresses are replying to arp requests

$ ping 192.168.1.255

then before the arp tables expire (15secs - 2mins depending on the OS)

$ arp -a

3.8 Arp caching defeats Heartbeat switchover

From: Claudio Di-Martino `claudio@claudio.csita.unige.it`

I've set up a VS using direct routing composed of two linux-2.2.9 boxes with the 0.4 patch applied. The load balancer acts as a local node too. I configured mon to monitor the state of the services and update the redirect table accordingly. I also configured heartbeat so that when the load balancer fails the second machine takes over the virtual ip, sets up the redirect table and starts mon. When the load balancer restarts, the backup reconfigures itself as a real server, drops the interface alias that carries the virtual ip, stops mon, clears the redirect table. Although the configuration of the two machines is set up correctly it fails to restore the load balancer due to arp caching problems.

It seems that the local gateway keeps routing requests for the virtual ip to the load balancer backup. Sending gratuitous arp packets from the load balancer doesn't have effect since the interface of the backup is still alive and responding.

Has anyone encountered a similar problem and is there a hack or a proper solution to take back control of the virtual ip?

From: "Antony Lee" `AntonyL@hwl.com.hk`

I am new to LVS and I have a problem in setting up two LVSes for failover issue. The problem is related to the ARP caching of the primary LVS' MAC address in the real servers and the router connected to the Internet. The problem leads all the Internet connections stalled until all ARP caching in Web Servers and router to be expired. Can anyone help to solve the problem by making some changes in the Linux LVS ? (It is because I am not able to change the router ARP cache time. The router is not owned by the Web hosting company not by me.)

In each LVS, there are two network card installed. The eth0 is connected to a router which is connected to the Internet. The eth1 is connected to a private network which is the same segment as the two NT IIS4.

```
The eth0 of the primary LVS is assigned an IP address 202.53.128.56
The eth0 of the backup LVS is assigned an IP address 202.53.128.57
The eth1 of the primary LVS is assigned an IP address 192.128.1.9
The eth1 of the primary LVS is assigned an IP address 192.128.1.10

In addition, both primary and backup LVS have enabled the IPV4 FORWARD and
IPV4 DEFRAG. In the file /etc/rc.d/rc.local the following command was also
added:
ipchains -A -j MASQ 192.168.1.0/24 -d 0.0.0.0/0
```

I use the piranha to configure the LVS so that the two LVS have a common IP address 202.53.128.58 in the eth0 as eth0:1. And have a IP address 192.128.1.1 in the eth1 as eth1:1

The pulse daemon is also automatically be run when the two LVSes were booted.

In my configuration, the Internet clients can still access to our Web server with one of the NT was disconnected from the LVS. The backup LVS –CAN AUTOMATICALLY– take up the role of the primary LVS when the primary LVS is shut down or disconnected from the backup LVS. However, I found that all the NT Web Servers cannot reach the backup LVS through the common IP address 192.128.1.1, and all the Internet clients stalled to connect to our web servers.

Later, I found that the problem may due to the ARP caching in the Web Servers and router. I tried to limit the ARP cache time to 5 seconds in the NT servers and half of the problem has solved ,i.e. the NT Web servers can reach the backup LVS through the common IP address 192.128.1.1 when the primary LVS was down. However, it is still cannot be connected through the Internet clients when the LVS failover occur.

> Wensong I just tried two LVS boxes with piranha 0.3.15. When the primary LVS stops or fails, the backup will take over and send out 5 Gratuitous Arp packets for the VIP and the NAT router IP respectively, which should clean the ARP caching in both the web servers and the external router.
>
> After the LVS failover occurs, the established connections from the clients will be lost in the current version, and the clients need to re-connection the LVS.
>
> ```
> .. 5 ARP packets for each IP address, and 10 for both the VIP and
> the NAT router IP. I saw the log file as follows:
>
> Mar  3 11:12:14 PDL-Linux2 pulse[4910]: running command "/sbin/ifconfig" "eth0:5" "192.168.10.1
> Mar  3 11:12:14 PDL-Linux2 pulse[4908]: running command "/usr/sbin/send_arp" "-i" "eth0" "192.1
> Mar  3 11:12:14 PDL-Linux2 pulse[4913]: running command  "/sbin/ifconfig" "eth0:1" "172.26.20.1
> Mar  3 11:12:14 PDL-Linux2 kernel: send_arp uses obsolete (PF_INET,SOCK_PACKET)
> Mar  3 11:12:14 PDL-Linux2 pulse[4909]: running command "/usr/sbin/send_arp" "-i" "eth0" "172.2
> Mar  3 11:12:17 PDL-Linux2 nanny[4911]: making 192.168.10.2:80 available
> ```
>
> I don't know if the target addresses of the 2 send_arp commands are set correctly. I am not sure if it is different when broadcast or source IP is used as target address, or any target address is OK.

Horms

Are there just 5 ARPs or 5 to start this and then more gratuitous ARPs at regular intervals. If the gratuitous ARPs only occur at fail-over then once the ARP caches on hosts expire there is a chance that a failed host - whose kernel is still functional - could reply to an ARP request.

`wanger@redhat.com` When we put this together, I talked to Alan Cox about this. His opinion was that send 5 ARPs out at 2 seconds apart. If there is something out there listening and cares, then it will pick it up.

The way piranha works, as long as the kernel is alive, the backup (or failed node) will not maintain any interfaces that are Piranha managed. In other words, it removes any of those IPs/interfaces from its routing table upon failure recovery.

3.9 The device doesn't reply to arp requests, the kernel does.

ARP requests/replies are thought of as coming from a device and people make statements like

"the dummy device in 2.0.x kernels does not reply to arp requests while the same device in 2.2.x kernels does reply".

It is the kernel that handles arp requests according to a set of rules and not the device. The code for the dummy device is the same in 2.0.x and 2.2.x kernels and is not responsible for the change in arp behaviour.

(The RPC for ARP is at ftp://ftp.isi.edu/in-notes/std/std37.txt. - also see rfc826 and rfc1122. The model system used there is 2 machines on a single ethernet. It doesn't shed any light on the implementation of ARP on multi-interface systems like LVS.)

3.10 Properties of devices for the VIP

In a previous version of the HOWTO I stated that the dummy0 device did not arp in 2.2.x kernels and therefore could be used as the device for the VIP on an unpatched 2.2.13 realserver. Julian Anastasov replied that they did arp (see below for his posting and the ensuing discussions).

I hadn't actually tested whether the dummy0 device arp'ed but had concluded that it wasn't arp'ing because I had a working LVS using the dummy0 interface for the VIP on unpatched 2.2.x realservers and because as everyone knows ;-) an LVS needs to have a non-arp'ing device on the VIP of the realservers.

I had a LVS-DR LVS which worked with dummy0, lo:0 and tunl0 as the VIP device and which on further testing, I found also worked with eth0:1 or eth1 as the VIP device on 2.2.13 realservers. Whatever the arp'ing status of dummy0, lo:0 or tunl0, clearly eth1 replies to arp requests, so despite the conventional wisdom, it is possible to build an LVS with arp'ing VIP's on the realservers.

On investigating why this LVS worked, I found that the MAC address for the VIP in the client's arp cache (# arp -a) was always the director. I assume this was because the director is 3-4x the speed of the other machines in the LVS and it replies to arp requests first for the VIP (another posting from Stephen WIlliams says that the address which replies last is stored in the arp cache - we'll figure out what's really going on here eventually). On another LVS where the realservers were all identical hardware with 2.2.13 unpatched kernels, one particular realserver always was the machine in the client's arp cache for the VIP (to check, delete entry for VIP with arp -d, then ping again, then look in arp cache).

I found that I could get a working LVS using almost anything to hold the VIP on the realservers, including eth0:1 and eth1 (another NIC in the realserver). These devices carrying the VIP were pingable from the client and I could get the corresponding MAC addresses in the arp table of the client if the director was not setup with a VIP. When I setup a working LVS this way, I found each time that the MAC address for the VIP in the client's arp cache was the director's MAC address. For some reason, that I don't know, whenever the client does an arp request for the VIP, it gets the director's MAC address.

Possible reasons for the MAC address of the director always being associated with the VIP in my LVS -

1. I configure the director first and then the realservers. I don't make requests for a service till the realservers are setup. (Still I can't imagine the client asking for the MAC address of the VIP until it makes a connect

request.)

2. The director is 3 times faster (CPU speed) than the next machine in the LVS and it always replies to arp request first.

3. I was lucky.

Since you can make a working LVS-DR LVS with the realserver VIP on an arp'ing eth0:1 device I decided that the relevent piece of information about arp'ing was (ta da!)

* *an LVS will work if the client always gets the MAC address of the director when it asks for the MAC address of the VIP* *

This provides an easy solution - you tell the client (or the router) the MAC address of the VIP with arp -s or arp -f .

here's my /etc/ethers

```
lvs.mack.net 00:A0:CC:55:7D:47
```

After installing the MAC address of the DIP (director) as the MAC address of the VIP (lvs) in the arp table ('$arp -f /etc/ethers') I get

```
client:/usr/src/temp/lvs# arp -a
realserver1.mack.net (192.168.1.1) at 00:90:27:66:CE:EB [ether] on eth0
lvs.mack.net (192.168.1.110) at 00:A0:CC:55:7D:47 [ether] PERM on eth0
director.mack.net (192.168.1.10) at 00:A0:CC:55:7D:47 [ether] on eth0
```

notice the "PERM" in the VIP entry on the client.

removing the permanent entry

```
client:/usr/src/temp/lvs# arp -d lvs.mack.net
client:/usr/src/temp/lvs# arp -a
realserver1.mack.net (192.168.1.1) at 00:90:27:66:CE:EB [ether] on eth0
lvs.mack.net (192.168.1.110) at <incomplete> on eth0
director.mack.net (192.168.1.10) at 00:A0:CC:55:7D:47 [ether] on eth0
```

If I edited /etc/ethers changing the MAC address of lvs to anything else, the LVS did not work anymore. So the arp information is coming from /etc/ethers rather than some uncontrolled variable I'm not aware of.

I had thought that in an LVS with the VIP on realservers on an arping device that the VIP would hop from one machine to another (see the postings in the MISC section). Since naturally occuring LVS's with arping VIP's on realservers existed and worked well (mine), I set up an LVS by making a permanent entry for the VIP of the director in the arp cache of the client (router). This can be done by

```
$ arp -f /etc/ethers
```

or

```
$ arp -s 192.168.1.110 MAC_ADDRESS
```

There are 2 results of this

1. the realservers can have the VIP on an an arp'ing device (eg eth0:1, eth1) - you don't need lo or dummy0, tunl0 for realservers with 2.0.36 and 2.2.x kernels.

2. If two (or more) directors are setup in failover mode, the mechanism by for changing the VIP from one to another is broken by making a permanent entry for VIP on the director in the arp cache of the router. This is not a problem for a test setup to demonstrate an LVS but may be a problem in a high availability environment (a solution may be found n the meantime too).

The normal method for changing directors (eg with heartbeat) includes a gratuitous arp. To force a gratuitous arp

> Julian You can use Yuri Volobuev's send_arp.c from the 'fake' package or Alexey Kuznetsov's arping from its iputils package:
>
> - fake - http://vergenet.net/linux/fake/
> - iputils - ftp://ftp.inr.ac.ru/ip-routing/iputils-ss991024.tar.gz iputils is also used for IPAT, IP address takeover

Here's some tests I did

```
LVS equipment: 2.2.13 client, and 0.9.4/2.2.13 director.
2 realservers
a) 2.0.36 kernel, libc5, gcc-2.7.2.3, net-tools 1.42.
b) 2.2.13 kernel, glibc, gcc-2.95,    net-tools 1.52
```

Experiment 1: Result - arp'ing is independant of [-]arp

Summary: the -arp/+arp option for ifconfig had no effect on any devices back to 2.0.36 kernels with net-tools 1.42. If it normally arps then -arp had no effect, if it normally doesn't arp, than "arp" doesn't turn it on (data below).

Method: IP=192.168.1.1/24 with VIP=192.168.1.110/32. The VIP was on dummy0. The test was to see if the VIP was pingable from another (external) machine on the 192.168.1.0/24 network or pingable from the machine itself (ie internally from the console). (I assume I had a route add -host for the VIP although I didn't record this). The test was done with ifconfig using arp or -arp (the output of ifconfig -a didn't change)

```
                 -----2.0.36------- -----2.2.13------
ping from        internal  external internal external
VIP device
dummy   ARP        +         -        +        +
        NOARP      +         -        +        +
        down       -         -        -        - (control)
```

Experiment2: Can the VIP be on a separate NIC?

Summary: yes, as long as the NIC doesn't have a cable plugged into it.

Method: same as above except VIP on eth1 (another NIC).

```
                 -----2.0.36-------
ping from        internal  external
VIP device
eth1 has cable connected to 192.168.1.0 network
eth1    ARP        +         +
        NOARP      +         +
eth1 cable to network removed
eth1    ARP        +         -
        NOARP      +         -
        works as realserver in LVS - yes
```

One of the reasons an no_arp interface is used on the realserver is that it is not visible to the rest of the network. Does the LVS work if the eth1 VIP on the realserver is not visible to the rest of the network?

Conclusion: for 2.0.36 dummy0 doesn't arp, and eth1 does arp. the arp/-arp option to ifconfig has no effect on arp behaviour. LVS works with both dummy0 and eth1, I assume since VIP need only be resolved as local on the realserver and does not need to be visible to the network.

Experiment 3: What devices and netmasks are neccessary for a working LVS?

Using the /etc/ethers approach for setting the MAC address of the VIP I then set up an LVS with pair of realservers serving telnet. All IPs are 192.168.1.x, all machines have a route to 192.168.1.0 via eth0. There is no default route.

```
1. 2.0.36, libc5, gcc 2.7.2.3, net-tools 1.42
2. 2.2.13, glibc-2.1.2, gcc-2.95, net-tools 1.52
```

with the following devices holding the VIP, tunl0, eth0:1, lo:0, dummy0, eth1. In each case there was no route entry for the VIP device and there was no cable connected to eth1 when it was used for the VIP. The table below shows whether the LVS worked. The VIP is installed with

ifconfig $DEVICE 192.168.1.110 netmask $NETMASK broadcast $BROADCAST

```
with $NETMASK="255.255.255.255" $BROADCAST="192.168.1.110"
or   $NETMASK="255.255.255.0"   $BROADCAST="192.168.1.255"
```

the result belong to 1 of 3 groups

```
+ works fine
- doesn't work (at $ prompt on client get
  "unable to connect to remote host.  Protocol not available"
  then client returns to regular unix $ prompt)
hang - client hangs, realserver cannot access network anymore,
  have to run rc.inet1 from console prompt on realserver to
  start network again.
```

netmask of VIP=255.255.255.255 (normal LVS setup)

```
LVS type  -----VS-Tun------     ----VS-DR------
kernel    2.0.36     2.2.13     2.0.36   2.2.13

VIP on
tunl0       +          +          +         +
eth0:1      +          -          +         +
lo:0        +          -          +         +
dummy0      +          -          +         +
eth1        +          -          +         +
```

netmask of VIP=255.255.255.0 (not normally used for LVS)

```
VIP on
tunl0      +           +         +         +
eth0:1     +           -         +         +
lo:0       +           hangs     +         hangs
dummy0     +           -         +         +
eth1       +           -         +         +
```

It would seem that any device and any netmask can be used for the VIP on a 2.0.36 realserver for both LVS-Tun and LVS-DR.

For 2.2.13 realserver, VS-Tun, VIP on a tunl0 device only, any netmask (ie you need tunl0 on LVS-Tun with 2.2.x kernels)

```
VS-DR,  lo:0 device netmask /32 only
       all other devices any netmask
```

For LVS-DR then on solaris/DEC/HP/NT... LVS can probably use a regular eth0 device rather than an lo:0 device (more work for Ratz to do :-).

Does anyone know why the lo:0 device has to be /32 for LVS-DR on kernel 2.2.13 while the other devices can be /24?

Jean-Francois Nadeau `jna@microflex.ca` 6 Dec 99

In kernel 2.2.1x with a virtual interface on lo:0 and netmask of 255.255.255.0 that the interface no longer arps.

Does anyone know why only the tunl0 device works for VS-Tun on 2.2.x kernels?

Experiment 4: Effect of route entry for VIP and connection to VIP. The VIP normally has an entry in the routing table eg

route add -host 192.168.1.110 $DEVICE

I found in Experiment 2 that a route entry was not neccessary for the LVS to work when the realserver had the VIP on eth0:1. Since I had always used a route entry for the VIP I wanted to find out when it was needed. The same LVS was used as for Experiment 3. The variables were

```
1) a route entry/no route entry for VIP/32
2) for eth1 whether the NIC was connected to the network by a cable.



kernel            ------2.0.36-------     -------2.2.13-------
VIP               eth1 eth1_nc eth0:1     eth1  eth1_nc eth0:1

no route
   LVS             +     +      +          +      +       +
   ping internal   -     -      -          +      +       +
   ping external   +     -      +          +      +       +

route
   LVS             +     +      +          +      +       +
   ping internal   +     +      +          +      +       +
   ping external   +     -      +          +      +       +
```

Conclusion 1: LVS works when for both cases of route/no_route for the VIP for eth0:1 and eth1 (ie you don't need a route entry for the VIP on the realservers).

Conclusion 2: having a network cable/no network cable does not affect whether the LVS works.

Conclusion 3: for 2.0.36 kernels you can choose to have the VIP pingable from the outside world but not pingable by the local host by having it on eth1 with a cable connection (this seems wierd and I can't think of any use for it just yet) or the reverse - pingable from the localhost but not by the external world by not have a cable connection.

(Note: using a hosts routable IP as the target - the IP on eth0 say - you can make a host unpingable from the console if you down the lo. The host is still pingable from elsewhere on the net.)

3.11 a 2nd NIC doesn't work for 2.4 kernels

> Jean Paul Piccato `j.piccato@studenti.to.it` I'm setting up a DR_LVS with a director and two servers... I've to handle the ARP problem so I've put two NIC on the two realservers...

Julian Anastasov `ja@ssi.bg` 16 Jan 2002

This works maybe only with Linux 2.0. For 2.2+ you need specific kind of ARP control:

http://www.linuxvirtualserver.org/ julian/#hidden

In Linux 2.2+ the operation of adding IP address involves the following 2 steps:

1. Define a local IP address as a host property - remote hosts can talk to it through any device
2. Define network link route on the specified device - you can talk with other hosts from this local network only through this device

(1) allows the Linux 2.2+ box to send ARP replies through any device that received the reply. Additionally, the user can provide some filtering by setting some device specific values:

```
/proc/sys/net/ipv4/conf/*/<FLAG>
```

These are explained in /usr/src/linux/Documentation/networking/ip-sysctl.txt

The LVS setups depend mostly on the FLAGs rp_filter, hidden, arp_filter, send_redirects. On problems check them after learning what they mean and how they can kill your setup.

By setting rp_filter or arp_filter on some device you can ignore the ARP requests (and the traffic if rp_filter is set) coming from addresses if we don't have a route to these addresses through the mentioned above device.

The send_redirects values must be checked for setups playing with NAT on one physical medium.

Information on using the hidden patch is in hidden.txt

> It seems that eth0 reply to the server instead of eth1

Any device can reply if the ARP probe is not filtered. See hidden.txt from the above URL

3.12 Topologies for LVS-DR and LVS-Tun LVS's

3.12.1 Traditional

The conventional LVS-DR/VS-Tun topology which allows maximum scalability has each realserver with its own default gateway (to a router). (In a routerless test setup, the client would be the default gateway for the realservers. In a setup which is not network bound, *i.e.* is disk- or compute-bound, only one router may be needed. The changes in topology/routing are made by changing the IP of the default gw for the realservers)

Some method of handling the arp problem is needed here.

The packets sent to the realservers from the director, generate replies which go directly to the client. Failure messages (eg if a realservers is not available) do not get returned to the director, who cannot tell if a realserver has failed (see discussion of 21.12 (monitoring agents)).

```
                       -------------clients-----------------------
                       |                         |       |       |
                    (router)                  (router)(router)(router)
                       |                         |       |       |
          ---------    |                         |       |       |
        |          |   |    VIP                  |       |       |
        | director |---     DIP                  |       |       |
        |__________|   |                         |       |       |
                       |                         |       |       |
                       |                         |       |       |
        ---------------------------------        |       |       |
        |              |                |        |       |       |
        |              |                |        |       |       |
       RIP1           RIP2             RIP3      |       |       |
       VIP            VIP              VIP       |       |       |
 -------------   -------------   -------------   |       |       |
|             | |             | |             |  |       |       |
| realserver  | | realserver  | | realserver  |  |       |       |
|_____________| |_____________| |_____________|  |       |       |
        |              |                |        |       |       |
        |              |                ----------       |       |
        |              -----------------------------------       |
        ----------------------------------------------------------
```

3.12.2 Director sees replies

(from Julian Anastasov)

This discussion led to Julian's 13.6 (martian modification).

If the default gw for each realserver is changed to the DIP (see the Martian modification section) then

1. The director has to handle the reply packets as well as in the incoming packets, doubling the network load.

2. The director sees all the reply packets. Connection failure can be detected (in principle).

```
                        clients
                           |
                         router
                           |
             ----------    |
            |          |   |    VIP
            | director |---     DIP
            |__________|   |
                           |
                           |
          ------------------------------------
          |                |                 |
          |                |                 |
         RIP1             RIP2              RIP3
         VIP              VIP               VIP
   -------------    -------------     -------------
  |             |  |             |   |             |
  | realserver  |  | realserver  |   | realserver  |
  |_____________|  |_____________|   |_____________|
```

Here's the original posting by Horms `horms@vergenet.net`

Hi, I have been setting up a test network to benchmark IPVS, the topology is as follows.

```
       node-1      node-6     node-7
       (client)   (client)   (client)
           |         |          |        client-net
  ---------+---------+----------+------ 192.168.2.0/24
                     |
                   node-3 (router)
                     |                   server-net
      ------+--------+----------+---     192.168.1.0/24
            |        |          |
         node-2    node-4     node-5
         (IPVS)   (server)   (server)
```

The question that I have is that the network I would really like to be testing is;

```
       node-1       node-6     node-7
       (client)   (client)   (client)
           |         |          |        client-net
  ---------+---------+----------+------ 192.168.2.0/24
                     |
                   node-2 (IPVS)
                     |                   server-net
      ---------+-----+----+---------     192.168.1.0/24
               |          |
             node-4     node-5
            (server)   (server)
```

.. other than using NAT, which has performance problems, is this possible? I tried this topology with direct routing and packets from the clients were multiplexed to the servers fine, but return packets from the servers to the client were not routed by the IPVS box.

Lars Yes. The LVS box silently drops the return packets, since they have a src ip which is also bound as a local interface on the LVS. This is meant to be a simple anti-spoofing protection.

Note from Joe: The return packet from the realserver has src=VIP, dest=CIP. If this packet is routed via the director, which also has the VIP, the director will be receiving a packet from another machine with the the src being an one of its own IPs and the director will drop the packet).

> You can enable logging these packets via
>
> ```
> echo 1 >/proc/sys/net/ipv4/conf/all/log_martians
> ```
>
> The only way around this with current Linux kernels is to disable the check in the kernel source or to use a separate box as the outward gateway. (Which is how DR is meant to be used for full performance)

This is not a problem as such as it probably makes a lot of sense on not to use an IPVS box as your gateway router,

> Actually it makes a lot of sense to do just that IMHO. Less points of failure, less hard- & software to duplicate in a failover configuration.

Ray Bellis `rpb@community.net.uk`

It needs to be made more explicit in the documentation that LVS-DR will *only* work if you have a different return path.

> Lars Marowsky-Bree `lmb@teuto.net` ... or if you have a suitably patched kernel.

We spent several man days trying to get this to work before figuring out why the packets were being dropped, at which point we had no alternative but to use LVS-NAT instead.

> I agree. We still assume too much knowledge on the network admin side.

FYI, we have our LVS system working now, with LVS redundancy achieved by running OSPF routing (gated) on the LVS-NAT servers and having the VIP within the same IP subnet as the RIPs so that IGP routing policies automatically determine which LVS router the packets arrive on.

> Yes, thats one option. Even better than heartbeat and IPAT, if all your systems support running a routing protocol. (IPAT = IP address takeover, part of heartbeat) In essence, heartbeat & IPAT is nothing but reinventing a subset of the functionality of a hardened routing protocol like OSPF/RIPv2/EIGRP.

3.12.3 On other schemes for director/realservers to exchange roles

Julian Anastasov `uli@linux.tu-varna.acad.bg` has pointed out on the mailing list that the prototype LVS can be redrawn as

```
                 --------
                |        |
                | client |
                |________|
                    |
                    |
                 (router)
                    |
```

```
                        |
        ----------------------------------
        |                |               |
        |                |               |
     DIP, VIP        RIP1, VIP       RIP2, VIP
   ------------    --------------    --------------
  |            |  |              |  |              |
  |  director  |  | realserver1  |  | realserver2  |
  |____________|  |______________|  |______________|
```

and that any realserver is in a position to replace a failed director. No-one has bothered to write the code for this. It seems it's easier do have extra boxes in the director role (ready for failover) and others in realserver role. It's easier to wheel in another box for a spare director than to configure realservers to do two jobs reliably.

Julian

The director and the backup are in a shared network for incoming traffic, the backup sniff packets and change its connection state the same as the director (because the director is just on half client-to-server connection in LVS/TUN and LVS/DR), then drop packets.

It needs some investigation and probably lots of additional code too. ;-)

Wensong Zhang `wensong@iinchina.net`

> I don't even think so - the main trick is getting the kernel to sniff the packets, which is probably quite easy with a little messing around. Not sending the packets out again (which would confuse the realservers) is easy with a ipchains output rule which silently drops them. This doesn't work with a switch though, you need a shared network like a hub.
>
> However, I have been talking with rusty about this. The problem is more general - HA shared-state firewalls are asked for all the time, so we want to do a generic thing for everything which builds upon Netfilter's state machine. This would not only cover LVS, but also masquerading and packet filtering in general. We intend to discuss this in greater detail at the Ottawa Linux Symposium latest.

You can see,the connections depend on the initalize status and realsevers realtime status. So another method is that when Director is down, backup-sever setup the ipvs with the connections,but it seems too late. How do you think about this?

> TCP/IP should be able to cope with a few seconds delay and lost packets. You want to heartbeat once per second and take over after 3-4s though - this usually means takeover is complete in <10s, which TCP/IP should swallow.

3.12.4 Geographically distributed LVS

This has 27 (moved).

3.13 A discussion about the arp problem

(Joe and Julian)

Julian Anastasov `uli@linux.tu-varna.acad.bg` There is no difference between devices in 2.2.x, all devices are reported in the ARP replies: lo, tunl and dummy. This can be tested using this configuration with any device:

```
Host A:
        eth:x 192.168.0.1

Host B:
        eth:x 192.168.0.2
        lo, dummy, tunl: 192.168.0.3
```

On host A try: ping 192.168.0.3

Host B replies for 192.168.0.3 through 192.168.0.2 device

The ARP problem means: "All local interfaces are reported" until the ARP patch is used. In fact, all ARP patches which use IFF_NOARP to hide the interface are incorrect. I don't expect them in the kernel.

ARP problem, some rules:

ARP responses

- all local IP addresses are replied: lo, eth, tunl*, dummy* but with some exceptions (see the next rules)
- 127.0.0.0/8(LOOPBACK) and 224.0.0.0/4(MULTICAST) are not replied
- there is one exception for the "lo" interface: it is possible the kernel to ignore the ARP request if the source IP is from the same net as the net used to configure "lo" alias. The specified network is treated as local.

For example:

realserver# ifconfig lo:0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up

"real" treats all packets with source addr from 192.168.1.0/24 which come from the other devices (eth0) as invalid, i.e. source address validation works in this case and the ARP request are not replied. The kernel thinks: "The incoming packet arrived with saddr=local_IP1 and daddr=local_IP2(VIP), so it is invalid". By this way the host from the LAN can't talk to the real server if its lo alias is configured with netmask != 255.255.255.255

```
        ifconfig dummy0 192.168.1.1 netmask 255.255.255.255
```

registers only 192.168.1.1 as local ip but:

```
        ifconfig lo:0 192.168.1.1 netmask 255.255.255.0
```

all 256 IPs are local. All IFF_LOOPBACK devices treat all IPs as local according to the used netmask.

> Joe I assume IFF_LOOPBACK devices are lo, lo:0..n?

Yes, currently only lo is marked as loopback. It is used to mark whole subnets as local.

> lo:0 is not marked as loopback?

lo:0 is just attached IP address to the same device "lo". You can try "ifconfig lo:0 192.168.0.1 netmask 255.255.255.255" and display the interfaces using "ifconfig". There is LOOPBACK flag for lo:0 which is inherited from the device "lo". In Linux 2.2 all aliases inherit the device flags. Only the IFF_UP flag is used to add/delete the aliases.

> Joe Assume LVS-DR with VIP, RIPs all on the same /24 network on eth0 devices, realservers all have lo:0 with VIP/24 and have the standard 2.2.x kernel (no patches to hide interfaces). Router says "who has VIP", the arp request arrives at the realservers via eth0. Device lo:0 finds arp request which arrived on eth0 from router is on the same subnet as lo:0 and does not reply to the arp request.

Before checking if to answer the ARP the routing tables are checked, i.e. the source validation of the packet is performed. If 192.168.0.2 asks "who-has 192.168.1.1 tell 192.168.1.2" the real servers assumes that this is invalid packet, i.e. from one local IP to another local IP (from me to me => drop).

> Joe I notice that with the 2.2.x kernel, that lo:0 has to have netmask=255.255.255.255 to work, whereas with the 2.0.x kernels (where lo:0 doesn't reply to arp requests), that lo:0 can have the VIP on a 255.255.255.0 netmask and still work.

The rule is to use netmask 255.255.255.255 and to hide lo. The ARP works in different way in 2.2. It looks the "local" table to validate the source of the ARP request and after that it lookups the same table to check if daddr of the ARP request is local ip.

ARP requests: - all local addresses can be used by the kernel to announce them as the source for the ARP request.

> is it OK to say the kernel can (does?) use all local addresses as the source of ARP requests

It can and does. The real server thinks that it can use any local ip address as saddr in the ARP request and the answer will be returned back if this ip is uniq in the LAN.

> Joe do you mean "the realserver will receive a reply if the s_addr is unique in the LAN"?

The real server will receive answer if it uses RIP as saddr in the ARP request because the VIP(HIP) is hidden or when using transparent proxy because it is not local (the VIP). Real server must know how to ask (using uniq IP) or the trafic for the asked IP (ROUTER) will be blocked.

But the hidden addresses are not used because they are not uniq (2.2.14) and the answer will be returned to the Director.

> Joe do you mean "the non-hidden VIP on the director"?

Yes, when the real server ask "who-has ROUTER tell VIP" the ARP reply is received in the Director and the transmission in the real servers is stopped. The ROUTER sends everything destined to VIP to the Director. This is true for all clients on the LAN too if they are not in this cluster (if they don't handle packets for VIP).

> Joe I would have thought that the main device on each NIC, eg eth0, eth1 would have been used as the source address.

No, it is extracted from the outgoing datagram and if saddr is local ip it is used. But if this is not local ip, i.e. when using transparent proxy or the address is marked as hidden the main device ip is used.

> Joe how is arping part of transparent proxy?

It is not. When VIP is not local IP address in the real server this IP is not used from the ARP code. It is not in the "local" table. But TCP, UDP and ICMP use it via transparent proxy support.

They are extracted from the outgoing packet.

> Joe what is "They"? the source addresses? When you say "extracted", do you mean "removed from packet" or "looked at/detected"

The saddr from the data packet is used to build the ARP request.

We tell the kernel that these addresses are not uniq by setting <interface>/hidden=1 (starting with kernel 2.2.14). By this way the kernel select the devices primary IP as the source of the ARP request.

> Joe the kernel can use any local address as s_addr but the code for hiding IPs from arp requests prevents the kernel from using hidden addresses as s_addr in an arp request?

Yes, the code to hide the addresses is already part of the source address autoselection (saddr in the ARP request in our case). We never autoselect hidden addresses, i.e. if the source address is not specified from the higher level. The code to hide interface:

```
- ignores ARP replies for hidden local addresses
- doesn't select hidden local addresses as source of the ARP request
- doesn't autoselect hidden local addresses for the IP level
```

> Joe When you say "We expect it is uniq in the LAN" do you mean - we expect you've set up your network properly and that you don't have the same RIP on 2 realservers? :-)

The LVS administrator must ensure that the RIPs are uniq, only the VIP is shared. We tell the kernel that the VIP addresses are not uniq by setting *interface*hidden=1 (2.2.14). By this way the kernel select the devices primary IP as the source of the ARP request. We expect it is uniq in the LAN.

So, the recommendation for using the "lo" interface in the real servers is:

- use netmask 255.255.255.255 when configuring lo alias. By this way source validation doesn't drop the incoming packets to this IP. LVS users usually define the net route through the eth interface, so we can talk to other hosts from this network, for example to send the packets to the client through the default gateway. It is not needed to configure the alias with mask != 255.255.255.255

So, the interfaces which can be used in the real servers to listen for VIP are:

```
- lo aliases with netmask 255.255.255.255
- tunl*
- dummy*
```

All these devices must be marked as hidden to solve the ARP problem when using Linux 2.2.

In the Director: there is no problem to configure the VIP even on lo alias or dummy interface. If the interface is not marked as hidden this VIP is visible for all hosts on the LAN.

3.14 ATM/ethernet and router problems

LVS has only been tested on ethernet. One person had an ATM setup which didn't work with LVS-DR as the ATM router expects packets from the VIP to have the same MAC address (in LVS-DR packets coming from the VIP could have the MAC address of any of the realservers). Apparently this is not easily fixable in the ATM world. It should be possible to use one of Julian's 13.6 (martian modifications) to make LVS-DR work on ATM, but the person with the ATM setup disappeared off the mailing list without us convincing him of the joy in having the first ATM LVS.

Other people have found similar problems with ethernet -

From: Kyle Sparger `ksparger@dialtoneinternet.net`

I don't know if someone has gone over this, but here's a consideration I've come across when setting up LVS in DR mode:

When the real servers reply, cisco routers (ours do, at least) will pick up on the fact that it's replying from a different MAC address, and will start arping soon thereafter. This is sub-optimal, as it causes a constant flood of arp requests on the network. Our solution has been to hardcode the MAC address into the router, but this can cause other issues, for example during failover. That can be worked around, as you can set the MAC address on most cards, but that in itself may cause other issues.

Has anyone else experienced this? Has anyone else come up with a better solution than hardcoding it into the router?

4 Collect Hardware

4.1 minimum setup

You will need a minimum of 3 machines.

(You can do it with 2, but this doesn't demonstrate how to scale up an LVS farm, read 15 (localnode) to see how a 2 node LVS is setup).

You need

- Director: running Linux patched kernel 2.2.x or 2.4.x (VS-NAT is still in development for 2.4.x, in Feb 2001, check on the mailing list for latest info).
- Client: any machine, any OS, with a client for the service (eg netscape, fetch/xterm/...)
- Realserver(s)
  - VS-NAT - any machine, any OS, running some service of interest (eg httpd, ftpd, telnetd, smtp, nntp, dns, daytime ...)
  - VS-Tun - realserver needs to run an OS that can tunnel (only Linux so far).
  - VS-DR - OS known to work are listed in 13 (VS-DR)

4.2 Gotchas

You need an outside client.

The LVS functions as one machine. You must access the LVS from a client that is NOT a member of the LVS. You cannot access an LVS controlled service (eg http, telnet) from any of the machines in the LVS; access from the director will hang, access from a realserver will connect to the service locally, bypassing the LVS.

Minimum 3 machines: client, director, realserver(s)

4.3 Test with telnet (or netcat)

All testing of your LVS should be done with simple services. Telnet has

- a simple client (available on all OSs)
- a simple protocol (one port)
- exchanges are all ascii (can be watched with tcpdump)
- non-persistent connection (you can test round robin scheduling)
- telnetd usually listens to all IP's on the server (*i.e.* to 0.0.0.0) at least under inetd
- most importantly, when you get a connection, you'll see an entry in the ActConn column of the output of ipvsadm.

For security reasons, you'll be turning off telnet later, but whenever you're testing your LVS or your new service, always look to see if telnet works if you're having trouble. If telnet is not being LVS'ed then you should go back and fix that first.

Another useful client is netcat. See 11.6 (examples using netcat as a client with ftp).

4.4 Test without filter rules

You can add them after you get the LVS working.

5 Choose LVS Forwarding Type

5.1 Comparison of LVS-NAT, LVS-DR and LVS-Tun

The instructions in the following sections show how to set up LVS in LVS-NAT, LVS-Tun and LVS-DR modes for the service telnet.

If you just want to demonstrate to yourself that you can setup an LVS, then LVS-NAT has the advantage that any OS can be used on the realservers, and that no modifications are needed for the kernel on the realserver(s).

If you have a linux machine with a 2.0.x kernel, then it can be used as a realserver for an LVS operating in any mode without any modifications.

Because LVS-NAT was the first mode of LVS developed, it was the type first used by people setting an LVS. For production work VS-DR scales to much higher throughput and is the most common setup for production. However for a simple test, LVS-NAT only requires patching 1 machine (the director) and an unmodified machine of any OS for a realservers. After a simple test, unless you need the features of LVS-NAT (ability to use realservers that provide services not found on Linux machines, port remapping, realservers with primitive tcpip stacks - e.g./printers, services that initiate connect requests such as identd), then it would be best to move to LVS-DR.

Here are the constraints for choosing the various flavors of LVS: LVS-NAT (network address translation), LVS-Tun (tunnelling) and LVS-DR (direct routing).

```
                        LVS-NAT        LVS-Tun             LVS-DR
```

realserver OS	any	must tunnel	most
realserver mods	none	tunl must not arp	lo must not arp
port remapping	yes	no	no
realserver network	private	on internet	local
	(remote or	local)	-
realserver number	low	high	high
client connnects to	VIP	VIP	VIP
realserver default gw	director	own router	own router

5.2 Expected LVS performance

unknown

what is the maximum number of servers I can have behind the LVS without any risk of failure.

> Horms `horms@vergenet.net` 03 Jul 2001 LVS does not set artificial limits on the number of servers that you can have. The real limitations are the number of packets you can get through the box, the amount of memory you have to store connection information and in the case of LVS-NAT the number of ports available for masquerading. These limitations effect the number of concurrent connections you can handle and your maximum through-put. This indirectly effects how many servers you can have.

(also see the section on 19.9 (port range limitations).)

Palmer J.D.F:

I know the JANet Web Cache Service does this, but I was hoping that someone had done it on a smaller scale.

> Martin Hamilton `martin@net.lut.ac.uk` Nov 14 2001 we (JWCS) also use LVS on our home institutional caches. These are somewhat smaller scale, e.g. some 10m URLs/day at the moment for Loughborough's campus caches vs. 130m per day typically on the JANET caches. The good news is that LVS in tunnelling mode is happily load balancing peaks of 120MBit/s of traffic on a 550MHz PIII.
>
> Folk in ac.uk are welcome to contact us at support@wwwcache.ja.net for advice on setting up and operating local caches. I'm afraid we can only provide this to people on the JANET network, like UK Universities and Colleges.

Michael McConnell:

Top doesn't display CPU usage of ipchains or ipvsadm. vmstat doesn't display CPU usage of ipchains or ipvsadm.

> Joe ipchains and ipvsadm are user tools that configure the kernel. After you've run them, they go away and the kernel does it's new thing (which you'll see in "system"). Unfortunately for some reason that no-one has explained to me "top/system" doesn't see everything. I can have a LVS-DR director which is running 50Mbps on a 100Mpbs link and the load average doesn't get above 0.03 and system to be negligable. I would expect it to be higher.

> Julian Anastasov `ja@ssi.bg` 10 Sep 2001 Yes, the column is named "%CPU", i.e. the CPU spend for one process related to all processes. As for the load average, it is based on the length (number of processes except the current one) of the queue with all processes in running state. As

> we know, LVS does not interract with any processes except the ipvsadm. So, the normal mode is the LVS box just to forward packets without spending any CPU cycles for processes. This is the reason we to see load average 0.00
>
> OTOH, vmstat reads /proc/stat and there are the counters for all CPU times. Considering the current value for jiffies (the kernel tick counter) the user apps can see the system, the user and the idle CPU time. LVS is somewhere in the system time. For more accurate measurement for the CPU cycles in the kernel there are some kernel patches/tools that are exactly for this job - to see what time takes the CPU in some kernel functions.

If you are just setting up an LVS to see if you can set one up, then you don't care what your performance is. When you want to put one on-line for other people to use, you'll want to know the expected performance.

On the assumption that you have tuned/tweeked your farm of realservers and you know that they are capable of delivering data to clients at a total rate of bits/sec or packets/sec, you need to design a director capable of routing this number of requests and replies for the clients.

Before you can do this, some background information on networking hardware is required. At least for Linux (the OS I've measured, see *performance data for single realserver LVS* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>`), a network rated at 100Mbps is not 100Mbps all the time. It's only 100Mbps when continuously carrying packets of mtu size (1500bytes). A packet with 1 bit of data takes as long to transmit as a full mtu sized packet. If your packets are <ack>s, or 1 character packets from your telnet editing session or requests for http pages and images, you'll barely reach 1Mbps on the same network. On the *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>`, you'll notice that you can get higher hit rates on a website as the size of the hit targets (in bytes) gets smaller. Hit rate is not neccessarily a good indicator of network throughput.

Tcpip can't use the full 100Mbps of 100Mbps network hardware, as most packets are paired (data, ack; request, ack). A link carrying full mtu data packets and their corresponding <ack>s, will presumably be only carrying 50Mbps. A better measure of network capacity is the packet throughput. An estimate of the packet throughput comes from the network capacity (100Mbps)/mtu size(1500bytes) = 8333 packets/sec.

Thinking of a network as 100Mbps rather than *ca.*8000packets/sec is a triumph of marketing. When offered the choice, everyone will buy network hardware rated at 100Mbps even though this capacity can't be used with your protocols, over another network which will run continuously at 8000packets/sec for all protocols. Only for applications like ftp will near full network capacity be reached (then you'll be only running at 50% of the rated capacity as half the packets are <ack>s).

A netpipe test (on my realservers are 75MHz pentiums and can't saturate the 100Mbps network) shows that some packets must be "small". Julian's 19.16.2 (show_traffic script) shows that for small packets (<128bytes), the throughput is constant at 1200packets/sec. As packets get bigger (upto mtu size), the packet throughput decreases to 700packets/sec, and then increases to 2600packets/sec for large packets.

The constant througput in packets/sec is a first order approximation of of tcpip network throughput and is the best information we have to predict director performance.

In the case where a client is in an exchange of small packets (<mtu size) with a realserver in a LVS-DR LVS, each of the links (client-director, director-realserver, realserver-client) would be saturated with packets, although the bps rate would be low. This is the typical case for non-persistent http when 7 packets are required for the setup and termination of the connection, 2 packets are required for data passing (eg the request GET /index.html and the reply) and an <ack> for each of these. Thus only 1 out of 11 packets is likely to be near mtu size, and throughput will be 10% of the rated bps throughput even though the network is saturated.

The first thing to determine then is the rate at which the realservers are generating/receiving packets. If the realservers are network limited, i.e. the realservers are returning data in memory cache (eg a disk-less squid) and have 100Mbps connections, then each realserver will saturate a 100Mbps link. If the service on the realserver requires disk or CPU access, then each realserver will be using proportionately less of the network. If the realserver is generating images on demand (and hence is compute bound) then it may be using very little of the network and the director can be handling packets for another realserver.

The forwarding method affects packet throughput. With LVS-NAT all packets go through the director in both directions. As well the LVS-NAT director has to rewrite incoming and reply packets for each realserver. This is a compute intensive process (12.14 (but less so for 2.4 LVS-NAT)). In a LVS-DR or LVS-Tun LVS, the incoming packets are just forwarded (requiring little intervention by the director's CPU) and replies from the realservers return to the client directly by a separate path (via the realserver's default gw) and aren't seen by the director.

In a network limited LVS, for the same hardware, because there are separate paths for incoming and returning packets with LVS-DR and LVS-Tun, the maximum (packet) throughput is twice that of LVS-NAT. Because of the rewriting of packets in LVS-NAT, the load average on a LVS-NAT director will be higher than for a VS-DR or LVS-Tun director managing twice the number of packets.

In a network bound situation, a single realserver will saturate a director of similar hardware. This is a relatively unusual case for the LVS's deployed so far. However it's the situation where replies are from data in the memory cache on the realservers (eg squids).

With a LVS-DR LVS, the realservers have their own connection to the internet, the rate limiting step is the NIC on the director which accepts packets (mostly <ack>s) from the clients. The incoming network is saturated for packets but is only carrying low bps traffic, while the realservers are sending full mtu sized packets out their default gw (presumably the full 100Mbps).

The information needed to design your director then is simply the number of packets/sec your realserver farm is delivering. The director doesn't know what's in the packets (being an L4 switch) and doesn't care how big they are (1 byte of payload or full mtu size).

If the realservers are network limited, then the director will need the same CPU and network capacity as the total of your realservers. If the realservers are not network limited, then the director will need correspondingly less capacity.

If you have 7 network limited realservers with 100Mbps NICs, then they'll be generating an average of 7x8000 = 50k packets/sec. Assuming the packets arrive randomly the standard deviation for 1 seconds worth of packets is +/- sqrt(50000)=200 (ie it's small compared to the rate of arrival of packets). You should be able to connect these realservers to a 1Gbps NIC via a switch, without saturating your outward link.

If you are connected to the outside world by a slow connection (eg T1 line), then no matter how many 8000packet/sec realservers you have, you are only going to get 1.5Mbps throughput (or half that, since half the packets are <ack>s).

Note: The carrying capacity of 100Mbps network of 8000packets/sec may only apply to tcpip exchanges. My 100Mbps network will carry 10,000 SYN packets/sec when tested with Julian's 19.16.1 (testlvs) program.

> Wayne `wayne@compute-aid.com` 03 Apr 2001 The performance page calculate the ack as 50% or so the total packets. I think that might not accurate. Since in the twist-pair and full duplex mode, ack and request are travelling on two different pairs. Even in the half duplex mode, the packets for two directions are transmit over two pairs, one for send, one for receive, only the card and driver can handle them in full duplex or half duplex mode. So the packets would be 8000 packets/sec all the times for the full duplex cards.

Unfortunately we only can approximately predict the performance of an LVS director. Still the best estimates come from comparing with a similar machine.

The *performance page* <http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html> shows that a 133MHz pentium director can handle 50Mbps throughput. With LVS-NAT the load average on the director is unusably high, but with LVS-DR, the director has a low load average.

Statements on the website indicate that a 300MHz pentium VS-DR director running a 2.2.x kernel can handle the traffic generated by a 100Mbps link to the clients. (A 5.2 (550MHz PIII can direct 120Mbps).)

Other statements indicate that single CPU high end (800MHz) directors cannot handle 1Gbps networks. Presumably multiple directors or SMP directors will be needed for Gbps networks.

> From: Jeffrey A Schoolcraft `dream@dr3amscap3.com` 7 Feb 2001 I'm curious if there are any known DR LVS bottlenecks? My company had the opportunity to put LVS to the test the day following the superbowl when we delivered 12TB of data in 1 day, and peaked at about 750Mbps.
>
> In doing this we had a couple of problems with LVS (I think they were with LVS). I was using the latest lvs for 2.2.18, and ldiretord to keep the machines in and out of LVS. The LVS servers were running redhat with an EEPro100. I had two clusters, web and video. The web cluster was a couple of 1U's with an acenic gig card, running 2.4.0, thttpd, with a somewhat performance tuned system (parts of the C10K). At peak our LVS got slammed with 40K active connections (so said ipvsadmin). When we reached this number, or sometime before, LVS became in-accessible. I could however pull content directly from a server, just not through the LVS. LVS was running on a single proc p3, and load never went much above 3% the entire time, I could execute tasks on the LVS but http requests weren't getting passed along.
>
> A similar thing occurred with our video LVS. While our real servers aren't quite capable of handling the C10K, we did about 1500 a piece and maxed out at about 150Mbps per machine. I think this is primarily modem users fault. I think we would have pushed more bandwidth to a smaller number of high bandwidth users (of course).
>
> I know this volume of traffic choked LVS. What I'm wondering is, if there is anything I could do to prevent this. Until we got hit with too many connections (mostly modems I imagine) LVS performed superbly. I wonder if we could have better performance with a gig card, or some other algorithm (I started with wlc, but quickly changed to wrr because all the rr calculations should be done initially and never need to be done again unless we change weights, I thought this would save us).
>
> Another problem I had was with ldirectord and the test (negotiate, connect). It seemed like I needed some type of test to put the servers in initially, then too many connections happened so I wanted no test (off), but the servers would still drop out from ldirectord. That's a snowball type problem for my amount of traffic, one server gets bumped because it's got too many connections, and then the other servers get over-loaded, they'll get dropped to, then I'll have an LVS directing to localhost.
>
> So, if anyone has pushed DR LVS to the limits and has ideas to share on how to maximize it's potential for given hardware, please let me know.

5.3 Initial setup steps

Here's some rules of thumb until you know enough to make informed decisions.

5.3.1 Choose forwarding type

choose in this order

- VS-DR default, has high throughput, can be setup on most OS's.

- VS-NAT, lower throughput, higher latency. Realserver only needs tcpip stack (ie any OS, even a network printer). Works for multiple instances of services (ie multiple copies of demon running on several different ports).
- VS-Tun, Linux only realservers. Same throughput as LVS-DR. Needed if realserver on different network to director (eg in another location).

5.3.2 Choose number of networks

The realservers are normally run on a private network (eg 192.168.1.0/24). They are not contacted by clients directly. Sometimes the realservers are machines (on the local network) also being used for other things, in which case leave them on their original network.

The director is contacted by client(s) on the VIP. If the VIP must be publically available, then it will usually be on a different network to the realservers, in which case you will have a two network LVS. The VIP then will be in the network that connects the director to the router (or test client). If the client(s) are on the same network as the realservers, then you'll only have a one network LVS.

5.3.3 Choose number of NICs on director

If the director is in a 2 network LVS, then having 2 NICs on the director (one for each network) will increase throughput (as long as something else doesn't become rate limiting, eg the CPU speed or the PCI bus).

You can have 1 NIC on the director with a 2 network LVS. This is easy to do for LVS-NAT (for which there is an example conf file). Doing the same this for VS-DR or LVS-Tun requires more though and is left as an exercise for the reader.

The 10.1 (configure script) will handle 1 or 2 NICs on the director. In the 1 NIC case, the NIC connects to the outside world and to the realserver network. In the 2 NIC case, these two networks are physically separated. To increase throughput further, the director could have a NIC for each realserver. The configure script doesn't handle this (yet - let me know if you'd like it).

5.3.4 Pick a configure script

If you're using the 10.1 (configure script) to setup your LVS, pick the appropriate lvs*.conf.* file and edit to suit you.

6 Install - General

(This section will eventually be folded into the LVS-mini-HOWTO and will disappear.)

You can setup a working LVS from the LVS-mini-HOWTO. Presumably it will be more upto date, clearer and simpler than this section.

Abreviations/conventions for setup/testing/configuring

```
client:      client's IP         =CIP
gateway:     gateway/router's IP =DGW (router will be the client in most test setups)
director:    director's IP       =DIP on director   eth0
             virtual IP          =VIP on director   eth0:x (eg eth0:1)
realserver:  realserver IP       =RIP on realserver eth0
             virtual IP          =VIP on realserver eth0:x/lo:0/tunl0/dummy0
             gateway             =SGW
```

Note: the DIP and the VIP must be different IPs (they can be on the same NIC).

> Julian Anastasov `ja@ssi.bg` 06 Nov 2001 If DIP==VIP the realserver will receive packets with saddr=local_ip (VIP) from the director. (This is the 13.6 (source martian) problem.) You have to add additional IP(DIP) in your director and when you execute ip route get x.y.z.230 you have to see that x.y.z.180 is not your preferred source. The usual way to achieve this is to configure VIP on another device (eg. lo) or to configure them after the DIP is configured. By this way DIP will become your preferred source when talking to your subnet.
>
> Adding DIP is not mandatory for any LVS setup to work but you will not be able to send non-LVS traffic between the director and the real servers (ping in you case).

6.1 Director

The kernel configuration options below are just for LVS. You will need other flags set (*e.g.* filesystems, hardware...).

6.1.1 2.2.x kernels

(with suggestions from John Cronin `jsc3@havoc.gtf.org`)

Get a fresh kernel from ftp.kernel.org and the matching ipvs patch from the software page on www.linuxvirtualserver.org. You must start from a clean source tree (just downloaded or that you've run "make mrproper" on). Apply the kernel patch 2.2.x using the instructions in the tarball. You'll do something like

director:/usr/src/linux# patch -p1 <../ipvs-0.9.8-2.2.14/ipvs-0.9.8-2.2.14.patch

Note: kernels from RedHat are not the standard kernels off ftp.kernel.org. They are pre-patched with (among other things) ipvs. The pre-applied patch on the RedHat kernel will be older than the one on the LVS website. If you are going to use RedHat files, follow RedHat's instructions, not ours. If you follow our instructions and try to patch a RH kernel you'll get

```
   Hunk #1 succeeded at 121 with fuzz 2 (offset 18 lines).
Hunk #2 FAILED at 153.
Hunk #3 FAILED at 163.
Hunk #4 FAILED at 177.
3 out of 4 hunks FAILED -- saving rejects to include/linux/ip_masq.h.rej

patching file ‘include/net/ip_masq.h’
Reversed (or previously applied) patch detected!  Assume -R? [n]
```

Compile the 2.2.x kernel and reboot.

The actual kernel compile instructions will vary with kernel patch number. Here's what I used for ipvs-0.9.9 on kernel 2.2.15pre9 in the Networking options. The relevant options are marked. Some of the options are not explicitely required for LVS to work, but you'll need them anyhow - e.g. ip aliasing if you need to constuct a director with only one NIC, or tunneling if you are going to run LVS-Tun. Until you know what you're doing activate all of the options with an '*' at the start of the line.

You need to turn on "Prompt for developmental code... " (or whatever) under the "Code Maturity" section. In the "Networking" section, you have to turn on IP:masquerading before you get the ipvs option.

Do all the other kernel stuff - make modules, install the modules, copy the new kernel (and optionally System.map, for the 10.1 (configure script)) into / or /boot, edit lilo.conf and run lilo. Make sure you leave the old kernel config in lilo too, so you can recover if all does not go well. When loading the new kernel, make sure the ip_vs* modules get loaded. The README in the kernel source tree has all the necessary info in there.

```
                [*] Kernel/User netlink socket
                [*] Routing messages
                < > Netlink device emulation
     *          [*] Network firewalls
                [*] Socket Filtering
                <*> Unix domain sockets
     *          [*] TCP/IP networking
                [ ] IP: multicasting
     *          [*] IP: advanced router
     *          [*] IP: policy routing
                [ ] IP: equal cost multipath
                [ ] IP: use TOS value as routing key
                [ ] IP: verbose route monitoring
                [ ] IP: large routing tables
                [ ] IP: kernel level autoconfiguration
     *          [*] IP: firewalling
                [ ] IP: firewall packet netlink device
     *          [*] IP: use FWMARK value as routing key (NEW)
     *          [*] IP: transparent proxy support
     *          [*] IP: masquerading
                --- Protocol-specific masquerading support will be built as modules.
     *          [*] IP: ICMP masquerading
                --- Protocol-specific masquerading support will be built as modules.
     *          [*] IP: masquerading special modules support
     *          <M> IP: ipautofw masq support (EXPERIMENTAL)
     *          <M> IP: ipportfw masq support (EXPERIMENTAL)
     *          <M> IP: ip fwmark masq-forwarding support (EXPERIMENTAL)
     *          [*] IP: masquerading virtual server support (EXPERIMENTAL)
     *          (12) IP masquerading VS table size (the Nth power of 2)
     *          <M> IPVS: round-robin scheduling
     *          <M> IPVS: weighted round-robin scheduling
     *          <M> IPVS: least-connection scheduling
     *          <M> IPVS: weighted least-connection scheduling
     *          [*] IP: optimize as router not host
     *          <M> IP: tunneling
                <M> IP: GRE tunnels over IP
                [*] IP: broadcast GRE over IP
                [ ] IP: multicast routing
                [*] IP: PIM-SM version 1 support
                [*] IP: PIM-SM version 2 support
     *          [*] IP: aliasing support
                [ ] IP: ARP daemon support (EXPERIMENTAL)
     *          [*] IP: TCP syncookie support (not enabled per default)
                --- (it is safe to leave these untouched)
                < > IP: Reverse ARP
                [*] IP: Allow large windows (not recommended if <16Mb of memory)
                < > The IPv6 protocol (EXPERIMENTAL)
```

For the moment accept the default hash table size. Here's the information about 19.1 (how the hash table works).

6.1.2 2.4.x kernels

The patches to the early versions of the 2.4.x kernels were configured and installed separately to the "make menuconfig" for the kernel. This required moving files into the /lib/modules directories and loading the modules by hand.

With later versions of the kernel, you can get a set of files where the patches are put into the source tree and configured by "make configure".

Here's the networking config

```
<*> Packet socket
[ ]   Packet socket: mmapped IO
[*] Kernel/User netlink socket
[*]   Routing messages
<*>   Netlink device emulation
[*] Network packet filtering (replaces ipchains)
[*]   Network packet filtering debugging
[*] Socket Filtering
<*> Unix domain sockets
[*] TCP/IP networking
[ ]   IP: multicasting
[*]   IP: advanced router
[*]     IP: policy routing
[*]       IP: use netfilter MARK value as routing key
[*]       IP: fast network address translation
[*]     IP: equal cost multipath
[*]     IP: use TOS value as routing key
[*]     IP: verbose route monitoring
[*]     IP: large routing tables
[*]   IP: kernel level autoconfiguration
[ ]     IP: BOOTP support
[ ]     IP: RARP support
<M>   IP: tunneling
< >   IP: GRE tunnels over IP
[ ]   IP: multicast routing
[ ]   IP: ARP daemon support (EXPERIMENTAL)
[ ]   IP: TCP Explicit Congestion Notification support
[ ]   IP: TCP syncookie support (disabled per default)
  IP: Netfilter Configuration  --->
  IP: Virtual Server Configuration  --->
< >   The IPv6 protocol (EXPERIMENTAL)
< >   Kernel httpd acceleration (EXPERIMENTAL)
[ ] Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)
```

Here's my config for the IP: Virtual Server configuration (turn it all on)

```
<M> virtual server support (EXPERIMENTAL)
[*]   IP virtual server debugging (NEW)
(12)   IPVS connection table size (the Nth power of 2) (NEW)
```

```
--- IPVS scheduler
<M>   round-robin scheduling (NEW)
<M>   weighted round-robin scheduling (NEW)
<M>   least-connection scheduling scheduling (NEW)
<M>   weighted least-connection scheduling (NEW)
<M>   locality-based least-connection scheduling (NEW)
<M>   locality-based least-connection with replication scheduling (NEW)
<M>   destination hashing scheduling (NEW)
<M>   source hashing scheduling (NEW)
--- IPVS application helper
<M>   FTP protocol helper (NEW)
```

Here is my config for the netfilter section

```
<M> Connection tracking (required for masq/NAT)
<M>   FTP protocol support
<M> Userspace queueing via NETLINK (EXPERIMENTAL)
<M> IP tables support (required for filtering/masq/NAT)
<M>   limit match support
<M>   MAC address match support
<M>   netfilter MARK match support
<M>   Multiple port match support
<M>   TOS match support
<M>   Connection state match support
<M>   Unclean match support (EXPERIMENTAL)
<M>   Owner match support (EXPERIMENTAL)
<M>   Packet filtering
<M>     REJECT target support
<M>     MIRROR target support (EXPERIMENTAL)
<M>   Full NAT
<M>     MASQUERADE target support
<M>     REDIRECT target support
<M>   Packet mangling
<M>     TOS target support
<M>     MARK target support
<M>   LOG target support
< > ipchains (2.2-style) support
< > ipfwadm (2.0-style) support
```

6.1.3 iptables/ipchains compatability problems

Note I have removed the ipchains option here. This was <M> in the last version of the HOWTO. However this raised problems for people who didn't understand the ipchains compatability problems.

ipchains in 2.2.x kernels has been replaced by iptables in 2.4.x kernels. For 2.4 kernels, ipchains is available for backwards compatibility. However ipchains and iptables can't be used at the same time.

ipchains under 2.4 makes 25.37 (conntrack slow)

The ip_tables module is incompatible with ipchains. If present, the ip_tables module must be unloaded for ipchains to work.

If you have ip_tables loaded, you'll get uninformative errors when you try to run ipchains commands with 2.4. Rather than saying that ipchains under 2.4 is there for compatibility, it would be more accurate to say that the ipchains commands available with 2.4 kernels will only cause you grief and it will be faster to rewrite your scripts to iptables, than to fall into all the holes you'll find using the compatibility. It won't take long before some script/program expects to run ip_tables on your 2.4 machine and as soon as that happens one or both (I don't know which) of your iptables or ipchains are hosed.

6.1.4 have working kernel before building separate ipvs code

Boot to the new kernel and make sure /usr/src/linux points to your kernel source tree (you can use rc.system_map, which comes with the 10.1 (configure script)), before building ipvs (if you're doing it separately) and ipvsadm.

Don't forget to run ‘depmod -a‘ again after you install the ip_vs modules.

6.1.5 ext3 filesystems

There is a variable name collision when patching the kernel with both ipvs and ext3 patches.

> Adam Kurzawa 4 Nov 2001 after applying BOTH, the lvs and ext3 patches to 2.4.13 I get compilation errors. Either of the patches applied exclusively works fine.

Wensong

Remove the line of "EXPORT_SYMBOL(buffermem_pages);" in the linux/kernel/ksyms.c, then compile the kernel.

6.1.6 Compile problems

> Kim Le, Jul 25, 2001
>
> I am having problem when trying to compile LVS as module. It keeps complaining "unresovled symbols - nf_register_hook". I though it is because I don't have NETFILTER select, so I did make menuconfig, select NETFILTER and recompile the kernel. I check the System.map and did see nf_register_hook symbol in there.

Horms

I believe that you are having a kernel symbols issue as discuessed in http://marc.theaimsgroup.com/?l=linux-virtual-server&m=98766822929703&w=2 You can find information on how to resolve this problem at http://lists.samba.org/listproc/netfilter/1999-November/002871.html Basically your kernel symbols are out of date and you need to rebuild the kernel back from "make mrproper". You may want to back up your kernel config before you do this. Once this is done go back into the ipvs directory and run

```
$ make clean all install
```

Wensong

My understanding is that as kernel/ksyms.c is patched we need to do a make mrproper to make sure the ksyms are up to date when the kernel is built. Is this correct? If so it wouldn't hurt to include this information in the README.

Here's the original posting for the fix from the samba list.

Erik Ratcliffe `erik@calderasystems.com` 29 Nov 1999

The symbols exported from the kernel should be stored in /usr/src/linux/include/linux/modules. Most notably, the ksyms.ver file. If your kernel was built from an existing source tree and you did not run 'make mrproper' first, odds are the files in the forementioned directory are stale. I have seen this happen on a number of systems where an SMP kernel is in use against a Linux source tree that contains non-SMP symbol versions (you can tell the SMP symbol versions by the "smp_" prefix in the CRC/version number; non-SMP kernel symbols do not have this prefix).

The best way to remedy this is to rebuild the kernel all they way from 'make mrproper' to 'make modules_install'. Be sure to state that versions should be assigned to module symbols in the kernel configuration before doing so.

6.2 Realservers

In general, nothing specific is done for the realservers. You can have any OS running on them (except LVS-Tun, which runs only on Linux realservers). You plug them in to the network, startup the services on the VIP (for LVS-DR, LVS-Tun) or the RIP (VS-NAT), setup the default gw (the router and the director respectively in the usual setups) and you're ready to go.

Except: You have to handle the 3 (arp problem) with LVS-DR (and possibly LVS-Tun). Unfortunately this turns what would be a trivial installation into one that requires clear thinking. If you don't want to deal with the arp problem for your first installation, then setup a LVS-NAT LVS.

Eventually you'll want the higher throughput and lower latency of LVS-DR, in which case you'll need to understand the arp problem. The simplest approach is to use the NOARP option of ifconfig to setup lo:0 on non-linux unix realservers. For linux,

- 2.0.x kernels: the NOARP option to ifconfig works as for other unices
- 2.2.x recent (*i.e.*x>12) kernels; activate arp hiding
- 2.4.x kernels; patch the kernel with Julian's "hidden patch" and then activate the arp hiding

For non-unix (ie Windows) realservers, look below for further instructions.

6.2.1 Linux Realservers

If you are handling the arp problem by hiding the VIP device on the realservers, then you need to patch the realservers if

- 2.2.x (where x<12)
- 2.4.x (all versions, see info about 3.2.3 (patching 2.4 realservers)).

The current version(s) of 2.2.x, *e.g.* 2.2.19 are already patched with the arp hiding code *i.e.* you don't have to patch the current 2.2.x kernels. For Linux-2.0.x realservers, you can use the NOARP option when setting up a device for the VIP.

6.2.2 Other unices

If you are running non-Linux unix realservers, you can handle the arp problem by configuring the device carrying the VIP with the -arp switch. This list of realservers is from Ratz `ratz@tac.ch` About the only thing he hasn't tried yet is Plan 9.

```
Solaris 2.5.1, 2.6, 2.7
Linux (of course): 2.0.36, 2.2.9, 2.2.10, 2.2.12
FreeBSD 3.1, 3.2, 3.3
NT (although Webserver would crash): 4.0 no SP
IRIX 6.5 (Indigo2)
HPUX 11
```

Ratz's code is now in the 10.1 (configure script). This part of the script has not been well tested (you might find that it doesn't setup your non-linux unix box properly yet, please contact me - Joe).

Here's the information for non-Linux unices. On some Unixes you have to plumb the interface before assigning an IP. The plumb instruction is not included here.

```
#uname      : FreeBSD
#uname -r   : 3.2-RELEASE
#<command>  : ifconfig lo0 alias <VIP> netmask 0xffffffff -arp up
#ifconfig -a: lo0: flags=80c9<UP,LOOPBACK,RUNNING,NOARP,MULTICAST>mtu 16837
#                  inet 127.0.0.1 netmask 0xff000000
#                  inet <VIP> netmask 0xffffffff

#uname      : IRIX
#uname -r   : 6.5
#<command>  : ifconfig lo0 alias <VIP> netmask 0xffffffff -arp up
#ifconfig -a: lo0: flags=18c9<UP,LOOPBACK,RUNNING,NOARP,MULTICAST,CKSUM>
#                  inet 127.0.0.1 netmask 0xff000000
#                  inet <VIP> netmask 0xffffffff

#uname      : SunOS
#uname -r   : 5.7
#<command>  : ifconfig lo0:1 <VIP> netmask 255.255.255.255 up
#ifconfig -a: lo0:  flags=849<UP,LOOPBACK,RUNNING,MULTICAST>mtu 8232
#                   inet 127.0.0.1 netmask ff000000
#             lo0:1 flags=849<UP,LOOPBACK,RUNNING,MULTICAST>mtu 8232
#                   inet <VIP> netmask ffffffff

#uname      : HP-UX
#uname -r   : B.11.00
#<command>  : ifconfig lan1:1 10.10.10.10 netmask 0xffffff00 -arp up
#ifconfig -a: lan0:   flags=842<BROADCAST,RUNNING,MULTICAST>
#                     inet <some IP> netmask ffffff00
#             lan0:1: flags=8c2<BROADCAST,RUNNING,NOARP,MULTICAST>
#                     inet <VIP> netmask ffffff00
#
```

Some unices aren't very cooperative and other methods (*e.g.* adding an extra NIC) should do it.

> Ratz 16 Apr 2001 in most cases (when using the NOARP option) you need alias support. Some Unices have no support for aliased interfaces or only limited, such as QNX, Aegis or Amoeba for example. Others have interface flag inheritance problems like HP-UX where it is impossible to give an aliased interface a different flag vector as for the underlying physical interface (as happens with Linux 2.2 and 2.4 - Joe). So for HP/UX you need a special setup because with the

standard depicted setup for DR it will NOT work. I've done most Unices as Realserver and was negatively astonished by all the different implementation variations of the different Unix flavours. This maybe resulted from unclear statements from the RFC's.

6.2.3 Windows/Microsoft/NT/W2K

This is *not* handled by the 10.1 (configure script).

Instructions for setting up windows realservers is one of the more common questions on the mailing list. This must be a difficult part of the HOWTO to find ;-\. Here are some of the answers.

Wensong's original recipe for setting up the lo device on a NT realserver.

> If you don't have MS Lookback Adapter Driver installed on your NT boxes, enter Network Control Panel, click the Adapter section, click to add a new adapter, select the MS Loopback Adapter. Your NT cdrom is needed here. Then add your VIP (Virtual IP) address on the MS Loopback Adapter, do not enter a gateway address on the Loopback Adapter. Since the netmask 255.255.255.255 is considered invalid in M$ NT, you just accept the default netmask, then enter MS-DOS prompt, remove the extra routing entry.

```
c:route delete <VIP's network> <VIP's netmask>
```

> This will make the packets destined for this network will go through the other network interface, not this MS Loopback interface. As I remember, setting its netmask to 255.0.0.0 also works.

Jerome Richard `jrichard@virtual-net.fr`

> On Windows NT Server, you just have to install a network adapter called "MS Loopback" (Provided on the Windows NT CDROM in new network section) and then you setup the VIP on this interface.

o1004g `o1004g@nbuster.com`

1. Click Start, point to Settings, click Control Panel, and then double-click Add/Remove Hardware.
2. Click Add/Troubleshoot a device, and then click Next.
3. Click Add a new device, and then click Next.
4. Click No, I want to select the hardware from a list, and then click Next.
5. Click Network adapters, and then click Next.
6. In the Manufacturers box, click Microsoft.
7. In the Network Adapter box, click Microsoft Loopback Adapter, and then click Next.
8. Click Finish.

`robert.gehr@web2cad.de`> 24 Oct 2001

The MS-Loopback adapter is a virtual device under Windows that does not answer any arp requests. It should be on a Server Edition CD of WinNT/2000. Install and assign it the appropriate IP Address. Because MS would not let you assign a "x.x.x.x/32" netmask to the MS-Loopback adapter, you will end up having two routes pointing into the same net. Lets say your RIP is 10.10.10.10 and your MS-Loopback VIP is 10.10.10.11. You will have two routes in your routing table both pointing to the 10.0.0.0 net. You delete the route that is bound to the VIP on the MS-Loopback adapter with a command like

```
route del 10.0.0.0 mask 255.0.0.0 10.10.10.11
```

Johan Ronkainen `jr@mpoli.fi>`

True with Windows NT4. However with Win2000 you can just configure high metric value for loopback interface. I tried this about year ago with metric value 254 without problems.

I think you could change netmask to /32 with regedit. Haven't tried that with WinNT4/2000 tho. I've used that trick with Win98SE and it worked.

7 Ipvsadm and Schedulers

ipvsadm is the user code interface to LVS. The schedulers are part of the ipvs kernel code which decide how to handle new connections.

There are patches for ipvsadm

- 29.1 (machine readable error codes for ipvsadm)
- 29.2 (stateless entry of ipvsadm commands)

7.1 Using ipvsadm

You use ipvsadm from the command line (or in rc files) to setup: -

- services/servers that the director directs (*e.g.* http goes to all realservers, while ftp goes only to one of the realservers).
- weighting given to each realserver - useful if some servers are faster than others.
- 7.3 (scheduling algorithm)

You use can also use ipvsadm to

- add services: add a service with weight >0
- shutdown services: set the weight to 0.

  This allows current connections to continue, untill they disconnect or expire, but will not allow new connections. When there are no connections remaining, you can bring down the service/realserver.
- delete services: this stops traffic for the service, but the entry in the connection table is not deleted till it times out. This allows deletion, followed shortly thereafter by adding back the service, to not affect established (but quiescent) connections.
- Once you have a working LVS, save the ipsvadm settings with ipvsadm-sav

  $ipvsadm-sav > ipvsadm.sav

- restore the ipvsadm settings, with ipvsadm-restore

  $ipvsadm-restore < ipvsadm.sav

7.2 Compile a version of ipvsadm that matches your ipvs

Compile and install ipvsadm on the director using the supplied Makefile. You can optionally compile ipvsadm with popt libraries, which allows ipvsadm to handle more complicated arguments on the command line. If your libpopt.a is too old, your ipvsadm will segv. (I'm using the dynamic libpopt).

Since you compile ipvs and ipvsadm independantly and you cannot compile ipvsadm until you have patched the kernel headers, a common mistake is to compile the kernel and reboot, forgetting to compile/install ipvsadm.

Unfortunately there is only rudimentary version detection code into ipvs/ipvsadm. If you have a mismatched ipvs/ipvsadm pair, many times there won't be problems, as any particular version of ipvsadm will work with a wide range of patched kernels. Usually with 2.2.x kernels, if the ipvs/ipvsadm versions mismatch, you'll get wierd but non-obvious errors about not being able to install your LVS. Other possibilities are that the output of ipvsadm -L will have IP's that are clearly not IPs (or not the IP's you put in) and ports that are all wrong. It will look something like this

```
[root@infra /root]# ipvsadm
IP Virtual Server version 1.0.4 (size=3D4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP  C0A864D8:0050 rr
  -> 01000000:0000      Masq    0      0          0
```

rather than

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.9.4 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:ssh rr
  -> bashfull.mack.net:ssh        Route   1      0          0
```

There was a change in the /proc file system for ipvs about 2.2.14 which caused problems for anyone with a mismatched ipvsadm/ipvs. The ipvsadm from different kernel series (2.2/2.4) do not recognise the ipvs kernel patches from the other series (they appear to not be patched for ipvs).

The later 2.2.x ipvsadms know the minimum version of ipvs that they'll run on, and will complain about a mismatch. They don't know the maximum version (produced presumably some time in the future) that they will run on. This protects you against the unlikely event of installing a new 2.2.x version of ipvsadm on an older version of ipvs, but will not protect you against the more likely scenerio where you forget to compile ipvsadm after building your kernel. The ipvsadm maintainers are aware of the problem. Fixing it will break the current code and they're waiting for the next code revision which breaks backward compatibility.

If you didn't even apply the kernel patches for ipvs, then ipvsadm will complain about missing modules and exit (*i.e.* you can't even do 'ipvsadm -h').

7.2.1 Other compile problems

Ty Beede tybeede@metrolist.net on a slackware 4.0 machine I went to compile ipvsadm and it gave me an error indicating that the iphdr type was undefined and it didn't like that when it saw the ip_fw.h header file. I added

```
#include <linux/ip.h>
```

in ipvsadm.c, which is where the iphdr #structure is defined and everything went ok

Doug Bagley doug@deja.com

The reason that it fails "out of the box" is because fwp_iph's type definition (struct iphdr) was

```
#ifdef'd out in <linux/ip_fw.h>
```

(and not included anywhere else) since the symbol __KERNEL_ was undefined.

```
Including <linux/ip.h> before <linux/ip_fw.h>
```

in the .c file did the trick.

7.3 RR and LC schedulers

On receiving a connect request from a client, the director assigns a realserver to the client based on a "schedule". The scheduler type is set with ipvsadm. The schedulers available are

- round robin (rr), weighted round robin (wrr) - new connections are assigned to each realserver in turn
- least connected (lc), weighted least connection (wlc) - new connections go to realserver with the least number of connections. This is not neccessarily the least busy realserver but is a step in that direction. Note: Doug Bagley doug@deja.com points out that *lc schedulers will not work properly if a particular realserver is used in two different LVSs.
- 8 (persistent connection)
- LBLC: a persistent memory algorythm
- DH: destination hash
- SH: source hash

The original schedulers are rr, and lc (and their weighted versions). Any of these will do for a test setup. In particular, round robin will cycle connections to each realserver in turn, allowing you to check that all realservers are functioning in the LVS. The rr,wrr,lc,wlc schedulers should all work similarly when the director is directing identical realservers with identical services. The lc scheduler will better handle situations where machines are brought down and up again (see 25.14 (thundering herd problem)). If the realservers are offering different services and some have clients connected for a long time while others are connected for a short time, or some are compute bound, while others are network bound, then none of the schedulers will do a good job of distributing the load between the realservers. LVS doesn't have any load monitoring of the realservers. Figuring out a way of doing this that will work for a range of different types of services isn't simple (see 21.12 (load and failure monitoring)).

7.4 LBLC, DH schedulers

The LBLC code (by Wensong) and the DH scheduler (by Wensong, inspired by code submitted by Thomas Proell `proellt@gmx.de`) are designed for web caching realservers (e.g. squids). For normal LVS services (eg ftp, http), the content offered by each realserver is the same and it doesn't matter which realserver the client is connected to. For a web cache, after the first fetch has been made, the web caches have different content. As more pages are fetched, the contents of the web caches will diverge. Since the web caches will be setup as peers, they can communicate by ICP (internet caching protocol) to find the cache(s) with the required page. This is faster than fetching the page from the original webserver. However, it would be better after the first fetch of a page from http://www.foo.com/*, for all subsequent clients wanting a page from http://www.foo.com/ to be connected to that realserver.

The original method for handling this was to make connections to the realservers persistent, so that all fetches from a client went to the same realserver.

The -dh (destination hash) algorythm makes a hash from the target IP and all requests to that IP will be sent to the same realserver. This means that content from a URL will not be retrieved multiple times from the remote server. The realservers (eg squids in this case) will each be retreiving content from different URLs.

> Wensong Zhang `wensong@gnuchina.org` 16 Feb 2001 Please see "man ipvsadm" for short description of DH and SH schedulers. I think some examples to use those two schedulers.
>
> Example: cache cluster shared by several load balancers.
>
> ```
> Internet
> |
> |------cache array
> |
> |-----------------------------
>    |                |
>    DH               DH
>    |                |
>  Access           Access
>  Network1         Network2
> ```
>
> The DH scheduler can keep the two load balancer redirect requests destined for the same IP address to the same cache server. If the server is dead or overloaded, the load balancer can use cache_bypass feature to send requests to the original server directly. (Make sure that the cache servers are added in the two load balancers in the same order)

7.4.1 scheduling 11.20 (squids)

The usual problem with squids not using a cache friendly scheduler is that fetches are slow. In this case the website is sending hits to several different RIPs. Some websites detect this and won't even serve you the pages.

> Palmer J.D.F. `J.D.F.Palmer@Swansea.ac.uk 18 Mar 2002` I tried an online banking site www.hsbc.co.uk. It seems that this site and undoubtedly many other secure sites don't like to see connections split across several IP addresses as happens with my cluster. Different parts of the pages are requested by different realservers, and hence different IP addresses.
>
> It gives an error saying... "...For your security, we have disconnected you from internet banking due to a period of inactivity..."

> I have had caching issues with HSBC before, they seem to be a bit more stringent than other sites. If I send the requests through one of the squids on it's own it works fine, so I can only assume it's because it is seeing fragmented requests, maybe there is a keepalive component that is requested. How do I combat this? Is this what persistence does or is there a way of making the realservers appear to all have the same IP address?

Joe

change -rr (or whatever you're running) to -dh.

Lars

Use a different scheduler, like lblc or lblcr.

7.5 Patches for multiple firewalls/gateways

If the LVS is protected by multiple firewall boxes and each firewall is doing connection tracking, then packets arriving and leaving the LVS from the same connection will need to pass through the same firewall box or else they won't be seen to be part of the same connection and will be dropped. An initial attempt to handle the firewall problem was by a 7.5.1 (patch sent in by Henrik Nordstrom), who is involved with developing *web caches (squids)* `<http://squid.sourceforge.net/hno>`.

This code isn't a scheduler, but it's in here awaiting further developements of code from Julian because it's addressing similar problems to the 7.5.2 (SH scheduler) in the next section.

> Julian 13 Jan 2002 Unfortunately Henrik's patch breaks the LVS fwmark code. Multiple gateway setups can be solved with routing and a solution is planned for LVS. Until then it would be best to contact *Henrik* `<mailto:hno@marasystems.com>` for his patch.

7.5.1 Henrik Nordstrom's patch

Here's some history of Henrik's patch.

> Henrik Nordstrom `hno@marasystems.com` 13 Jan 2002 My use of the MARK is for routing purposes of return traffic only, not at all related to the scheduling onto the farm. This to solve complex routing problems arising in borders between networks where it is impractical to know full routing of all clients. One example of what I do is like this:
>
> I have a box connected to three networks (firewall, including LVS-NAT load balancing capabilities for published services)
>
> - a - Internet
> - b - DMZ, where the farm members are
> - c - Large intranet
>
> For simplicity both Internet and intranet users connect to the same LVS IP addresses. Both networks 'a' and 'c' is complex, and maintaining a complete and correct routing table covering one of the networks (i.e. the 'c' network in the above) is on the border to impossible and error prone as the use of addresses change over time.
>
> To simplify routing decisions I simply simply want return traffic to be routed back the same way as from where the request was received. This covers 99.99% of all routing needed in such situation regardless of the complexity of the networks on the two (or more) sides without the need of any explicit routing entries. To do this I MARK the session when received using netfilter, giving it a routing mark indicating which path the session was received from. My small patch modifies

LVS to memorize this mark in the LVS session, and then restore it on return traffic received FROM the real servers. This allows me to route the return traffic from the farm members to the correct client connection using iproute fwmark based routing rules.

As farm distribution algorithms I use different ones depending on the type of application. The MARK I only use for routing of return traffic. I also have a similar patch for Netfilter connection tracking (and NAT), for the same purpose of routing return traffic. If interested search for CONNMARK in the netfilter-devel archives. The two combined allows me to make multihomed boxes who do not need to know the networks on any of the sides in detail, besides it's own IP addresses and suitable gateways to reach further into the networks.

Another use of the connection MARK memory feature is a device connected to multiple customer networks with overlapping IP addresses, for example two customers both using 192.168.1.X addresses. In such case making a standard routing table becomes impossible as the customers are not uniquely identified by their IP addresses. The MARK memory however deals with such routing at ease since it do not care about the detailed addressing as long as it possible to identify the two customer paths somehow. i.e. interface originally received on, source MAC of the router who sent us the request, or anything uniquely identifying the request as coming from a specific path.

The two problems above (not wanting to known the IP routing, or not being able to IP route) are not mutually exclusive. If you have one then the other is quite likely to occur.

Here's Henrik's announcement and the replies.

Henrik Nordstrom 14 Feb 2001 Here is a small patch to make LVS keep the MARK, and have return traffic inherit the mark.

We use this for routing purposes on a multihomed LVS server, to have return traffic routed back the same way as from where it was received. What we do is that we set the mark in the iptables mangle chain depending on source interface, and in the routing table use this mark to have return traffic routed back in the same (opposite) direction.

The patch also moves the priority of LVS INPUT hook back to infront of iptables filter hook, this to be able to filter the traffic not picked up by LVS but matchin it's service definitions. We are not (yet) interested of filtering traffic to the virtual servers, but very interested in filtering what traffic reaches the Linux LVS-box itself.

Julian - who uses NFC_ALTERED ?

Netfilter. The packet is accepted by the hook but altered (mark changed).

Julian - Give us an example (with dummy addresses) for setup that require such fwmark assignments.

For a start you need a LVS setup with more than one real interface receiving client traffic for this to be of any use. Some clients (due to routing outside the LVS server) comes in on one interface, other clients on another interface. In this setup you might not want to have a equally complex routing table on the actual LVS server itself. Regarding iptables/ipvs I currently "only" have three main issues.

- As the "INPUT" traffic bypasses most normal routes, the iptables conntrack will get quite confused by return traffic..
- Sessions will be tracked twice. Both by iptables conntrack and by IPVS.
- There is no obvious choice if IPVS LOCAL_IN sould be placed before or after iptables filter hook. Having it after enables the use of many fancy iptables options, but instead requires

> one to have rules in iptables for allowing ipvs traffic, and any mismatches (either in rulesets or IPVS operation) will cause the packets to actually hit the IP interface of the LVS server which in most cases is not what was intended.

7.5.2 Wensong's SH scheduler

which handles this firewall problem. The -sh (source hash) scheduler is for directors with multiple firewalls. The director hashes on the MAC address of the firewall. Here's Wensong's announcement:

> Wensong Zhang `wensong@gnuchina.org` 16 Feb 2001 Please see "man ipvsadm" for short description of DH and SH schedulers. I think some examples to use those two schedulers.
>
> Example: Firewall Load Balancing

```
                    |-- FW1 --|
Internet ----- SH --|         |-- DH -- Protected Network
                    |-- FW2 --|
```

> Make sure that the firewall boxes are added in the load balancers in the same order. Then, request packets of a session are sent to a firewall, e.g. FW1, the DH can forward the response packets from protected network to the FW1 too. However, I don't have enough hardware to test this setup myself. Please let me know if any of you make it work for you. :)

For initial discussions on the -dh and -sh scheduler see on the mailing list under "some info for DH and SH schedulers" and "LVS with mark tracking".

7.6 What is an "active connection"

> Joe, 14 May 2001 according to the ipvsadm man page, for "lc" scheduling, the new connections are assigned according to the number of "active connections". Is this the same as "ActConn" in the output of ipvsadm? If the number of "active connections" used to determine the scheduling is "ActConn", then for services which don't maintain connections (*e.g.* http or UDP services), the scheduler won't have much information, just "0" for all realservers?

Julian, 14 May and 23 May

It is a counter and it is incremented when new connection is created. The formula is:

```
active connections = ActConn * K + InActConn
```

where K can be 32 to 50 (I don't remember the last used value), so it is not only the active conns (which would break UDP).

> Is "active connections" incremented if the client re-uses a port?

No, the reused connections are not counted.

7.7 does rr and lc weighting equally distribute the load? - clients reusing ports.

I ran the *polygraph* `<http://www.polygraph.org/>` simple.pg test on a LVS-NAT LVS with 4 realservers using rr scheduling. Since the responses from the realservers should average out I would have expected the number of connection and load average on the realservers to be equally distributed over the realservers.

Here's the output of ipvsadm shortly after the number of connections had reached steady state (about 5 mins).

```
IP Virtual Server version 0.2.12 (size=16384)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:polygraph rr
  -> doc.mack.net:polygraph         Masq    1      0          883
  -> dopey.mack.net:polygraph       Masq    1      0          924
  -> bashfull.mack.net:polygraph    Masq    1      0          1186
  -> sneezy.mack.net:polygraph      Masq    1      0          982
```

The servers were identical hardware. I expect (but am not sure) that the utils/software on the machines is identical (I set up doc,dopey about 6 months after sneezy,bashfull). Bashfull was running 2.2.19, while the other 3 machine were running 2.4.3 kernels. The number of connections (all in TIME_WAIT) at the realservers was different for each (otherwise apparently identical) realserver and was in the range 450-500 for the 2.4.3 machines and 1000 for the 2.2.19 machine (measured with netstat -an | grep $polygraph_port |wc) and varied about 10% over a long period.

This run had been done immediately after another run and InActConn had not been allowed to drop to 0. Here I repeated this run, after first waiting for InActConn to drop to 0

```
IP Virtual Server version 0.2.12 (size=16384)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:polygraph rr
  -> doc.mack.net:polygraph         Masq    1      0          994
  -> dopey.mack.net:polygraph       Masq    1      0          994
  -> bashfull.mack.net:polygraph    Masq    1      0          994
  -> sneezy.mack.net:polygraph      Masq    1      1          992
TCP  lvs2.mack.net:netpipe rr
```

Bashfull (the 2.2.19 machine) had 900 connections in TIME_WAIT while the other (2.4.3) machines were 400-600. Bashfull was also delivering about 50% more hits to the client.

Repeating the run using "lc" scheduling, the InActConn remains constant.

```
IP Virtual Server version 0.2.12 (size=16384)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:polygraph lc
  -> doc.mack.net:polygraph         Masq    1      0          994
  -> dopey.mack.net:polygraph       Masq    1      0          994
  -> bashfull.mack.net:polygraph    Masq    1      0          994
  -> sneezy.mack.net:polygraph      Masq    1      0          993
```

The number of connections (all in TIME_WAIT) at the realservers did not change.

I've been running the polygraph simple.pg test over the weekend using rr scheduling on what (AFAIK) are 4 identical realservers in a LVS-NAT LVS. There are no ActConn and a large number of InActConn. Presumably the client makes a new connection for each request.

> The implicit persistence of TCP connection reuse can cause such side effects even for RR. When the setup includes small number of hosts and the used rate is big enough to reuse the client's port, the LVS detects existing connections and new connections are not created. This is the reason you can see some of the rs not to be used at all, even for such method as RR.

the client is using ports from 1025-4999 (has about 2000 open at one time) and it's not going above the 4999 barrier. ipvsadm shows a constant InActConn of 990-995 for all realservers, but the number of connections on each of the realservers (netstat -an) ranges from 400-900.

So if the client is reusing ports (I thought you always incremented the port by 1 till you got to 64k and then it rolled over again), LVS won't create a new entry in the hash table if the old one hasn't expired?

> Yes, it seems you have (5000-1024) connections that never expire in LVS.

Presumably because the director doesn't know the number of connections at the realservers (it only has the number of entries in its tables), and because even apparently identical realservers aren't identical (the hardware here is the same, but I set them up at different times, presumably not all the files and time outs are the same), the throughput of different realservers may not be the same.

7.8 Changing weights with ipvsadm

When setting up a service, you set the weight with a command like (default for -w is 1).

```
ipvsadm -a -t $VIP:$SERVICE -r $REALSERVER_NAME:$SERVICE $FORWARDING -w 1
```

If you set the weight for the service to "0", then no new connections will be made to that service (see also man ipvsadm, about the -w option).

> Lars Marowsky-Bree `lmb@suse.de` 11 May 2001 Setting weight = 0 means that no further connections will be assigned to the machine, but current ones remain established. This allows to smoothly take a real server out of service, ie for maintenance.
>
> Removing the server hard cuts all active connections. This is the correct response to a monitoring failure, so that clients receive immediate notice that the server they are connected to died so they can reconnect.

> Laurent Lefoll `Laurent.Lefoll@mobileway.com` 11 May 2001 Is there a way to clear some entries in the ipvs tables ? If a server reboots or crashes, the connection entries remains in the ipvsadm table. Is there a way to remove manually some entries? I have tried to remove the real server from the service (with ipvsadm -d ....), but the entries are still there.
>
>> Joe After a service (or realserver) failure, some agent external to LVS will run ipvsadm to delete the entry for the service. Once this is done no new connections can be made to that service, but the entries are kept in the table till they timeout. (If the service is still up, you can delete the entries and then re-add the service and the client will not have been disconnected). You can't "remove" those entries, you can only change the timeout values.
>>
>> Any clients connected through those entries to the failed service(s) will find their connection hung or deranged in some way. We can't do anything about that. The client will have to disconnect and make a new connection. For http where the client makes a new connection almost every page fetch, this is not a problem. Someone connected to a database may find their screen has frozen.

If you are going to set the weight of a connection, you need to first know the state of the LVS. If the service is not already in the ipvsadm table, you add (-a) it. If the service is already in the ipvsadm table, you edit (-a) it. There is no command to just set the weight no matter what the state. A patch exists to do this (from Horms) but Wensong doesn't want to include it. Scripts which dynamically add, delete or change weights on services will have to know the state of the LVS before making any changes, or else trap errors from running the wrong command.

7.9 experimental scheduling code

This section is a bit out of date now. See the 7 () new schedulers by Thomas Prouell for web caches and by Henrik Norstrom for firewalls. Ratz `ratz@tac.ch` has produced a scheduler which will keep activity on a particular realserver below a fixed level.

For this next code write to Ty or grab the code off the list server

> Ty Beede `tybeede@metrolist.net` 23 Feb 2000 This is a hack to the ip_vs_wlc.c schedualing algorithm. It is curently implemnted in a quick, ad hoc fashion. It's purpose is to support limiting the total number of connections to a real server. Currently it is implmented using the weigh value as the upper limit on the number of activeconns(connections in an established TCP state). This is a very simple implementation and only took a few minutes after reading through the source. I would like, however, to develop it further.
>
> Due to it's simple nature it will not function in several types of enviroments, those based on connectionless protocals (UDP, this uses the inactconns variable to keep track of things, simply change the activeconns varible-in the weigh check- to inactconns for UDP) and it may impose complecations when persistance is implemented. The current algorimthm simply checks that weight > activeconns before including a server in the standard wlc scheduling. This works for my enviroment, but could be changed to perhaps (weight * 50) > (activeconns * 50) + inactconns to include the inactconns but make the activeconns more important in the decison.
>
> Currently the greatest weight value a user may specify is approimalty 65000, independant of this modification. As long as the user keeps most importanly the weight values correct for the total number of connections and in porportion to one another the things should function as expected.
>
> In the event that the cluster is full, all real severs have maxed out, then it might be neccessary for overflow control, or the client's end will hang. I haven't tested this idea but it could simply be implemented by specifing the over flow server last, after the real severs using the ipvsadm tool. This will work because as each real server is added using ipvsadm it is put on a list, with the last one added being last on the list. The scheduling algorithm traverses this list linearly from start to finish and if it finds that all severs are maxed out, then the last one will be the overflow and that will be the only one to send traffic to.
>
> Anyway this is just a little hack, read the code and it should make sense. It has been included as an attachment. If you would like to test this simply replace the old ip_vs_wlc.c scheduling file in /usr/src/linux/net/ipv4 with this one. Compile it in and set the weight on the real severs to the max number of connections in an established TCP state or modifiy the source to your liking.
>
> From: Ty Beede `tybeede@metrolist.net` 28 Feb 2000
>
> I wrote a little patch and posted it a few days ago... I indicated that overflow might be accomplished by adding the overflow server to the lvs last. This statement is completely off the wall wrong. I'm not really sure why I thought that would work but it won't, first of all the linked list adds each new instance of a real sever to the start of the real servers list, not the end like I though. Also it would be impossible do distingish the overflow server from the real servers in the case that not all the realservers were busy. I don't know where I got that idea from but I'm going to blame it on my "bushy eyed youth". In responce to needing overflow support I'm thinking about implementing "prority groups" into the lvs code. This would logically group the real severs into different groups, though with a higher priority group would fillup before those with a lower grouping. If anybody could comment on this it would be nice to hear what the rest of you think about overflow code.

7.9.1 Theoretical issues in developing better scheduling algorithms

Julian

It seems to me it would be useful in some cases to use the total number of connections to a real server in the load balancing calculation, in the case where the real server participates in servicing a number of different VIPs.

> Wensong Yeah, it is true. Sometimes, we need tradeoff between simplicity/performance and functionality. Let me think more about this, and probably maximum connection scheduling together together too. For a rather big server cluster, there may be a dedicated load balancer for web traffic and another load balancer for mail traffic, then the two load balancers may need exchange status periodically, it is rather complicated.

Yes, if a real server is used from two or more directors the "lc" method is useless.

> Actually, I just thought that dynamic weight adaption according to periodical load feedback of each server might solve all the above problems.

Joe - this is part of a greater problem with LVS, we don't have good monitoring tools and we don't have a lot of information on the varying loads that realservers have, in order to develope strategies for informed load regulation. See 21.12 (load and failure monitoring).

> Julian From my experience with real servers for web, the only useful parameters for the real server load are:
>
> - cpu idle time
>
>   If you use real servers with equal CPUs (MHz) the cpu idle time in percents can be used. In other cases the MHz must be included in a expression for the weight.
>
> - free ram
>
>   According to the web load the right expression must be used including the cpu idle time and the free ram.
>
> - free swap
>
>   Very bad if the web is swapping.
>
> The easiest parameter to get, the Load Average is always < 5. So, it can't be used for weights in this case. May be for SMTP ? The sendmail guys use only the load average in sendmail when evaluating the load :)
>
> So, the monitoring software must send these parameters to all directors. But even now each of the directors use these weights to create connections proportionally. So, it is useful these parameters for the load to be updated in short intervals and they must be averaged for this period. It is very bad to use current value for a parameter to evaluate the weight in the director. For example, it is very useful to use something like "Average value for the cpu idle time for the last 10 seconds" and to broadcast this value to the director on each 10 seconds. If the cpu idle time is 0, the free ram must be used. It depends on which resource zeroed first: the cpu idle time or the free ram. The weight must be changed slightly :)
>
> The "*lc" algorithms help for simple setups, eg. with one director and for some of the services, eg http, https. It is difficult even for ftp and smtp to use these schedulers. When the requests are very different, the only valid information is the load in the real server.
>
> Other useful parameter is the network traffic (ftp). But again, all these parameters must be used from the director to build the weight using a complex expression.

I think the complex weight for the real server based on connection number (lc) is not useful due to the different load from each of the services. May be for the "wlc" scheduling method ? I know that the users want LVS to do everything but the load balancing is very complex job. If you handle web traffic you can be happy with any of the current scheduling methods. I didn't tried to balance ftp traffic but I don't expect much help from *lc methods. The real server can be loaded, for example, if you build new Linux kernel while the server is in the cluster :) Very easy way to switch to swap mode if your load is near 100%.

8 Persistent connection

The term "persistence" has 2 meanings in setting up an LVS. There is "persistent connection" a term used for connecting to webservers and databases and "persistent connection" used in LVS. These are quite different.

8.1 netscape/database/tcpip persistence

Persistant connection outside of LVS is described in *http persistent connection* `<http://www.research.compaq.com/wrl/techreports/abstracts/95.4.html>` and is an application level protocol. It works this way:

In a normal http (or database connection), after the server has sent it's reply, it shuts down the tcpip connection. This makes your session with the server stateless - the server has no record of previous packets/data/state sent to it. If the payload is small (eg 1 packet), then you've gone through a lot of handshakes and packet exchanges to deliver one packet. To solve this, http persistent connection was invented. Both the client and server must be persistence-enabled for this to work. At connect time, the client and server notify each other that they support persistent connection. The server uses an algorithm to determine when to drop the connection (timeout, needs to recover file handles...). The client can drop the connection at anytime without consulting the server. This requires more resources from the server as file handles can be open for much longer than the time needed for a tcpip transfer.

8.2 LVS persistence

LVS persistence makes a client connect to the same realserver for *different* tcpip connections. This is used when

- state must be maintained on the server, *e.g.* for https key exchanges
- a client requests related services which are not on the same port using a pair of ports
  - 20,21 for active ftp
  - 21 and some high port for passive ftp
  - port 80,443 for an e-commerce site
- a client comes through a proxy which presents different IPs for the CIP for different tcpip connections from the same CIP.

Another writeup on persistence is on the *LVS persistence page* `<http://www.linuxvirtualserver.org/docs/persistence.html>`.

The default timeout for LVS persistence is 360secs (used to be 600 secs). The default timeout for a regular LVS connection via LVS-DR is TIME_WAIT (about 1 minute). This means that LVS persistent connections will stay in the LVS connection table for 6 times longer for persistent connection. As a consequence the hash table (and memory requirements) will be 6 times larger for the same number of connections/sec. Make sure you have enough memory to hold the increased table size if you're using persistent connections. If the persistence is being used to hold state (*e.g.* shopping cart), then you must allow a long enough timeout for the client to surf to another site for a better price, make a cup of coffee, think about it and then go find their credit card. This is going to be much longer than any reasonable timeout for LVS persistence and the state information will have to be held on a disk somewhere on the realservers and you'll have to allow for the client to appear on a different realserver later with their credit card information.

The LVS persistant (or sticky) connection is at the layer 4 protocol level. This is not the same as the persistent connection described above for netscape persistence or database persistence of a single tcpip connection. Unfortunately, both features are persistent and can reasonably claim the name "persistent", and this causes some confusion. LVS could alternately be described as connection affinity or port affinity.

> Wensong Zhang `wensong@gnuchina.org` 11 Jan 2001 The working principle of persistence in LVS is as follows:
>
> - a persistent template is used to keep the persistence between the client and the server.
> - when the first connection from a client, the LVS box will select a server according to the scheduling algoriths, then create a persistent template and the connection entry. the control of the connection entry is the template.
> - The late connections from the clients will be forwarded to the same server, as long as the template doesn't expire. The control of their connection entries are the template.
> - If the template has its controlled connections, it won't expire.
> - If the template has no controlled connections, it expires in its own time.

You can trace your system in the following way. For example:

```
[root@kangaroo /root]# ipvsadm -ln
IP Virtual Server version 1.0.3 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InInActConn
TCP  172.26.20.118:80 wlc persistent 360
  -> 172.26.20.91:80             Route   1      0          0
  -> 172.26.20.90:80             Route   1      0          0
TCP  172.26.20.118:23 wlc persistent 360
  -> 172.26.20.90:23             Route   1      0          0
  -> 172.26.20.91:23             Route   1      0          0

[root@kangaroo /root]# ipchains -L -M -n
IP masquerading entries
prot expire   source             destination         ports
TCP  02:46.79 172.26.20.90       172.26.20.222       23 (23) -> 0
```

Although there is no connection, the template isn't expired. So, new connections from the client 172.26.20.222 will be forwarded to the server 172.26.20.90.

> Bowie Bailey If I start a service with:
>
> ```
> ipvsadm -A -f 1 -s wlc -p 180
> ```
>
> and then change the persistence flag with:
>
> ```
> ipvsadm -E -f 1 -s wlc -p 180 -M 255.255.255.0
> ```

> how does that affect the connections that have already been made?

Julian 30 Jul 2001

The connections are already established. But the persistence is broken and after changing the netmask you can expect the next connections to be established to another real servers (not to the same as before the change).

(also see 8.8 (persistence netmask)).

> If IP address 1.2.3.4 was connected to RIP1 before I changed the persistence and then 1.2.3.5 tries to connect afterwards, would he be sent to RIP1, or would it be considered a new connection and possibly be sent to either server since the mask was 255.255.255.255 when the first connection happened?

New realserver will be selected.

8.3 persistent client connection, pcc (for kernel =<2.2.10)

All connections from an IP go to same realserver. Timeout for inactive connections is 360 sec. pcc is designed for https and 11.25 (cookie) serving. With ppc, after the first connection (say to port 80) any subsequent connections requests from the same client but from another port (eg 443) will be sent to the same realserver. The problem with this is that about 25% of the people on the internet have the same IP (AOL customers are connected to the internet via a server in Virginnia, USA). If you have pcc set, then after the first client connects from AOL, then all subsequent connections from AOL will go to the same realserver, until the last AOL client disconnects. This effect will override attempts to distribute the load between realservers.

8.4 persistent port connection (ppc) (for kernel >= 2.2.12)

With kernel 2.2.12, the persistent connection feature has been changed from a scheduling algorythm (you get rr|wrr|lc|wlc|pcc) to a switch (you can have persistent connection with rr|wrr|lc|wlc). If you do not select a scheduling algorithm when asking for a persistent connection, ipvsadm will default to wlc.

The difference between pcc and ppc is probably of minor consequence to the LVS admin (if you want persistent connection, you have to have it and you don't care how you got it). With ppc, connections are assigned on a port by port basis. With ppc, if both port 80 and 443 were persistant, then connections from the same client would not neccessarily go to the same realserver. This solves the AOL problem.

If you are handing out 11.25 (cookies) to a client on port 80 and they need to go to port 443 to give their credit card, you want them going to the same realserver. There is no way to make ports sticky by groups (or pairs), so for the moment you emulate the pcc connection by using port 0.

8.5 Problems: Removing persistant connections after a realserver crash

> Patrick Kormann `pkormann@datacomm.ch` I have the following problem: I have a direct routed 'cluster' of 4 proxies. My problem is that even if the proxy is taken out of the list of real servers, the persistent connection is still active, that means, that proxy is still used.

> Andres Reiner Now I found some strange behaviour using 'mon' for the high-availability. If a server goes down it is correctly removed from the routing table. BUT if a client did a request prior to the server's failure, it will still be directed to the failed server afterwards. I guess this

got something to do with the persistent connection setting (which is used for the cold fusion applications/session variables).

In my understanding the LVS should, if a routing entry is deleted, no longer direct clients to the failed server even if the persistent connection setting is used.

Is there some option I missed or is it a bug ?

Wensong Zhang wrote:

No, you didn't miss anything and it is not a bug either. :)

In the current design of LVS, the connection won't be drastically removed but silently drop the packet once the destination of the connection is down, because monitering software may marks the server temporary down when the server is too busy or the monitering software makes some errors. When the server is up, then the connection continues. If server is not up for a while, then the client will timeout. One thing is gauranteed that no new connections will be assigned to a server when it is down. When the client reestablishs the connection (e.g. press reload/refresh in the browser), a new server will be assigned.

`jacob.rief@tis.at` wrote: Unfortunately I have the same problem as Andres (see below) If I remove a real server from a list of persistent virtual servers, this connection never times out. Not even after the specified timeout has been reached.

Wensong

The persistent template won't timeout until all its connections timeout. After all the connections from the same client connection expires, new connections can be assigned to one of the remaining servers. You can use "ipchains -M -L -n" (or netstat -M) to check the connection table (for 2.4.x use cat /proc/net/ip_conntrack).

Only if I unset persisency the connection will be redirected onto the remaining real servers. Now if I turn on persistency again, a prevoiusly attached client does not reconnect anymore - it seems as if LVS remembers such clients. It does not even help, if I delete the whole virtual service and restore it immediately, in the hope to clear the persistency tables.

```
ipvsadm -D -t <VIP>; ipvsadm -A -t <VIP> -p; ipvsadm -a -t <VIP> -R <alive real server>
```

And it also does not help closing the browser and restarting it. I run LVS in masquerading mode on a 2.2.13-kernel patched with ipvs-0.9.5. Would'nt it be a nice feature to flush the persistent client connection table, and/or list all such connections?

Wensong

There are several reasons that I didn't do it in the current code. One is that it is time-consuming to search a big table (maybe one million entries) to flush the connections destined for the dead server; the other is that the template won't expire until its connection expire, the client will be assigned to the same server as long as there is a connection not expired. Anyway, I will think about better way to solve this problem.

valery brasseur I would like to to load balancing based on cookie and/or URL,

Wensong

Have a look at http://www.LinuxVirtualServer.org/docs/persistence.html :-)

Joe

also see 11.25 (cookie)

> matt matt@paycom.net
>
> I have run into a problem with the persistant connection flag, I'm hoping that someone can help me. First off, I don't think there is anything like this out now, but, is there anyway to load-balance via URL? Such as http://www.matthew.com being balanced among 5 servers without persistant connections turned on, and http://www.matthew.com/dynamic.html being flagged with persistance? Second question is this; I don't exactly need a persistant connection, but I do need to make sure that requests from a particular person continue to go to the same server. Is there any way to do this?

James CE Johnson jcej@tragus.org Jul 2001

We ran into something similar a while back. Our solution was to create a simple Apache module that pushes a cookie to the browser the when the "session" begins (eg – when no cookie exists). The content of the cookie is some indicator of the realserver. On the second and subsequent requests the Apache module sees the cookie and uses the Apache proxy mechanism to forward the request to the appropriate realserver and return the results.

> unknown Let's say: I have 1000 http requests (A) through a firewall of a customer (so in fact all requests have the same Source IP for Loadbalancer, because of NAT) and then one request (B) from the Intranet and then again 1000 Request (C) from that firewall, what does LB do? I have three Realservers r1, r2, r3 (ppc with rr)

```
a) A to r1, B to r2, C to r1 (because of SourceIP) [Distribution:2000:1:0.0000001]
b) A to r1, B to r2, C to r3 (because r3 is free) [Distribution:1000:1:1000]
c) A to r1, B to r2, C to r2 (due to the low load of r2) [Distribution:1000:1000:0.000001]
A to r1 && r2 && r3 (depending on source port),
B to r1 || r2 || r3,
C to r1 && r2 && r3 [Distribution: 667:667:666]
```

Ratz ratz@tac.ch 12 Sep 1999

If C reachs the load balancer before all the 1000 requests of A expire, then the requests of C will be sent to r1, and the distribution is 2000:1:0.

If all the requests of A expires, the requests of C will be forwarded to a server that is selected by a scheduler.

BTW, persistent port is used to solve the connection affinity problem, but it may lead to dynamic load imbalance among servers.

> Jean-Francois Nadeau I will use LVS to load balance web servers (Direct Routing and WRR algo). I use persitency with a big timeout (10 minutes). Many of our clients are behind big proxies and I fear this will unbalance our cluster because of the persitent timeout.

Wensong persistent virtual services may lead to the load imbalance among servers. Using some weight adapation approaches may help avoid that some servers are overloaded for a long time. When the server is overloaded, decrease its weight so that connections from new clients won't be sent to that server. When the server is underloaded, increase its weight.

> Can we alter directly /proc/net/ip_masquerade ?

No, it is not feasible, because directly modifying masq entries will break the established connection.

8.6 Persistent and regular services are possible on the same realserver.

If you setup a 2 realserver LVS-DR LVS with persistence,

```
ipvsadm -A -t $VIP -p -s -rr
ipvsadm -a -t $VIP -R $realserver1 $VS_DR -w 1
ipvsadm -a -t $VIP -R $realserver2 $VS_DR -w 1
```

giving the ipvsadm output

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.5 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:0 rr persistent 360
  -> bashfull.mack.net:0         Route   1      0          0
  -> sneezy.mack.net:0           Route   1      0          0
```

then (as expected) a client can connect to any service on the realservers (always getting the same realserver).

If you now add an entry for telnet to both realservers, (you can run these next instructions before or after the 3 lines immediately above)

```
ipvsadm -A -t $VIP:telnet -s -rr
ipvsadm -a -t $VIP:telnet -R $realserver1 $VS_DR -w 1
ipvsadm -a -t $VIP:telnet -R $realserver2 $VS_DR -w 1
```

giving the ipvsadm output

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.5 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:0 rr persistent 360
  -> bashfull.mack.net:0         Route   1      0          0
  -> sneezy.mack.net:0           Route   1      0          0
TCP  lvs2.mack.net:telnet rr
  -> sneezy.mack.net:telnet      Route   1      0          0
```

the client will telnet to both realservers in turn as would be expected for an LVS serving only telnet, but all other services (ie !telnet) go to the same first realserver. All services but telnet are persistent.

The director will make persistent all ports except those that are explicitely set as non-persistent. These two sets of ipvsadm commands do not overwrite each other. Persistent and non-persistent connections can be made at the same time.

> JulianThis is part of the LVS design. The templates used for persistence are not inspected when scheduling packets for non-persistent connections.

8.7 Examples of persistence

- ftp (VS-NAT): connections to both ftp ports (for both active and passive ftp) are handled by the module ip_masq_ftp. You don't need to add persistence for ftp with LVS-NAT.
- ftp (VS-DR or LVS-Tun): you need persistence on the realservers. Run the first set of commands above.
- ftp and http (VS-NAT): persistence not needed (ip_masq_ftp handles the ftp ports for active and passive ftp).
- ftp and http (VS-DR or LVS-Tun): persistence needed to handle the two port protocol ftp. You can set this up by running the first set of commands above. A secondary consequence of this arrangement is that a client connecting to the http service of the LVS will always get the same realserver (this may not be a great problem). If you want to make the http service non-persistent but leaving all other services persistent, then run the second set of commands (above) as well.
- http and https (all forwarding methods): Normally an https connection is made after the client has made selections on an http connection when data is stored on the realserver for the client. In this case the realserver should be made persistent (first set of commands above). Do not run ipvsadm commands for http (like the second set of commands above) as this will make the http connections non-persistent.

Note: making realserver connections persistent allows _all_ ports to be forwarded by the LVS to the realservers. Non-persistent LVS connections are only for the nominated service. An open, persistently connected realserver then is a security hazard. You should run ipchains commands on the director to block all services on the VIP except those you want forwarded to the realservers.

8.8 AOL and proxies, persistence netmask

(also see 9.8 (persistence granularity with fwmark).)

Because of the way proxies can work, a client can come from one IP for one connection (eg port 80) and from another IP for the next connection (eg port 443) and will appear to be two different clients. Usually these two connections will come from the same /24 netmask. You can set the netmask for persistence to /24 and all clients from the same class C network will be sent to the same realserver.

valery brasseur

I have seen some discussion about "proxy farm" such as AOL or T-Online,

> Wensong If you want to build a persistent proxy cluster, you just need set a LVS box at the front of all proxy servers, and use the persistent port option in the ipvsadm commands. BTW, you can have a look at *how to build a big JANET cache cluster using LVS* `<http://wwwcache.ja.net/JanetServices/PilotServices.html>`.
>
> If you want to build a persistent web service but some proxy farms are non-persistent at client side, then you can use the persistent granularity so that clients can be grouped, for example you use 255.255.255.0 mask, the clients from the same /24 network will go to the same server.

While persistence can be used for services that require multiple ports eg ftp/ftp-data, http/https it can be useful for ssl services.

Here's an example of using persistence granularity (from Ratz 3 Jan 2001). The -M 255.255.255.255 sets up /32 granularity. Here port 80 and port 443 are being linked by fwmarks.

```
ipchains -A input -j ACCEPT -p tcp -d 192.168.1.100/32 80 -m 1 -l
ipchains -A input -j ACCEPT -p tcp -d 192.168.1.100/32 443 -m 1 -l
ipvsadm -A -f 1 -s wlc -p 333 -M 255.255.255.255
ipvsadm -a -f 1 -r 192.168.1.1 -g -w 1
ipvsadm -a -f 1 -r 192.168.1.2 -g -w 1
```

For more information on persistence granularity see the section on 9.8 (persistence granularity with fwmark). It's use for fwmark is the same as for VIP.

> Francis Corouge wrote: I made a LVS-DR lvs. All services work well, but with IE 4.1 on secured connection, pages are received randomly. when you make several requests, sometime the page is displayed, but sometimes a popup error message is displayed
>
> ```
> Internet Explorer can't open your Internet Site <url>
> An error occured with the secured connexion.
> ```
>
> I did not test with other versions of IE, but netscape works fine. It works when I connect directly to the real server (realserver disconnected from the LVS, and the VIP on the realserver allowed to arp).

Julian

Is the https service created persistent? *i.e.* using ipvsadm -p

> Joe Why does persistence fix this problem? (also see http://www.linuxvirtualserver.org/docs/persistence.html)

Julian

I assume the problem is in the way SSL is working: cached keys, etc. Without persistence configured, the SSL connections break when they hit another real server.

> what is (or might be) different about IE4 and Netscape?

Maybe in the way the bugs are encoded. But I'm not sure how the SSL requests are performed. It depends on that too.

Example 1. https only

This is done with persistent connection.

lvs_dr.conf config file excerpt (Oct 2001, this syntax doesn't work anymore, persistence is set by the services in @persistent_services).

SERVICE=t https ppc 192.168.1.1

output from ipvsadm

```
ipvsadm settings
IP Virtual Server version 0.9.4 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  ssl.mack.net:https wlc persistent 360
  -> di.mack.net:https           Route   1      0          0
```

Example 2. All ports sticky, timeout 30mins, wrr scheduling

lvs_dr.conf config file excerpt

SERVICE=t 0 wrr ppc -t 1800 192.168.1.1 (Oct 2001, this syntax doesn't work anymore, persistence is set by the services in @persistent_services).

which specifies tcp (t), service (all ports = 0), weighted round robin scheduling (wrr), timeout 1800 secs (-t 1800), to realserser 192.168.1.1.

Here's the code generated by the 10.1 (configure script)

```
#ppc persistent connection, timeout 1800 sec
/sbin/ipvsadm -A -t 192.168.1.110:0 -s wrr -p 1800
echo "adding service 0 to realserver 192.168.1.1 using connection type dr weight 1"
/sbin/ipvsadm -a -t 192.168.1.110:0 -R 192.168.1.1 -g -w 1
```

here's the output of ipvsadm

```
# ipvsadm
IP Virtual Server version 0.9.4 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  ssl.mack.net:https wrr persistent 1800
  -> di.mack.net:https           Route   1      0          0
```

> Jeremy Johnson jjohnson@real.com how does LVS handles a single client that uses multiple proxies... for instance aol, when an aol user attempts to connect to a website, each request can come from a different proxy so, how/if does LVS know that the request is from the same client and bind them to the same server?

Joe

if this is what aol does then each request will be independant and will not neccessarily go to the same realserver. Previous discussions about aol have assumed that everyone from aol was coming out of the same IP (or same class C network). Currently this is handled by making the connection persistant and all connections from aol will go to one realserver.

> Michael Sparks zathras@epsilon3.mcc.ac.uk If ISP user (eg AOL) has a proxy array/farm then the requests are _likely_ to come from two possibilities:
>
> - A single subnet (if using an L4/L7 switch that rewrites ether frames, or using several NAT based L4/L7 switches)
>
>   or
> - A single IP (If using the common form of L4/L7 switch)
>
> The former can be handled using a subnet mask in the persistance settings, the latter is handled by normal persistance.
>
> *However* In the case of our proxy farm neither of these would work since we have 2 subnet ranges for our systems - 194.83.240/24 & 194.82.103/24, and an end user request may come out of each subnet totally defeating the persistance idea... (in fact dependent on our clients configuration of their caches, the request could appear to come from the above two subnets or the above 2 subnets and about 1000 other ones as well)
>
> Unfortunately this problem is more common that might be obvious, due to the NLANR hierarchy, so whilst persistance on IP/subnet solves a large number of problems, it can't solve all of them.

> Billy Quinn bquinn@ifleet.com 05 Jun 2001 I've come to conclusion that I need an expensive (higher layer) load balancer node , which load balances port 80 (using persistence because of sessions) to 3 real servers which each run an apache web server, and tomcat servlet engine. Each of the 3 servers is independent and no tomcat load balancing occurs.
>
> This has worked great for about a year, while we only had to support certain IP address ranges. Now, however, we have to support clients using AOL and their proxy servers, which completely messes up the session handling in tomcat. In other words, one client comes from multiple different IP addresses based on which proxy server it comes through.
>
> It seems the thing to do is to adjust the persistence granularity . However, if I adjust the netmask, all of our internal network traffic will go to one server, which kind of defeats the purpose.
>
> What I'm concluding is, that I'll need to change the network architecture (since we are all on one subnet), or buy a load balancer which will look at the actual data in the packets (layer 7?).

Joe

There has been comments by people dealing with this problem (not many), but they seem to be still able to use LVS. We don't hear of anyone who is having lots of trouble with this, but it could be because no-one on this list is dealing with AOL as a large slice of their work.

If 1/3 of your customers are from AOL you could sacrifice one server to them, but it's not ideal. If all your customers are from AOL, I'd say we can't help you at the moment.

> My concern with that would be anyone else doing proxying ... now or in the future . I would not be opposed to routing all of the AOL customers to one server for now though . I guess we could have to deal with each case of proxying individually. I wonder how many other ISP's do proxying like that

How many different proxy IPs do AOL customers arrive on the internet from? How many will appear from multiple IP's in the same session and how big is the subnet they come from? (/24?)

> Good question, I'm not sure about that one. The customer that reported the problem seemed to be coming from about 2-4 different IP addresses (for the same session).

If AOL customers come from at least 3 of these subnets and you have 3 servers, then you can use LVS as a balancer.

> Peter Mueller pmueller@sidestep.com Over here we also need layer-7 'intelligent' balancing with our apache/jakarta setup. We utilize two tiers of 'load-balancing'. One is the initial LVS-DR round-robin type setup, while the second layer is our own creation, layer-7. Currently we round-robin the first connection to one server, then that server calls a routine that will ask the second-tier layer-7 java monitor boxes which box to send the connection to. (If for some reason the second layer is down, standard round-robin occurs).
>
> We're about 50% done with migration from cisco LD (yuck!) to LVS-DR. After the migration is fully complete the goal is to have the two layers interacting more efficiently and hopefully merged into one 'layer' eventually.. for example, if we tell our java-monitor second-tier controllers to shutdown a server, the first tier will then mark the node out of service automatically.
>
> PS - we found the added layer-7 intelligent balancing to be about 30-50% (?) added effectiveness to cisco round robin LD.. I think the analogy of a hub versus a switch works fairly well here..

> Chris Egolf cegolf@refinedsolutions.net> We're having the exact same problem with WebSphere cookie-based sessions. I was testing this earlier today and I think I've solved this particular problem by using firewall marks.

Basically, I'm setting everything from our internal network with one FWMARK and everything else with another. Then, I setup the ipvsadm rules with the default client persistence for our internal network(/32) and a class C netmask granularity (/24) for everything from the outside to deal w/ the AOL proxy farms.

Here's the iptables script I'm using to set the marks:

```
iptables -F -t mangle
iptables -t mangle -A PREROUTING  -p tcp -s 10.3.4.0/24 -d $VIP/32 \
            --dport 80 -j MARK --set-mark 1
iptables -t mangle -A PREROUTING  -p tcp -s ! 10.3.4.0/24 -d $VIP/32 \
            --dport 80 -j MARK --set-mark 2
```

Then, I have the following rules setup for ipvsadm:

```
ipvsadm -C
ipvsadm -A -f 1 -s wlc -p 2000
ipvsadm -a -f 1 -r $RIP1:0 -g -w 1
ipvsadm -a -f 1 -r $RIP2:0 -g -w 1

ipvsadm -A -f 2 -s wlc -p 2000 -M 255.255.255.0
ipvsadm -a -f 2 -r $RIP1:0 -g -w 1
ipvsadm -a -f 2 -r $RIP2:0 -g -w 1
```

FWMARK #1 doesn't have a persistent mask specified, so each client on the 10.3.4.0/24 network is seen as an individual client. FWMARK #2 packets are seen as a class C client network to deal with the AOL proxy farm problem. (for more on persistent netmask see the section in fwmark on 9.8 (fwmark persistence granularity)).

Like I said, I just did this today, and based on my limited testing, I think it works. I'm thinking about maybe setting a whole bunch of rules to deal w/ each of the published AOL cache-proxy server networks (http://webmaster.info.aol.com/index.cfm?article=15&sitenum=2), but I think that would be too much of an administrative nightmare if they change it.

The ktcpvs project implements some level of layer-7 switching by matching URL patterns, but we need the same type of cookie based persistence for our WebSphere real servers. Hopefully, it won't be too long before that gets added.

8.9 PPC (persistent port connection) (kernels >2.2.12)

(Note: Jul 2001, this is quite old now, from at least 2yrs ago. If you've got this far, you probably don't need to go further.)

In earlier kernels, persistence was implemented with PCC (persistent client connection).

PPC is used for clients who must maintain a session with the same realserver throughout a session (eg for various SSL protocols, to an http server sending 11.25 (cookies)...). The default session timeout is 5mins (ip_vs_pcc.h).

PCC (kernel <2.2.12) was removed in 2.2.12 and has resurfaced as a more general persistance feature called persistant port. PCC connects (some or all) ports from a client IP through to the same ports on a single realserver (the realserver is selected on the first connection, after than all subsequent port requests from the same IP go to the same realserver). With persistant port, the persistant connection is on a port by port basis and not by IP. If persistant port is called with a port of "0" then the connection will be the same as PCC.

here's the syntax for 2.2.12 kernels

Wensong To use persistent port, the commands are as follows:

```
ipvsadm -A -t <VIP>:<port> [-s <scheduler>] -p
ipvsadm -a -t <VIP>:<port> -R <real server> ...
        ...
```

if port=0 then all ports from the CIP will be mapped through to the realserver (the old PCC behaviour). If port=443, then only port 443 from the CIP will be mapped through to the realserver as a persistant connection.

If the virtual service port is set persistent, connections from the same clients are gauranteed to direct to the same server. When a client sends request for the service at the first time, the load balancer (director) selects a server by the scheduling method and creates a connection and the template. Then, the following connections from the same client will be forwarded to the same server according to the template in the specified time.

The source address of an incoming packet is used to lookup connection template.

from Peter Kese (who implemented pcc)

PCC (persistent client connection) scheduling algorithm needs some more explanation. When PCC scheduling is used, the connections are scheduled on a per client base instead of per connection. That means, the scheduling is performed only the first time a certain client connects to the virtual IP. Once the real server is chosen, all further connections from the same client will be forwarded to the same real server.

PCC scheduling algorithm can either be attached to a certain port or to the server as whole. By setting the service port to 0 (example: ipvscfg -A -t 192.168.1.10:0 -s pcc) the scheduler will accept all incoming connections and will schedule them to the same real server no matter what the port number is.

As Wensong had noted before, the PCC scheduling algorithm might produce some imbalance of load on real servers. This happens because the number of connections established by clients might vary a lot. (There are some large companies for example, that use only one IP address for accessing the internet. Or think about what happens when a search engine comes to scan the web site in order to index the pages.) On the other hand, the PCC scheduler resolves some problems with certain protocols (e.g. FTP) so I think it is good to have it.

and a comment about load balancing using pcc/ssl. (the problem: once someone comes in from aol.com to one of the realservers, all subsequent connections from aol.com will also go to the same server) -

Lars

Lets examine what happens now with SSL session comeing in from a big proxy, like AOL. Since they are all from the same host, they get forwarded to the same server - *thud*.

Now, SSL carries a "session id" which identifies all requests from a browser. This can be used to separate the multiple SSL sessions, even if comeing in from one big proxy and load balance them.

> (from unknown) SSL connections will not come from the same port, since the clients open many of them at once, just like with normal http. So would we be able to differentiate all the people coming from aol by the port number?

No. A client may open multiple SSL connections at once, which obviously will not come from the same port - but I think they will come in with the same SSL id.

> (unknown again) But like I said: really hard to get working, and even harder to get right ;-)

Wensong

No, not really! As I know, the PCC (Persistent Client Connection) scheduling in the VS patch for kernel 2.2 can solve connection affinity problem in SSL.

When a SSL connection is made (crypted with server's public key), port 443 for secure Web servers and port 465 for secure mail server, a key (session id) must be generated and exchanged between the server and the client. The later connections from the same client are granted by the server in the life span of the SSL key.

So, the PCC scheduling can make sure that once SSL "session id" is exchanged between the server and the client, the later connections from the same client will be directed to the same server in the life span of the SSL key.

However, I haven't tested it myself. I will download ApacheSSL and test it sometime. Anyone who have tested or are going to test it, please let me know the result, no matter it is good or bad. :-)

(a bit later)

> (unknown) I tested LVS with servers running Apache-SSL. LVS uses the VS patch for kernel 2.2.9, and uses the PCC scheduling. It worked without any problem.

SSL is a little bit different.

In use, the client will send a connection request to the server. The server will return a signed digital certificate. The client then authenticates the certificate using the digital signature and the public key of the CA.

If the certificate is not authentic the connection is dropped. If it is authentic then the client sends a session key (such as a) and encrypts the data using the servers public key. This ensures only the server can read it since decrypting requires knowing the server private key. The server sends its session key (such as b) and encrypts with its private key, the client decrypt it with server's public key and get b.

Since both the client and the server get a and b, they can generate the same session key based on a and b. Once they have the session key, they can use this to encrypt and decrypt data in communication. Since the data sent between the client and server is encrypted, it can't be read by anyone else.

Since the key exchange and generating is very time-consuming, for performance reasons, once the SSL session key is exchanged and generated in a TCP connection, other TCP connections can also use this session key between the client and the server in the life-span of the key.

So, we have make the connections from the same client is sent to the same server in the life-span of the key. That's why the PCC scheduling is used here.

About longer timeouts

felix k sheng `felix@deasil.com` and Ted Pavlic

> 2. The PCC feature....can I set the permanent connection for something else than the default value (I need to maintain the client on the same server for 30 minutes at maximum) ?
>
> If people connecting to your application will contact your web server at least once every five minutes, setting that value to five minutes is fine. If you expect people to be idle for up to thirty minutes before contacting the server again, then feel free to change it to thirty minutes. Basically remember that the clock is reset every time they contact the server again. Persistence lasts for as long as it's needed. It only dies after the amount of seconds in that value passes without a connection from that address.
>
> So if you really want to change it to thirty minutes, check out ip_vs_pcc.h – there should be a constant that defines how many seconds to keep the entry in the table. (I don't have access to a machine with IPVS on it at this location for me to give you anything more precise)

I think this 30 minute idea is a web specific time out period. That is, default timeout's for 11.25 (cookies) are 30 minutes, so many web sites use that value as the length of a given web "session". So if a user hits your site, stops and does nothing for 29 minutes, and then hits your site again, most places will consider that the same session - the same session cookies will still be in place. So it would probably be a nice to have them going to the same server.

8.10 Related to PPC - Sticky connections

(Joe, Jul 2001 - This is also quite old now too)

Wensong

Since there are many messages about passive ftp problem and sticky connection problem, I'd better send a separate message to make it clear.

In LinuxDirector (by default), we have assumed that each network connection is independent of every other connection, so that each connection can be assigned to a server independently of any past, present or future assignments. However, there are times that two connections from the same client must be assigned to the same server either for functional or for performance reasons.

FTP is an example for a functional requirement for connection affinity. The client establishs two connections to the server, one is a control connection (port 21) to exchange command information, the other is a data connection (usually port 20) that transfer bulk data. For active FTP, the client informs the server the port that it listens to, the data connection is initiated by the server from the server's port 20 and the client's port. LinuxDirector could examine the packet coming from clients for the port that client listens to, and create any entry in the hash table for the coming data connection. But for passive FTP, the server tells the clients the port that it listens to, the client initiates the data connection connectint to that port. For the LVS-Tunneling and the LVS-DRouting, LinuxDirector is only on the client-to-server half connection, so it is imposssible for LinuxDirector to get the port from the packet that goes to the client directly.

SSL (Secure Socket Layer) is an example of a protocol that has connection affinity between a given client and a particular server. When a SSL connection is made, port 443 for secure Web servers and port 465 for secure mail server, a key for the connection must be chosen and exchanged. The later connections from the same client are granted by the server in the life span of the SSL key.

Our current solution to client affinity is to add persistent client connection scheduling in LinuxDirector. In the PCC scheduling, when a client first access the service, LinuxDirector will create a connection template between the give client and the selected server, then create an entry for the connection in the hash table. The template expires in a configurable time, and the template won't expire if it has its connections. The connections for any port from the client will send to the server before the template expires. Although the PCC scheduling may cause slight load imbalance among servers, it is a good solution to connection affinity.

The configuration example of PCC scheduling is as follows:

```
ipvsadm -A -t <VIP>:0 -s pcc
ipvsadm -a -t <VIP>:0 -R <your server>
```

BTW, PCC should not be considered as a scheduling algorithm in concept. It should be a feature of virtual service port, the port is persistent or not. I will write some codes later to let user to specify whether port is persistent or not.

(and what if a realserver holding a sticky connection crashes?)

Ted Pavlic `tpavlic_list@netwalk.com`

Is this a bug or a feature of the PCC scheduling...

A person connects to the virtual server, gets direct routed to a machine. Before the time set to expire persistent connections, that real machine dies. mon sees that the machine died, and deletes the real server entries until it comes back up.

But now that same person tries to connect to the virtual server again, and PCC *STILL* schedules them for the non-existent real server that is currently down. Is that a feature? I mean – I can see how it would be good for small outages... so that a machine could come back up really quick and keep serving its old requests... YET... For long outages those particular people will have no luck.

> Wensong You can set the timeout of template masq entry into a small number now. It will be expired soon.
>
> Or, I will add some codes to let each real server entry keep a list of its template masq entries, remove those template masq entries if the real server entry is deleted.

To me, this seems most sensible. Lowering the timeouts has other effects, affecting general session persistence...

> I agree with this. This was what I was hoping for when I sent the original message. I figure, if the server the person was connecting to went down, any persistence wouldn't be that useful when the server came back up. There might be temporary files in existence on that server that don't exist on another server, but otherwise... FTP or SSL or anything like that – it might as well be brought up anew on another server. Plus, any protocol that requires a persistent connection is probably one that the user will access frequently during one session. It makes more sense to bring that protocol up on another server than waiting for the old server to come back up – will be more transparent to the user. (Even though they may have to completely re-connect once)
>
> So, yes, deleting the entry when a real server goes down sounds like the best choice. I think you'll find most other load balancers do something similar to this.

Andres Reiner `areiner@nextron.ch`

I found some strange behaviour using 'mon' for the high-availability. If a server goes down it is correctly removed from the routing table. BUT if a client did a request prior to the server's failure, it will still be directed to the failed server afterwards. I guess this got something to do with the persistent connection setting (which is used for the cold fusion applications/session variables). In my understanding the LVS should, if a routing entry is deleted, no longer direct clients to the failed server even if the persistent connection setting is used. Is there some option I missed or is it a bug ?

> Wensong No, you didn't miss anything and it is not a bug either. :) In the current design of LVS, the connection won't be drastically removed but silently drop the packet once the destination of the connection is down, because monitering software may marks the server temporary down when the server is too busy or the monitering software makes some errors. When the server is up, then the connection continues. If server is not up for a while, then the client will timeout. One thing is gauranteed that no new connections will be assigned to a server when it is down. When the client reestablishs the connection (e.g. press reload/refresh in the browser), a new server will be assigned.

9 Fwmarks

9.1 Introduction

The original method for setting up an LVS was to use the VIP as the target for ipvsadm commands. A more flexible method, of using firewall marks (fwmarks) was introduced by Horms in Apr 2000. Ted Pavlic then showed how used fwmarks to group arbitary services. This had previously been possible, to a limited extent, with LVS persistence. The fwmarks method is more flexible and simpler to administer for large numbers of services than is the VIP method.

fwmark is used

- to group services together within a single LVS *e.g.*
  - group 1 - port 80,443 for an e-commerce site
  - group 2 - port 20,21 for an ftp server
- group large numbers of VIPs together

Setting up an LVS on fwmarks rather than the VIP is now the method of choice for anything but a collection of simple one port non-persistent services. Fwmark should be used instead of the VIP when persistence is required or multiport services are involved.

Fwmarks are numbers but can be translated into names using the 29.5 (fwmark name translation table) patch.

Some history (Horms)

> The impeteus origionally came out of a VA Linux Systems Professional Services customer who I was called onsite to help sway towards using LVS. My original proposal and implementation was to allow virtual services based on netmasks. Wensong rejected this because of some potential performance issues.
>
> I distinctly remember working on the original implementation on a train trip from the Blue Mountains to Sydney's Central Station. By the time I had to change trains go to Wynyard the code was working :)
>
> A few days latter Julian came up with the idea of using a fwmark, a feature of the ip_masq code that had been around for a while but wasn't heavily used. Wensong passed this on to me. I wrote the kernel, ipvsadm and ldirectord changes and largely have maintained them ever since. I believe they were included with ipvs-0.9.9.
>
> It is of note that as a part of the work that came out of this customer the -R and -S options to ipvsadm were suggested and implemented by myself. These were released just before the inclusion of the fwmark code.
>
> This customer was also the impetus for putting together what is now known as Ultra Monkey. All in all quite an interesting outcome for a couple of days on site. Pleasingly I believe that the customer in question is using Ultra Monkey with the fwmark support in LVS.

Sample configurations/topologies for fwmarks are at *Ultramonkey* `<http://ultramonkey.org/>`.

9.2 single port service: telnet with fwmarks

Assuming you already have setup the networks and default gw for the machines in your LVS, here's how you'd setup telnet *without* fwmarks (*i.e.* the "normal" method, using the VIP as the target for ipvsadm commands) on a two realserver LVS-DR.

```
#make a table for connections to VIP:telnet, with round robin scheduling
#schedule realserver bashfull for connections to VIP:telnet, \
            weight=1, forwarding method=DR
#schedule realserver sneezy for connections to VIP:telnet, \
            weight=1, forwarding method=DR
director:# ipvsadm -A -t VIP:telnet -s rr
```

```
director:# ipvsadm -a -t VIP:telnet -r bashfull:telnet -g -w 1
director:# ipvsadm -a -t VIP:telnet -r sneezy:telnet -g -w 1
```

Here's how to do the same thing with fwmarks. You first mark the packets with ipchains or iptables.

9.2.1 ipchains for 2.2.x director

Here's the recipe for setting a fwmark with ipchains:

```
#flush ipchains tables
#mark with value=1, tcp packets from anywhere,
#arriving on eth1 (holds the VIP on my setup),
#with dst_addr=192.168.2.110 (the VIP) for port telnet
#show ipchains tables
director:# ipchains -F
director:# ipchains -A input -p tcp -i eth1 -s 0.0.0.0/0 \
       -d 192.168.2.110/32 --dport telnet -m 1
director:# ipchains -L input
Chain input (policy ACCEPT):
target     prot opt     source                destination           ports
-          tcp  ------  anywhere             lvs2.mack.net         any ->   telnet
```

9.2.2 iptables for 2.4.x director

Here's the recipe for setting a fwmark with iptables:

The iptables parameters are taken from an example by *Paul Schulz* <http://www.foursticks.com.au/~pschulz/qos/pfifo.sample>, which I found through *google* <http://www.google.com>.

First put a mark of value=1 on tcp packets which arrive from anywhere with dst_addr=VIP:telnet (the VIP is on eth1 in my setup).

```
#flush the mangle table
#in the skb, put mark=1 on all tcp packets arriving on eth1 from anywhere, with dest=VIP:telnet
#output the mangle table, just for a look
director:# iptables -F -t mangle
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
         -d 192.168.2.110/32 --dport telnet -j MARK --set-mark 1
director:/etc/lvs# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:telnet MARK set 0x1

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

The fwmark is only associated with the packet while it is in the director skb (socket buffer). The packet which emerges from the director and is forwarded to the realserver is a normal (unmarked) packet. (You can't use the director's fwmark information when the packet arrives on the realserver to decide on how to handle the packet.)

9.2.3 install the LVS with ipvsadm

```
#setup an ipvsadm table for packets with mark=1,
#schedule them with round robin.
#schedule realserver sneezy for connections with mark=1, \
          forwarding method=DR, weight=1
#schedule realserver bashfull for connections with mark=1, \
          forwarding method=DR, weight=1
director:# ipvsadm -A -f 1 -s rr
director:# ipvsadm -a -f 1 -r sneezy.mack.net:telnet -g -w 1
director:# ipvsadm -a -f 1 -r bashfull.mack.net:telnet -g -w 1
```

Here's the output of ipvsadm

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.7 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr
  -> bashfull.mack.net:23           Route   1      0          0
  -> sneezy.mack.net:23             Route   1      0          0
```

You can now telnet to the VIP. You'll get the expected round robin scheduling of your connections to bashfull and sneezy.

9.3 Grouping services: single group, active ftp(20,21)

The telnet example above could equally well be done using the VIP or a fwmark as the target for ipvsadm commands. The same is true for any one port service, where connections to services are made independantly of each other. Sometimes we need to group services together, e.g. port 20,21 for an ftp server or port 80, 443 for an e-commerce site. With persistence, you can only make ports persistent singly (but you can make persistent as many or as few as you want, they will be persistent independently); or make all ports persistent at once (with the :0 option), in which case persistence of the ports will be linked. There is no way to make pairs (or groups) of ports persistence with the current persistence code. The current method for handling this, 8 (persistent connections), links all ports on the VIP, and the director will forward connections to all ports, not just the two we are interested in. For security purposes, if persistence is used to group services, then connection requests to the other ports will have to be blocked. Although workable, it's an ugly solution.

For background on how the specifications for fwmarks were set to allow services to be grouped, see 9.22 (Appendix 1) for the initial discussion between Ted and the LVS developers (Horms and Julian), 9.23 (Appendix 2) where Ted let me know that he'd had it working, and 9.24 (Appendix 3) for Ted's announcement to the mailing list.

9.3.1 grouping using VIP and persistence

Here's an example grouping ports 20,21 for ftp. This uses persistence and the VIP as the target for ipvsadm commands (this is the original, VIP way of setting up ftp).

```
#make a table for connections to all ports on VIP
#with round robin scheduling, persistence timeout=360secs
#schedule realserver bashfull for connections to all ports on VIP, \
           weight=1, forwarding method=DR
#schedule realserver sneezy for connections to all ports on VIP, \
           weight=1, forwarding method=DR
director:# ipvsadm -A -t VIP:0 -s rr -p 360
director:# ipvsadm -a -t VIP:0 -r sneezy.mack.net:0 -g -w 1
director:# ipvsadm -a -t VIP:0 -r bashfull.mack.net:0 -g -w 1
```

Here's the output of ipvsadm

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.7 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:0 rr persistent 360
  -> sneezy.mack.net:0              Route   1      0          0
  -> bashfull.mack.net:0            Route   1      0          0
```

After the client has made the initial connection on port 21, then any subsequent connection on port 20 (within the 360sec timeout period) will go to the same realserver.

The problem is that the director will forward to the same realserver, connection requests made to any port by the client. If we have listeners on port 80 and 443 on the realserver, then these services will be linked to each other (which we may want), and they will also be linked to the ftp service (which we may not want). If you telnet to the VIP, this request will be forwarded to the realservers too (in production you'll have to block this).

9.3.2 grouping with fwmarks

Here's how to setup an ftp server with fwmarks. First mark the packets of interest with ipchains or iptables (*i.e* mark all tcp packets destined for VIP:ftp and VIP:ftp-data arriving on eth1).

ipchains for 2.2 director

```
#flush ipchains tables
#mark ftp packets
#put the same mark on ftp-data packets
#show ipchains tables
director:# ipchains -F
director:# ipchains -A input -p tcp -i eth1 -s 0.0.0.0/0 \
                         -d 192.168.2.110/32 --dport ftp -m 1
director:# ipchains -A input -p tcp -i eth1 -s 0.0.0.0/0 \
                         -d 192.168.2.110/32 --dport ftp-data -m 1
director:# ipchains -L input
Chain input (policy ACCEPT):
target     prot opt     source                destination           ports
-          tcp  ------  anywhere             lvs2.mack.net         any ->   ftp
-          tcp  ------  anywhere             lvs2.mack.net         any ->   ftp-data
```

iptables for 2.4 director

```
#clear mangle table
#mark ftp packets
#put the same mark on ftp-data packets
#show mangle table
director:# iptables -F -t mangle
director:/etc/lvs# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
                                -d 192.168.2.110/32 --dport ftp -j MARK --set-mark 1
director:/etc/lvs# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
                                -d 192.168.2.110/32 --dport ftp-data -j MARK --set-mark 1
director:/etc/lvs# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp MARK set 0x1
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp-data MARK set 0x1

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

install LVS with ipvsadm Next setup ipvsadm to schedule packets marked with fwmark=1 to your realservers. You need persistence (here timeout set to 600secs).

```
director:# ipvsadm -A -f 1 -s rr -p 600
director:# ipvsadm -a -f 1 -r sneezy.mack.net:0 -g -w 1
director:# ipvsadm -a -f 1 -r bashfull.mack.net:0 -g -w 1
```

Here's the output of ipvsadm with two current connections to the LVS and 3 expiring ones. Note they are all to the same realserver, as expected for a persistent connection. Since forwarding is by LVS-NAT, the ip_vs_ftp module automatically loads.

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.7 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0            Route   1      2          3
  -> sneezy.mack.net:0              Route   1      0          0
```

A netpipe test showed the same latency and throughput for a connection based on fwmark or based on VIP.

What happens now when you telnet from the client to the VIP? (pause to let you think.) The director is only forwarding packets with fwmark=1 to the LVS, so a telnet request to the VIP is accepted by the director and not forwarded to the realservers. If telnetd is running on the director, you'll get a login prompt from the director. In production you'll have to block this too (just like you had to when setting up on a VIP).

So what's the difference, you ask, between setting up an ftp server with persistence on the VIP on one hand (which requires you to block all other packets with iptables rules), and grouping 20,21 with fwmarks on the other (which requires exactly the same blocking of unwanted packets)? Not a lot. At the moment you're at least even

> Lars Marowsky-Brée `lmb@suse.de` 2000-05-11 When using the LVS box as a firewall/router, the fwmark technique is a perfectly adequate solution, which doesn't cost anything.

But look at the next example.

9.4 Grouping services: two groups, active ftp(20,21) and e-commerce(80,443)

Setup 2 groups of services, group 1 - ftp(20,21), group 2 - ecommerce(80,443).

First mark packets in 2 groups.

9.4.1 ipchains for 2.2 director

```
director:# ipchains -F
director:# ipchains -A input -p tcp -i eth1 -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport ftp -m 1
director:# ipchains -A input -p tcp -i eth1 -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport ftp-data -m 1
director:# ipchains -A input -p tcp -i eth1 -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport http -m 2
director:# ipchains -A input -p tcp -i eth1 -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport https -m 2
director:# ipchains -L input
Chain input (policy ACCEPT):
target     prot opt     source               destination          ports
-          tcp  ------  anywhere             lvs2.mack.net        any ->   ftp
-          tcp  ------  anywhere             lvs2.mack.net        any ->   ftp-data
-          tcp  ------  anywhere             lvs2.mack.net        any ->   www
-          tcp  ------  anywhere             lvs2.mack.net        any ->   https
```

9.4.2 iptables for 2.4 director

```
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
            -d 192.168.2.110/32 --dport ftp -j MARK --set-mark 1
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
            -d 192.168.2.110/32 --dport ftp-data -j MARK --set-mark 1
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
            -d 192.168.2.110/32 --dport http -j MARK --set-mark 2
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
            -d 192.168.2.110/32 --dport https -j MARK --set-mark 2
director:/etc/lvs# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp MARK set 0x1
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp-data MARK set 0x1
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:www MARK set 0x2
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:https MARK set 0x2

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

9.4.3 setup LVS to schedule (with persistence) 2 groups of packets

Note: The ipvs code in Apr 2001 needed a patch to get the expected behaviour. This section describes the function of LVS before and after this patch. As a result of these tests, the patch will be applied to future releases. ipvs-1.0.7-2.2.19 is already patched (Apr 2001). The 2.4.3 series are not patched yet. To see if the code has been patched look in ipvs/Changelog for something like this

> Julian changed persistent connection template for fwmark-based service from <CIP,VIP,RIP> to <CIP,FWMARK,RIP>, so that different fwmark-based services that share the same VIP can work correctly.

If your ipvs code is pre-patched, then you can skip down to the part where the behaviour after applying the patch is described. If your code isn't patched, you should just go get the patch and skip to the part where 9.4.5 (the expected behaviour is described).

9.4.4 unexpected behaviour

Here's what happened with the original code.

```
director:# ipvsadm -A -f 1 -s rr -p 600
director:# ipvsadm -a -f 1 -r sneezy.mack.net:0 -g -w 1
director:# ipvsadm -a -f 1 -r bashfull.mack.net:0 -g -w 1
director:# ipvsadm -A -f 2 -s rr -p 600
director:# ipvsadm -a -f 2 -r sneezy.mack.net:0 -g -w 1
director:# ipvsadm -a -f 2 -r bashfull.mack.net:0 -g -w 1

IP Virtual Server version 0.2.7 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0             Route   1      0          0
  -> sneezy.mack.net:0               Route   1      0          0
FWM  2 rr persistent 600
  -> bashfull.mack.net:0             Route   1      0          0
  -> sneezy.mack.net:0               Route   1      0          0
```

If you ftp and http to the VIP, you'd expect the ftp connections to go to fwmark 1 (presumably to the first realserver bashfull) and the http connections to go to fwmark 2 (again presumably to bashfull).

With the director running 1.0.6-2.2.19 (ipvs/kernel version), all connections (ftp, http) go to group 1. With the director 0.2.7-2.4.2, all connections go to group 2. Here's the output from ipvsadm for the 2.2.19 example immediately after downloading a webpage. You would expect the http InActConn to be associated with FWM2.

```
IP Virtual Server version 1.0.6 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port            Forward Weight ActiveConn InActConn
FWM  1 rr persistent 30
  -> bashfull.mack.net:0           Route   1      0          2
  -> sneezy.mack.net:0             Route   1      0          0
FWM  2 rr persistent 30
  -> bashfull.mack.net:0           Route   1      0          0
  -> sneezy.mack.net:0             Route   1      0          0
director:/etc/lvs#
```

It appears (Apr 2001) that the ipvs code doesn't really follow the 9.22 (persistent fwmarks spec). When there is a collision between VIP space and fwmark space (eg in these examples, where all packets are going to the same VIP), then the VIP takes precedence and the two fwmark groups are not differentiated. The collision arises because there is only one set of templates for the connection tables.

9.4.5 expected behaviour

Note: May 2001: the ipvs code now has the persistent-fwmark behaviour.

> (The code to produce the expected behaviour requires a separate set of templates for fwmarks and VIP. The patch to do this is on *Julian's patch page* `<http://www.linuxvirtualserver.org/~julian>` and has names like persistent-fwmark-0.2.8-2.4-1.diff, persistent-fwmark-1.0.5-2.2.18-1.diff. (Note: the 0.2.8 patch had DOS carriage control and wouldn't patch till I removed the ^M characters). (Note: as of ipvs-0.9.0, this patch has been applied to the source tree.)
>
> After patching the ip_vs code to produce the new ip_vs.o module (rmmod the old one first), you get the expected fwmark behaviour.)

Here's the output of ipvsadm after ftp'ing and http'ing from a client. Note that the ftp connection is to fwmark=1. The InActConn is the expiring connection from the http client to fwmark=2.

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 30
  -> bashfull.mack.net:0            Route   1      1          0
  -> sneezy.mack.net:0              Route   1      0          0
FWM  2 rr persistent 30
  -> bashfull.mack.net:0            Route   1      0          1
  -> sneezy.mack.net:0              Route   1      0          0
```

9.4.6 The original idea from Ted Pavlic

Ted Pavlic `tpavlic@netwalk.com` 2000-10-08

Just another persistence option that you may or may not have thought of... LVS does support port-group sticky persistance. Before FWMARK support was added to LVS, the only types of persistance one could do were:

- One port persistence (all queries to 80 return to the same real server per CIP)
- ALL port persistence (all queries to all ports return to the same RIP per CIP)

But now that FWMARK support exists in LVS, it is easy to create group-based sticky persistence. That is... It adds the option where:

- Only these two ports (443 and 80) return to the same RIP per CIP
- Meanwhile, another persistence table keeps track of 20, 21, and 1024:65535
- Any other port is not persistent

Just have ipchains keep track of flagging the incoming packets with the correct port group identifier:

```
ipchains -A input -D VIPNET/VIPMASK PORT -p PROTOCOL -m FWMARK
```

And have IPVS stop looking at IPs and start look at FWMARKs:

```
ipvsadm -A -f FWMARK
ipvsadm -a -f FWMARK -r RIP:0
```

9.4.7 ssl and cookies

Ted Pavlic `tpavlic@netwalk.com` 2000-10-13

LVS DIRECTLY supports two types of persistence and INDIRECTLY supports another. If you are just asking how to make port 443 persistent so that those who receive a cookie on 443 will come back to the same real server on 443, simply:

```
/sbin/ipvsadm -A -t 192.168.1.110:443 -p
/sbin/ipvsadm -a -t 192.168.1.110:443 -R 192.168.2.1
/sbin/ipvsadm -a -t 192.168.1.110:443 -R 192.168.2.2
/sbin/ipvsadm -a -t 192.168.1.110:443 -R 192.168.2.3
...
```

Will setup persistence just for port 443.

However, say someone gets a cookie on port 80 and gives it back on port 443 – in that case you want to have persistence between multiple ports. Using port 0 accomplishes this:

```
/sbin/ipvsadm -A -t 192.168.1.110:0 -p
/sbin/ipvsadm -a -t 192.168.1.110:0 -R 192.168.2.1
/sbin/ipvsadm -a -t 192.168.1.110:0 -R 192.168.2.2
/sbin/ipvsadm -a -t 192.168.1.110:0 -R 192.168.2.3
...
```

In this setup, anyone who visits ANY service will continue to go back to the same real server. So requests which come in on 80 or 443 will continue to come in to the same real server regardless of port.

This is an OK solution, but it basically makes all services persistent which might mess up scheduling. That is, this is a decent solution but sometimes not extremely desirable.

If you want to simply group ports 80 and 443 together, you need to do something more intuitive. Use FWMARK...

```
ipchains -A input -d 192.168.1.110/32 80 -p tcp -m 1
ipchains -A input -d 192.168.1.110/32 443 -p tcp -m 1
/sbin/ipvsadm -A -f 1 -p
/sbin/ipvsadm -a -f 1 -R 192.168.2.1
/sbin/ipvsadm -a -f 1 -R 192.168.2.2
/sbin/ipvsadm -a -f 1 -R 192.168.2.3
...
```

Now only port 80 and 443 will be grouped together via persistence. Any other ipvsadm rules will be completely separate. This means that you can make 80 and 443 persistence by their own little "port group" and leave ports 25 and 110 (for example) not persistent. OR... You could group all the FTP ports together as well on a completely different persistence group... i.e.

```
ipchains -A input -d 192.168.1.110/32 80 -p tcp -m 1
ipchains -A input -d 192.168.1.110/32 443 -p tcp -m 1
/sbin/ipvsadm -A -f 1 -p
/sbin/ipvsadm -a -f 1 -R 192.168.2.1
/sbin/ipvsadm -a -f 1 -R 192.168.2.2
/sbin/ipvsadm -a -f 1 -R 192.168.2.3
# Really adding port 20 isn't needed
ipchains -A input -d 192.168.1.110/32 20 -p tcp -m 2
ipchains -A input -d 192.168.1.110/32 21 -p tcp -m 2
ipchains -A input -d 192.168.1.110/32 1024:65535 -p tcp -m 2
/sbin/ipvsadm -A -f 2 -p
/sbin/ipvsadm -a -f 2 -R 192.168.2.1
/sbin/ipvsadm -a -f 2 -R 192.168.2.2
/sbin/ipvsadm -a -f 2 -R 192.168.2.3
...
```

and again

> Wayne wrote Is there a easy way to relating server in both port 80 and port 443 (with LVS-NAT)?
>
> Say I have two farms, each with same three servers. One farm load balancing HTTP requests and another farm load balancing HTTPS farms. To make sure the user in the persistent mode connected to the HTTP server always go to the same server for HTTPS service, we would like to have some way to relate the services between the two farms, is there a easy way to do it?

ratz `ratz@tac.ch` 2001-01-03

Two possibilities to solve this with LVS

1. Use port 0 in your setup. (advantage: easy to set up and easy understand)
2. Use fwmark and group them together. (advantage: finer port granularity possible)

Example (1):

```
ipvsadm -A -t 192.168.1.100:0 -s wlc -p 333 -M 255.255.255.255
ipvsadm -a -t 192.168.1.100:0 -r 192.168.1.1 -g -w 1
ipvsadm -a -t 192.168.1.100:0 -r 192.168.1.2 -g -w 1
```

Example (2):

```
ipchains -A input -j ACCEPT -p tcp -d 192.168.1.100/32 80 -m 1 -l
ipchains -A input -j ACCEPT -p tcp -d 192.168.1.100/32 443 -m 1 -l
ipvsadm -A -f 1 -s wlc -p 333 -M 255.255.255.255
ipvsadm -a -f 1 -r 192.168.1.1 -g -w 1
ipvsadm -a -f 1 -r 192.168.1.2 -g -w 1
```

9.5 passive ftp

You can setup 11.5 (passive ftp) with the VIP as the target using persistence. This is not a particular satisfactory solution, as connect requests to all ports will be forwarded. As well, if another service on the realserver fails (eg http), then all services have to be failed out together.

Here's a solution to passive ftp from Ted Pavlic using fwmark. This allows setting up passive ftp independantly of other services. Passive ftp listens on an unknown and unpredictable high port on realserver. This is handled by forwarding requests to all high ports (it's still ugly, but at least this way, we can fail out ftp independently of other services).

9.5.1 test session with active ftp

Here's ftp setup in active mode, as a control.

```
director:# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source              destination
MARK       tcp  --  anywhere            lvs2.mack.net      tcp dpt:ftp MARK set 0x1
MARK       tcp  --  anywhere            lvs2.mack.net      tcp dpt:ftp-data MARK set 0x1
#
#setup ipvsadm, making all packets with mark=1 persistent
director:# ipvsadm -A -f 1 -s rr -p 600
director:# ipvsadm -a -f 1 -r sneezy:0 -g -w 1
director:# ipvsadm -a -f 1 -r bashfull:0 -g -w 1
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0            Route   1      0          0
  -> sneezy.mack.net:0              Route   1      0          0
```

Here's netstat -an on the client and the realserver (bashfull) immediately after an ftp file transfer (with the client still connected).

```
#client:
client:~# netstat -an | grep 110 #110 is part of the VIP
tcp        0      0 client:1176   VIP:21          ESTABLISHED
#realserver
bashfull:/home/ftp/pub# netstat -an | grep 254 #254 is part of the client IP
tcp        0      0 VIP:20        client:1180     TIME_WAIT
tcp        0      0 VIP:20        client:1178     TIME_WAIT
tcp        0      0 VIP:20        client:1177     TIME_WAIT
tcp        0      0 VIP:21        client:1176     ESTABLISHED
```

Only port 20,21 are involved here.

Here's the command line at the client during the active ftp transfer (all expected output).

```
ftp> get tulip.c
```

```
local: tulip.c remote: tulip.c
200 PORT command successful.
150 Opening BINARY mode data connection for tulip.c (104241 bytes).
226 Transfer complete.
104241 bytes received in 0.0232 secs (4.4e+03 Kbytes/sec)
```

The iptables rules on the director do not allow passive ftp connection. To test this put the ftp client into passive mode.

```
ftp> pass
Passive mode on.
ftp> dir
227 Entering Passive Mode (192,168,2,110,4,72)
ftp: connect: Connection refused
ftp>
```

connection is not allowed. To check that the system is still functioning, put the client back into active mode.

```
ftp> pass
Passive mode off.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 155178
.
.
-rw-r--r--   1 root     root       104241 Nov 10  1999 tulip.c
226 Transfer complete.
ftp>
```

9.5.2 test session with passive ftp

Here's the setup for passive ftp (2.4.x director) (you can leave ipvsadm untouched).

```
director:# iptables -F -t mangle
#mark ftp packets
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport ftp -j MARK --set-mark 1
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport 1024: -j MARK --set-mark 1
director:# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp MARK set 0x1
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpts:1024:65535 MARK set 0x1
```

Here's the command line from the ftp client still in active mode

```
ftp>  dir
200 PORT command successful.
```

The session is hung, the server shows an established connection to port 21 and the client session has to be killed.

Here's the passive session.

```
client:~# ftp VIP
Connected to VIP.
220 bashfull.mack.net FTP server (Version wu-2.4.2-academ[BETA-15](1) Wed May 20
 13:45:04 CDT 1998) ready.
Name (VIP:root): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pass
Passive mode on.
ftp> cd pub
250 CWD command successful.
ftp> dir *.c
227 Entering Passive Mode (192,168,2,110,4,75)
150 Opening ASCII mode data connection for /bin/ls.
-rw-r--r--   1 root     root       104241 Nov 10  1999 tulip.c
226 Transfer complete.
ftp> mget *.c
mget tulip.c? y
227 Entering Passive Mode (192,168,2,110,4,78)
150 Opening BINARY mode data connection for tulip.c (104241 bytes).
226 Transfer complete.
104241 bytes received in 0.0233 secs (4.4e+03 Kbytes/sec)
ftp>
```

Here's the connections at the realserver immediately after the file transfer. There is the regular connection at the ftp port (21) and a connection timing out to a high port on the realserver.

```
bashfull:/home/ftp/pub# netstat -an | grep 254 #254 is part of the client IP
tcp        0      0 VIP:1104      client:1191      TIME_WAIT
tcp        0      0 VIP:21        client:1184      ESTABLISHED
```

Here's the output from ipvsadm after connecting to the URL ftp://vip/ using a web-browser

```
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0             Route   1      1          5
  -> sneezy.mack.net:0               Route   1      0          0
```

9.5.3 LVS with 2 groups: group 1 = ftp(active and passive), group 2 = http

```
#fwmark rules
```

```
director:# iptables -F -t mangle
#active and passive ftp in group 1
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
               -d 192.168.2.110/32 --dport ftp -j MARK --set-mark 1
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
               -d 192.168.2.110/32 --dport ftp-data -j MARK --set-mark 1
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
               -d 192.168.2.110/32 --dport 1024: -j MARK --set-mark 1
#http as group 2
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
               -d 192.168.2.110/32 --dport http -j MARK --set-mark 2
director:/etc/lvs# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp MARK set 0x1
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp-data MARK set 0x1
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpts:1024:65535 MARK set 0x1
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:www MARK set 0x2
#
#setup LVS for 2 groups
director:# ipvsadm -C
#ftp (active and passive) are persistent as group 1
director:# ipvsadm -A -f 1 -s rr -p 600
director:# ipvsadm -a -f 1 -r sneezy:0 -g -w 1
director:# ipvsadm -a -f 1 -r bashfull:0 -g -w 1
#http as group 2 (not persistent)
director:# ipvsadm -A -f 2 -s rr
director:# ipvsadm -a -f 2 -r sneezy:http -g -w 1
director:# ipvsadm -a -f 2 -r bashfull:http -g -w 1
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0            Route   1      0          0
  -> sneezy.mack.net:0              Route   1      0          0
FWM  2 rr
  -> sneezy.mack.net:80             Route   1      0          0
  -> bashfull.mack.net:80           Route   1      0          0
```

The client connected (in order) ftp://VIP/, http://VIP/ (passive ftp) and then by active (command line) ftp to VIP. Here's the ipvsadm output.

```
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0            Route   1      2          3
  -> sneezy.mack.net:0              Route   1      0          0
FWM  2 rr
```

```
  -> sneezy.mack.net:80             Route   1      2          2
  -> bashfull.mack.net:80           Route   1      4          0
```

Here's the connections showing on the realserver. The most recent ones are at the top of the list. The connection list shows (from the bottom, *i.e.* in the order of connection), passive ftp, http, and active ftp.

```
bashfull:/home/ftp/pub# netstat -an | grep 254 #254 is part of the CIP
tcp        0      0 VIP:21        client:1207      ESTABLISHED
tcp        0      0 VIP:80        client:1206      FIN_WAIT2
tcp        0      0 VIP:80        client:1204      FIN_WAIT2
tcp        0      0 VIP:1108      client:1202      TIME_WAIT
tcp        0      0 VIP:21        client:1201      ESTABLISHED
```

The whole point of this setup is to make ftp and http, which belonged to one persistence group when setup on a VIP, into two groups. Now you can bring the httpd and the ftpd up and down independantly (if you want to fail them out, to change the configuration or software).

9.6 fwmark with LVS-NAT

(based on a posting by Horms on 14 Jul 2000)

Here we setup a LVS-NAT LVS on a 2.4.x director. (Note: With 2.4 LVS, the masquerading is setup by the ipvs code, *i.e.* you don't have to masquerade the packets back from the realservers). These examples assume that the VIP is on eth1 and your network is already setup (ie the realservers are using the director as the default gw etc).

Mark packets for the VIP and setup the LVS for telnet. (Warning: this first example is not going to get you anything you want.)

```
#
#mark packets
director:# iptables -F -t mangle
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
                -d 192.168.2.110/32 -j MARK --set-mark 1
director:# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs2.mack.net      MARK set 0x1
#
#Setup ipvsadm
director:# ipvsadm -C
director:# ipvsadm -A -f 1 -s rr
director:# ipvsadm -a -f 1 -r sneezy.mack.net:telnet -m -w 1
director:# ipvsadm -a -f 1 -r bashfull.mack.net:telnet -m -w 1
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr
  -> bashfull.mack.net:23           Masq    1      0          0
  -> sneezy.mack.net:23             Masq    1      0          0
```

You can connect with telnet to the VIP and you'll be forwarded to both realservers in the expected way.

All packets from the client will be marked and processed by the ipvsadm rules. What happens if you attempt to connect to VIP:80 (pause to think)?

Here's the answer.

```
client:~# telnet VIP 80
Trying 192.168.2.110...
Connected to lvs2.mack.net.
Escape character is '^]'.

Welcome to Linux 2.2.19.



bashfull login: root
Linux 2.2.19.
Last login: Fri Apr 13 11:43:52 on ttyp1 from client2.mack.net.
No mail.
```

If you connect to VIP:80 using a browser for a client, it sits there showing the watch symbol for quite a while.

What happened? The explanation is that you told the director to mark all packets (*i.e.* from any port) from the client, rewrite them to have dest_addr=RIP:telnet and forward the rewritten packets to the realserver. So when you telnet'ed to VIP:80, the packets were forwarded to RIP:23.

Just to make sure that I'd interpretted this correctly, here's the first packets seen by tcpdump running on the client and the realserver during the connect attempts. (These are from different sessions, so the ports shown on the client are different.)

client: here the client is connecting to VIP:80 (lvs2.www)

```
12:09:44.449566 client2.1118 > lvs2.www: S 2887976275:2887976275(0) win 5840 <mss 1460,sackOK,timest
12:09:44.450453 lvs2.www > client2.1118: S 1441372470:1441372470(0) ack 2887976276 win 32120 <mss 14
12:09:44.450579 client2.1118 > lvs2.www: . ack 1 win 5840 <nop,nop,timestamp 118456418 117741798> (D
```

realserver (bashfull): here the realserver is receiving packets to the RIP:23 (bashfull.telnet)

```
11:44:28.319675 client2.1116 > bashfull.telnet: S 2722509719:2722509719(0) win 5840 <mss 1460,sackOK
11:44:28.319974 bashfull.telnet > client2.1116: S 1283414485:1283414485(0) ack 2722509720 win 32120
11:44:28.320681 client2.1116 > bashfull.telnet: . ack 1 win 5840 <nop,nop,timestamp 118440378 117725
```

If you want only telnet requests to be forwarded to the realservers, you should mark only packets for VIP:telnet. If you want both telnet and http forwarded then you should give them each their own mark. Here's how to setup LVS-NAT with fwmark for both telnet and http.

```
director:# iptables -F -t mangle
#telnet packets to the VIP get fwmark=1
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport telnet -j MARK --set-mark 1
#http packets to the VIP get fwmark=2
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
                        -d 192.168.2.110/32 --dport http -j MARK --set-mark 2
```

```
director:# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source              destination
MARK       tcp  --  anywhere            lvs2.mack.net      tcp dpt:telnet MARK set 0x1
MARK       tcp  --  anywhere            lvs2.mack.net      tcp dpt:www MARK set 0x2
#
#setup ipvsadm
director:# ipvsadm -C
#forward packets with mark=1 to the telnet port
director:# ipvsadm -A -f 1 -s rr
director:# ipvsadm -a -f 1 -r sneezy.mack.net:telnet -m -w 1
director:# ipvsadm -a -f 1 -r bashfull.mack.net:telnet -m -w 1
#forward packets with mark=2 to the httpd port
director:# ipvsadm -A -f 2 -s rr
director:# ipvsadm -a -f 2 -r sneezy.mack.net:http -m -w 1
director:# ipvsadm -a -f 2 -r bashfull.mack.net:http -m -w 1
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
FWM  1 rr
  -> bashfull.mack.net:23            Masq    1      0          0
  -> sneezy.mack.net:23              Masq    1      0          0
FWM  2 rr
  -> bashfull.mack.net:80            Masq    1      0          0
  -> sneezy.mack.net:80              Masq    1      0          0
```

Here's the (expected) output of ipvsadm showing the client with 2 telnet sessions and having just downloaded a webpage from the LVS.

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
FWM  1 rr
  -> bashfull.mack.net:23            Masq    1      1          0
  -> sneezy.mack.net:23              Masq    1      1          0
FWM  2 rr
  -> bashfull.mack.net:80            Masq    1      0          1
  -> sneezy.mack.net:80              Masq    1      0          0
```

9.7 Collisions between fwmark and VIP rules

Since it's possible to write iptables rules that include many different types of packets, it's possible to write VIP and fwmark rules that would conflict by accepting the same packet. Here's a setup that would accept telnet by both VIP and fwmarks.

```
director:# iptables -t mangle -A PREROUTING -i eth1 -p tcp -s 0.0.0.0/0 \
               -d 192.168.2.110/32 --dport ftp -j MARK --set-mark 1
director:# ipvsadm -A -t lvs2.mack.net:telnet -s rr
```

```
director:# ipvsadm -a -t lvs2.mack.net:telnet -r sneezy.mack.net:telnet -g -w 1
director:# ipvsadm -a -t lvs2.mack.net:telnet -r bashfull.mack.net:telnet -g -w 1
director:# ipvsadm -A -f 1 -s rr
director:# ipvsadm -a -f 1 -r sneezy.mack.net:telnet -g -w 1
director:# ipvsadm -a -f 1 -r bashfull.mack.net:telnet -g -w 1
#
director:# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs2.mack.net      tcp dpt:ftp MARK set 0x1
#
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:telnet rr
  -> bashfull.mack.net:telnet        Route   1      0          0
  -> sneezy.mack.net:telnet          Route   1      0          0
FWM  1 rr
  -> bashfull.mack.net:telnet        Route   1      0          0
  -> sneezy.mack.net:telnet          Route   1      0          0
```

Here's the ipvsadm output after 4 telnet connections from a client

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:telnet rr
  -> bashfull.mack.net:telnet        Route   1      2          0
  -> sneezy.mack.net:telnet          Route   1      2          0
FWM  1 rr
  -> bashfull.mack.net:telnet        Route   1      0          0
  -> sneezy.mack.net:telnet          Route   1      0          0
```

All connections go to the first (here VIP) entries. The same ipvsadm table and connection pattern results if you feed the VIP and fwmarks rules into ipvsadm in the reverse order. This behaviour is not part of the spec (yet). You might want to check the behaviour, if you are doing this sort of setup.

9.8 persistence granularity with fwmark

9.8.1 introduction

Persistence granularity was added to LVS by Lars `lmb@suse.de` 1999-10-13

> This patch adds netmasks to persistent ports, so you can adjust the granularity of the templates. It should help solve the problems created with non-persistent cache clusters on the client side."

The problem being addressed is that some clients (eg AOL customers) connect to the internet via large proxy farms. The IP they present to the server will not neccessarily be the same for different sessions (tcp connections), even though they remain connected to their proxy machine. Persistence granularity makes all clients from a network equivalent as far as persistence is concerned. Thus a client could appear as CIP=x.x.x.13 for their http connections, but CIP=x.x.x.14 for their https connections. With persistence granularity set to /24, all CIPs from the same class C network will be sent to the same realserver. The default behaviour (*i.e.* persistence granularity is /32) has the effect that all connections from the same CIP to be sent to the one realserver but other connections from the same network will be scheduled to other realservers.

Persistence granularity is applied to the CIP and works the same whether you are using fwmark or the VIP to setup the LVS.

You set the netmask (granularity) for 8.4 (persistence granularity) with ipvsadm. If the LVS was setup with the following command, the persistence granularity is 255.255.255.0.

```
ipvsadm -A -t 192.168.1.100:0 -s wlc -p 333 -M 255.255.255.0
```

Let's say a client from a class C network (*e.g.* with IP=100.100.100.2) connects to the LVS. If any other client connects from 100.100.100.0/24 they will also connect to the same realserver as long as the original client's entry in the persistence table has not expired (*i.e.* the first client is still connected, or disconnected < 333 secs ago).

9.8.2 examples

Here's an example LVS-DR LVS set to mark packets for an IP on the outside of the director (this IP serves as the VIP in the usual LVS setup, but there's no such thing as a VIP with fwmarks) with –dport telnet. Persistence granularity is set to the default (-M 255.255.255.255).

```
director:# ipvsadm -C
director:# ipvsadm -A -f 1 -s rr -p 600
director:# ipvsadm -a -f 1 -r sneezy:0 -g -w 1
director:# ipvsadm -a -f 1 -r bashfull:0 -g -w 1
```

Two clients (192.168.2.254, 192.168.2.253) connect to the LVS. Each host connects to different realservers but multiple connects from each client go to the same realserver (*i.e.* client A always goes to realserver A; client B always goes to realserver B, at least till the persistence timeout clears). Here both clients have connected twice.

```
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0            Route   1      2          0
  -> sneezy.mack.net:0              Route   1      2          0
```

This is the connection pattern expected if the connections were based on the CIP/32 and fwmark (ie all clients are scheduled independently).

Here's the same setup with persistence granularity set to /24.

```
director:# ipvsadm -C
director:# ipvsadm -A -f 1 -s rr -p 600 -M 255.255.255.0
```

```
director:# ipvsadm -a -f 1 -r sneezy:0 -g -w 1
director:# ipvsadm -a -f 1 -r bashfull:0 -g -w 1
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600 mask 255.255.255.0
  -> bashfull.mack.net:0             Route   1      0          0
  -> sneezy.mack.net:0               Route   1      0          0
```

Here's what happens when the 2 clients, both of who belong to the same CIP/24 persistence group, connect twice - all connections go to the same realserver.

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port              Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600 mask 255.255.255.0
  -> bashfull.mack.net:0             Route   1      4          0
  -> sneezy.mack.net:0               Route   1      0          0
```

9.8.3 Discussion with Julian about persistence granularity

Joe

> I expect if you were using persistence with fwmark, then any connection requests arriving with the same fwmark will be treated as belonging to that persistence group. Presumably any combination of client IPs and/or networks could have been used to make the rules which marks the packets.

Julian

Yes, it is for the same group but in one fwmark group there are many templates created. These templates are different for the client groups. The template looks like this:

CIPNET:0 -> SERVICE(FWMARK/VIP):0 -> RIP:0

All ports 0 for the fwmark-based services

So, for client 10.1.2.3/24 (24=persistent granularity) the template looks like this:

10.1.2.0:0 -> VIP:0 -> RIP:0

LVS patched with the persistent_fwmark patch:

10.1.2.0:0 -> FWMARK:0 -> RIP:0

So, the templates are created with CIP/GRAN in mind and the lookup uses CIPNET too. We use CIPNET = CIP & CNETMASK before creation and lookup.

> so if I did
>
> ```
> iptables -s 10.1.2.3 -m 1
> ipvsadm -A -f 1 -s rr -p 600 -M 255.255.255.0
> ```
>
> only packets from 10.1.2.3 will have a fwmark on them, but the director would forward all packets from 10.1.2.0/24, even those without fwmarks?

The patched LVS will accept only the marked packets for this fwmark service, from the same /24 client subnet. If only one client IP sends packets that are marked then the real service will receive packets only from 10.1.2.3. The current LVS versions don't consider the service and all packets CIPNET -> VIP will be forwarded using the first created template for CIPNET:0->VIP:0, i.e. these packets will randomly hit one of the many services that accept packets for the same VIP (just like in your setup) and then may be a wrong real server.

> The current LVS versions don't consider the service and all packets CIPNET -> VIP
>
> but there is no VIP here, I'm using fwmark only. what does the -M 255.255.255.0 do in this case?

The current LVS versions (*i.e.* without the persistent_fwmark patch) assume the VIP is the iphdr->daddr, i.e. the destination address in the datagram and this addresses is used to lookup/create the template.

> how about your persistent-patch, which I've been working with?

The patch ignores this daddr when creating or looking for templates. Instead, the service fwmark values is used when the service is fwmark-based: CIPNET:0 -> FWMARK:0 -> RIP:0

The normal services use daddr as VIP when looking for or creating templates: CIPNET:0 -> daddr:0 -> RIP:0

The persistence is associated with the client address (CIP). The sequence is this:

- packet comes from CIP to VIP1

- fw marking, optional

- lookup for existing connection CIP:CPORT->VIP1:VPORT, if yes => forward, if not found:

- lookup service => fwmark 1, persistent

- try to select real service in context of the virtual service

Apply the persistence granularity to the client address CIPNET = CIP & svc->netmask

Now lookup for template

1. not patched: check for existing template CIPNET:0, VIP1:0
2. patched: check for existing template CIPNET:0, 1(fwmark):0

if there is template, bind the new connection to the template's destination

if there is no existing template, get one destination using the scheduler and bind it to the newly created template and the new connection. The created template is

1. CIPNET:0, VIP1:0, DEST_RIP:0
2. CIPNET:0, 1(fwmark):0, DEST_RIP:0

- forward the packet

> Persistence granularity was designed for people coming in from large proxy servers (eg AOL). With fwmarks, this can be handled by iptables rules.

Yes, the fact that we group the clients using this netmask is not related to the virtual service type: normal or fwmark-based.

Yes, each different IP is treated as different client. When a netmask < 32 is used, the group of addresses is treated as one client when applying the persistence rules. This is not related to the packet marking and virtual service type.

9.9 fwmark allows LVS-DR director to be default gw for realservers

If a LVS-DR director is accepting packets by fwmarks, then it does not have a VIP. The director can then be the default gw for the realservers (see 13.4 (VS-DR director is default gw for realservers)).

9.10 Routing to director and realservers in an LVS setup with fwmark

If you are using fwmark as the target for the LVS, the destination addresses could be any arbitary grouping of IPs. Since there is no interface on the director with any of these IPs, then some method is needed so that packets destined for the LVS get to the director. This same problem occurs with 16.5 (transparent proxy). The solution is to configure the router to send all those packets to the director.

Once the packets have been forwarded from the director, in the case of LVS-DR some method is needed for the realserver to accept the packets. With LVS-DR there is no routing problem; the director sends the packets to the interface with the MAC address of the RIP. However the realserver must recognise that the packets are to be processed locally. Several methods are available.

- The realserver can accept the packets by transparent proxy
- an alias can be set to accept a network of addresses, without assigning an IP to the device

  Horms `horms@vergenet.net` 10 Apr 2000

  ```
  ifconfig lo:0 192.168.1.0 netmask 255.255.255.0
  ```

  You can now ping anything in the 192.168.1.0/24 network from the console.

  Note: You can't ping any of those IP's from a remote host (after adding a route on the remote host to this network). If you put this network onto an eth0 alias (rather than an lo alias), it doesn't reply to pings from the console - presumably the ping replies in the lo:0 case are coming from 127.0.0.1.

  For another example of routing to an interface without an IP, see 13.9 (routing to realservers from director in LVS-DR).

9.11 fwmark simplifies configuration for large numbers of addresses

If a fwmark rule accepts packets for a /24 network, then 254 IPs are configured in one instruction. The next sections are examples.

9.12 Example: firewall farm

Horms `horms@vergenet.net` 2000-12-06

Assume that packets from out local network (192.168.0.0/23) are outgoing traffic.

Mark all outgoing packets with fwmark 1

```
ipchains -A input  -s 192.168.0.0/23 -m 1
# Now, set up a virtual service to act on the marked packets
ipvsadm -A -f 1
ipvsadm -a -f 1 -r 192.168.1.7
ipvsadm -a -f 1 -r 192.168.1.8
ipvsadm -a -f 1 -r 192.168.1.9
```

Where 192.168.1.7, 192.168.1.8 and 192.168.1.9 are your firewall boxen.

9.13 Example: LVS'ing a CIDR block

Matthew S. Crocker wrote:

> would like to put a CIDR block of addresses (/25) through my LVS server. Is there a way I can set one entry for a VIP range and then the load balancing will be handled over the entire range.

Horms `horms@vergenet.net` 2001-01-13

Set up fwmark rules on the input chain to match incoming packets for the CIDR and mark them with a fwmark.

e.g.

```
ipchains -A input -d 192.168.192.0/24 -m 1
```

Use the fwmark (1 in this case) as the virtual service.

```
ipvsadm -A -f 1
ipvsadm -a -f 1 -r 10.0.0.1
ipvsadm -a -f 1 -r 10.0.0.2
```

9.14 Example: forwarding based on client source IP

client A (from 192.x.x.x) should go to realserver 1..3, and client B (from 10.x.x.x) should go to realserver 4..6.

(Julian, 10-05-2000)

Write fwmark rules based on the source IP of the packets. Then create two virtual services, one for each fwmark.

9.15 Example: load balancing multiple class C networks

Ian Courtney wrote:

> Basically here at our ISP, we tend to have 2-3 Class C's worth of hosting per server. We would like to move the the LVS, but I'm not exactly sure how I should be setting it up.

Chris `chris@isg.de` 2001-01-15

You can use the fwmark option for the loadbalancing

```
#mark the incoming packets with ipchains
ipchains -A input -s 0.0.0.0/0 -d 192.168.0.0/24 -m 1
#then you can setup your LVS like
ipvsadm -A -f 1 -s wlc
ipvsadm -a -f 1 -r 10.10.10.15 -g
ipvsadm -a -f 1 -r 10.10.10.16 -g
```

the router should point to the director.

Ian Courtney wrote back:

> It didn't work until I aliased all 3 class C's to my director. Do I have to do this?

Julian Anastasov `ja@ssi.bg` 2001-01-16

Yes, only the packets destined for local addresses/networks are accepted. The others are dropped or forwarded to another box.

> the next project involves redoing our standard linux web space, which so far consists of about 8 webservers, each hosting atleast 2 class C's worth of hosting. I some how don't think Linux will take nicely to have 16 or more class C's aliased to it.

If possible use netmask <24. I assume you execute (replace with the right Class C nets):

```
ifconfig lo:1 207.228.79.0  netmask 255.255.254.0
ifconfig lo:2 207.148.155.0 netmask 255.255.255.0
ifconfig lo:3 207.148.151.0 netmask 255.255.255.0
```

on the director and on each real server and solve the arp problem using:

```
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
```

in the real servers. If you don't want to advertise these addresses using ARP to the Cisco LAN, you can execute the above two commands in the director too.

9.16 Example: proxy server

Thomas Proell, 16 Aug 2000

> How do you use fwmark if you want the director to accept packets for a wide range of addresses, for which is doesn't have IPs.

(Horms)

Here's a setup I used...

```
                                               Internet
                                                  |
                                                Router 192.168.128.1
"client"                  Linux Director          |
  va2-------------------------va3-----------------+--------- proxy (va4)
192.168.16.3      192.168.16.1   192.168.128.2        192.168.128.5
```

I have used 192.168/16, but these could be real addresses too. I have only put one proxy server in the diagram but I did test it with 2

```
Client: default gw va3 (192.168.16.1)

Linux Director:
eth0: 192.168.128.2             (internet/proxy side)
eth1:  192.168.16.1             (client side)
Default gw: Router ,192.168.128.1
IPV4 forwarding enabled.

Ipvsadm rules - these can be translated into ldirectord configuration.
ipvsadm -A -f 1 -s wlc
ipvsadm -a -f 1 -r 192.168.128.3:0 -g -w 1
... add additonal proxy servers
```

Interestingly enough if you add a proxy that just forwads traffic then it will end up going direct. This may be useful as a failback server if the proxy servers fail.

```
ipchains -A input -s 0.0.0.0/0.0.0.0 -d 127.0.0.1/255.255.255.255 -j ACCEPT
ipchains -A input -s 0.0.0.0/0.0.0.0 -d 192.168.128.2/255.255.255.255 -j ACCEPT
ipchains -A input -s 0.0.0.0/0.0.0.0 -d 192.168.16.2/255.255.255.255 -j ACCEPT
ipchains -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 80 -p tcp -j REDIRECT 80 -m 1
```

The -m 1 means that IPVS will regognise packets patched by this filter as belonging to the virtual service as long as it sees the packets as local. -j REDIRECT 80 makes the packets appear as local. It is of note that the port you redirect to is _ignored_ because of the way IPVS works - paickets using fwmark are sent to the port they arrived on. This means that packets will be sent to proxy servers as port 80 traffic.

```
Proxy:
eth0: 192.168.128.5
Default gw: 192.168.128.1 (router)
IPV4 forwarding enabled.

ipchains -A input -s 0.0.0.0/0.0.0.0 -d 127.0.0.1/255.255.255.255 -j ACCEPT
ipchains -A input -s 0.0.0.0/0.0.0.0 -d 192.168.128.5/255.255.255.255 -j ACCEPT
ipchains -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 80:80 -p 6 -j REDIRECT +8080
```

Note, this is where the redirection to port 8080 takes place.

9.17 Example: transparent web cache

`Pongsit@yahoo.samart.co.th` May 08, 2000

If i would like to use LVS to balance 3 transparent proxy is this how i do it ?

```
                 Internet
                    |
                     |
 ------------------------------------------- hub 1
          |          |           |
          |eth0      |           |          proxy1 ,2 and 3 set as a
       proxy1     proxy2      proxy3        transparent proxy with firewall
          |eth1      |           |          where eth0 connect to internet
          |          |           |          and eth1 to the internal network
 -------------------------------------------
             |          |     |     |     |    hub 2
             |          |     |     |     |
          LVS/DR       client machines    |
                                          |
                                          |
 -------------------------------------------   hub 3 if i have more internel
                                                    users
```

Horms `horms@vergenet.net` 2000-05-08

If you want to do transparent proxying then I would suggest a topology more along the lines of:

```
                 Internet
                    |
                    |
------------------------------------------------- hub 1
                    |
                    |
                 LVS/DR
                    |
                    |
------------------------------------------------
   |      |      |      |     |     |     |    hub 2
   |      |      |      |     |     |     |
 proxy1 proxy2 proxy3  client machines    |
                                          |
                                          |
-------------------------------------------------
                                                 hub 3 if i have more internel users
```

Use IP chains mark all outgoing port 80 traffic, other than from the 3 proxy servers with firewall mark 1 (ipchains -m 1...).

Set up a IPVS virtual service matching of fwmark 1 (ipvsadm -A -f 1...).

The proxy servers will need to be set up to recognise all port 80 traffic forwarded to them as local.

This way all outgoing traffic hits the LVS box. If it is for port 80 and isn't from one of the proxy servers then it gets load balanced and forwarded to one of the proxy servers.

You may want to consider a hot standby LVS/DR host to eliminate a single point of failure on your network.

I haven't tested this but I think it should work.

9.18 Example: Multiply-connected router

(Joe: my initial -apparently incorrect- reaction was that routing protocols would handle this better.)

Martin Sk?tt `martin@xenux.dk` 19 Jun 2001

I have several ADSL connections to the Internet (same ISP) and I wan't all the users on my network to be using them. I would like it to work in a way so that all the lines are utilised all the time and without assigning groups of users to specific gateways.

```
                                              Internet
                                            /
My users ---- Linux box with LVS - Internet
                                            \
                                              Internet
```

What I want to do is assign one default gateway, the LVS box.

Joe

> ..for doing by LVS, you could set up a director to be a router and setup like it was infront of 4 squid boxes (you'll need the IP's of the other end of the ADSL link). There's an 9.17 (example proxy) above.

Alexandre Cassen `Alexandre.Cassen@wanadoo.fr`

> I have tried some time ago that kind of setup. I have test 4 differents topology
>
> - Using a dynamic routing protocol like BGP. Using BGP you can use cost onto your routing path. To setup a multipath Internet connection using BGP all the ISP connected to your BGP setting must be informed to add BGP their side. This setup is recommanded by ISP for eavy corporate Internet use. It is mostly expensive due to ISP router side reconfiguration.
> - Implementing a loadsharing topology like discripbed into the "Linux 2.4 advanced routing HOWTO" section 9.5. You need here to use the same ISP for all your Internet connections because your ISP must implement the symetric config. This mean that ISP must support linux 2.4 loadsharing over multiple interface. This is rarely implemented by ISP because it is much more interresting implementing constructor integration that is more expensive. This is my feedback in France :/
> - Setting up router with multiple default gateway. That way you will loadbalance by TCP conversation. I have only implemented this on CISCO, your are limited to the max default gw number implemented (3 or 4 for CISCO).
> - Implement the solution discribed in the LVS HOWTO (above). Loadbalancing a squid server pool, each squid directly connected to your ADSL line.
>
> Personally, I prefer the LVS solution which is much more easy and recommanded because it is ISP configuration independent. I have tested that on a RTSP proxy pool.

9.19 httpd clients (browsers)

Initially when testing you should use a non-persistent (in the netscape sense) client, *e.g.* telnet VIP 80, or lynx VIP. Or else revert to these if you don't understand what you're seeing with netscape.

9.19.1 Opera

Peter Mastren `Peter.Mastren@chron.com` 18 Dec 2001

For the past several weeks, we have experienced almost daily denial of service attacks/events on our www servers. A remote client somewhere has opened a number of TCP connections to LVS that have absolutely no traffic whatsoever, save a single keepalive packet every two minutes. I have seen few as 3 and as many as 120 connections in the various incidents over the weeks.

These open connections are counted in the algorithm LVS uses to schedule servers, so the server that has all these open connections receives proportionately fewer new connections, in most cases taking the target server completely out of rotation.

Yesterday, I noticed an event coming from 130.80.XXX.XXX, our firewall address. Three connections were being held open from a machine inside our network. The culprit was my own workstation. I killed my browser and the connections went away. I fired up my browser again and tried to retrace my steps to duplicate the situation.

To make a long story short, it appears that Opera version 6.0 beta will leave a connection open to a server even after the window that was used for that connection has been closed. The only time the connection is closed is when Opera exits.

I will submit a problem report to Opera, in the meantime, there could be hundreds if not thousands of beta Opera browsers out there that could lock up ports on our servers for hours or days or longer.

This morning I made a configuation change to LVS that seems to have solved the problem. The masquerading portion of the Linux kernel (using LVS_NAT) uses default times to keep connections open, one for TCP connections, one for closed TCP connections that have received a FIN, and one for UDP connections. These defaults are 15, 2, and 5 minutes respectively. I changed the TCP timeout from 15 minutes to 110 seconds, which is shorter that the two minute intervals that the keepalive packets occur, yet long enough for any imaginable connection to a web server.

The change I made was:

```
	ipchains -M -S 110 0 0
```

9.20 Example: Dynamically generated images in webpages

One of the assumptions of setting up an LVS is that the content presented on the realservers is identical. This is required because the client can be sent to any of the realservers. This requirement is not handled if the client fills in a form which produces a gif on the realserver.

Alois Treindl `alois@astro.ch` 30 Apr 2001

> If a page is created by a CGI and contains dynamically created GIFs, the requests for these gifs will land on a different realserver than the one where the cgi runs. Will I need persistence?
>
> I am running an astrology site; a typical request is to a CGI which creates an astrological drawing, based on some form data; this drawing is stored as a temporary GIF file on the server. A html page is output by the CGI which contains a reference to this GIF.
>
> The browser receives the html, and then requests the GIF file from the server. It will mostly hit a different server than the one who created the GIF.
>
> So either we make sure that the new client request for the GIF hits the same realserver which ran the CGI (i.e. have persistence) or we must create the GIF on a shared directory, so that each realserver sees it.
>
> I have not tested it yet (not ported the CGIs yet to the new LVS box) but I think things are not so simple. In a 'rr' scheduling configuration, for example, the scheduler could play dirty, depending on the number of http requests for the given page, and the number of realservers.

> Both could be incommensurable in a way that the http request for the GIF never reaches the same realserver as the one which ran the CGI request.
>
> I had already decided that I need shared directories between all real servers for our CGI environment which does computationally expensive things all the time. Some CGIs create also data files which are used by later CGIs. It is either shared directories for such files, or a shared database (which we also use).
>
> These temp files will be sitting in the RAM cache of the NFS server, so that only network bandwidth between the realservers and the NFS server is the limiting factor. This is why I give the NFS server 2 gb of RAM, the max it will physically take, and this is why I chose 2.2.19 as the kernel because it contains NFS-3, which is said to be faster than NFS-2.

(Joe)

I tested it here on a page which generates a gif for the client. I found that I could never get the gif. Presumably after downloading the page containing the reference to the gif, the round robin scheduler sends the request for the gif to another realserver.

Presumably even page counters will have this problem. Writing to a shared directory should work.

Here's a solution with persistent fwmark using ip_tables to setup on a 2.4.x kernel. (Note: for page counters, this method will increment for each realserver, and not for the total page count over all the realservers as would happen with a shared directory.)

```
#put fwmark=1 on all tcp packets for VIP:http arriving on eth0
director:# iptables -t mangle -A PREROUTING -i eth0 -p tcp -s 0.0.0.0/0 \
            -d 192.168.1.110/32 --dport http -j MARK --set-mark 1
#setup a 2 realserver LVS to persistently forward packets with fwmark=1 using rr scheduling.
director:# ipvsadm -A -f 1 -s rr -p 600
director:# -a -f 1 -r sneezy.mack.net:0 -g -w 1
director:# -a -f 1 -r bashfull.mack.net:0 -g -w 1
#output setup
director:# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             lvs.mack.net       tcp dpt:http MARK set 0x1
director:# ipvsadm
IP Virtual Server version 0.2.11 (size=16384)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0            Route   1      0          0
  -> sneezy.mack.net:0              Route   1      0          0
```

Here's the output of ipvsadm after the successful generation and display of the dynamically generated gif. Note all connections went to one realserver.

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.11 (size=16384)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
FWM  1 rr persistent 600
  -> bashfull.mack.net:0            Route   1      5          3
  -> sneezy.mack.net:0              Route   1      0          0
```

9.21 Example: Balancing many IPs/services as one block

The simplest LVS balances requests to a VIP:port amongst a group of realservers. If you are servicing many VIPs, then few requests may be present for any particular IP at any time and a disproportionate number of requests will be sent to the first realserver. In this case you should balance all the different IPs as one group.

Josh Marcus josh@serve.com> 02 Oct 2001

I'm using LVS to serve a few thousand domains, but I don't see how I can setup LVS to load balance all of the domains as if they were all a single ip. In my ideal world, I would have a single entry *:80 that would forward all of our ips at port 80 to our set of realservers, and load balance all requests coming in. The way LVS is working for us now, the vast majority of all of our requests are going to the server that is for some reason being listed first. Only sites with heavy traffic get pushed along to the other servers.

Michael E Brown michael_e_brown@dell.com>

> fwmarks

9.22 Appendix 1: Specificiations for grouping of services with fwmarks

Here are the discussions that has resulted in the current specifications for handling of persistence with fwmarks in LVS.

Ted Pavlic Jul 14, 2000

> What I was asking about would be something like this:

```
virtual=192.168.6.2-192.168.6.30:80
        real=192.168.6.240:80 gate
        service=http
        request="index.html"
        receive="Test Page"
        scheduler=rr
```

> I have 1029 virtual servers – that is I have 1029 hosts which need to be load balanced.

Horms horms@vergenet.net 2000-07-14

(fwmark) has the advantage of simplfying the amount of _kernel_ configuration that has to be done which is a big win, even if this is automated by a user space application. The basic idea is that this provides a means for LVS to have virtual services that have more than one host/port/protocol triplet. In your situation this means that you can have a single virtual service that handles many virtual IP addresses and all ports and protocols (UDP, TCP and

You should take a look at *ultramonkey* <http://ultramonkey.sourceforge.net/> (note from Joe, UM is now 1.0.2, look for examples there). My understanding is that this is quite similar to how your LVS topology will be set up, though I understand you will be having more than one of these configured.

Basically what happens is that you set up LVS to consider any packets like other LVS virtual services other than that no VIP is specified.

e.g.

```
ipvsadm -A -f 1 -s rr
ipvsadm -a -f 1 -r 192.168.6.3:80 -m
ipvsadm -a -f 1 -r 192.168.6.2:80 -m
ipvsadm -L -n
IP Virtual Server version 0.9.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
FWM  1 rr
  -> 192.168.6.3:80              Masq    1      0          0
  -> 192.168.6.2:80              Masq    1      0          0
```

The other half of the equation is that ipchains is used to match incoming traffic for virtual IP addresses and mark them with fwmark 1. Say you have 8 contiguous class C's of virtual addresses beginning at 192.168.0.0/24. The ipchains command to set up matching of these packets would be:

```
ipchains -A input -d 192.168.0.0/21 -m 1
```

You also need to set up a silent interface so that the LVS box sees traffic for the VIPs as local. To do this use:

```
ifconfig lo:0 192.168.0.0 netmask 255.255.248.0 mtu 1500
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
```

Now, as long as 192.168.0.0/21 is routed to the LVS box, or more particularly the floating IP address of the LVS box brought up by heartbeat, traffic for the VIPs will be routed to the LVS box, the ipchains rules will mark it with fwmark 1 and LVS will see this fwmark and consider the traffic as destined for a virtual service.

Ted Jul 14, 2000

> for me to enable persistent connections to every port using direct routing, would this work?

```
ipvsadm -A -f 1 -s rr -p 1800
ipvsadm -a -f 1 -r 216.69.192.201:0 -g
ipvsadm -a -f 1 -r 216.69.192.202:0 -g
```

Horms

Yes, that would work. The port in the "ipvsadm -a" commands is ignored if the real servers are being added to a fwmark service. Connections will be sent to the port on the real server that they will be recieved on the virtual server. So port 80 traffic will go to port 80, port 443 traffic will go to port 443 etc...

As a caveat you should really make sure that your ipchains statments catch all traffic for the given addresses including ICMP traffic so ICMP traffic is handled correctly by LVS.

(Julian on catching ICMP traffic)

IIRC, this is already not a requirement in the last LVS versions. If we look in skb->fwmark for ICMP packets it is impossible to use normal and fwmark virtual services to same VIP because we can't create such ipchains rules. The good news is that in 2.4 (0.0.3) the virtual service lookup (the fwmark field) is used only for the new connections. In 2.2 the service is looked up even for existing entries but we don't want to break the MASQ code entirely

Ted Pavlic `tpavlic@netwalk.com` 19 Jul 2000

When using fwmark to assign real servers to virtual servers, how is scheduling and persistence handled?

In my particular example, I have: 216.69.196.0/22 (ie 4 class C networks) all marked with a fwmark of 1. ipvsadm setup is

```
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port          Forward Weight ActiveConn InActConn
FWM  1 lc persistent 600
-> nw01:0                      Route   1      0          0
-> nw02:0                      Route   1      0          0
```

Say someone connects to 216.69.196.1 and the connection is assigned to nw01. At this point ipvsadm shows

```
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port          Forward Weight ActiveConn InActConn
FWM  1 lc persistent 600
-> nw01:0                      Route   1      1          0
-> nw02:0                      Route   1      0          0
```

A new person connects to another IP in 216.69.196.0/22 (say 216.69.196.2). Will this new connection to 216.69.196.2 go to nw02 because it has the least number of TOTAL connections, or will it go to nw01 because for that PARTICULAR IP, both have 0 connections?

Now then say that the person who just connected to 216.69.196.1 makes a connection (within the 600 persistence seconds) to 216.69.196.3. Will this new connection go to nw01 because it's being persistent? Or will it go to either server depending on the number of connections?

Here's what I think would be the best way to do things...

If multiple IPs are marked with FWMARK 1, LVS should consider them all one entry in its active/inactive table. I don't believe that's how things are currently being handled.

(Julian)

The templates are not accounted in the active/inactive counters.

> (Joe, almost a year later - Julian, what do you mean here?) (Julian 13 Apr 2001)
>
> Ted here thinks that the templates are accounted in the inactive/active counters. And before the persistent-fwmark patch we can have many templates for one fwmark-based service:
>
> ```
> CIPNET:0 -> VIP1:0 -> RIP_A:0
> CIPNET:0 -> VIP2:0 -> RIP_B:0
> ```
>
> where VIP1 and VIP2 are marked with same fwmark.
>
> Ted recommends these two templates to be replaced with one, i.e. just like in the persistent-fwmark patch:
>
> CIPNET:0 -> FWMARK:0 -> RIP1:0
>
> We can't see the templates (which are normal connection entries with some reserved values in the connections structure fields) accounted in the inactive/active counters. The reason for this is that the inactive/active counters are used to represent the real server load but our templates don't lead to any load in the real servers, we use them only to maintain the persistence.

When a service is marked persistent all connections from CIP to VIP go to same RIP for the specified period. Even for the fwmark based services. This works for many independent VIPs.

The other case is fwmark service covering a DNS name. I expect comments from users with SSL problems and persistent fwmark service. Is there a problem or may be not?

I agree, may be the both cases can be useful:

```
1. CIP->VIP
2. CIP->FWMARK
```

Any examples where/why (2) is needed?

But switching the LVS code always to use (2) for the persistent fwmark services is possible.

(Ted)

In my opinion, here are some pros and cons of case 2:

Pros:

Improves scheduling, I think, and true load balancing. If someone is using [W]RR or [W]LC, the LVS box will actually look at the real servers as a whole rather than separate real server entries for EACH VIP. Does that make sense?

For example, in my particular configuration I have over one thousand VIPs which are load balanced onto four RIPs. When I configure the LVS server to use LC scheduling, I'd like it to look at how many TOTAL connections are being made to each RIP not how many connections are being made to each RIP PER VIP. I would like to load balance all one thousand VIPs as a WHOLE onto the four RIPs rather than load balance EACH VIP.

That is, in some of my less active sites, most of their traffic will probably hit one VIP just because not much traffic will need to be load balanced. However, more active sites will hit both servers. The load will then not be distributed equally among the servers as one server will probably get not only the active traffic but also the less active traffic and the other server will only get the more active traffic (in the case of having two RIPs).

Cons:

One person on the Internet will keep connecting to the same RIP for many different VIPs if persistence is turned on.

If this causes a problem, the LVS administrator can do one of two different things:

1) Rather than load balancing a fwmark template, go back to load balancing specific VIPs. The scheduling will then be unique for those particular VIPs.

2) Create multiple fwmark templates. The scheduling for each template will be unique.

In my opinion if you group a bunch of IPs together by marking them with an fwmark, that you say that you want to load balance all of those COLLECTIVELY – almost like load balancing one site.

I'm just saying, are there any examples where CIP->FWMARK is not needed?

As far as the LVS is concerned, if someone connects to a VIP marked with fwmark 1, it should treat it just like every other VIP marked with fwmark 1 – as if they were all one VIP.

But today on my LVS (where I have a ten minute persistence setup) I connected to one virtual server marked with fwmark 1 and got a certain real server. I then expected to connect to another virtual server also marked with fwmark 1 and get that same real server. I did not, however. If what you're telling me is correct, the persistence should have connected me to the same real server as long as I was connecting within that ten minute window.

Now in this particular example – connecting to DIFFERENT virtual servers – it isn't so necessary for persistence to be carried through PER virtual server. I'm just worried that least connection scheduling and round-robin scheduling aren't working at the fwmark level – I'm worried that they are working at the VIP level as if I had setup hundreds of explicit VIP rules inside IPVSADM.

Julian I hope this feature (2) will be implemented in the next LVS version (if Wensong don't

> see any problems). I.e. the templates can be changed to case (2) for the persistent fwmark services. For now we (I and Horms) don't see any problems after this change. Then connections from one client IP to different VIPs (from the same fwmark service) will go to the same real server (only for the persistent fwmark services).

Do you see any reason why enabling CIP->FWMARK for all cases would be a bad thing?

That is, not only using case 2 for persistent fwmark, but just whenever fwmark was used. Personally, I cannot ever forsee a scenerio when a person would setup an fwmark for load balancing and want each VIP associated with that fwmark to act independently.

> Web cluster for independent domains (VIPs). fwmark service is used only to reduce the amount of work for configuration.

I've always thought that the scheduling algorithms should look directly at the real servers rather than the real server stats for each particular virtual server. That is, least connection scheduling would look at the total number of connections on a real server, not just the connections from that particular VIP. Round-robin would go round-robin from real server to real server based on the last connection from ANY VIP to the real servers... However, before fwmark I realized that this would probably very difficult to do especially in cases where an LVS administrator was load balancing to a number of different real server clusters that may overlap.

> This is a job for the user space tools: WRR scheduling method + weights derived from the real server load. Yes, one real server can be loaded from:
>
> - many directors
> - many virtual services
> - other processes not part from the real service
>
> In this case the director's opinion (for each virtual service) about the real server load is wrong. The only way to handle such case properly is to use WRR method. In the other cases WLC, LC and RR can do their job.

fwmark, to me, just by causing all VIPs marked with a particular fwmark to look like one big VIP makes it possible to do basically that which I just described. I don't see why anyone would not want such functionality with the fwmark services. If one did want such functionality, he would probably partition the VIPs associated with his fwmark into separate fwmarks or even explicit VIP entries anyway.

> Yes. IMO, this can be a problem only for the balancing but I don't think so. The problems will come when one real server dies and the client can't access any VIP part from the fwmark service for a period of time.

9.23 Appendix 2: Demonstration of grouping services with fwmarks

Here's the original e-mail between Ted `tpavlic@netwalk.com` 3 Aug 2000 and Joe

One of the things it fwmarks lets me do is make ports sticky by groups.

Basically I setup ipchains rules that say all packets to ports 80/tcp and 443/tcp mark with a 1. All packets to ports 20/tcp and 21/tcp as well as 1024:65535/tcp mark with a 2. Voila... I just made ports stick by groups.

I then go into IPVS and setup my real servers under FWMARK1 and FWMARK2. Ports 80 and 443 are now persistent as a group just as 20 and 21 and 1024:65535 are persistent as a group. If my HTTP goes down on one of my real servers, I do not have to take my FTP down as well. I only have to remove the real server from the FWMARK1 group. It's great!

> Joe most people don't program their own on-line transaction processing program and the point of an LVS is for the realservers to be running the same code as when they're stand alone.

My users run PHP scripts as well as ASPs that keep session information. That session information is unique per server and usually is stored in a local /tmp directory. Users are handed cookies which tie them to their session information. If they go to the wrong real server, that session information won't exist and a number of things could go wrong.

most of my real servers run a lot of services... HTTP, HTTPS, FTP, SMTP, POP3, IMAP, DNS, And when one of them went down (with persistence set up), I would have to take the entire real server down.

Several problems:

*) One little thing goes down... POP3, for example. Now the load increases a great deal on all my other real servers... Perhaps causing the load to become so high that sendmail starts rejecting connections... and then THAT real server also is taken COMPLETELY down... domino effect. If I could have just taken POP3 down off of that server, it would have been perfect.

*) Say something horrible happens causing sendmail to go down on all the servers... or HTTP... or POP3... any one service – just as long as it goes down on all servers. Rather than just causing that service to be affected, ALL of my services go down because every real server was taken completely off-line until that ONE service is fixed. :(

But I figured that those two problems wouldn't be that big of a deal... I could probably put such a system in production.

Well – I put such a system in production and those problems weren't that big of a deal... Except for a COUPLE of times when all services went down and caused a BIG hassle. So my superiors wanted something better – needed in fact.

So at first I came up with the interim idea of separating persistent services and non-persistent services by IP. All of my persistent services were basically on one supernet and all of my non-persistent services were on another subnet. Consequently, I could tie the one supernet to one FWMARK and the other subnet to another FWMARK. Now if a persistent service went down, it would bring down only all of the persistent services. Also, if a non-persistent service went down, it would only bring down all of the non-persistent services.

This was definitely an interim solution because it required a lot more IPs that any one administrator should need, and it still was far from perfect.. BUT... I started to realize that just as I could mark different supernets and subnets with different FWMARKs, I could go farther down the TCP/IP layers and mark things at their protocol and port level. That's where I realized that we COULD do persistents by port group just with a little help from ipchains.

> Joe I asked Horms if there was any point in having multiple fwmarks. His only example was if you had duplicate sets of realservers. Eg the paying customers get the fast servers, while the people coming into the free site get the 486 with 16M.

Similar idea here... except rather than setting up your policies like:

- Paying customers -> fast server
- Free -> slow server

You have:

- SMTP -> a realserver
- POP3 -> another realserver
- HTTP/HTTPS -> yet another realserver
- FTP -> and another realserver

The key of it all is the fact that you can group by about any parameter that ipchains can see. If ipchains can segregate it, you can group it. Anything that ipchains can do IPVS can then add onto itself.

Joe Have you solved passive ftp without using persistance?

I really don't think there's any way to get around it... In order to get passive FTP to work, you need to make TCP port 21 persistent with every TCP port above 1024. I mean – how else could you do it without putting some big brother software inside of LVS which would keep an eye on FTP and see what port it tells the end-user to connect to.

Still, putting 21 and 1024:65535 together is a lot better than putting everything together. Personally I only plan on load balancing things in the < 1024 range anyway, so I have no problem including that huge group above 1024.

This is my setup

```
FWMARK1 => HTTP/HTTPS (persistent)
FWMARK2 => FTP (persistent)
FWMARK3 => SMTP
FWMARK4 => POP3
FWMARK5 => DOMAIN
FWMARK6 => IMAP
FWMARK7 => ICMP (for kicks)

================
IP Virtual Server version 0.9.12 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
FWM  1 lc persistent 600
  -> nw04:0                      Route   1      58         121
  -> nw03:0                      Route   1      49         76
  -> nw02:0                      Route   1      60         98
  -> nw01:0                      Route   1      61         44
FWM  2 lc persistent 600
  -> nw04:0                      Route   1      0          2
  -> nw03:0                      Route   1      0          2
  -> nw02:0                      Route   1      1          13
  -> nw01:0                      Route   1      1          0
FWM  3 lc
  -> nw04:0                      Route   1      4          11
  -> nw03:0                      Route   1      4          12
  -> nw02:0                      Route   1      3          20
  -> nw01:0                      Route   1      3          16
```

```
FWM  4 lc
  -> nw04:0                     Route   1      3          54
  -> nw03:0                     Route   1      1          74
  -> nw02:0                     Route   1      3          51
  -> nw01:0                     Route   1      2          73
FWM  5 lc
  -> nw03:0                     Route   1      0          46
  -> nw01:0                     Route   1      0          44
  -> nw02:0                     Route   1      0          45
  -> nw04:0                     Route   1      0          45
FWM  6 lc
  -> nw04:0                     Route   1      0          0
  -> nw03:0                     Route   1      0          0
  -> nw02:0                     Route   1      1          0
  -> nw01:0                     Route   1      0          0
FWM  7 lc
  -> nw04:0                     Route   1      0          0
  -> nw03:0                     Route   1      0          0
  -> nw02:0                     Route   1      0          0
  -> nw01:0                     Route   1      0          0
==============
```

Is this anything new?

> Joe It's new to me and Horms didn't have any other ideas for multiple fwmarks 3 weeks ago, so I expect it will be new to him.

I've been thinking of ways of combining different programs which already exist out there to get L7 scheduling working. For example – you have some program (sorta like policy routing but one more layer up) that filters packets at the application layer and does something to them... routes them to a particular IP... something like that... and then have ipchains mark each one of those packets with a particular mark... and have LVS work from there.

You see – using multiple fwmarks makes me think that you can do a lot more with LVS.

We could probably borrow some of the ideas used for some of the dynamic routing protocols, like BGP or RIP. A master could advertise its IPVS hash table. If it didn't advertise within a given interval of time, other LVS's could take over.

During the failover, rather than trading an IP like we were talking about, all LVSs could know which one is the active one and ICMP redirect to that LVS or something like that.

Right now I'm routing every virtual server through the active LVS. This lets me do a lot of nifty things (for me at least):

* Very little has to happen on the LVS during failovers. They basically just trade an IP. In fact – I COULD do the failover right at the router before the LVS's – just have it route to another IP.

* I do not have to bring every IP up on my realservers – I just have to bring the network that they're on up on a hidden loopback device. When you route an entire network to a loopback device, the loopback device answers every IP on that subnet automatically. So even with 1024+ IPs, I have to setup very few interfaces/aliases because a great deal of them are on the same subnet.

9.24 Appendix 3: Announcement of grouping services with fwmarks

Ted Pavlic `tpavlic@netwalk.com` 4 Aug 2000

Periodically the issue comes up regarding wanting to do persistence by groups of ports. Until now, an LVS administrator could make a single port persistent or all ports persistent.

Single port persistence was nice for quite a few things. However, things like HTTP and HTTPS caused complications with it. Someone who connected to a webpage on HTTP and started a session tied to them with a cookie would want to return to that same real server when they went to the HTTPS version of that site. FTP would also cause a problem with single port persistence as someone who wanted to use passive FTP wouldn't be gauranteed the same server when they returned on a random TCP port above 1024. There are other examples as well.

So the solution to these problems would be to make every port persistent. This works pretty well, but now anytime a user of a large network behind a firewall would connect to a real server on ANY service, everyone behind that firewall would hit that same real server. Plus, if an administrator wanted to stop scheduling a single service to a single real server, he would have to take all services down on that single real server. This causes many problems as well... especially if one small service dies on every real server – brings down every service on every real server.

So there has been the need for persistence by port GROUPS. Rather than saying all ports are persistent, it would be nice to tell LVS to tie just 80/tcp and 443/tcp together or just 21/tcp and 1024:65535/tcp together. Before the wonderful FWMARK additions to LVS, this was not possible.

But now that LVS listens to FWMARKs, it becomes possible to group ports together inside ipchains with different FWMARKs and then tell LVS to listen to those FWMARKs.

For example, one can setup a rule inside FWMARK to do this...

```
80/tcp, 443/tcp --> FWMARK1
21/tcp, 1024:65535/tcp --> FWMARK2
25/tcp --> FWMARK3
110/tcp --> FWMARK4
```

Then inside LVS (assume on this setup all of these services are served by the same real server cluster), say:

```
FWMARK1 -> PERSISTENT -> real1,real2,real3,real4
FWMARK2 -> PERSISTENT -> real1, real2, real3, real4
FWMARK3 -> real1, real2, real3, real4
FWMARK4 -> real1, real2, real3, real4
```

Not only have you now setup persistence by port groups, but you've also split your services back up into autonomous services that will not bring EVERY server down for the sake of persistence. If FTP goes down on real1, real1 only needs to be stopped scheduling for FTP.

9.24.1 another explanation

Ted Pavlic `tpavlic@netwalk.com` 2000-09-15

Using fwmark, you can setup something which used to be a big desire in LVS, persistence by port groups.

For example... Say you were serving HTTP and HTTPS. In this case, you would probably want calls to one HTTP server to end up hitting the same HTTPS server. This way session information and such would be accessable no matter how the end-user was accessing the website.

Say you also wanted all forms of FTP to work... You would need persistence there, but not necessarily the same persistence as HTTP/HTTPS.

And other protocols do not need to be persistent.

Back in the olden days before fwmark, to do any of this you would have to make ALL ports persistent. You couldn't simply say "Group 80 and 443 together and make them persistent and then make 21, 20, AND 1024:65535 persistent." If one service went down, you would have to bring down ALL services. Some sort of persistence by port groups would allow you to only need to take down whatever went down and the affected server could still serve other services.

FWMARK allows you to do this by way of setting up multiple FWMARKs.

That is – you can use ipchains to say that:

```
HTTP,HTTPS --> FWMARK1
FTP --> FWMARK2
SMTP --> FWMARK3
POP --> FWMARK4
```

Then in LVS, setup:

```
FWMARK1 --> WLC Persistent 600
FWMARK2 --> WLC Persistent 300
FWMARK3 --> WLC
FWMARK4 --> WLC
```

And if FTP went down, all you'd have to do is stop scheduling FTP rather than stop scheduling EVERYTHING.

Also note that FWMARK makes setting up MASS VIPs really easy (of course because of recent ARIN policy changes, this probably won't be done much more anymore). That is, if you wanted to load balance 1000 VIPs, it might be easy to setup one single rule in ipchains to cover them all, where it would be 1000 rules for EACH real server in ipvsadm.

It makes me think that if there was a utility already out there that could sit on a director and figure out where name-based packets were going it might be able to mark each name-based host with a different FWMARK and pass that right back to LVS... Then LVS wouldn't have to worry about handling name-based stuff ITSELF. Of course the name-based challenge is even more challenging considering how much data needs to be looked at to figure out if a TCP stream is a name-based HTTP session going to specific name X.... but that's a completely other argument... Just food for thought.

10 Configure tools

There are some tools to help you configure an LVS.

- Gui for LVS-DR lvs-gui http://www.us.vergenet.net/linux/lvs-gui. This only sets up a LVS-DR LVS and is awaiting further work. (I believe this is no longer being maintained and should not be used for current work.)
- 10.1 (configure script) for all LVS types which does sanity checking on setup but which doesn't handle director failover

- 20 (tools which include director failover) *e.g.*Ultra Monkey by Horms, which handles director failover, but has to be setup by hand and vrrpd/keepalived by Alexandre Cassen, which sets everything up for you.

Unfortunately several of these scripts produce files with the name rc.lvs (just to make things interesting when reading the mailing list).

A recent survey of production LVS's (by Lorn Kay `lorn_kay@hotmail.com`, results on the *LVS website* `<http://www.linuxvirtualserver.org/survey/>`) showed that 10 out of 19 were setup by methods described as "other", ie they didn't use any of the scripts on the LVS website. Clearly the above scripts are not regarded as production quality by our users.

10.1 Configure script

(The section in the LVS-mini-HOWTO may be more recent than this.)

I have written a configure script, available at *the LVS software page* `<http://www.linuxvirtualserver.org/sofware/index.html>` (down at the bottom). The script will have a name like configure-x.x-lvs.tar.gz. The script handles realserver failure with Mon, but does not handle director failover. This is sufficient for testing. For production you will want to 20 (handle director failure).

The configure script was designed to set up LVS's quickly so I could do testing. The current version (0.9.x) does a wide range of checks, hopefully catching the usual errors. The configure script uses mon to handle failure of services on realservers.

The configure script reads a conf file and sets up LVS-DR, LVS-NAT, LVS-Tun, with single or multi-NIC machines using regular ethernet devices or transparent proxy to accept packets. Elementary checks are done on routing and connectivity and the script has always produced a working LVS when no errors are reported. I tried it with as many deliberately pathological cases as I can think of. The output is a well commented rc.lvs and rc.mon file to run at bootup (or when you're setting up an LVS).

The configure script detects the kernel version of the director (for old versions of ipvs/kernels) and is run on the director. The resulting rc.lvs (and rc.mon) files are run on the director and then the realservers (the configure files/directory nfs exported to the realservers, where I run the same rc.lvs file to configure the realservers). The rc.lvs file knows whether it's running on a director or realserver. After setup, the network connections neccessary to export this directory can be removed.

Running the script on the realservers after it is run on the director allows the rc.lvs script to check the connections to the now configured director. Running the rc.lvs script on the realserver first will achieve the same result, but you will get error messages, and won't know if the script succeeded even if it did.

Pick an appropriate template conf file (lvs_nat.conf, lvs_tun.conf or lvs_dr.conf) for the mode you are operating the LVS under (VS-NAT, LVS-Tun or LVS-DR respectively) and edit the IPs and services for your situation. The files come preconfigured for telnet as the service. This is the simplest service for initial tests: the client is readily available and on connection, the login prompt will tell you which server you have connected to. Use round robin scheduling, so that you will connect to each server in turn, confirming that they are all working.

Other services can be added by uncommenting/adding or editing the entries in the example *.conf files. The *.conf files give suggested IP's (192.168.1.x/24, 10.1.1.x/24) for the various machines.

For those that don't like to debug disasters from trivial mistakes, the configure script will use either the name (eg telnet) or port (eg 23) of the service in the *.conf file, using the /etc/services file to translate one to another. Likely additions needed to your /etc/services will be

```
ftp-data        20/tcp
```

```
ssh             22/tcp
domain          53/tcp  nameserver      dns #the string "dns" needs to be added here
domain          53/ucp  nameserver      dns #and here
shell           514/tcp cmd             rsh #add rsh here
https           443/tcp
https           443/udp
printer         515/tcp spooler         lpd #add the string "lpd" here
nfs             2049/udp        nfs
mysql           3306/tcp
mysql           3306/udp
netpipe         5002/tcp
```

In several instances, a machine will need multiple IPs. You can put multiple IP's on a single NIC with IP aliasing (an option when building the kernel) - just enter the aliased interface (eg eth0:12) for that IP into the *.conf file.

run the configure script

$ ./configure.pl lvs_nat.conf

This produces an rc.lvs_xxx script (eg rc.lvs_nat, rc.lvs_tun, rc.lvs_dr), and a mon_xxx.cf script. (After testing put rc.lvs_xxx into /etc/rc.d or /etc/init.d and put mon_xxx.cf in /etc/mon)

Run the rc.lvs script on the director and then the realservers with the command

$ . ./rc.lvs_dr

or possibly (if you get weird errors, eg it doesn't detect correctly whether it's running on a director or a realserver - this happens on my 2.0.36 realserver)

$sh rc.lvs_dr

The rc.lvs script -

- adds ethernet devices and routes to director, realservers (including non-Linux)
- checks connections with fping.
- runs ipchains
- turns ipforwarding on (VS-NAT) or off (VS-DR, LVS-Tun)
- turns off icmp redirects (VS-NAT)
- adds services with ipvsadm.

The rc.lvs script does not -

- know about director failover. It assumes there is only one director.

Check the output from ipvsadm, ifconfig -a and netstat -rn on the director and then the realserver(s), to see that the services/IP's are correct. If not re-edit and re-run the script(s).

10.1.1 Test with telnet

Telnet is a non-persistent service (in the http sense, rather than the LVS sense) and will allow you to check that all realservers are present (you'll get the login prompt from each realserver in turn).

Check that the service(s) are running on each server at the IP of the VIP (use netstat -an). Some services (eg telnet listen to 0.0.0.0, ie to all IPs on a machine).

If there are no errors on running the rc.lvs_xxx script, then telnet from the client to the VIP (here 192.168.1.110). You will be routed through to one of the realservers and you will get the login prompt (and real name) of a realserver and can login.

On the director look at the output of ipvsadm, you should see a connection to a realserver on port:23.

On the realserver do

$ netstat -an | grep 23

and look for connections to the telnet port.

Logout and telnet again to the VIP. You will get the login prompt from the next realserver.

10.1.2 Test with something else eg http

Edit lvs_nat.conf activating http, rerun configure.pl and rerun the new rc.lvs_nat files.

http: Point your browser to the VIP http://192.168.1.110. You will get the DocumentRoot of one of the realservers. Open another copy of the browser and connect again. You should get the other server (this will be easier to see if the webpages are different). Look at the output of ipvsadm on the director for connections to the httpd ports, and on the server look at the output from netstat -an | grep 80 (or 8080 if you are running LVS-NAT and have remapped the ports) for connections.

If your httpd is persistent (in the http sense), you will/may connect to the same website each time.

11 Services

In principle setting up a service on an LVS is simple - you run the service on the realserver and forward the packets from the director. The simplest service to LVS is telnet: the client types a string of characters and the server returns a string of characters. In practice some services interact more with their environment. Ftp needs another port. With http, the server needs to know its name (it will have the IP of realserver, but will need to proclaim to the client that it has the VIP). https is not listening to an IP, but to requests to a nodename. This section shows the steps needed to get the common services working.

When trying something new on an LVS, always have the service telnet LVS'ed. If something is not working with your service, check how telnet is doing. Telnet has the advantages

- telnetd listens on 0.0.0.0 on the realserver (at least under inetd)
- the exchange between the client and server is simple, well documented,
- the connection is non-persistence (new sessions initiated from a client will make a new connection with the LVS) unencrypted and in ascii (you can follow it with tcpdump)
- the telnet client is available on most OS's

11.1 setting up a new service

When setting up an LVS on a new service, the client-server semantics are maintained

- the client thinks it is connecting directly to a server

- the realserver thinks it is being contacted directly by the client

Example: nfs over LVS, realserver exports its disk, client mounts disk from LVS (this example taken from *performance data for single realserver LVS* <http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>),

realserver:/etc/exportfs (realserver exports disk to client, here a host called client2)

```
/       client2(rw,insecure,link_absolute,no_root_squash)
```

The client mounts the disk from the VIP. Here's client2:/etc/fstab (client mounts disk from machine with an /etc/hosts entry of VIP=lvs).

```
lvs:/   /mnt            nfs     rsize=8192,wsize=8192,timeo=14,intr 0 0
```

The client makes requests to VIP:nfs. The director must forward these packets to the realservers. Here's the conf file for the director.

```
#lvs_dr.conf for nfs on realserver1
.
.
VIP=eth1:110 lvs 255.255.255.255 192.168.1.110
DIP=eth0 dip 192.168.1.0 255.255.255.0 192.168.1.255
DIRECTOR_DEFAULT_GW=client2
SERVICE=t telnet rr realserver1 realserver2     #for sanity check on LVS
#to call NFS the name "nfs" put the following in /etc/services
#nfs             2049/udp
#note the 'u' for "udp" in the next line
SERVICE=u nfs rr realserver1                    #the service of interest
SERVER_VIP_DEVICE=lo:0
SERVER_NET_DEVICE=eth0 Why do you need persistence?
SERVER_DEFAULT_GW=client
#----------end lvs_dr.conf------------------------------------
```

11.2 services must be setup for forwarding type

The services must be setup to listen on the correct IP. With telnet, this is easy (telnetd listens on 0.0.0.0 under inetd), but most other services need to be configured to listen to an IP.

For LVS-NAT, the packets will arrive with dst_addr=RIP, i.e. the service will be listening to the RIP of the realserver. When the realserver replies, then name of the machine returned will be the realserver, but the src_addr will be rewritten by the director to be the VIP.

With LVS-DR and LVS-NAT the packets will arrive with dst_addr=VIP, i.e. the service will be listening to an IP which is *NOT* the IP of the realserver. Configuring the httpd to listen to the RIP rather than the VIP is a common cause of problems for people setting up http/https.

In both cases, in production, you will need to make the name of the machine given by the realserver to be the name associated with the VIP.

Note: if the realserver is Linux 2.4 and is accepting packets by transparent proxy, then see the section on 16 (TP) for the IP the service should listen on.

11.3 ftp general

ftp is a 2 port service in both active and passive modes. In general multiport services or services which need to run together on the one realserver (eg http/https), can be handled by persistance or by Ted Pavlic's adaption of fwmark (see 9.5 (fwmark for passive ftp)).

ftp comes in 2 flavors active/passive

11.4 ftp (active) - the classic command line ftp

This is a 2 port service.

- port 20 - data (the files you want)
- port 21 - commands (eg ls)

11.4.1 ip_vs_ftp/ip_masq_ftp module helpers

As part of the ip_vs build, the modules ip_masq_ftp (2.2.x) and ip_vs_ftp (2.4.x) are produced. The ip_masq_ftp module is a patched version of the file which allowed ftp through a NAT box. This patch stopped its original function (at least in early kernels).

The 2.2.x ftp module is only available as a module (*i.e.* it can't be built into the kernel).

> Juri Haberland `juri@koschikode.com` 30 Apr 2001 AFAIK the IP_MASQ_* parts can only be built as modules. They are automagically selected if you select CONFIG_IP_MASQUERADE.

> Julian Anastasov May 01, 2001 Starting from 2.2.19 the following module parameter is required:
>
> ```
> modprobe ip_masq_ftp in_ports=21
> ```
>
> > Joe I don't see this mentioned in /usr/src/linux/Documentation, ipvs-1.0.7-2.2.19/Changelog, google or dejanews. Is this an ip_vs feature or is it a new kernel feature?
>
> I see info only in the source. This is a new 2.2.19 feature.
>
> > ratz It's /usr/src/linux/net/ipv4/ip_masq_ftp.c:

```
* Multiple Port Support
*      The helper can be made to handle up to MAX_MASQ_APP_PORTS (normally 12)
*      with the port numbers being defined at module load time.  The module
*      uses the symbol "ports" to define a list of monitored ports, which can
*      be specified on the insmod command line as
*              ports=x1,x2,x3...
*      where x[n] are integer port numbers.  This option can be put into
*      /etc/conf.modules (or /etc/modules.conf depending on your config)
*      where modload will pick it up should you use modload to load your
*      modules.
* Additional portfw Port Support
*      Module parameter "in_ports" specifies the list of forwarded ports
*      at firewall (portfw and friends) that must be hooked to allow
```

```
         *      PASV connections to inside servers.
         *      Same as before:
         *              in_ports=fw1,fw2,...
         *      Eg:
         *              ipmasqadm portfw -a -P tcp -L a.b.c.d 2021 -R 192.168.1.1 21
         *              ipmasqadm portfw -a -P tcp -L a.b.c.d 8021 -R 192.168.1.1 21
         *              modprobe ip_masq_ftp in_ports=2021,8021
```

And it is a new kernel feature, not LVS feature.

what are these modules for: from ipvsadm(8) (ipvs 0.2.11)

> If a virtual service is to handle FTP connections then persistence must be set for the virtual service if Direct Routing or Tunnelling is used as the forwarding mechanism. If Masquerading is used in conjunction with an FTP service than persistence is not necessary, but the ip_vs_ftp kernel module must be used. This module may be manually inserted into the kernel using insmod(8)

> From Julian 3 May 2001, the modules are required for
>
> - VS-NAT
> - recommended for active ftp and mandatory for passive ftp, if persistence tricks are not used when setting up the LVS.

The modules are *NOT* used for LVS-DR or LVS-Tun: in these cases persistence is used (or fwmarks version of persistence).

> Jeremy Kusnetz: although Julian says that all you need for ftp with LVS-NAT is the ip_masq_ftp module, it doesn't work for me (director 2.2.19-1.0.7 with ip_masq_ftp in_ports=21) my ftp client just hangs.

Julian: Hm, what a day, let's try them tomorrow :)

> Joe 23 May 2001: I run these rules on the director (without the ftp module) and ftp works fine
>
> ```
> $ ipchains -A forward -p tcp -j MASQ -s RIP ftp -d 0.0.0.0/0
> $ ipchains -A forward -p tcp -j MASQ -s RIP ftp-data -d 0.0.0.0/0
> $ ipchains -A forward -p tcp -j MASQ -s RIP 1024:65535 -d 0.0.0.0/0
> ```

These rules are risky. What happens with ICMP? It is not masqueraded. I hope there is similar rule for ICMP.

11.4.2 VS-NAT, 2.2.x director

I found that ftp worked just fine without the module for 2.2.x (1.0.3-2.2.18 kernel).

11.4.3 VS-NAT, 2.4.x director

For 2.4.x you can connect with ftp without any extra modules, but you can't "ls" the contents of the ftp directory. For that you need to load the ip_vs_ftp module. Without this module, your client's screen won't lock up, it just does nothing. If you then load the module, you can list the contents of the directory.

11.4.4 VS-DR, LVS-Tun

For LVS-DR, LVS-Tun active ftp needs persistence. Otherwise it does not work, with or without ip_masq_ftp loaded. You can login, but attempting to do a 'ls' will lockup the client screen. Checking the realserver, shows connections on ports 20,21 to paired ports on the client.

11.5 ftp (passive)

Passive ftp is used by netscape to get files from an ftp url like ftp://ftp.domain.com/pub/ . Here's an explanation of passive ftp from http://www.tm.net.my/learning/technotes/960513-36.html

> If you can't open connections from Netscape Navigator through a firewall to ftp servers outside your site, then try configuring the firewall to allow outgoing connections on high-numbered ports. Usually, ftp'ing involves opening a connection to an ftp server and then accepting a connection from the ftp server back to your computer on a randomly-chosen high-numbered telnet port. the connection from your computer is called the "control" connection, and the one from the ftp server is known as the "data" connection. All commands you send and the ftp server's responses to those commands will go over the control connection, but any data sent back (such as "ls" directory lists or actual file data in either direction) will go over the data connection.
>
> However, this approach usually doesn't work through a firewall, which typically doesn't let any connections come in at all; In this case you might see your ftp connection appear to work, but then as soon as you do an "ls" or a "dir" or a "get", the connection will appear to hang.
>
> Netscape Navigator uses a different method, known as "PASV" ("passive ftp"), to retrieve files from an ftp site. This means it opens a control connection to the ftp server, tells the ftp server to expect a control connection to the ftp server, tells the ftp server to expect a second connection, then opens the data connection to the ftp server itself on a randomly-chosen high-numbered port. This works with most firewalls, unless your firewall retricts outgoing connections on high-numbered ports too, in which case you're out of luck (and you should tell your sysadmins about this).
>
> "Passive FTP" is described as part of the ftp protocol specification in RFC 959 ("http://www.cis.ohio-state.edu/htbin/rfc/rfc959.html").

If you are setting up an LVS ftp farm, it is likely that users will retrieve files with a browser and you will need to setup the LVS to handle passive ftp. You will either need 8 (persistence) (also see on the LVS website under documentation; persistence handling in LVS) or 9.5 (fwmark persistent connection for ftp).

For passive ftp, the ftpd sets up a listener on a high port for the data transfer. This problem for LVS is that the IP for the listener is the RIP and not the VIP.

> Wenzhuo Zhang 1 May 2001 I've been using 2.2.19 on my dialup masquerading box for quite some time. It doesn't seem to me that the option is required, whether in PASV or PORT mode. We can actually get ftp to work in NAT mode without using the ip_masq_ftp module. The trick is to tell the real ftp servers to use the VIP as the passive address for connections from outside; e.g. in wu-ftpd, add the following lines to the /etc/ftpaccess:
>
> ```
> passive address RIP <localnet>
> passive address 127.0.0.1 127.0.0.0/8
> passive address VIP 0.0.0.0/0
> ```
>
> Of course, the ftp virtual service has to be persistent port 0.

> Alois Treindl, 3 May 2001 I found (with kernel 2.2.19) that I needed the command

> ```
> modprobe ip_masq_ftp in_ports=21
> ```
>
> so that (passive mode) ftp from Netscape would work. without the in_ports=21 it did not work.

Julian Anastasov `ja@ssi.bg` 03 May 2001

Yes, it seems this option is not useful for the active FTP transfers because if the data connection is not created while the client's PORT command is detected in the command stream then it is created later when the internal real server creates normal in->out connection to the client. So, it is not a fatal problem for active FTP to avoid this option. The only problem is that these two connections are independent and the command connection can die before the data connection, for long transfers. With the in_ports option used this can not happen.

The fatal problems come for the passive transfers when the data connection from the client must hit the LVS service. For this, the ip_masq_ftp module must detect the 227 response from the real server in the in->out packets and to open a hole for the client's data connection. And the "good" news is that this works only with in_ports/in_mark options used.

> Alois I am using proftpd as ftp server, which does not seem to have on option so that I could configure on the server that it gives the VIP to clients making a PASV request; it always gives the realserver IP address in replies to such requests.

Bad ftpd :) It seems the follwing rules are valid:

- active ftp always works through stupid balancers (for external clients) that have minimum support for masquerading, with some drops in the command connection
- passive ftp always works through stupid masq boxes (for internal clients). The passive ftp setup is useful because the data connection can be marked as a slave to the command connection and in this way avoid connection reconnects.

11.5.1 passive ftp client/server miss-match with LVS-NAT

Julian

The Netfilter guys use another approach when detecting the 227 message in Linux 2.4, i.e. they try to ignore the message and to use only the code (I'm not sure what is the final status of this handling there). But in Linux 2.2 the word "Entering" may be a requirement :(You have to select another FTPd, IMO.

> Jeremy Kusnetz `JKusnetz@nrtc.org` 24 May 2001 It was my ftp server. When going into passive mode it said:
>
> ```
> Passive mode on (x,x,x,x,x,x)
> ```
>
> instead of:
>
> ```
> Entering Passive Mode (x,x,x,x,x,x)
> ```

11.6 ftp is difficult to secure

Roberto Nibali `ratz@tac.ch` 06 May 2001

If you are trying to secure the LVS using the LVS as a packetfilter, will have no big success in doing it for the ftp protocol, because it is so open. You can do a lot to minimize full breaches. At least put the ftp daemon in a chroot environment.

We have multiple choices if we want to narrow down the input ipchains rules on the front interface of director

- Use ftp via LVS. (this is not a solution actually, we still need special input rules on the EXT_IF for 1024:65535)
- Use ftp without LVS but with SNAT. (difficult to setup)
- Use 11.7 (SuSE ftp proxy suite).
- Use 2.4 kernel and ip_conntrack_ftp (don't know much about this, ask Rusty)
- Don't use ftp at all (this is what we want)
- The *ftpfs project* `<http://ftpfs.sourceforge.net/>` I haven't fully tested it and it's a very dangerous approach but it is worth to a look.

The biggest problem is with the ip_masq_ftp module. It should create an ip_fw entry in the masq_table for the PORT port. It doesn't do this and we have to open the whole port range. For PASV we have to DNAT the range.

```
ipchains -A forward -i $EXT_IF -s $INTERNAL_NET $UNPRIV_PORTS -d $DEP -j MASQ
```

FTP is made up of two connections, the Control- and the Data- Connection.

- ftp Control Connection

  The Client contacts the Servers port 21 from an UNPRIV Port. No trouble, standard, plain, vanilla TCP-Connection, we all love it. Over this connection the client sends commands to the server. We will see examples later.

- FTP Data Connection "Data" can be either the content of a file (sent as e.g. the result of a "get" or "put" command) or the content of a directory-listing (i.e. the result of a "ls" or "dir" command).

  The data connection is where the trouble starts. To transfert data, a second connection is opened.

  Usually the client opens this second connection to the server. But for active ftp, the server opens this second connection, using the well-known port 20 (called ftp-data) as sourceport. But which port on the client should he connect to? The client announces the port via a "port"-command over the control connection. This is nasty: Ports are negotiated on application-level where L4 switches like LVS can see what's going on.

  For passive ftp, the server announces the port the client should connect to in its reply to the client's "pasv"-command (this command starts passive FTP, active is the default). The client then opens the data-connection to the server. The port that the server listens on is an unprivileged port (rather than a privileged port as is normal for internet services). A passive ftp transfer then requires that connections be allowed between all 63000 unprivileged ports on both the client and realservers rather than just one. A passive ftp server is difficult to secure with packet filter rules.

If we have to protect a client, we would like to only allow passive ftp, because then we do not have to allow incoming connections. If we have to protect a server, we would like to only allow active ftp, because then we only have to allow the incoming control-connection. This is a deadlock.

Example ftp sessions with *netcat* <http://l0pht.com/~weld/netcat/nc110.tgz>. (Note this URL now refers through to "atstake.com" which has the NT binary of netcat, but not the unix source. If you know where it is now or want the original code, let me know.)

We need 2 xterms (x1, x2), netcat and an ftp-server (here "zar" 172.23.2.30).

First passive mode (because it is conceptionally easier)

```
#x1: Open the control-connection to the server,
#and sent the command "pasv" to the server.
$ netcat zar 21
220 zar.terreactive.ch FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.16) ready.
user ftp
331 Guest login ok, send your complete e-mail address as password.
pass ftp
230 Guest login ok, access restrictions apply.
pasv
227 Entering Passive Mode (172,23,2,30,169,29)
```

The server replied with 6 numbers:

- 172,23,2,30 is the IP I have to connect to
- (169*256+29=43293) is the Port

In x2 I open a second connection with a second netcat

```
$ netcat 172.23.2.30 43293
# x2 will now display output from this connection
```

Now in x1 (the control-connection)

```
$ list
list
150 Opening ASCII mode data connection for '/bin/ls'.
226 Transfer complete.
```

and in x2 the listing appears.

Active ftp

I use the same control-connection in x1 as above, but I want the server to open a connection. Therefore I first need a listener. I do it with netcat in x2:

```
$ netcat -l -p 2560
```

Now I tell the server on the control connection to connect (2560=10*256+0)

```
port 172,23,2,8,10,0
200 PORT command successful.
```

```
<verb>
```

```
Now you see, why I used port 2560.
172.23.2.8 is, of course, my own IP-address.
And now, using x1, I ask for a directory-listing
with the list command, and it appears in x2.
For completeness sake, here is the the full in/output.

First the xterm 1:

<verb>
netcat zar 21
220 zar.terreactive.ch FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.16) ready.
user ftp
331 Guest login ok, send your complete e-mail address as password.
pass ftp
230 Guest login ok, access restrictions apply.
pasv
227 Entering Passive Mode (172,23,2,30,169,29)
list
150 Opening ASCII mode data connection for '/bin/ls'.
226 Transfer complete.
port 172,23,2,8,10,0
200 PORT command successful.
list
150 Opening ASCII mode data connection for '/bin/ls'.
226 Transfer complete.
quit
221 Goodbye.
```

xterm 2:

```
netcat 172.23.2.30 43293
total 7
dr-x--x--x   2 root     root         1024 Jul 26  2000 bin
drwxr-xr-x   2 root     root         1024 Jul 26  2000 dev
dr-x--x--x   2 root     root         1024 Aug 20  2000 etc
drwxr-xr-x   2 root     root         1024 Jul 26  2000 lib
drwxr-xr-x   2 root     root         1024 Jul 26  2000 msgs
dr-xr-xr-x  11 root     root         1024 Mar 15 14:26 pub
drwxr-xr-x   3 root     root         1024 Mar 11  2000 usr

netcat -l -p 2560
total 7
dr-x--x--x   2 root     root         1024 Jul 26  2000 bin
drwxr-xr-x   2 root     root         1024 Jul 26  2000 dev
dr-x--x--x   2 root     root         1024 Aug 20  2000 etc
drwxr-xr-x   2 root     root         1024 Jul 26  2000 lib
drwxr-xr-x   2 root     root         1024 Jul 26  2000 msgs
dr-xr-xr-x  11 root     root         1024 Mar 15 14:26 pub
drwxr-xr-x   3 root     root         1024 Mar 11  2000 usr
```

11.6.1 mail on securing ftp

> JoeI see that ftp is hard to make secure and your prime recommendation is to have an ftp server isolated from all other machines. Do you recommend that people not use ftp and say instead use http for LVSs that are delivering files? I don't like http for file download. At home (28k phone ppp link) if I do anything else over the line (like load a webpage) while doing a download, the download stalls and doesn't start up again. This is pain as a 10M file takes 2hrs and I have to start again.

Joe Cooper `joe@swelltech.com` 07 May 2001

```
'wget -c http://url' will solve that problem.
```

sftp is now available as part of the openssh packages I believe, but requires clients to have a recent version of openssh – probably not what folks want if they have enough clients to justify an LVS cluster. I don't think LVS really has anything to do with whether someone should use ftp for security reasons or not. Securing ftp is a separate issue from securing LVS.

11.7 evaluation of SuSE ftp proxy

Roberto Nibali `ratz@tac.ch` 08 May 2001

There has been some talks about ftp, security and LVS recently and different opinions appeared. I wasn't aware of the fact the people still heavily use the ftp protocol through a firewall, rather then putting a completely secluded box in a corner. Back here at terreActive we have been fighting with the ftp problem since 4 years already and we have not yet the ultimate solution. As such we also evaluated *the SuSE FTP proxy* `<http://www.suse.de/en/support/proxy_suite/>`

What follows is an evaluation mostly done by one of our coworkers Martin Trampler and me (ratz). We're not yet finished with testing everything and all possible setups (NAT, non-NAT, client behind a firewall, etc.) but the result looks rather good in terms of improving security. A better paper will probably follow, but we're too busy right now and ftp is anyway not allowed in our company policy unless the customer has a special SLA.

11.7.1 Motivated by

- the general problems related to allowing FTP-traffic through a non-stateful packetfilter
- the special problems we encountered with the Firewall-1 software on one of our packetfilter which in the end forced us to deinstall FW1
- the broken Linux FTP-masquerading module

I started in March/01 with a search for a FTP Proxy-software, which could be used as a drop-in on pab1/2-machines to increase the security of the machines (clients and servers) behind the packetfilters. Since it was a requirement that this Software should be able to transparently proxy external clients (i.e. the clients don't realize that there is a proxy inbetween), there was only one package which deserved a closer look: The FTP-Proxy from the SuSE Proxy Suite (which actually consists of nothing but this FTP-Proxy). This Proxy now includes support for transparent proxying mode.

11.7.2 Mode of operation

The Proxy consists of a single binary (ftp-proxy, stripped about 50k). All configuration-options can be set in a single configuration-file which by default is named ftp-proxy.conf and searched for in whatever directory was given with the configure-option –sysconfdir=. If this option was not given it is searched for in /usr/local/proxy-suite/etc/, which shows SuSEs BSD-heritage. Best way is, to give the config-file at runtime with the -f cmdline-option.

It is very useful to compile debugging-support into the binary during evaluation and to run it with the cmdline-option -v 4 for maximum debugging. Debugging output is then appended to /tmp/ftp-proxy.debug. It can be run from (x)inetd or in standalone-mode as daemon. I only evaluated the daemon.

It reads the config-file (which must exist) and binds to some local port (e.g. 3129, which is IANA-unassigned and squid+1). The Packetfilter has to be configured to redirect all packets which come in on port 21 to this port (more later). As soon as it gets a request it handles it by first replying to the client only. After the initial USER <username> command it connects to the server (or, more exactly, to port 21 of the host whose IP was the destination of the redirected package).

The configfile may contain user-specific sections which direct special users to special servers. This feature may be very useful but was not evaluated either. It then continues as an agent between client and server; checking either side's communication for correct syntax, as a good application-proxy should.

As soon as the client prepares a data connection (either by sending a PASV or a PORT command, it acknowledges it and, in case of a requested passive connection, establishes a listener which binds to the server's IP (!!). This came rather as a surprise to me. It actually works and means, that the data connection is transparent as well for the client. The range of port on which it listens is configureable as well as the range of ports it uses for outgoing connection (to either the server or to the client in the active-ftp case).

As soon as the client actually wants to retrieve data, the connection to the server is established and the data is shuffeled around. Since the connection to the server is completely seperate from the client connection, its mode doesn't have to be the one the client requests (also by default it is). The data-connection to the server may also be configured to always be active or passive. Here it is clearly desireable to always use passive mode to avoid opening another listener on the packetfilter.

11.7.3 Evaluation

After initially having some minor problems to compile the proxy (it has to be configured –with-regex) and to get it running (by default it thinks it is started by inetd, i.e. standalone-mode is not the default) it ran without problems and also wrote informative messages into the debugging file. Almost everything can be configured in the configuration file (also not everything is documented unambigously) but in general the quality of the documentation, the logging and the debugging messages seems quite high.

The proxy is, as already mentioned, completely transparent for the client and of course intransparent for the server (i.e. the server sees the connections coming from the client).

First more extensive stress-testing and code-review performed on 04/Apr/01 showed the following irregularities:

- Using active ftp, the data-connection on the server-side often failed. This problem could be circumvented by forcing the Proxy to always use passive mode on the server side (which should be used anyways, see below)
- The control-connection on the server side often could not be established.
- When running with debugging-output enabled, we found strange characters in the debugging as well as in the logging-output.

The failed connections result from portnumbers being reused where they should be increased. I think, this problem would also be found on the client-side of the connection if the stress-test would issue more than 1 data-retrieving command. It may help to undefine the Destination[Min|Max]Port configuration directive to get a port assigned by the system.

The - attack-behaviour vanished after disabling debugging output but may nevertheless being an issue. We found a questionable use of a static char* in a formatting routine.

11.7.4 Integration into a firewall suite

Packetfilter-ruleset First the Packetfilterport on which the proxy listens must be closed. It is well possible to bind the proxy to e.g. localhost, but the ipchains ... -j REDIRECT (see below) only allows the specification of a port, not of port+IP. If the proxy is bound to an IP it doesn't get the packets. It has to be universally bound and therefore its port must be closed.

In the following I use:

- PASV_PORTS: Configuration-directives "Passive[Min|Max]DataPort"; Ports on which Listeners for passive connections (to Clients) may be installed
- SC_PORTS: Configuration-directives "Destination[Min|Max]Port"; Ports which are used for connecting to the server (control-conn., data-conn. if server in passive mode) or on which the proxy listens for the server's data (if server in active mode, not documented).

  For client-connections it is necessary that:

  - In the input-chain (for the IF on which client packets arrive) there must be a redirect for packets from the client to the server, port 21 to the port on which the proxy listens.
  - In the input-Chain (for the IF on which client packets arrive) connections from the client to the server on port (UNPRIV->21) and (UNPRIV->PASV_PORTS) are accepted (as well as the reply-packets in the output-chain). The redirect-rule already acts as accepting rule for the incoming packets for port 21. Furthermore in the output-chain connections from the proxy (but with IP-adress of the server!), port 20, to the client (UNPRIV) must be allowed (as well as the corresponging reply-packets in the input-chain).

  For server-connections it is necessary that:

  - In the output-chain of the server-side IF connections from the proxy (SC_PORTS) to port 21 of the server are allowed (and the reply packets in the input-chain)
  - In the output-chain of the server-side IF connections from the proxy (SC_PORTS) to the server (UNPRIV) are allowed (and the reply packets in the input-chain)

The latter pair of rules covers the case that all data connections from the proxy to the server are passive. Note, that no rules for the forward-chain are necessary at all.

This diagram shows the control-connection.

```
                      +--------------------------+
                      |      |     Proxy      |  |
                      |      |3129____________|  |
+--------+  tcp/21    |-----   ^         |  -----|  tcp/21   +--------+
| Client |----------->|eth0|   |         +->|eth1|---------->| Server |
+--------+ to server  |--------+redirect    -----|           +--------+
                              |Packetfilter             |
                              +-------------------------+
```

The obvious problem is, to formulate a fw-ftp-proxy script which accepts 2 NEs (external client, internal server) as input and does not generate redundant rules. Because the server-side connection is completely independent from the client, its rules must only be added once for each server while the client-side rules (including the redirect) are dependent of both server and client. Probably the best way would be to add a script which only handles the client-side and to add each ftp-server seperately with a "tcp@fw"-rule. Since tcp@fw does not allow specification of source-ports, this rule would then be wider as necessary.

In the client-side script, the portrange used in the proxy-configfile would then have to be hardwired. It would be necessary to verify, that for every server used as target in a client-side script there is at least (or even better exactly one) tcp@fw as described above.

11.7.5 Security considerations

We had a swift look at the code and it looks rather clean and well documented to me. Unfortunately some features are incorrectly documented or not documented at all while some features are already documented but not yet implemented.

As already mentioned, the port on which the proxy listens has to be closed. The servers should only be driven in passive mode, which should be possible for any server. The PASV_PORTS should be restricted to a dozen or so (depending on the load).

For the maintainers of the servers, the major drawback is the proxy's intransparency.

11.7.6 Features not yet evaluated

- User-specific configurations
- Use with LDAP and the TCP-Wrapper library (configure-options –with-ldap and –with-libwrap
- Limiting the set of allowed commands
- Proc-Filesystem Interface (module)
- chroot of forked processes

11.7.7 Conclusion

General Aspects Given the current situation, where we shoot huge holes in the firewall to fully enable (passive) ftp connections to servers located inside, the use of this proxy would greatly increase the security of these systems.

Prior to deployment I think the code should be reviewed more closely (remember that the proxy opens listeners on the PF!) and some more efforts should be undertaken to find a configuration which is as tight as possible by providing the required functionality (cf. the section above).

Extensions It should, in general, be possible to have a second proxy running for inside clients. There we still have the problem, that we have to open the whole UNPRIV-Range for connections coming from sourceport 20. Basically I think, that this problem should be handled differently: Providing the functionality is the business of the server (hence the name) . The FTP-Protocol provides passive mode exactly for this case (firewalled client). So we should in general not allow clients behind our Firewalls/Packetfilters to make active FTP connections.

We found out that it should not be too difficult to enable "bidirectional transparency".

11.8 sshd

surprisingly (considering that it negotiates a secure connection) nothing special either. You do not need persistent port/client connection for this.

`jeremy@xxedgexx.com`

I'm using ipvs to balance ssh connections but for some reason ipvs is only using one real server and persists to use that server until I delete its arp entry from the ipvs machine and remove the virtual loopback on the real server. Also, I noticed that connections do not register which this behavior.

> Wensong do you use the persistent port for your VIP:22? If so, the default timeout of persistent port is 360 seconds, once the ssh session finishes, it takes 360 seconds to expire the persistent session. (In ipvs-0.9.1, you can flexibly set the timeout for the persistent port.) There is no need to use persistent port for ssh service, because the RSA keys are exchanged in each ssh session, and each session is not related.

The director will timeout an idle tcp connection (*e.g.* ssh, telnet) in 15mins, independantly of any settings on the client or server. You will want to 11.30 (change these timeouts).

11.8.1 keys for realservers running sshd

If you install sshd and generate the host keys for the realservers using the default settings, you'll get a working LVS'ed sshd. However you should be aware of what you've done. The default sshd listens to 0.0.0.0 and you will have generated host keys for a machine whose name corresponds to the RIP (and not the VIP). Since the client will be displaying a prompt with the name of the realserver (rather than the name associated with the VIP) this will work just fine. However the client will get a different realserver each connection (which is OK too) and will accumulate keys for each realserver. If instead you want the client to be presented with one virtual machine, you will need each machine to have its hostname being the name associated with the VIP, the sshd will have to listen to the VIP (if LVS-DR, LVS-Tun) and the hostkeys will have to be generated for the name of the VIP.

11.9 telnet

Simple one port service. Use telnet (or 11.6 (netcat)) for initial testing of your LVS. It is a simpler client/service than http (it is not persistent) and a connection shows up as an ActConn in the ipvsadm output.

(Also note the director timeout problem, explained in the 11.8 (ssh) section).

11.10 dns

This is from Ted Pavlic. Two (independant) connections, tcp and udp to port 53 are needed.

(from the IPCHAINS-HOWO) DNS doesn't always use UDP; if the reply from the server exceeds 512 bytes, the client uses a TCP connection to port number 53, to get the data. Usually this is for a zone transfer.

Here is part of an lvs.conf file which has dns on two realservers.

```
#dns, note: need both udp and tcp
#A realserver must be able to determine its own name.
#(log onto machine from console and use nslookup
# to see if it knows who it is)
# and to do DNS on the VIP and name associated with the VIP
#To test a running LVS, on client machine, run nslookup and set server = VIP.
SERVICE=t dns wlc 192.168.1.1 192.168.1.8
SERVICE=u dns wlc 192.168.1.1 192.168.1.8
```

If the LVS is run without mon, then any setup that allows the realservers to resolve names is fine (ie if you can sit at the console of each realserver and run nslookup, you're OK).

If the LVS is run with mon (eg for production), then dns needs to be setup in a way that dns.monitor can tell if the LVS'ed form of dns is working. When dns.monitor tests a realserver for valid dns service, it first asks for the zone serial number from the authoritative (SOA) nameserver of the virtualserver's domain. This is compared with the serialnumber for the zone returned from the realserver. If these match then dns.monitor declares that the realserver's dns is working.

The simplest way of setting up an LVS dns server is for the realservers to be secondaries (writing their secondary zone info to local files, so that you can look at the date and contents of the files) and some other machine (eg the director) to be the authoritative nameserver. Any changes to the authoritative nameserver (say the director) will have to be propagated to the secondaries (here the realservers) (delete the secondary's zone files and HUP named on the realservers). After the HUP, new files will be created on the secondary nameservers (the realservers) with the time of the HUP and with the new serial numbers. If the files on the secondary nameservers are not deleted before the HUP, then they will not be updated till the refresh/expire time in the zonefile and the secondary nameservers will appear to dns.monitor to not be working.

There is no reason to create an LVS to do DNS. DNS has its own cacheing and hierachial method of loadbalancing. However if you have an LVS already running serving http, ftp... then it's simple to throw in dns as well (Ted).

11.11 sendmail/smtp/pop3/qmail

For mail which is being passed through, LVS is a good solution.

If the realserver is the final delivery target, then the mail will arrive randomly at any one of the realservers and write to the different filesystems. This is the many reader/many writer problem that LVS has. Since you probably want your mail to arrive at one place only, the only way of handling this right now is to have the /home directory nfs mounted on all the realservers from a backend fileserver which is not part of the LVS. (an nfs.monitor is in the works.) Each realserver will have to be configured to accept mail for the virtual server DNS name (say lvs.domain.com).

It should be possible to use Coda (http://www.coda.cs.cmu.edu/) to keep /home directories synchronised, or inter-mezzo or gfs all of which look nice, but we haven't done a lot of testing with it (however see 19.20 (Filesystems for realserver content) and the related problem 11.29 (Synchronising realserver content)).

11.11.1 Maintaining user passwds on the realservers

Gabriel Neagoe `Gabriel.Neagoe@snt.ro` for syncing the passwords - IF THE ENVIRONMENT IS SAFE- you could use NIS or rdist

11.11.2 identd (auth) problems with MTAs

You will not be explicitely configuring identd in an LVS. However 17 (identd) is used by sendmail and tcpwrappers and will cause problems. Sendmail can't use identd when running on an LVS (see 17.8 (identd and sendmail)). Running identd as an LVS service doesn't fix this.

To fix, in sendmail.cf file set the value

```
Timeout.ident=0
```

Also see http://www.sendmail.org/faq/section3.html - Why do connections to the smtp port take such a long time?

for *qmail* <http://www.qmail.org>:

> Martin Lichtin lichtin@bivio.com
>
> - If invoked with tcp-env in inetd.conf - use the -R option
> - If spawned using svc and DJ's daemontools packages -
>
> ```
> #/usr/local/bin/tcpserver -u 1002 -g 1001 -c 500 -v 0 smtp /var/qmail/bin/qmail-smtpd
> ```
>
> tcpserver is the recommended method of running qmail, where you use the -R option for tcpserver
>
> > -R: Do not attempt to obtain $TCPREMOTEINFO from the remote host. To avoid loops, you must use this option for servers on TCP ports 53 and 113.

11.11.3 testing an LVS'ed sendmail

Here's an LVS-DR with sendmail listening on the VIP on the realserver. Notice that the reply from the MTA includes the RIP allowing you to identify the realserver.

Connect from the client to VIP:smtp

```
client:~# telnet lvs.cluster.org smtp
 trying 192.168.1.110...
 Connected to lvs.cluster.org
 Escape character is '^]'.
220 lvs.cluster.org ESMTP Sendmail 8.9.1a/8.9.0; Sat 6 Nov 1999 13:16:30 GMT
 HELO client.cluster.org
250 client.cluster.org Hello root@client.cluster.org [192.168.1.12], pleased to meet you
 quit
221 client.cluster.org closing connection
```

check that you can access each realserver in turn (here the realserver with RIP=192.168.1.12 was accessed).

11.11.4 pop3

pop3 - as for smtp. The mail agents must see the same /home file system, so /home should be mounted on all realservers from a single file server.

Using qmail -

Abe Schwartz sloween@hotmail.com Has anyone used LVS to balance POP3 traffic in conjunction with qmail?

Wayne wayne@compute-aid.com 13 Feb 2002

We used LVS to balance POP3 and Qmail without any problem.

11.11.5 Thoughts about sendmail/pop

(another variation on the many reader/many writer problem)

loc@indochinanet.com wrote:

I need this to convince my boss that LVS is the solution for very scalable and high available mail/POP server.

Rob Thomas rob@rpi.net.au This is about the hardest clustering thing you'll ever do. Because of the constant read/write access's you -will- have problems with locking, and file corruption.. The 'best' way to do this is (IMHO):

1. NetCache Filer as the NFS disk server.
2. Several SMTP clients using NFS v3 to the NFS server.
3. Several POP/IMAP clients using NFS v3 to the NFS server.
4. At least one dedicated machine for sending mail out (smarthost)
5. LinuxDirector box in front of 2 and 3 firing requests off

Now, items 1 2 -and- 3 can be replaced by Linux boxes, but, NFS v3 is still in Alpha on linux. I -believe- that NetBSD (FreeBSD? One of them) has a fully functional NFS v3 implementation, so you can use that.

The reason why I emphasize NFSv3 is that it -finally- has 'real' locking support. You -must- have atomic locks to the file server, otherwise you -will- get corruption. And it's not something that'll happen occasionally. Picture this:

```
[client]  --  [ l.d ] -- [external host]
                  |
   [smtp server]-+-[pop3 server]
                  |
              [filesrv]
```

Whilst [client] is reading mail (via [pop3 server]), [external host] sends an email to his mailbox. the pop3 client has a file handle on the mail spool, and suddenly data is appended to this. Now the problem is, the pop3 client has a copy of (what it thinks) is the mail spool in memory, and when the user deletes a file, the mail that's just been received will be deleted, because the pop3 client doesn't know about it.

This is actually rather a simplification, as just about every pop3 client understands this, and will let go of the file handle.. But, the same thing will happen if a message comes in -whilst the pop3d is deleting mail-.

```
                         POP Client    SMTP Client
I want to lock this file <--
I want to lock this file               <--
```

```
You can lock the file    -->
You can lock the file                  -->
Consider it locked       <--
File is locked           -->
Consider it locked                     <--
Ooh, I can't lock it                   -->
```

The issue with NFS v1 and v2 is that whilst it has locking support, it's not atomic. NFS v3 can do this:

```
                         POP Client    SMTP Client
I want to lock this file <--
I want to lock this file               <--
File is locked           -->
Ooh, I can't lock it                   -->
```

That's why you want NFSv3. Plus, it's faster, and it works over TCP, rather than UDP 8-)

This is about the hardest clustering thing you'll ever do.

Stefan Stefanov `sstefanov@orbitel.bg` I think this might be not-so-hardly achieved with CODA and Qmail.

Coda (http://www.coda.cs.cmu.edu) allows "clustering" of file system space. Qmail's (http://www.qmail.org) default mailbox format is Maildir, which is very lock safe format (even on NFS without lockd).

(I haven't implemented this, it's just a suggestion.)

11.12 Mail farms

Peter Mueller `pmueller@sidestep.com` 10 May 2001

what open source mail programs have you guys used for SMTP mail farm with LVS? I'm thinking about Qmail or Sendmail?

Michael Brown `Michael_E_Brown@Dell.com`, Joe and Greg Cope `gjjc@rubberplant.freeserve.co.uk` 10 May 2001 You can do load balancing against multiple mail servers without LVS. Use multiple MX records to load balance, and mailing list management software (Mailman, maybe?). DNS responds with all MX records for a request. The MTA should then choose one at random from the same piority. (A cache DNS will also return all MX records.) You don't get persistent use of one MX record. If the chosen MX record points to a machine that's down, the MTA will choose another MX record.

Wensong

I think that central load balancing is more efficient in resource utilization than randomly picking up servers by clients, basic queuing theory can prove this. For example, if there are two mail servers grouped by multiple DNS MX records, it is quite possible that a mail server of load near to 1 still receiving new connections (QoS is bad here), in the mean while the other mail server just has load 0.1. If the central load balancing can keep the load of two server around 0.7 respectively, the resource utilization and QoS is better than that of the above case. :)

Michael Brown `Michael_E_Brown@Dell.com` 15 May 2001 I agree, but... :-)

> 1. You can configure most mail programs to start refusing connections when load rises above a certain limit. The protocol itself has built-in redundancy and error-recovery. Connections will automatically fail-over to the secondary server when the primary refuses connections. Mail will _automatically_ spool on the sender's side if the server experiences temporary outage.
> 2. Mail service is a special case. The protocol/RFC itself specified application-level load balancing, no extra software required.
> 3. Central load balancer adds complexity/layers that can fail.
>
> I maintain that mail serving (smtp only, pop/imap is another case entirely) is a special case that does not need the extra complexity of LVS. Basic Queuing theory aside, the protocol itself specifies load-balancing, failover, and error-recovery which has been proven with years of real-world use.
>
> LVS is great for protocols that do not have the built-in protocol-level load-balancing and error recovery that SMTP inherently has (HTTP being a great example). All I am saying is use the right tool for the job.

Note this discussion applies to mail which is being forwarded by the MTA. The final target machine has the single-writer, many-reader problem as before (which is fine if it's a single node).

Joe

How would someone like AOL handle the mail farm problem? How do users get to their mail? Does everyone in AOL get their mail off one machine (or replicated copies of it) or is each person directed to one of many smaller machines to get their mail?

> Michael Brown Tough question... AOL has a system of inbound mail relays to receive all their user's mail. Take a look:

```
[mebrown@blap opt]$ nslookup
Default Server:  ausdhcprr501.us.dell.com
Address:  143.166.227.254

> set type=mx
> aol.com
Server:  ausdhcprr501.us.dell.com
Address:  143.166.227.254

aol.com preference = 15, mail exchanger = mailin-03.mx.aol.com
aol.com preference = 15, mail exchanger = mailin-04.mx.aol.com
aol.com preference = 15, mail exchanger = mailin-01.mx.aol.com
aol.com preference = 15, mail exchanger = mailin-02.mx.aol.com
aol.com nameserver = dns-01.ns.aol.com
aol.com nameserver = dns-02.ns.aol.com
mailin-03.mx.aol.com    internet address = 152.163.224.88
mailin-03.mx.aol.com    internet address = 64.12.136.153
mailin-03.mx.aol.com    internet address = 205.188.156.186
mailin-04.mx.aol.com    internet address = 152.163.224.122
mailin-04.mx.aol.com    internet address = 205.188.158.25
mailin-04.mx.aol.com    internet address = 205.188.156.249
mailin-01.mx.aol.com    internet address = 152.163.224.26
mailin-01.mx.aol.com    internet address = 64.12.136.57
```

```
mailin-01.mx.aol.com    internet address = 205.188.156.122
mailin-01.mx.aol.com    internet address = 205.188.157.25
mailin-02.mx.aol.com    internet address = 64.12.136.89
mailin-02.mx.aol.com    internet address = 205.188.156.154
mailin-02.mx.aol.com    internet address = 64.12.136.121
dns-01.ns.aol.com       internet address = 152.163.159.232
dns-02.ns.aol.com       internet address = 205.188.157.232
```

So that is on the recieve side. On the actual user reading their mail side, things are much different. AOL doesn't use normal SMTP mail. They have their own proprietary system, which interfaces to the normal internet SMTP system through gateways. I don't know how AOL does their internal, proprietary stuff, but I would guess it would be massively distributed system.

Basically, you can break down your mail-farm problem into two, possibly three, areas.

```
1) Mail receipt (from the internet)
2) Users reading their mail
3) Mail sending (to the internet)
```

Items 1 and 3 can normally be hosted on the same set of machines, but it is important to realize that these are separate functions, and can be split up, if need be.

For item #1, the listing above showing what AOL does is probably a good example of how to set up a super-high-traffic mail gateway system. I normally prefer to add one more layer of protection on top of this: a super low-priority MX at an offsite location. (example: aol.com preference = 100, mail exchanger = disaster-recovery.offsite.aol.com)

For item #2, that is going to be a site policy, and can be handled many different ways depending on what mail software you use (imap, pop, etc). The good IMAP software has LDAP integration. This means you can separate groups of users onto separate IMAP servers. The mail client then can get the correct server from LDAP and contact it with standard protocols (IMAP/POP/etc).

For item #3, you will solve this differently depending on what software you have for #2. If the client software wants to send mail directly to a smart gateway, you are probably going to DNS round-robin between several hosts. If the client expects it's server (from #2) to handle sending email, then things will be handled differently.

Wenzhuo Zhang `wenzhuo@zhmail.com`

Here's an *article on paralleling mail servers* `<http://lw.itworld.com/linuxworld/lw-1999-09/lw-09-sendmail.html>` by Derek Balling.

Shain Miley 25 May 2001 I am planning on setting up an LVS IMAP cluster. I read some posts that talk about file locking problems with NFS that might cause mailbox corruption. Do you think NFS will do the trick or is there a better (faster, journaling) file system out there that will work in a production environment.

Matthew S. Crocker `matthew@crocker.com` 25 May 2001

NFS will do the trick but you will have locking problems if you use mbox format e-mail. You *must* use MailDir instead of mbox to avoid the locking issues.

You can also use GFS (www.globalfilesystem.org) which has a fault tolerant shared disk solution.

Don Hinshaw `dwh@openrecording.com` I do this. I use Qmail as it stores the email in Maildir format, which uses one file per message as opposed to mbox which keeps all messages in a single file. On a cluster this is an advantage since one server may have a file locked for writing while another is trying to write. Since they are locking two different files it eases the problems with NFS file locking.

Courier also supports Maildir format as I believe does Postfix.

I use Qmail+(many patches) for SMTP, Vpopmail for a single UID mail sandbox (shell accounts my ass, not on this rig), and Courier-Imap. Vpopmail is configured to store userinfo in MySQL and Courier-Imap auths out of Vpopmail's tables.

Joe I've always had the creeps about pop and imap sending clear text passwds. How do you handle passwds?

It's a non-issue on that particular system, which is a webmail server. There is no pop just imapd and it's configured to allow connections only from localhost. The webmail is configured to connect to imapd on localhost. No outside connections allowed. But, this is another reason that I started using Vpopmail. Since it is a mail sandbox that runs under a single UID, email users don't get a shell account, so even if their passwords are sniffed, it only gets the cracker a look into that user's mailbox, nothing more.

At least on our system. If a cracker grabs someone's passwd and then finds that the user uses the same passwd on every account they have, there's not much I can do about that.

On systems where users do have an ftp or shell login, I make sure that their login is not the same as their email login and I also gen random passwords for all such accounts, and disallow the users changing it.

I'm negotiating a commercial contract to host webmail for a company (that you would recognize if I weren't prohibited by NDA from disclosing the name), and if it goes through then I'll gen an SSL cert for that company and auth the webmail via SSL.

You can also support SSL for regular pop or imap clients such as Netscape Messanger or MS Outlook or Outlook Express.

Everything is installed in /var/qmail/* and that /var/qmail/ is an NFS v3 export from a RAID server. All servers connect to a dedicated MySQL server that also stores it's databases on another NFS share from the RAID. Also each server mounts /www from the RAID.

Each realserver runs all services, smtpd, imapd, httpd and dns. I use TWIG as a webmail imap client, which is configured to connect to imapd on localhost (each server does this). Incoming smtp, httpd and dns requests are load balanced, but not imapd, since they are local connections on each server. Each server stores it's logs locally, then they are combined with a cron script and moved to the raid.

It's been working very well in a devel environment for over a year (round- robin dns, not lvs). I've recently begun the project to rebuild the system and scale it up into a commercially viable system, which is quite a task since most of the software packages are at least a year old, and I'll be using a pair of LVS directors instead of the RRDNS.

Users will also be using some sort of webmail (IMP/HORDE) to get their mail when they are off site...other than that standard Eudora/Netscape will be used for retrieval.

I settled on TWIG mainly because of it's vhost support. With Vpopmail, I can execute

```
# /var/qmail/vpopmail/bin/vadddomain somenewdomain.com <postmaster passwd>
```

and add that domain to dns and begin adding users and serving it up. I had to tweak TWIG just a bit to get it to properly deal with the "user@domain" style login that Vpopmail requires, but otherwise it works great. Each vhost domain can have it's own config file, but there is only one copy of TWIG in /www. TWIG uses MySQL, and though it doesn't require it, I also create a seperate database for each vhost domain. IMP's development runs along at about Mach

> 0.00000000004 and I got tired of waiting for them to get a working addressbook. That plus it doesn't vhost all that well. SquirrelMail is very nice, but again not much vhost support. Plus TWIG includes the kitchen sink. Email, contacts, schedule, notes, todo, bookmarks and even USENET (which I don't use), each module can be enabled/disabled in the config file for that domain, and it's got a very complete advanced security module (which I also don't use). It's all PHP and using `mod_gzip` is pretty fast. I tested the APC optimizer for PHP, but every time I made a change to a script I had to reload Apache. Not very handy for a devel system, but it did add some noticable speed increases, until I unloaded it.

The real servers would need access to both the users home directories as well as the /var/mail directory. I am not too familiar with the actual locking problems...I understand the basics but I also hear that NFS V3 was supposed to fix some of the locking issues with V2...I also saw some links to GFS,AFS,etc not too sure how they would work...

> Just a quick note: About a week ago I tried compiling a kernel that had been patched by SGI for XFS. The kernel (2.4.2) compiled fine, but choked once the LVS patches had been applied. Not having a lot of time to play around with it, I simply moved to 2.4.4+lvs 0.9 and decided not to bother with XFS on the director boxes.

also I thought about samba and only found one post from last year where someone was going to try it but there was no more info there.

> Well, there's how I do it. I've tried damned near every combination of GPL software available over about the last 2 years to finally arrive at my current setup. Now if I could just load balance MySQL...

> Greg Cope MySQL connections / data transfere work much faster (20% ish) when on local host - so how about running mysql on each host, which is a select only system, and each localhost uses replication to a mster DB that is used for inserts and updates ?

Ultimately I think I'll have to. After I get done rebuilding the system to use kernel 2.4 and LVS and everything is stabilized, then I'll be looking very hard at just this sort of thing.

Joe, 04 Jun 2001

SMTP servers need access to DNS for reverse name lookup. If they are LVS'ed in a LVS-DR setup, won't this be a problem?

> Matthew S. Crocker `matthew@crocker.com` You only need to make sure you have the proper forward and reverse lookup set. inbound mail to an SMTP server gets load balanced by the LVS but it still sees the orginal from IP of the sender and can do reverse lookups as normal. outbound mail from an SMTP server makes connections from its real IP address which can be NAT'd by a firewall or not. That IP address can also be reverse looked up.

normally the realservers in a LVS-DR setup have private IPs for the RIPs and hence they can't receive replies from calls made to external name servers.

I would also assume that people would write filter rules to only allow packets in and out of the realservers that belong to the services listed in the director's ipvsadm tables.

I take it that your LVS'ed SMTP servers can access external DNS servers, either by NAT through the director, or in the case of LVS-DR by having public IPs and making calls from those IPs to external nameservers via the default gw of the realservers?

We currently have our real servers with public IP addresses.

Bowie Bailey Bowie_Bailey@buc.com You can also do this by NAT through a firewall or router. I am not doing SMTP, but my entire LVS setup (VIPs and all) is private. I give the VIPs a static conduit through the firewall for external access. The realservers can access the internet via NAT, the same as any computer on the network.

11.13 http name and IP-based (with LVS-DR or LVS-Tun)

http with name- and ip-based http is a simple one port service. Your httpd must be listening to the VIP which will be on lo:0 or tunl0:0. The httpd can be listening on the RIP too (on eth0) for mon, but for the LVS you need the httpd listening to the VIP.

Thanks to Doug Bagley doug@deja.com for getting this info on ip and name based http into the HOWTO.

Both ip-based and name-based webserving in an LVS are simple. In ip-based (HTTP/1.0) webserving, the client sends a request to a hostname which resolves to an IP (the VIP on the director). The director sends the request to the httpd on a realserver. The httpd looks up its httpd.conf to determine how to handle the request (e.g. which DOCUMENTROOT).

In named-based (HTTP/1.1) webserving, the client passes the HOST: header to the httpd. The httpd looks up the httpd.conf file and directs the request to the appropriate DOCUMENTROOT. In this case all URL's on the webserver can have the same IP.

The difference between ip- and name-based web support is handled by the httpd running on the realservers. LVS operates at the IP level and has no knowledge of ip- or name-based httpd and has no need to know how the URLs are being handled.

For the definitive word on ip-based and name-based web support see

http://www.apache.org/docs/vhosts/index.html

Here are some excerpts.

The original (HTTP/1.0) form of http was IP-based, ie the httpd accepted a call to an IP:port pair, eg 192.168.1.110:80. In the single server case, the machine name (www.foo.com) resolves to this IP and the httpd listens to calls to this IP. Here's the lines from httpd.conf

```
Listen 192.168.1.110:80
<VirtualHost 192.168.1.110>
        ServerName lvs.mack.net
        DocumentRoot /usr/local/etc/httpd/htdocs
        ServerAdmin root@chuck.mack.net
        ErrorLog logs/error_log
        TransferLog logs/access_log
</VirtualHost>
```

To make an LVS with IP-based httpds, this IP is used as the VIP for the LVS and if you are using LVS-DR/VS-Tun, then you set up multiple realservers, each with the httpd listening to the VIP (ie its own VIP). If you are running an LVS for 2 urls (www.foo.com, www.bar.com), then you have 2 VIPs on the LVS and the httpd on each realserver listens to 2 IPs.

The problem with ip-based virtual hosts is that an IP is needed for each url and ISPs charge for IPs.

Doug Bagley doug@deja.com With HTTP/1.1, a client Name based virtual hosting uses the HTTP/1.1 "Host:" header, which HTTP/1.1 clients send. This allows the server to know what host/domain, the client thinks it is connecting to. A normal HTTP request line only has the request path in it, no hostname, hence the new header. IP-based virtual hosting works for older browsers that use HTTP/1.0 and don't send the "Host:" header, and requires the server to use a separate IP for each virtual domain.

The httpd.conf file then has

```
NameVirtualHost 192.168.1.110

<VirtualHost 192.168.1.110>
ServerName www.foo.com
DocumentRoot /www.foo.com/
..
</VirtualHost 192.168.1.110>

<VirtualHost 192.168.1.110>
ServerName www.bar.com
DocumentRoot /www.bar.com/
..
</VirtualHost 192.168.1.110>
```

DNS for both hostnames resolves to 192.168.1.110 and the httpd determines the hostname to accept the connection from the "Host:" header. Old (HTTP/1.0) browsers will be served the webpages from the first VirtualHost in the httpd.conf.

For LVS again nothing special has to be done. All the hostnames resolve to the VIP and on the realservers, VirtualHost directives are setup as if the machine was a standalone.

Ted Pavlic `pavlic@netwalk.com`.

Note that in 2000, *ARIN* `<http:/www.arin.net/announcements/>` (look for "name based web hosting" announcements, the link changes occasionally) announced that IP based webserving would be phased out in favor of name based webserving for ISPs who have more that 256 hosts. This will only require one IP for each webserver. (There are exceptions, ftp, ssl, frontpage...)

11.13.1 / terminated urls

Noah Roberts wrote: When I use urls like www.myserver.org/directory/ everything works fine. But if I don't have the ending / then my client attempts to find the realserver and ask it, and it uses the hostname that I have in /etc/hosts on the director which is to the internal LAN so it fails badly.

Scott Laird `laird@internap.com` 02 Jul 2001

Assuming that you're using Apache, set the ServerName for the real server to the virtual server name. When a user does a 'GET /some/directory', Apache returns a redirect to 'http://$servername/some/directory/'.

11.14 http with LVS-NAT

Summary: make sure the httpd on the realserver is listening on the RIP not the VIP (this is the opposite of what was needed for LVS-DR or LVS-Tun). (Remember, there is no VIP on the realserver with LVS-NAT).

tc lewis had an (ip-based) non-working http LVS-NAT setup. The VIP was a routable IP, while the realservers were virtual hosts on the non-routable 192.168.1.0/24 network.

> Michael Sparks michael.sparks@mcc.ac.uk What's happening is a consequence of using NAT. Your LVS is accepting packets for the VIP, and re-writing them to either 192.168.123.3 or 192.168.123.2. The packets therefore arrive at those two servers marked for address 192.168.123.2 or 192.168.123.3, not the VIP.
>
> As a result when apache sees this:
>
> ```
> <VirtualHost w1.bungalow.intra>
> ...
> </VirtualHost>
> ```
>
> It notices that the packets are arriving on either 192.168.123.2 or 192.168.123.3 and not w1.bungalow.intra, hence your problem.
>
> Solutions
>
> - If this is the only website being serviced by these two servers, change the config so the default doc root is the one you want.
> - If they're servicing many websites, map a realworld IP to an aliases on the realservers and use that to do the work. IMO this is messy, and could cause you major headaches.
> - Use LVS-DR or LVS-Tun - that way the above config could be used without problems since the VS address is a local address as well. This'd be my choice.

Joe 10 May 2001

It just occured to me that a realserver in a LVS-NAT LVS is listening on the RIP. The client is sending to the VIP. In an HTTP 1.1 or name based httpd, doesn't the server get a request with the URL (which will have the VIP) in the payload of the packet (where an L4 switch doesn't see it)? Won't the server be unhappy about this? This has come up before with name based service like 11.22 (https) and for 11.18.7 (indexing of webpages). Does anyone know how to force an HTTP 1.1 connection (or to check whether the connection was HTTP 1.0 or 1.1) so we can check this?

> Paul Baker pbaker@where2getit.com 10 May 2001 The HTTP 1.1 request (and also 1.0 requests from any modern browser) contain a Host: header which specifies the hostname of the server. As long as the webservers on the realservers are aware that they are serving this hostname. There should be no issue with 1.1 vs 1.0 http requests.

so both virtualHost and servername should be the reverse dns of the VIP?

> Yes. Your Servername should be the reverse dns of the VIP and you need to have a Virtualhost entry for it as well. In the event that you are serving more than one domain on that VIP, then you need to have a VirtualHost entry for each domain as well.

what if instead of the name of the VIP, I surf to the actual IP? There is no device with the VIP on the LVS-NAT realserver. Does there need to be one? Will an entry in /etc/hosts that maps the VIP to the public name do?

> Ilker Gokhan IlkerG@sumerbank.com.tr If you write URL with IP address such as http://123.123.123.123/, the Host: header is filled with this IP address, not hostname. You can see it using any network monitor program (tcpdump).

11.15 httpd normally closes connections

If you look with ipvsadm to see the activity on an LVS serving httpd, you won't see much. A non-persistent httpd on the realserver closes the connection after sending the packets. Here's the output from ipvsadm, immediately after retrieving a gif filled webpage from a 2 realserver LVS.

```
director:# ipvsadm
IP Virtual Server version 0.2.5 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:www rr
  -> bashfull.mack.net:www       Masq    1      2          12
  -> sneezy.mack.net:www         Masq    1      1          11
```

The InActConn are showing the connections that transferred hits that have been closed and are in the FIN state waiting to timeout. You may see "0" in the InActConn column, leading you to think that you are not getting the packets via the LVS.

11.16 Persistence with http; browser opens many connections to httpd

With the first version of the http protocol, HTTP/1.0, a client would request a hit/page from the httpd. After the transfer, the connection was dropped. It is expensive to setup a tcp connection just to transfer a small number of packets, when it is likely that the client will be making several more requests immediately afterwards (*e.g.* if the client downloads a page with references to gif images in it, then after parsing the html page, it will issue requests to fetch the gifs). With HTTP/1.1 persistent connection was possible. The client/server pair negotiate to see if persistent connection is available. The httpd will keep the connection open for a period (KeepAliveTimeout, 15sec usually) after a transfer in case further transfers are requested. The client can drop the connection any time it wants to (*i.e.* when it has got all the hits on a page).

> Alois Treindl `alois@astro.ch` 30 Apr 2001 when I reload a page on the client, the browser makes several http hits on the server for the graphics in the page. These hits are load balanced between the real servers. I presume this is normal for HTTP/1.0 protocol, though I would have expected Netscape 4.77 to use HTTP/1.1 with one connection for all parts of a page.

Joe

Here's the output of ipvsadm after downloading a test page consisting of 80 different gifs (80 lines of <img src="foo.gif">.

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 1.0.7 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs.mack.net:http rr
  -> bashfull.mack.net:http         Route   1      2          0
  -> sneezy.mack.net:http           Route   1      2          0
```

It would appear that the browser has made 4 connections which are left open. The client shows (netstat -an) 4 connections which are ESTABLISHED, while the realservers show 2 connections each in FIN_WAIT2. Presumably each connection was used to transfer an average of 20 requests.

If the client-server pair were using persistent connection, I would expect only one connection to have been used.

> Andreas J. Koenig `andreas.koenig@anima.de` 02 May 2001 Netscape just doesn't use a single connection, and not only Netscape. All major browsers fire mercilessly a whole lot of

> connections at the server. They just don't form a single line, they try to queue up on several ports simultaneously...
>
> ...and that is why you should never set KeepAliveTimeout to 15 unless you want to burn your money. You keep several gates open for a single user who doesn't use them most of the time while you lock others out.

Julian

Hm, I think the browsers fetch the objects by creating 3-4 connections (not sure how many exactly). If there is a KeepAlive option in the httpd.conf you can expect small number of inactive connections after the page download is completed. Without this option the client is forced to create new connections after each object is downloaded and the HTTP connections are not reused.

The browsers reuse the connection but there are more than one connections.

KeepAlive Off can be useful for banner serving but a short KeepAlive period has its advantages in some cases with long rtt where the connection setups costs time and because the modern browsers are limited to the number of connections they open. Of course, the default period can be reduced but its value depends on the served content, whether the client is expected to open many connections for short period or just one.

> Peter Mueller `pmueller@sidestep.com` 01 May 2001 I was searching around on the web and found the following relevant links..
>
> ```
> http://thingy.kcilink.com/modperlguide/performance/KeepAlive.html
> http://httpd.apache.org/docs/keepalive.html -- not that useful
> http://www.apache.gamma.ru/docs/misc/fin_wait_2.html -- old but interesting
> ```

Andreas J. Koenig `andreas.koenig@anima.de` 02 May 2001

> If you have 5 servers with 15 secs KeepAliveTimeout, then you can serve
> 60*60*24*5/15 = 28800 requests per day

Joe

don't you actually have MaxClients=150 servers available and this can be increased to several thousand presumably?

Peter Mueller

I think a factor of 64000 is forgotten here (number of possible reply ports), plus the fact that most http connections do seem to terminate immediately, despite the KeepAlive.

> Andreas (?) Sure, and people do this and buy lots of RAM for them. But many of them servers are just in 'K' state, waiting for more data on these KeepAlive connections. Moreover, they do not compile the status module into their servers and never notice.
>
> Let's rewrite the above formula:
>
> MaxClients / KeepAliveTimeout
>
> denotes the number of requests that can be satisfied if all clients *send* a keepalive header (I think that's "Connection: keepalive") but *do not actually use* the kept-alive line. If they actually use the kept-alive line, you can serve more, of course.
>
> Try this: start apache with the -X flag, so it will not fork children and set the keepalivetimeout to 60. Then load a page from it with Netscape that contains many images. You will notice that many pictures arive quickly and a few pictures arive after a long, long, long, looooong time.
>
> When the browser parses the incoming HTML stream and sees the first IMG tag it will fire off the first IMG request. It will do likewise for the next IMG tag. At some point it will reach

an IMG tag and be able to re-use an open keepalive connection. This is good and does save time. But if a whole second has passed after a keepalive request it becomes very unlikely that this connection will be re-used ever, so 15 seconds is braindead. One or two seconds is OK.

In the above experiment my Netscape loaded 14 images immediately after the HTML page was loaded, but it took about a minute for each of the remaining 4 images which happened to be the first in the HTML stream.

Joe

Here's the output of ipvsadm after downloading the same 80 gif page with the -X option on apache (only one httpd is seen with ps, rather than the 5 I usually have).

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.11 (size=16384)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs.mack.net:http rr
  -> bashfull.mack.net:http         Route   1      1          1
  -> sneezy.mack.net:http           Route   1      0          2
```

The page shows a lot of loading at the status line, then stops, showing 100% of 30k. However the downloaded page is blank. A few seconds later the gifs are displayed. The client shows 4 connections in CLOSE_WAIT and the realservers each show 2 connections in FIN_WAIT2.

Paul J. Baker pbaker@where2getit.com 02 May 2001 The KeepAliveTimeout value is NOT the connection time out. It says how long Apache will keep an active connection open waiting for a new request to come on the SAME connection after it has fulfilled a request. Setting this to 15 seconds does not mean apache cuts all connections after 15 seconds.

I write server load-testing software so I have do quiet a bit of research in the behaviour of each browser. If Netscape hits a page with a lot of images on it, it will usually open about 8 connections. It will use these 8 connections to download things as quickly as it can. If the server cuts each connection after 1 request is fullfilled, then Netscape browser has to keep reconnecting. This costs a lot of time. KeepAlive is a GOOD THING. Netscape does close the connections when it is done with them which will be well before the 15 seconds since the last request expire.

Think of KeepAliveTimeout as being like an Idle Timeout in FTP. Imagine it being set to 15 seconds.

11.17 Dynamically generated images on web pages

On static webpages, all realservers serve identical content. Dynamically generated images are only generated on the webserver that gets the request. The director will send the clients request for that image to any of the realservers and not neccessarily the realserver that generated the images.

Solutions are

- generate the images in a shared directory/filesystem
- use fwmark to setup the LVS.

Both methods are described in the section using fwmark for 9.20 (dynamically generated images).

11.18 http: sanity checks, logs, shutting down, cookies, url_parsing, mod_proxy, indexing programs, htpasswd

11.18.1 sanity checks

1. put something different on each realserver's web page (*e.g.* the string "realserver_1" at the top of the homepage).
2. use rr for your scheduler
3. make sure you're rotating through the different web pages (each one is different) and look at the output of ipvsadm to seen a new connection (probably InActConn)
4. ping the VIP from a machine directly connected to the outside of the director. Then check the MAC address for the VIP with arp -a

11.18.2 Logs

The logs from the various realservers need to be merged. From the postings below, using a common nfs filesystem doesn't work and no-one knows whether this is a locking problem fixable by NFS-v3.0 or not. The way to go seems to be *merglog* `<http://mergelog.sourceforge.net/>`

Emmanuel Anne `emanne@absysteme.fr`

..the problem about the logs. Apparently the best is to have each web server process its log file on a local disk, and then to make stats on both all files for the same period... It can become quite complex to handle, is there not a way to have only one log file for all the servers

> Joe - (this is quite old *i.e.* 2000 or older and hasn't been tested). log to a common nfs mounted disk? I don't know whether you can have httpds running on separate machines writing to the same file. I notice (using truss on Solaris) that apache does write locking on files while it is running. Possibly it write-locks the log files. Normally multiple forked httpds are running. Presumably each of them writes to the log files and presumably each of them locks the log files for writing.

Webstream Technical Support `mmusgrove@webstream.net` 18 May 2001

I've got 1 host and 2 real servers running apache(ver 1.3.12-25). The 2nd server NFS exports a directory called /logs. The 1st acts as a client and mounts that drive. I have apache on the 1st card saving the access_log file for each site into that directory as access1.log. The 2nd server saves it as access2.log in the same directory. Our stats program on another server looks for *.log files in that directory. The problem is that whenever I access a site (basically browse through all the pages of a site), the 2nd card adds the access info into the access2.log file and everything is fine. The 1st card saves it to the access1.log file for a few seconds, then all of a sudden the file size goes down to 0 and its empty.

> Alois Treindl `alois@astro.ch` I am running a similar system, but with Linux 2.4.4 which has NFS version 3, which is supposed to have safe locking. Earlier NFS version are said to have buggy file locking, and as Apache must lock the access_log for each entry, this might be the cause of your problem.
>
> I have chosen not to use a shared access_log between the real servers, i.e. not sharing it via NFS. I share the documents directory and a lot else via NFS between all realservers, but not the logfiles.
>
> I use remote syslog logging to collect all access logs on one server.

1. On server w1, which holds the collective access_log and error_log, I have in /etc/syslog.conf the entry: local0.=info /var/log/httpd/access_log local0.err /var/log/httpd/error_log

2. on all other servers, I have an entry which sends the messages to w1: local0.info @w1 local0.err @w1

3. On all servers, I have in http.conf the entry: CustomLog "|/usr/local/bin/http_logger" common

4. and the utility http_logger, which sends the log messages to w1, contains:

```
#!/usr/bin/perl
# script: logger
use Sys::Syslog;
$SERVER_NAME = 'w1';
$FACILITY = 'local0';
$PRIORITY = 'info';
Sys::Syslog::setlogsock('unix');
openlog ($SERVER_NAME,'ndelay',$FACILITY);
while (<>) {
  chomp;
  syslog($PRIORITY,$_);
}
closelog;
```

5. I also to error_log logging to the same central server. This is even easier, because Apache allows to configure in httpd.conf: ErrorLog syslog:local0

On all realservers, except w1, thse log entries are sent to w1 by the syslog.conf given above.

I think it is superior to using NFS. the access_log entries of course contain additional fields in from of the Apache log lines, which originate from the syslogd daemon.

It is also essential that the realservers are well synchonized, so that the log entries appear in correct timestamp sequence.

I have a shared directory setup and both Real Servers have their own access_log files that are put into that directory (access1.log and access2.log...i do it this way so the Stats server can grab both files and only use 1 license), so i dont think its a file locking issue at all. Each apache server is writing to its own separate access log file, it's just that they happen to be in the same shared directory. How would httpd daemon on server A know to LOCK the access log from server B.

Alois Why do you think it is NOT a file locking problem? On each realserver, you have a lot of httpd daemons running, and to write into the same file without interfering, they will have to use file locking, to get exclusive access. On one each server, you do not have just one httpd daemon, but many forked copies. All these processes on ONE server need to write to the SAME logfile. For this shared write access, they use file locking.

If this files sits on a NFS server, and NFS file locking is buggy (which I only know as rumor, not as experience), then it might well be the cause of your problem.

Why don't you keep your access_log local on each server, and rotate them frequently, to collect them on one server (merge-sorted by date/time), and then use your Stats server on it?

If you use separate log files anyway, I cannot see the need to create them on NFS. Nothing prevents you from rotating them every 6 hours, and you will probably not need more current stats.

So the log files HAVE to be on a local disk or else one may run into such a problem as I am having now?

Alois I don't now. I only have read the NFS file locking before NFS 3.0 is broken. It is not a problem related to LVS. You may want to read http://httpd.apache.org/docs/mod/core.html#lockfile

Thanks but Ive seen that before. Each server saves that lock file to its own local directory.

Anyone have a quick and dirty script to merge-sort by date/time the combined apache logs?

Martin Hierling `mad@cc.fh-lippe.de` try *merglog* `<http://mergelog.sourceforge.net/>`

Alois assuming that all files contain only entries from the same month, I think you can try: sort -m -k 4 file1 file2 file3 ...

Arnaud Brugnon `arnaud.brugnon@24pmteam.com` We successfuly use mergelog (you can find on freshmeat or SourceForge) for merging logs (gz or not) from our cluster nodes. With use a simple perl script for downloading them to a single machine.

Juri Haberland `list-linux.lvs.users@spoiled.org` Jul 13 2001

I'm looking for a script that merges and sorts the access_log files of my three real servers running apache. The logs can be up to 500MB if combined.

Michael Stiller `ms@2scale.net` Jul 13 2001 You want to look at *mod_log_spread* `<http://www.backhand.org/mod_log_spread/>`

Stuart Fox `stuart@fotango.com` cat one log to the end of the other then run

```
sort -t - -k 3 ${WHEREVER}/access.log > new.log
```

then you can run webalizer on it.

Thats what I use, doesnt take more than about 30 seconds. If you can copy the logs from your real servers to another box and run sort there, it seems to be better

Heck, here's the whole(sanitized) script

```
#!/bin/bash

##
## Set constants
##

DATE=‘date "+%d-%b-%Y"‘
YESTERDAY=‘date --date="1 day ago" "+%d-%b-%Y"‘
ROOT="/usr/stats"
SSH="/usr/local/bin/ssh"

## First(1) Remove the tar files left yesterday
find ${ROOT} -name "*.tar.bz2" |xargs -r rm -v

##
## First get the access logs
## Make sure some_account has read-only access to the logs
```

```
su - some_account -c "$SSH some_account@real.server1 \"cat
/usr/local/apache/logs/access.log\" >  ${ROOT}/logs/$DATE.log"
su - some_account -c "$SSH some_account@real.server2 \"cat
/usr/local/apache/logs/access.log\" >> ${ROOT}/logs/$DATE.log"

##
## Second sort the contents in date order
##

sort -t - -k 3 ${ROOT}/logs/$DATE.log > ${ROOT}/logs/access.log

##
## Third run webalizer on the sorted files
## Just set webalizer to dump the files in ${ROOT}/logs

/usr/local/bin/webalizer -c /usr/stats/conf/webalizer.conf

##
## Forth remove all the crud
## You still got the originals on the real servers

find ${ROOT} -name "*.log"|xargs -r rm -v

##
## Fifth tar up all the files for transport to somewhere else

cd ${ROOT}/logs && tar cfI ${DATE}.tar.bz2 *.png *.tab *.html && chown
some_account.some_account ${DATE}.tar.bz2
```

Stuart Fox `stuart@fotango.com` Ok scrub my last post, i just tested mergelog. On a 2 x 400mb log it took 40 seconds, my script did it in 245 seconds.

Juri Haberland `list-linux.lvs.users@spoiled.org`

Ok, thanks to you all very much! That was quick and successful :-)

I tried mergelog, but I had some difficulties to compile it on Solaris 2.7 until I found that I was missing GNU make...

But now: Happy happy, joy joy!

karkoma `abambala@genasys.es` Another posibility... http://www.xach.com/multisort/

Stuart Fox `stuart@fotango.com` mergelog seems to be 33% faster than multisort using exactly the same file

11.18.3 Shutting down http

You need to shut down httpd gracefully, by bringing the weight to 0 and letting connections drop, or you will not be able to bind to port 80 when you restart httpd. If you want to do on the fly modifications to your httpd, and keep all realservers in the same state, you may have problems.

> Thornton Prime thornton@jalan.com 05 Jan 2001 I have been having some problems restarting apache on servers that are using LVS-NAT and was hoping someone had some insight or a workaround.
>
> Basically, when I make a configuration change to my webservers and I try to restart them (either with a complete shutdown or even just a graceful restart), Apache tries to close all the current connections and re-bind to the port. The problem is that invariably it takes several minutes for all the current connections to clear even if I kill apache, and the server won't start as long as any socket is open on port 80, even if it is in a 'CLOSING' state.

Michael E Brown wrote:

Catch-22. I think the proper way to do something like this is to take the affected server out of the LVS table _before_ making any configuration changes to the machine. Wait until all connections are closed, then make your change and restart apache. You should run into less problems this way. After the server has restarted, then add it back into the pool.

> I thought of that, but unfortunately I need to make sure that the servers in the cluster remain in a near identical state, so the reconfiguration time should be minimal.

Julian wrote

Hm, I don't have such problems with Apache. I use the default configuration-time settings, may be with higher process limit only. Are you sure you use the latest 2.2 kernels in the real servers?

> I'm guessing that my problem is that I am using LVS persistent connections, and combined with apache's lingering close this makes it difficult for apache to know the difference between a slow connection and a dead connection when it tries to close down, so the time it takes to clear some of the sockets approaches my LVS persistence time. I haven't tried turning off persistence, and I haven't tried re-compiling apache without lingering-close. This is a production cluster with rather heavy traffic and I don't have a test cluster to play with. In the end rebooting the machine has been faster than waiting for the ports to clear so I can restart apache, but this seems really dumb, and doesn't work well because then my cluster machines have different configuration states.

One reason for your servers to block is a very low value for the client number. You can build apache in this way:

CFLAGS=-DHARD_SERVER_LIMIT=2048 ./configure ...

and then to increase MaxClients (up to the above limit). Try with different values. And don't play too much with the MinSpareServers and MaxSpareServers. Values near the default are preferred. Is your kernel compiled with higher value for the number of processes:

/usr/src/linux/include/linux/tasks.h

> Is there any way anyone knows of to kill the sockets on the webserver other than simply wait for them to clear out or rebooting the machine? (I tried also taking the interface down and bringing it up again ... that didn't work either.) Is there any way to 'reset' the MASQ table on the LVS machine to force a reset?

No way! The masq follows the TCP protocol and it is transparent to the both ends. The expiration timeouts in the LVS/MASQ box are high enough to allow the connection termination to complete. Do you remove the real servers from the LVS configuration before stopping the apaches? This can block the traffic and can delay the shutdown. It seems the fastest way to restart the apache is apachectl graceful, of course, if you don't change anything in apachectl (in the httpd args).

11.18.4 Cookies

see 11.25 (cookie)

11.18.5 URL parsing

unknown

Is there any way to do URL parsing for http requests (ie send cgi-bin requests to one server group, static to another group?)

> John Cronin `jsc3@havoc.gtf.org` 13 Dec 2000
>
> Probably the best way to do this is to do it in the html code itself; make all the cgis hrefs to cgi.yourdomain.com. Similarly, you can make images hrefs to image.yourdomain.com. You then set these up as additional virtual servers, in addition to your www virtual server. That is going to be a lot easier than parsing URLs; this is how they have done it at some of the places I have done consulting for; some of those places were using Extreme Networks load balancers, or Resonate, or something like that, using dozens of Sun and Linux servers, in multiple hosting facilities.

> Horms What you are after is a layer-7 switch, that is something that can inspect HTTP packets and make decisions bassed on that information. You can use squid to do this, there are other options. A post was made to this list about doing this a while back. Try hunting through the archives.
>
> LVS on the other hand is a layer-4 switch, the only information that it has available to it is IP address and port and protocol (TCP/IP or UDP/IP). It cannot inspect the data segment and see even understand that the request is an HTTP request, let alone that the URL requested is /cgi-bin or whatever.
>
> There has been talk of doing this, but to be honest it is a different problem to that which LVS solves and arguably should live in user space rather than kernel space as a _lot_ more proccessing is required.

11.18.6 mod_proxy

Sean, 25 Dec 2000

I need to forward request using the Direct Routing method to a server. However I determine which server to send the request to depending on the file it has requested in the HTTP GET not based on it's load.

> Michael E Brown Use LVS to balance the load among several servers set up to reverse-proxy your realservers, set up the proxy servers to load-balance to realservers based upon content.

> Atif Ghaffar `atif@4unet.net` On the LVS servers you can run apache with mod_proxy compiled in, then redirect traffic with it.
>
> `Example`

```
        ProxyPass /files/downloads/ http://internaldownloadserver/ftp/
        ProxyPass /images/ http://internalimagesserver/images/
```

more on Proxy pass: http://www.linuxfocus.org/English/March2000/article147.html

or you can use mod_rewrite, in that case, your REAL servers should be reachable from the net.

there is also a transparent proxy module for apache http://www.stevek.com/projects/mod_tproxy/

11.18.7 Running indexing programs (eg htdig) on the LVS

(From Ted I think)

Setup -

realservers are node1.foobar.com, node2.foobar.com... nodeN.foobar.com, director has VIP=lvs.foobar.com (all realservers appear as lvs.foobar.com to users).

Problem -

if you run the indexing program on one of the (identical) realservers, the urls of the indexed files will be

http://nodeX.foobar.com/filename

These urls will be unuseable by clients out in internetland since the realservers are not individually accessable by clients.

If instead you run the indexing program from outside the LVS (as a user), you will get the correct urls for the files, but you will have to move/copy your index back to the realservers.

Solution (from Ted Pavlic, edited by Joe).

On the indexing node, if you are using LVS-NAT add a non-arping device (eg lo:0, tunl0, ppp0, slip0 or dummy) with IP=VIP as if you were setting up LVS-DR (or LVS-Tun). With LVS-DR/VS-Tun this device with the VIP is already setup. The VIP is associated in dns with the name lvs.foobar.com. To index, on the indexing node, start indexing from http://lvs.foobar.com and the realserver will index itself giving the URLs appropriate for the user in the index.

Alternately (for LVS-NAT), on the indexing node, add the following line to /etc/hosts

127.0.0.1 localhost lvs.foobar.com

make sure your resolver looks to /etc/hosts before it looks to dns and then run your indexing program. This is a less general solution, since if the name of lvs.foobar.com was changed to lvs.bazbar.com, or if lvs.foobar.com is changed to be a CNAME, then you would have to edit all your hosts files. The solution with the VIP on every machine would be handled by dns.

There is no need to fool with anything unless you are running VS-NAT.

11.18.8 htpasswd with http

Noah Roberts wrote:

If anyone has had success with htpasswords in an LVS cluster please tell me how you did it.

> Thornton Prime `thornton@jalan.com` Fri, 06 Jul 2001 We have HTTP authentication working on dozens of sites through LVS with all sorts of different password storage from old fashioned htpasswd files to LDAP. LVS when working properly is pretty transparent to HTTP between the client and server.

11.19 HTTP 1.0 and 1.1 requests

Joe

Does anyone know how to force an HTTP 1.1 connection?

> Patrick O'Rourke `orourke@missioncriticallinux.com` *httperf* `<ftp://ftp.hpl.hp.com/pub/httperf>` has an 'http-version' flag which will cause it to generate 1.0 or 1.1 requests.

11.20 Squids

A squid realserver for the most part can be regarded as just another httpd realserver. There are some exceptions (see 11.31 (3 tier setups) and 7.4.1 (scheduling squids)).

> Palmer J.D.F `J.D.F.Palmer@swansea.ac.uk` Nov 05, 2001 With the use of IP-Tables etc on the directors can you route various URLs/IPs (such as ones requiring NTLM authentication like FrontPage servers etc) not to go through the caches, but just to be transparently routed to their destination.

Horms

This can only be done if the URLS to be passed directly through can be identified by IP address and/or Port. LVS only understands IP addresses and Ports, weather it is TCP or UDP, and other spurious low level data that can be matched using ipchains/iptables.

In particular LVS does _not_ understand HTML, it cannot differentiate between, for instance http://blah/index.html and http://blah/index.asp. rather you would need to set up something along the lines of http://www.blah/index.html and http://asp.blah/index.asp, and have www.blah and asp.blah resolve to different IP addresses.

Further to this you may want to take a look at http://wwwcache.ja.net/JanetService/PilotService.html

11.21 authd/identd (port 113) and tcpwrappers (tcpd)

You do not explicitely set authd (==identd) as an LVS service. It is used with some services (eg sendmail and services running inside tcpwrappers). authd initiates calls from the realservers to the client. LVS is designed for services which receive connect requests from clients. LVS does not allow authd to work anymore and this must be taken into account when running services that cooperate with authd. The inability of authd to work with LVS is important enough that there is a separate 17 (section on authd).

11.22 https

http is an *IP based* protocol, while https is a *name based* protol.

http: you can test an httpd from the console by configuring it to listen on the RIP of the realserver. Then when you bring up the LVS you can re-configure it to listen on the VIP.

https: requires a certificate with the official (DNS) name of the server as the client sees it (the DNS name of the LVS cluster which is associated with the VIP). The https on the realserver then must be setup as if it had the name of the LVS cluster. To do this, activate the VIP on a device on the realserver (it can be non-arping or arping - make sure there are no other machines with the VIP on the network or disconnect your realserver from the LVS), make sure that the realserver can resolve the DNS name of the LVS to the VIP (by dns or /etc/hosts), setup the certificate and conf file for https and startup the httpd. Check that a netscape client running on the realserver (so that it connects to the realserver's VIP and not to the arping VIP on the director) can connect to https://lvs.clustername.org. Since the certificate is for the URL and not for the IP, you only need 1 certificate for an LVS running LVS-DR serving https.

Do this for all the realservers, then use ipvsadm on the director to forward https requests to each of the RIPs.

The scheduling method for https must be persistent for keys to remain valid.

> When compiling in Apache.. What kind of certificate should I create for a real application with Thawte?

Alexandre Cassen `Alexandre.Cassen@wanadoo.fr`

When you generate your CSR use the CN (Common Name) of the DNS entry of your VIP.

11.23 Named Based Virtual Hosts for https

Dirk Vleugels `dvl@2scale.net` 05 Jul 2001

I want to host several https domains on a single LVS-DR cluster. The setup of http virtual hosts is straightforward, but what about https? The director needs to be known with several VIP's (or it would be impossible to select the correct server certificate).

> Matthew S. Crocker `matthew@crocker.com` SSL certs are labelled with the URL name but the SLL session is established before any HTTP requests. So, you can only have one SSL cert tied to an IP address. If you want to have a single host handle multiple SSL certs you need a seperate IP for each cert. You also need to setup the director to handle all the IP's
>
> named based HTTP DO NOT WORK with SSL because the SSL cert is sent BEFORE the HTTP so the sever won't know what cert to send.

Martin Hierling `mad@cc.fh-lippe.de`

You can´t do Name Based VHosts, because the SSL Stuff is done before HTTP snaps in. So at the Beginning there is only the IP:Port and no www.domain.com. Look at *Why can't I use SSL with name-based/non-IP-based virtual hosts?* `<http://www.modssl.org/docs/2.4/ssl_faq.html>`

(Here reproduced in its entirety)

> The reason is very technical. Actually it's some sort of a chicken and egg problem: The SSL protocol layer stays below the HTTP protocol layer and encapsulates HTTP. When an SSL connection (HTTPS) is established Apache/mod_ssl has to negotiate the SSL protocol parameters with the client. For this mod_ssl has to consult the configuration of the virtual server (for instance it has to look for the cipher suite, the server certificate, etc.). But in order to dispatch to the correct virtual server Apache has to know the Host HTTP header field. For this the HTTP request header has to be read. This cannot be done before the SSL handshake is finished. But the information is already needed at the SSL handshake phase. Bingo!

Dirk

With LVS-NAT this would be no problem (targeting different ports on the RS's). But with direct routing i need different virtual IP's on the RS. The question: will the return traffic use the VIP-IP by default? Otherwise the client will notice the mismatch during the SSL handshake.

> "Matthew S. Crocker" `matthew@crocker.com` Yes, on the real servers you will have multiple dummy interfaces, on for each VIP. Apache will bind itself to each interface. The sockets for the SSL session are also bound to the interface. The machine will send packets from the IP address of the interface the packet leaves the machine on. So, it will work as expected. The clients will see packets from the IP address they connected to.

Julian Anastasov `ja@ssi.bg`

```
NAT:    one RIP for each name/key
DR/TUN: one VIP for each name/key
```

Is this correct?

> James Ogley `james.ogley@pinnacle.co.uk` The realservers also need a VIP for each https URL, as they need to be able to resolve that URL to themself on a unique IP (this can be achieved with /etc/hosts of course)

Joe

Are you saying that https needs its own IP:port rather than just IP?

> Dirk Nope. A unique IP is sufficient. Apache has to decide which csr to use _before_ seeing the 'Host' header in the HTTP request (SSL handshake comes first). A unique port is also sufficient to decide which virtual server is meant though (and via NAT easier to manage imho).

(I interpret Dirk as saying that the IP:port must be unique. Either the IP is unique and the port is the same for all urls, or the IP is common and there is a unique port for each url.)

11.24 Databases

Normal databaseds (eg mysqld, *i.e.* anything but Oracle's parallel database server for several 100k$) running under LVS suffer the same restrictions of single writer/many readers as does any other service (eg smtp) where the user can write to files on the realserver.

Databases running independantly on several realservers have to be kept synchronised for content, just as do webservers. If the database files are read-only as far as the LVS clients are concerned, and the LVS administrator can update each copy of the database on the realservers at regular intervals (eg a script running at 3am) then you can run a copy of the databased on each realserver, reading the files which you are keeping synchronised.

(see 19.20 (Filesystems for realserver content) and 11.29 (Synchronising realserver content)).

Online transaction processing requires that LVS clients be able to write to the database.

If you try to do this by setting up an LVS where each realserver has a databased and its own database files, then writes from a particular user will go to only one of the realservers. The database files on the other realservers will not be updated and subsequent LVS users will be presented with inconsistent copies of the database files.

The Linux Scalable Database project http://lsdproject.sourceforge.net/ is working on code to serialise client writes so that they can be written to all realservers by an intermediate agent. Their code is experimental at the moment, but is a good prospect in the long term for setting up multiple databased and file systems on separate realservers. (Note: May 2001, the 11.24.1 (replication feature of mysql) is functionally equivelant.)

Currently most databased are deployed in a multi-tier setup. The clients are out in internet land; they connect to a web-server which has clients for the database; the web-server database client connects to a single databased. In this arrangement the LVS should balance the webservers/database clients and not balance the database directly.

Production LVS databases, eg the service implemented by Ryan Hulsker `RHulsker@ServiceIntelligence.com` (sample load data at http://www.secretshopnet.com/mrtg/) have the LVS users connect to database clients (perl scripts running under a webpage) on each realserver. These database clients connect to a single databased running on a backend machine that the LVS user can't access. The databased isn't being LVS'ed - instead the user connects to LVS'ed database clients on the realserver(s) which handle intermediate dataprocessing, increasing your throughput.

The approach of having databaseds on each realserver accessing a common filesystem on a back-end server, fails. Tests with mysqld running on each of two realservers working off the same database files mounted from a backend machine, showed that reads were OK, but writes from any realserver either weren't seen by the other mysqld or corrupted the database files. Presumably each mysqld thinks it owns the database files and keeps copies of locks and pointers. If another mysqld is updating the filesystem at the same time then these first set of locks and pointers are invalid. Presumably any setup in which multiple databaseds were writing to one file system (whether NFS'ed, GFS'ed, coda, intermezzo...) would fail for the same reason.

In an early attempt to setup this sort of LVS jake buchholz `jake@execpc.com` setup an LVS'ed mysql database with a webinterface. LVS was to serve http and each realserver to connect to the mysqld running on itself. Jake wanted the mysql service to be lvs'ed as well and for each realserver to be a mysql client. The solution was to have 2 VIPs on the director, one for http and the other for mysqld. Each http realserver makes a mysql request to the myqslVIP. In this case no realserver is allowed to have both a mysqld and an httpd. A single copy of the database is nfs'ed from a fileserver. This works for reads.

11.24.1 mysql replication

MySQL `<http://www.mysql.com>` (and most other databases) supports replication of databases.

> Ted Pavlic `tpavlic@netwalk.com` 23 Mar 2001 When used with LVS, a replicated database is still a single database. The MySQL service is not load balanced. HOWEVER, it is possible to put some of your databases on one server and others on another. Replicate each SET of databases to the OTHER server and only access them from the other server when needed (at an application or at some fail-over level).

> Doug Sisk `sisk@coolpagehosting.com` 9 May 2001 An *article on mysql's built in replication facility* `<http://www.phpbuilder.com/columns/tanoviceanu20000912.php3>`

Michael McConnell `michaelm@eyeball.com`> 13 Sep 2001

Can anyone see a down side or a reason why one could not have two System in a Failover Relationship running MySQL. The Database file would be synchronized between the two

Can anyone see a down side or a reason why one could not have two Systems in a Failover Relationship running MySQL. The Database file would be synchronized between the two systems via a crontab and rsync. Can anyone see a reason why rsync would not work? I've on many occasions copied the entire mysql data directory to another system and start it up without a problem. I recognize that there are potential problems that the rsync might take place while the master is writing and the sync will only have part of a table, but mysql's new table structure is supposed to account for this. If anything, a quick myismfix should resolve these problems.

> Paul Baker `pbaker@where2getit.com`> Why not just use MySQL's built in replication?

There are many fundamental problems with MySQL Replication. MySQL's Replication requires that two systems be setup with identical data sources, activate in a master / slave relationship. If the Master fails all requests can be directed to the Slave. Unfortunately this Slave does not have a Slave, and the only way to give it a slave, is to turn it off, synchronize it's data with another system and then Activate them in a Master/Slave relationship, resulting in serious downtime when databases are in excess of 6 gigs (-:

This is the most important problem, but there are many many more, and I suggest people take a serious look at other options. Currently I use a method of syncing 3 systems using BinLog's.

> Paul Baker `pbaker@where2getit.com>` What is the downtime when you have to run myisamchk against the 6 gig database because rsync ran at exactly the same time as mysql was writting to the database files and now your sync'd image is corrupted?
>
> There is no reason you can not set up the slave as a master in advance from the beginning. You just use the same database image as from the original master.
>
> When the master master goes down, set up a new slave by simple copying the original master image over to the new slave, then point it to the old slave that was already setup to be a master. You wouldn't need to take the original slave down at all to bring up a new one. You would essentially be setting up a replication chain but only with the first 2 links active.

Michael McConnell `michaelm@eyeball.com>`

In the configuration I described using Rsync, the MyISMchk would take place on the slave system, I recognize the time involved would be very large, but this is only the slave. This configuration would be setup so an Rsync between the master and slave takes place every 2 hours, and then the Slave would execute a MyISMchk to ensure the data is ready for action.

I recognize that approximately 2 hours worth of data could be lost, but I would probably us the MySQL BinLogs rotated at 15 minutes interval and stored on the slave to allow this to be manually merged in, and keep the data loss time down to only 15 minutes.

Paul, you said that I could simply copy the Data from the Slave to a new Slave, but you must keep in mind, in order to do this MySQL requires that the Master and Slave data files be IDENTICAL, that means the Master must be turned off, the data copied to the slave, and then both systems activated. Resulting in serious downtime.

> Paul You only have to make a copy of the data one time when you initial set up your Master the first time. As long as it takes to do this is your downtime:
>
> ```
> kill mysqlpid
> tar cvf mysql-snapshot.tar /path/to/mysql/datadir
> /path/to/mysql/bin/safe_mysqld
> ```
>
> Your down time is essentially only how long it takes to copy your 6 gigs of data NOT across a network, but just on the same machine. (which is far less than a myisamchk on the same data) Once that is done, at your leisure you can copy the 6 gigs to the slave while the master is still up and serving requests.
>
> You can then continue to make slave after slave after slave just by copying the original snap shot to each one. The master never has to be taking offline again.

Michael McConnell

You explained that I can kill the MySQL on the Master, tar it up, copy the data to the Slave and activate it as the Slave. Unfortunately this is not how MySQL works. MySQL requires that the Master and Slave be identical data files, *IDENTICAL* that means the Master (tar file) cannot change before the Slave comes online.

> Paul Well I suppose there was an extra step that I left out (because it doesn't affect the amount of downtime). The complete migration steps would be:
>
> 1. Modify my.cnf file to turn on replication of the master server. This is done while the master mysql daemon is still running with the previous config in memory.
> 2. shutdown the mysql daemon via kill.
> 3. tar up the data.

4. start up the mysql daemon. This will activate replication on the master and cause it to start logging all changes for replication since the time of the snapshot in step 3. At this point downtime is only as long as it takes you to do steps 2, 3, and 4.
5. copy the snapshot to a slave server and active replication in the my.cnf on the slave server as both a master and a slave.
6. start up the slave daemon. at this time the slave will connect to the master and catch up to any of the changes that took place since the snapshot.

> So as you see, the data can change on the master before the slave comes online. The data just can't change between when you make the snapshot and when the master daemon comes up configured for replication as a master.

Michael McConnell

Paul you are correct. I've just done this experiment.

```
A(master) -> B(slave)
B(master -> C(slave)
```

A Died. Turn off C's Database, tar it up, replicated the Data to A, Activate A as Slave to C. No data loss, and 0 downtime.

(there appears to have been an offline exchange in here.)

Michael McConnell `michaelm@eyeball.com>`

I've just completed the Rsync deployment method, this works very well. I find this method vastly superiors to both other methods we discussed. Using Rsync allows me to only use 2 HOSTS and I can still provide 100% uptime. In the other method I need 3 systems to provide 100% uptime.

In addition the Rsync method is far easier to maintain and setup.

I do recognize this is not *perfect* I run the Rsync every 20 minutes, and then I run myismchk on the slave system immediately afterwards. I run the MyISMChk to only scan Tables that have changed since the last check. Not all my tables change every 20 minutes. I will be timing operations and lowering this Rsync down to approximately 12 minutes. This method works very effectively for managing a 6 Gig Database that is changing approximately 400 megs of data a day.

Keep in mind, there are no *real time* replication methods available for MySQL. Running with MySQL's building Replication commonly results (at least with a 6 gig / 400 megs changing) in as much as 1 hour of data inconsistency. The only way to get true real time is to use a shared storage array.

> Paul Baker MySQL builtin replication is supposed to be "realtime" AFAIK. It should only fall behind when the slave is doing selects against a database that causes changes to wait until the selects are complete. Unless you have a select that is taking an hour, there is no reason it should fall that far behind. Have you discussed your findings with the MySQL developers?

Michael McConnell

I do not see MySQL making any claims to Real-time. There are many situations where a high load will result in systems getting backed up, especially if your Slave system performs other operations.

MySQL's built-in replication functions like so;

1. Master writes updates to Binary Log
2. Slave checks for Binary Updates

3. Slave Downloads new Bin Updates / Installs

Alexander N. Spitzer`aspitzer@3plex.com`

how often are you running rsync? since this is not realtime replication, you run the risk of losing information if the master dies before the rsync has run (and new data has been added since the last rsync.)

> Don Hinshaw `dwh@openrecording.com`> I have one client where we do this. At this moment (I just checked) their database is 279 megs. An rsync from one box to another across a local 100mbit connections takes 7-10 seconds if we run it at 15 minute intervals. If we run it at 5 minute intervals it takes < 3 seconds. If we run it too often, we get an error of "unexpected EOF in read_timeout".
>
> I don't know what causes that error, and they are very happy with the current situation, so I haven't spent any time to find out why. I assume it has something to do with write caching or filesystem syncing, but that's just a wild guess with nothing to back it up. For all I know, it could be ssh related.
>
> We also do an rsync of their http content and httpd logs, which total approximately 30 gigs. We run this sync hourly, and it takes about 20 minutes.

Benjamin Lee `benjaminlee@consultant.com`>

For what it's worth, I have also been witness to the EOF error. I have also fingered ssh as the culprit.

> John Cronin What kind of CPU load are you seeing? rsync is more CPU intensive than most other replication methods, which is how it gains its bandwidth efficiency. CPU Load? How many files are you syncing up - a whole filesystem, or just a few key files? From you answer, I assume you are not seeing a significant CPU load.

Michael McConnell

I RSync 6 Gigs worth of data, Approximately 50 files (tables). Calculating Checksum's is really a very simple calculation, the cpu used to do this is less than 0% - 1% of a PIII 866. (care of vmstat)

I believe all of these articles you have found are related to RSync Servers that serve in the function one would see a system as a major FTP Server. For example ftp.linuxberg.org or ftp.cdrom.com

11.25 Cookies

Cookies are not a service. Cookies are an application level protocol for maintaining state for a client when using the stateless http/https protocols. Other methods for maintaining state involve passing information to the client in the URL. (This can be done with *e.g. php* `<http://www.php.org/>`.) Cookies are passed between servers and clients which have http, https and/or database services and need to be considered when setting up an LVS.

For the cookie specification see *netscape site* `<http://home.netscape.com/newsref/std/cookie_spec.html>`.

Being a layer 4 switch, LVS doesn't inspect the content of packets and doesn't know what's in them. A cookie is contained in a packet and the packet looks just like any other packet to an LVS.

> Eric Brown wrote: Can LVS in any of its modes be configured to support cookie based persistent sessions?

Horms `horms@vergenet.net` 3 Jan 2001

No. This would require inspection of the TCP data secion, and infact an understanding of HTTP. LVS has access only to the TCP headers.

Roberto Nibali `ratz@tac.ch` 19 Apr 2001

LVS is a Layer4 load balancer and can't do content based (L7) load balancing.

You shouldn't try to solve this problem by changing the TCP Layer to provide a solution which should be handled by the Application Layer. You should never touch/tweak TCP settings out of the boundaries given in the various RFC's and their implementations.

If your application passes a cookie to the client, these are the general approaches:

- buy an L7 load balancer (and don't use LVS).
- Set a very high persistency timeout and hope it is higher than the period a client will wait to come back after he found his credit card, or look at other sites, or have a cup of coffee.

  This is not a good solution.

  - Increased persistency timeout increases the number of concurrent connections possible, which increases the amount of memory required to hold the connection table. A persistency timeout of 30min, with clients connecting at 500 connections/s you would need a memory pool of at least: 30*60*128*500/(1024*1024) = 109 MBytes. With the standard timeout of 300 seconds, you'd only need 109/6 = 18 Mbytes.
  - Long persistency times are incompatible with the DoS defense strategies employeed by 19.12 (secure_tcp).
- Have a 2-tier architecture where you have the application directly on the webserver itself and maybe helped by a database. The problem of the cookies storage is not solved however. You have to deal with the replication problem. Imagine following setup:

```
                       ---->  Web1/App -->
                     /                    \
  Clients  ----> director ->  Web2/App ---> DB Server
                     \                    /
                       ---->  Web3/App -->
```

  Cookies are generated and stored locally on each WebX server. But if you have a persistency timeout of 300s (default LVS setting) and the client had his cup of coffee while entering his visa numbers, he would get to a new server. This new server whould then ask the client to reauthenticate. There are solutions to this *e.g*

  - NFS export a dedicated cookie directory over the back-interfaces. Cookies are quickly distributed among the servers.
  - the application is written to handle cookie replication and propagation between the WebX servers (you have at least 299 seconds time to replicate the cookie on all web servers. This should be enough even for distributing over serial line and do a crosscheck :)

    This does not work (well) for geographically distributed webserver.
- 3-Tier architecture

```
                          -->  Web1 --
                        /               \
Clients  ----> LVS ---->   Web2 ----> Application Server <---> DB Server
                        \               /
                          -->  Web3 -->
```

The cookies are generated by the application server and either stored there or on the database server. If a request comes in, the LVS assigns the request f.e to Web1 and sets the persistency timeout. Web1 does a short message exchange with the application server which generates the sessionID as a cookie and stores it. The webserver sends the cookie back and now we are safe. Again this whole procedure has t_pers_timeout (300 seconds normally) amout of time. Let's assume the client times out (has gone for a cup of coffee). When he comes back normally on a Layer4 load balancer he will be forwarded to a new server, (say Web2). The CGI script on Web2 does the same as happened originally on Web1: it generates a cookie as sessionID. But the application server will tell the script that there is already a cookie for this client and will pass it to Web2. In this way we have unlimited persistency based on cookies but limited persistency for TCP.

Advantages

– set your own persistency timeout values
– TCP state timeout values are not changed.
– table lookup is faster
– it's cheaper than buying an L7 load balancer

Disadvantages:

– more complex setup, more hardware
– you have to write some software

• If a separate database is running on each webserver, use replication to copy the cookie between servers. (You have 300 secs to do this). This was also mentioned by Ted Pavlic in connection with 11.24 (databases).

11.26 r commands; rsh, rcp, and their ssh replacements

An example of using rsh to copy files is in *performance data for single realserver LVS* <http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html> Sect 5.2,

Caution: The matter of rsh came up in a private e-mail exchange. The person had found that rshd, operating as an LVS'ed service, initiated a call (rsh client request) to the rshd running on the LVS client. (See Stevens "Unix Network Programming" Chapter 14, which explains rsh.) This call will come from the RIP rather than the VIP. This will require rsh to be run under LVS-NAT or else the realservers must be able to contact the client directly. Similar requests from the 17 (identd) client and 11.5 (passive ftp) on realservers cause problems for LVS.

11.27 NFS

11.27.1 failover of NFS

It is possible with LVS to export directories from realservers to a client, making an nfs fileserver (see *performance data for single realserver LVS* <http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>), near the end). This is all fine and dandy except that there is no easy way to fail-out the nfs service.

> Joseph Mack One of the problems with running NFS as an LVS'ed service (ie to make an LVS fileserver), that has come up on this mailing list is that a filehandle is generated from disk geometry and file location data. In general then the identical copies of the same file that are on different realservers will have different file handles. When a realserver is failed out (e.g. for maintenance) and the client is swapped over to a new machine (which he is not supposed to be able to detect), he will now have an invalid file handle.
>
> Is our understanding of the matter correct?

Dave Higgen `dhiggen@valinux.com` 14 Nov 2000

In principle. The file handle actually contains a 'dev', indicating the filesystem, the inode number of the file, and a generation number used to avoid confusion if the file is deleted and the inode reused for another file. You could arrange things so that the secondary server has the same FS dev... but there is no guarantee that equivalent files will have the same inode number; (depends on order of file creation etc.) And finally the kicker is that the generation number on any given system will almost certainly be different on equivalent files, since it's created from a random seed.

> If so is it possible to generate a filehandle only on the path/name of the file say?

Well, as I explained, the file handle doesn't contain anything explicitly related to the pathname. (File handles aren't big enough for that; only 32 bytes in NFS2, up to 64 in NFS3.)

Trying to change the way file handles are generated would be a MASSIVE redesign project in the NFS code, I'm afraid... In fact, you would really need some kind of "universal invariant file ID" which would have to be supported by the underlying local filesystem, so it would ramify heavily into other parts of the system too...

NFS just doesn't lend itself to replication of 'live' filesystems in this manner. It was never a design consideration when it was being developed (over 15 years ago, now!)

There HAVE been a number of heroic (and doomed!) efforts to do this kind of thing; for example, Auspex had a project called 'serverguard' a few years ago into which they poured millions in resources... and never got it working properly... :-(

Sorry. Not the answer you were hoping for, I guess...

11.27.2 shared scsi solution for NFS

(from a discussion with Horms at OLS 2001)

It seems that the code which calculates the filehandle in NFS is so entrenched in NFS, that it can't be rewritten to allow disks with the same content (but not neccessarily the same disk geometry) to act as failovers in NFS. The current way around this problem is for a reliable (eg RAID-5) disk to be on a shared scsi line. This way two machines can access the same disk. If one machine fails, then the other can supply the disk content. If the RAID-5 disk fails, then you're dead.

> John Cronin `jsc3@havoc.gtf.org` 08 Aug 2001 You should be able to do it with shared SCSI, in an active-passive failover configuration (one system at a time active, the second system standing by to takeover if the first one fails). Only by using something like GFS could both be active simultaneously, and I am not familiar enough with GFS to comment on how reliable it is. If you get the devices right, you can have a seamless NFS failover. If you don't, you may have to umount and remount the file systems to get rid of stale file handle problems. For SMB, users will have to reconnect in the event of a server failure, but that is still not bad.

11.27.3 detecting failed NFS

> Don Hinshaw Would a simple TCP_CONNECT be the right way to test NFS?

Horms

If you are running NFS over TCP/IP then perhaps, but in my experience NFS deployments typically use NFS over UDP/IP. I'm suspecting a better test would be to issue some rpc calls to the NFS server and look at the response, if any. Something equivalent to what showmount can do might not be a bad start.

> Joe
> how about exporting the directory to the director as well and doing an 'ls' every so often

Malcolm Cowe `malcolm_cowe@agilent.com` 7 Aug 2001

HP's MC/ServiceGuard cluster software monitors NFS using the "rpcinfo" command – it can be quite sensitive to network congestion, but it is probably the best tool for monitoring this kind of service.

The problem with listing an NFS exported directory is that when the service bombs, ls will wait for quite a while – you don't want the director hanging because of an NFS query that's gone awry.

Nathan Patrick `np@sonic.net` 09 Aug 2001

Mon includes a small program which extends what "rpcinfo" can do (and shares some code!) Look for rpc.monitor.c in the mon package, available from kernel.org among other places. In a nutshell, you can check all RPC services or specific RPC services on a remote host to make sure that they respond (via the RPC null procedure). This, of course, implies that the portmapper is up and responding.

From the README.rpc.monitor file:

> This program is a monitor for RPC-based services such as the NFS protocol, NIS, and anything else that is based on the RPC protocol. Some general examples of RPC failures that this program can detect are:
>
> ```
> - missing and malfunctioning RPC daemons (such as mountd and nfsd)
> - systems that are mostly down (responding to ping and maybe
>   accepting TCP connections, but not much else is working)
> - systems that are extremely overloaded (and start timing out simple
>   RPC requests)
> ```
>
> To test services, the monitor queries the portmapper for a listing of RPC programs and then optionally tests programs using the RPC null procedure. At Transmeta, we use:
>
> ```
> "rpc.monitor -a" to monitor Network Appliance filers
> "rpc.monitor -r mountd -r nfs" to monitor Linux and Sun systems
> ```

Michael E Brown `michael_e_brown@dell.com` 08 Aug 2001

how about rpcping?

(Joe - rpcping is at *nocol_xxx.tar.gz* `<ftp://ftp.navya.com/pub/>` for those of us who didn't know rpcping existed.)

11.27.4 NFS locking and persistence

Steven Lang Sep 26, 2001

The primary protocol I am interested in here is NFS. I have the director setup with DR with LC scheduling, no persistence, with UDP connections timing out after 5 seconds. I figured the time it would need to be accessing the same host would be when reading a file, so they are not all in contention for the same file, which seems to cost preformance in GFS. That would all come in a series of accesses. So there is not much need to keep the traffic to the same host beyond 5 seconds.

Horms `horms@vergenet.net>`

> I know this isn't the problem you are asking about, but I think there are some problems with your architecture. I spent far to much of last month delving into NFS - for reasons not related to laad balancing - and here are some of the problems I see with your design. I hope they are useful. As far as I can work out you'll need the persistance to be much longer than 5s. NFS is stateless, but regardless, when a client connects to a server a small ammount of state is established on both sides. The stateless aspect comes into play in that if either side times out, or a reboot is detected then the client will attempt to reconnect to the server. If the client doesn't reconnect, but rather its packets end up on a different server because of load balancing the server won't know anything about the client and nothing good will come of the sitiation. The solution to this is to ensure a client consistently talks to the same server, by setting a really long persistancy.
>
> There is also the small issue of locks. lockd should be running on the server and keeping track of all the locks the client has. If the client has to reconnect, then it assumes all its locks are lost, but in the mean time it assumes everything is consistent. If it isn't accessing the same server (which wouldn't work for the reason given above) then the server won't know about any locks it things it has.
>
> Of course unless you can get the lockd on the different NFS servers to talk to each other you are going to have a problem if different clients connected to different servers want to lock the same file. I think if you want to have any locking you're in trouble.

I actually specifically tested for this. Now it may just be a linux thing, not common to all NFS platforms, but in my tests, when packets were sent to the server other than the one mounted, it happily serves up the requested data without even blinking. So whatever state is being saved (And I do know there is some minimal state) seems unimportant. It actually surprised me how seamlessly it all worked, as I was expecting the non-mounted server to reject the requests or something similar.

> That is quite suprising as the server should maintain state as to what clients have what mounted. There is also the small issue of locks. lockd should be running on the server and keeping track of all the locks the client has. If the client has to reconnect, then it assumes all its locks are lost, but in the mean time it assumes everything is consistent. If it isn't accessing the same server (which wouldn't work for the reason given above) then the server won't know about any locks it things it has.

This could indeed be an issue. Perhaps setting up persistence for locks. But I don't think it is all bad. Of course, I am basing this off several assumptions that I have not verified at the moment. I am assuming that NFS and GFS will Do The Right Thing. I am assuming the NFS lock daemon will coordinate locks with the underlying OS. I am also assuming that GFS will then coordinate the locking with the rest of the servers in the cluster. Now as I understand locking on UNIX, it is only advisory and not enforced by the kernel, the programs are supposed to police themselves. So in that case, as long as it talks to a lock daemon, and keeps talking to the same lock daemon, it should be okay, even if the lock daemon is not on the server it is talking to, right?

> That should be correct, the locks are advisor. As long as the lock daemons are talking to the underlying file system (GFS) then the behaviour should be correct, regarldess of which lock

daemon a client talks to, as long as the client consistently talks to the same lock daemon for a given lock.

Of course, in the case of a failure, I don't know what will happen. I will definately have to look at the whole locking thing in more detail to know if things work right or not, and maybe get the lock daemons to coordinate locking.

Given that lockd currently lives entirely in the kernel that is easier said than done.

11.28 RealNetworks streaming protocols

Jerry Glomph Black `black@real.com` August 25, 2000

RealNetworks' streaming protocols are

- PNM (TCP on port 7070, UDP from server -> player on ports 6970-7170). PNM was the original protocol in version 1 through 5. It's now mostly legacy.
- RTSP (TCP on port 554, similar UDP as above, but often on multiple ports) With the G2 release, we adopted the RTSP delivery standard. The current version, RealPlayer 8 came out about two weeks ago. A free one is available to run on just about any platform in common use today. The Linux versions are great.
- There's also a HTTP/TCP-only fallback mode which is (usually) on port 8080.

The server configuration can be altered to run on any port, but the above numbers are the customary, and almost universally-used ones.

Mark Winter, a network/system engineer in my group wrote up the following detailed recipe on how we do it with LVS:

add IP binding in the G2 server config file

```
<List Name="IPBindings">
     <Var Address_1="<real ip address>"/>
     <Var Address_2="127.0.0.1"/>
     <Var Address_3="<virtual ip address>"/>
</List>

On the LVS side
./ipvsadm -A -u <VIP>:0  -p
./ipvsadm -A -t <VIP>:554  -p
./ipvsadm -A -t <VIP>:7070  -p
./ipvsadm -A -t <VIP>:8080  -p

./ipvsadm -a -u <VIP>:0 -r <REAL IP ADDRESS>
./ipvsadm -a -t <VIP>:554 -r <REAL IP ADDRESS>
./ipvsadm -a -t <VIP>:7070 -r <REAL IP ADDRESS>
./ipvsadm -a -t <VIP>:8080 -r <REAL IP ADDRESS>
```

(Ted)

I just wanted to add that if you use FWMARK, you might be able to make it a little simpler and not have to worry about forwarding EVERY UDP port.

```
# Mark packets with FWMARK1
ipchains -A input -d <VIP>/32 7070 -p tcp -m 1
ipchains -A input -d <VIP>/32 554 -p tcp -m 1
ipchains -A input -d <VIP>/32 8080 -p tcp -m 1
ipchains -A input -d <VIP>/32 6970:7170 -p udp -m 1

# Setup the LVS to listen to FWMARK1
ipvsadm -A -f 1 -p

# Setup the real server
ipvsadm -a -f 1 -r <RIP>
```

Not only is this only six lines rather than eight, but now you've setup a persistent port grouping. You do not have to forward EVERY UDP port, and you're still free to setup non-persistent services (or other persistent services that are persistent based on other ports).

When you want to remove a real server, you now do not have to remove FOUR real servers, you just remove one. Same thing with adding. Plus, if you want to change what's forwarded to each real server, you can do so with ipchains and not bother with taking up and down the LVS. ALSO... if you have an entire network of VIPs, you can setup IPCHAINS rules which will forward the entire network automatically rather than each VIP one by one.

Jerry Glomph Black black@prognet.com 07 Jun 2001

Following is a currently-operational configuration for LVS balancing of a set of 3 RealServers (or Real Servers, in LVS-terminology) It has been running at very high loads (thousands of simultaneous connections) for months, in addition to numerous conventional LVS setups for more familiar web load-balancing at massive loads.

```
#!/bin/sh
# LVS initialization for RealNetworks streaming.
#
# client connects on TCP ports 554 (rtsp) or 7070 (pnm, deprecated)
# data returns to client either as UDP on port-range 6970-7170, or
# via the initial TCP socket, if the client cannot receive the UDP stream.

# written and tested to very high (several thousand simultaneous) client load by
# Mark Winter, network department, RealNetworks
# additional LVS work by Rodney Rutherford & Glen Raynor, internet operations
# with random comments by Jerry Black, former Director of Internet Operations
# supplied with no warranty, support, or sympathy, but it works great for us

# Setup IP Addresses
VIP="publicly-advertised-IP-number.mynet.com"
RIP_1="RealServer-1.mynet.com"
RIP_2="RealServer-2.mynet.com"
RIP_3="RealServer-3.mynet.com"

# Load needed modules
BALANCE="wrr"
# Load LVS fwmark module
/sbin/modprobe ip_masq_mfw
# Load appropriate LVS load-balance algorithm module
```

```
/sbin/modprobe ip_vs_$BALANCE

# Mark packets with FWMARK1
/sbin/ipchains -F
/sbin/ipchains -A input -d ${VIP}/32 7070 -p tcp -m 1
/sbin/ipchains -A input -d ${VIP}/32 554 -p tcp -m 1
/sbin/ipchains -A input -d ${VIP}/32 8080 -p tcp -m 1
/sbin/ipchains -A input -s 0.0.0.0/0 6970:7170 -d ${VIP}/32 -p udp -m 1

# Setup the LVS to listen to FWMARK1
/sbin/ipvsadm -C
/sbin/ipvsadm -A -f 1 -p -s $BALANCE

# Setup the real servers
/sbin/ipvsadm -a -f 1 -r ${RIP_1}
/sbin/ipvsadm -a -f 1 -r ${RIP_2}
/sbin/ipvsadm -a -f 1 -r ${RIP_3}
```

Roberto Nibali `ratz@tac.ch` 08 Jun 2001

there is no fwmark module, and the ip_vs module is loaded by ipvsadm now. Why do you need persistence?

11.29 Synchronising content and backing up realservers.

Realservers should have indentical filesi/content for any particular service (since the client can be connected to any of them). This is not a problem for slowing changing sites (*e.g.* ftp servers), where the changes can be made by hand, but sites serving webpages have to be changed daily or hourly. Some semi-automated method is needed to stage the content in a place where it is reviewed and then moved to the realservers.

For a 11.24 (database) LVS, the changes have to be propagated in seconds. In an e-commerce site you have to either keep the client on the same machine when they transfer from http to https (using persistence), which may be difficult if they do some comparative shopping or have lunch in the middle of filling their shopping cart, or propagate the information 11.25 (<em>e.g.</em> a cookie) to the other realservers.

Here are comments from the mailing list.

Wensong

> If you just have two servers, it might be easy to use rsync to synchronize the backup server, and put the rsync job in the crontab in the primary. See http://rsync.samba.org/ for rsync.
>
> If you have a big cluster, you might be interested in Coda, a fault-tolerant distributed filesystem. See http://www.coda.cs.cmu.edu/ for more information.

Joe

> from comments on the mailing list, Coda now (Aug 2001) seems to be a usable project. I don't know what has happened to the sister project *Intermezzo* `<http://www.inter-mezzo.org>`.

J Saunders 27 Sep 1999

> I plan to start a frequently updated web site (potentially every minute or so).

Baeseman, Cliff `Cliff.Baeseman@greenheck.com`

I use mirror to do this!. I created a ftp point on the director. All nodes run against the director ftp directory and update the local webs. It runs very fast and very solid. upload to a single point and the web propagates across the nodes.

Paul Baker `pbaker@where2getit.com` 23 Jul 2001

PFARS Project on SourceForge I have just finished commiting the latest revisions to the PFARS project CVS up on SourceForge. PFARS prounced 'farce' is the "PFARS For Automatic Replication System."

PFARS is currently used to handle server replication for Where2GetIt.com's LVS cluster. It has been in the production environment for over 2 months so we are pretty confident with the code stability. We decided to open source this program under the GPL to give back to the community that provided us with so many other great FREE tools that we could not run our business without (especially LVS). It is written in Perl and uses rsync over SSH to replicate server file systems. It also natively supports Debian Package replication.

Although the current version number is 0.8.1 it's not quite ready for release yet. It is seriously lacking in documentation and there is no installation procedure yet. Also in the future we would like add support for RPM based linux distros, many more replication stages, and support for restarting server processes when certain files are updated. If anyone would like to contribute to this project in any way do not be afraid to email me directly our join the development mailing list at pfars-devel@lists.sourceforge.net.

Please visit the project page at http://sourceforge.net/projects/pfars/ and check it out. You will need to check it out from CVS as there are no files released yet. Any feedback will be greatly appreciated.

Zachariah Mully wrote:

I am having a debate with one of my software developers about how to most efficiently sync content between realservers in an LVS system. The situation is this... Our content production software that we'll be putting into active use soon will enable our marketing folks to insert the advertising into our newsletters without the tech and launch teams getting involved (this isn't necessarily a good thing, but I'm willing to remain open minded ;). This will require that the images they put into newsletters be synced between all the webservers... The problem though is that the web/app servers running this software are load-balanced so I'll never know which server the images are being copied to.

Obviously loading the images into the database backend and then out to the servers would be one method, but the software guys are convinced that there is a way to do it with rsync. I've looked over the documentation for rsync and I don't see anyway to set up cron jobs on the servers to run an rsync job that will look at the other servers content, compare it and then either upload or download content to that server. Perhaps I am missing an obvious way of doing this, so can anyone give me some advice as to the best way of pursuing this problem?

Bjoern Metzdorf `bm@turtle-entertainment.de` 19 Jul 2001

You have at least 3 possibilities: 1. You let them upload to all RIPs (uploading to each real server) 2. You let them upload to a testserver, and after some checks you use rsync to put the images onto the RIPs. 3. You let them upload to one defined RIP instead of the VIP and rsync from there (no need for a testserver)

Stuart Fox `stuart@fotango.com` 19 Jul 2001 nfs mount one directory for upload and server the images from there. Write a small perl/bash script to monitor both upload directories remotely then rsync the differences when detected.

Don Hinshaw `dwh@openrecording.com` 19 Jul 2001

You can use rsync, rsync over ssh or scp. You can also use partition syncing with a network/distributed filesystem such as Coda or OpenAFS or drbd (drbd is still too experimental for me). Such a setup creates partitions which are mirrored in real-time. I.e., changes to one reflect on them all.

We use a common NFS share on a RAID array. In our particular setup, users connect to a "staging" server and make changes to the content on the RAID. As soon as they do this, the real-servers are immediately serving the changed content. The staging server will accept FTP uploads from authenticated users, but none of the real-servers will accept any FTP uploads. No content is kept locally on the real-servers so they never need be synced, except for config changes like adding a new vhost to Apache.

> `jik@foxinternet.net` 19 Jul 2001 If you put the conf directory on the NFS mount along with htdocs then you only need to edit one file, then ssh to each server and "apachectl graceful"
>
> Don Hinshaw `dwh@openrecording.com` 20 Jul 2001
>
> Um, no. We're serving a lot of: <VirtualHost x.x.x.x> and the IP is different for each machine. In fact the conf files for all the real-servers are stored in an NFS mounted dir. We have a script that manages the separate configs for each real-server.

I'm currently building a cluster for a client that uses a pair of NFS servers which will use OpenAFS to keep synced, then use linux-ha to make sure that one of them is always available. One thing to note about such a system is that the synced partitions are not "backups" of each other. This is really a "meme" (way of thinking about something). The distinction is simply that you cannot rollback changes made to a synced filesystem (because the change is made to them both), whereby with a backup you can rollback. So, if a user deletes a file, you must reload from backup. I mention this because many people that I've talked to think that if you have a synced filesystem, then you have backups.

What I'm wondering is why you would want to do this at all. From your description, your marketing people are creating newsletters with embedded advertising. If they are embedding a call for a banner (called a creative in adspeak) then normally that call would grab the creative/clickthrough link from the ad server not the web servers. For tracking the results of the advertising, this is a superior solution. Any decent ad server will have an interface that the marketing dept. can access without touching the servers at all.

Marc Grimme `grimme@comoonics.atix.de` 20 Jul 2001

Depending on how much data you need to sync, you could think about using a Cluster Filesystem. So that any node in the LVS-Cluster could concurrently access the same physically data. Have a look at http://www.sistina.com/gfs/ . We have a clustered webserver with 3 to 5 nodes with GFS underneath and it works pretty stable. If you are sure on what server has latest data is uploaded to, no problem with rsync. If not I would consider to use a Network - or Cluster Filesystem. That should save a lot of scripting work and is more storage efficient.

`jgomez@autocity.com`

We are using rsync as a server. We are using a function that uploads the contents to the server and sync the uploaded file to the other servers.The list of servers we have to sync is in a file like:

```
192.168.0.X
192.168.0.X
192.168.0.X
```

When a file is uploaded,the server reads this file and make the sync to all the other servers.

11.30 Timeouts for TCP/UDP connections

11.30.1 2.2 kernels

Julian, 28 Jun 00

LVS uses the default timeouts for idle connections set by MASQ for (EST/FIN/UDP) of 15,2,5 mins (set in /usr/src/linux/include/net/ip_masq.h). These values are fine for ftp or http, but if you have people sitting on a LVS telnet connection they won't like the 15min timeout. You can't read these timeout values, but you can set them with ipchains. The format is

```
$ipchains -M -S tcp tcpfin udp
```

example:

```
$ipchains -M -S 36000 0 0
```

sets the timeout for established connections to 10hrs. The value "0" leaves the current timeout unchanged, in this case FIN and UDP.

11.30.2 2.4 kernels

Julian Anastasov `ja@ssi.bg` 31 Aug 2001

The timeout is set by ipvsadm. Although this feature has been around for a while, it didn't work till ipvsadm 1.20 (possibly 1.19) and ipvs-0.9.4 (thanks to Brent Cook for finding this bug).

```
$ipvsadm --set tcp tcpfin udp
```

The default timeout is 15 min for the LVS connections in established state. For any NAT-ed client connections ask iptables.

To set the tcp timeout to 10hrs, while leaving tcpfin and udp timeouts unchanged, do

```
#ipvsadm --set 36000 0 0
```

> Brent Cook `busterb@mail.utexas.edu` 31 Aug 2001 I found the relevant code in the kernel to modify this behavior in 2.4 kernels without using ipchains. I got this info from http://www.cs.princeton.edu/ jns/security/iptables/iptables_conntrack.html In /usr/src/linux/net/ipv4/netfilter/ip_conntrack_proto_tcp.c , change TCP_CONNTRACK_TIME_WAIT to however long you need to wait before a tcp connection timeout.
>
> Does anyone foresee a problem with other tcp connections as a result of this? An regular tcp program will probably close the connection anyway.

```
static unsigned long tcp_timeouts[]
= { 30 MINS,    /*      TCP_CONNTRACK_NONE,     */
    5 DAYS,     /*      TCP_CONNTRACK_ESTABLISHED,      */
    2 MINS,     /*      TCP_CONNTRACK_SYN_SENT, */
    60 SECS,    /*      TCP_CONNTRACK_SYN_RECV, */
    2 MINS,     /*      TCP_CONNTRACK_FIN_WAIT, */
    2 MINS,     /*      TCP_CONNTRACK_TIME_WAIT,        */
    10 SECS,    /*      TCP_CONNTRACK_CLOSE,    */
    60 SECS,    /*      TCP_CONNTRACK_CLOSE_WAIT,       */
    30 SECS,    /*      TCP_CONNTRACK_LAST_ACK, */
    2 MINS,     /*      TCP_CONNTRACK_LISTEN,   */
};
```

In the general case you cannot change the settings at the client. If you have access to the client, you can you can arrange for the client to send keepalive packets often enough to reset the timer above and keep the connection open.

Kyle Sparger ksparger@dialtone.com> 5 Oct 2001

You can address this from the client side by reducing the tcp keepalive transmission intervals. Under Linux, reduce it to 5 minutes:

```
echo 300 > /proc/sys/net/ipv4/tcp_keepalive_time
```

where '300' is the number of seconds. I find this useful in all manner of situations where the OS times out connections.

11.30.3 director with many entries in FIN state

With the FIN timeout being about 1 min (2.2.x kernels), if most of your connections are non-persistent http (only taking 1 sec or so), most of your connections will be in the InActConn state.

Hendrik Thiel, 20 Mar 2001 we are using a lvs in NAT Mode and everything works fine ... Probably, the only Problem seems to be the huge number of (idle) Connection Entries.

ipvsadm shows a lot of inActConn (more than 10000 entries per Realserver) entries. ipchains -M -L -n shows that these connections last 2 minutes. Is it possible to reduce the time to keep the Masquerading Table small? e.g. 10 seconds ...

Joe

For 2.2 kernels, you can use netstat -M instead of ipchains -M -L. For 2.4.x kernels use 'cat /proc/net/ip_conntrack'.

Julian

One entry occupies 128 bytes. 10k entries mean 1.28MB memory. This is not a lot of memory and may not be a problem.

To reduce the number of entries in the ipchains table, you can reduce the timeout values. You can edit the TIME_WAIT, FIN_WAIT values in ip_masq.c, or enable the *secure_tcp strategy* <http://www.linuxvirtualserver.org/docs/defense.html> and alter the proc values there. FIN_WAIT can also be changed with ipchains.

11.31 3-Tier LVSs

(this section left blank for the moment).

12 VS-NAT

12.1 Introduction

(see also *Julian's layer 4 LVS-NAT setup* <http://www.linuxvirtualserver.org/~julian/L4-NAT-HOWTO.txt>).

VS-NAT is based on cisco's LocalDirector.

This method was used for the first LVS. If you want to set up a test LVS, this requires no modification of the realservers and is still probably the simplest setup.

With LVS-NAT, the incoming packets are rewritten by the director to have the destinatation address of one of the realservers and then forwarded to the realserver. The replies from the realserver are sent to the director where they are rewritten have the source address of the VIP.

Unlike the other two methods of forwarding used in an LVS (VS-DR and LVS-Tun) the realserver only needs a functioning tcpip stack (eg a networked printer). I.e. the realserver can have any operatining system and no modifications are made to the configuration of the realservers (except setting their route tables).

12.2 Example 1-NIC, 2 Network LVS-NAT (VIP and RIPs on different network)

Here the client is on the same network as the VIP (in a production LVS, the client will be coming in from an external network via a router). (The director can have 1 or 2 NICs - two NICs will allow higher throughput of packets, since the traffic on the realserver network will be separated from the traffic on the client network).

```
Machine                      IP
client                       CIP=192.168.1.254
director VIP                 VIP=192.168.1.110 (the IP for the LVS)
director internal interface  DIP=10.1.1.1
realserver1                  RIP1=10.1.1.2
realserver2                  RIP2=10.1.1.3
realserver3                  RIP3=10.1.1.4
.
.
realserverN                  RIPn=10.1.1.n+1
dip                          DIP=10.1.1.9 (director interface on the LVS-NAT network)
```

```
                         --------
                        |        |
                        | client |
                        |________|
                        CIP=192.168.1.254
                            |
                         (router)
```

```
                           |
             ----------    |
            |          |   |   VIP=192.168.1.110 (eth0:110)
            | director |---|
            |__________|   |   DIP=10.1.1.9 (eth0:9)
                           |
                           |
          -----------------------------------
          |                |                |
          |                |                |
   RIP1=10.1.1.2      RIP2=10.1.1.3   RIP3=10.1.1.4 (all eth0)
   -------------     -------------    -------------
  |             |   |             |  |             |
  | realserver  |   | realserver  |  | realserver  |
  |_____________|   |_____________|  |_____________|
```

here's the lvs.conf file for this setup

```
LVS_TYPE=VS_NAT
INITIAL_STATE=on
VIP=eth0:110 lvs 255.255.255.0 192.168.1.255
DIP=eth0 dip 192.168.1.0 255.255.255.0 192.168.1.255
DIRECTOR_DEFAULT_GW=client
SERVICE=t telnet rr realserver1:telnet realserver2:telnet realserver3:telnet
SERVER_NET_DEVICE=eth0
SERVER_DEFAULT_GW=dip
#----------end lvs_nat.conf------------------------------------
```

The VIP is the only IP known to the client. The RIPs here are on a different network to the VIP (although with only 1 NIC on the director, the VIP and the RIPs are on the same wire).

In normal NAT, masquerading is the rewriting of packets originating behind the NAT box. With LVS-NAT, the incoming packet (src=CIP,dst=VIP, abbreviated to CIP->VIP) is rewritten by the director (becoming CIP->RIP). The action of the LVS director is called demasquerading. The demasqueraded packet is forwarded to the realserver. The reply packet (RIP->CIP) is generated by the realserver.

12.3 All packets from the realserver to the outside world must go through the director

For LVS-NAT to work

- *all packets from the realservers to the client must go through the director.*

Forgetting to set this up is the single most common cause of failure when setting up a LVS-NAT LVS.

For a 2 NIC director with different networks for the realservers and the clients, it is enough for the default gw of the realservers to be the director. For a 1 NIC, two network setup, in addition, the realservers must only have routes to the director. For a 1 NIC, 1 network setup, ICMP redirects must be turned off on the director (the 10.1 (configure script) does this for you).

In a normal server farm, the default gw of the realserver would be the router to the internet and the packet RIP->CIP would be sent directly to the client. In a LVS-NAT LVS, the default gw of the realservers must be the director. The director masquerades the packet from the realserver (rewrites it to VIP->CIP) and the client receives a rewritten packet with the expected source IP of the VIP.

Note: the packet must be routed via the director, there must be no other path to the client. A packet arriving at the client directly from the realserver will not be seen as a reply to the client's request and the connection will hang. If the director is not the default gw for the realservers, then if you use tcpdump on the director to watch an attempt to telnet from the client to the VIP (run tcpdump with 'tcpdump port telnet'), you will see the request packet (CIP->VIP), the rewritten packet (CIP->RIP) and the reply packet (RIP->CIP). You will not see the rewritten reply packet (VIP->CIP). (Remember if you have a switch on the realserver's network, rather than a hub, then each node only sees the packets to/from it. tcpdump won't see packets to between other nodes on the same network.)

Part of the setup of LVS-NAT then is to make sure that the reply packet goes via the director, where it will be rewritten to have the addresses (VIP->CIP). In some cases (e.g. 12.12 (1 net NS-NAT)) icmp redirects have to be turned off on the director so that the realserver doesn't get a redirect to forward packets directly to the client.

In a production system, a router would prevent a machine on the outside exchanging packets with machines on the RIP network. As well, the realservers will be on a private network (eg 192.168.x.x/24) and replies will not be routable.

In a test setup (no router), these safeguards don't exist. All machines (client, director, realservers) are on the same piece of wire and if routing information is added to the hosts, the client can connect to the realservers independantly of the LVS. This will stop LVS-NAT from working (your connection will hang), or it may appear to work (you'll be connecting directly to the realserver).

In a test setup, traceroute from the realserver to the client should go through the director (2 hops in the above diagram). The 10.1 (configure script) will test that the director's gw is 2 hops from the realserver and that the route to the director's gw is via the director, preventing this error.

(Thanks to James Treleaven `jametrel@enoreo.on.ca` 28 Feb 2002, for clarifying the write up on the ping tests here.)

In a test setup with the client connected directly to the director (in the setup above with 1 or 2 NICs, or the 12.12 (one NIC, one network LVS-NAT) setup), you can ping between the client and realservers. However in production, with the client out on internet land, and the realservers with unroutable IPs, you should not be able to ping between the realservers and the client. The realservers should not know about any other network than their own (here 10.1.1.0). The connection from the realservers to the client is through ipchains (for 2.2.x kernels) and LVS-NAT tables setup by the director.

In my first attempt at LVS-NAT setup, I had all machines on a 192.168.1.0 network and added a 10.1.1.0 private network for the realservers/director, without removing the 192.168.1.0 network on the realservers. All replies from the servers were routed onto the 192.168.1.0 network rather than back through LVS-NAT and the client didn't get any packets back.

Here's the general setup I use for testing. The client (192.168.2.254) connects to the VIP on the director. (The VIP on the realserver is present only for LVS-DR and LVS-Tun.) For LVS-DR, the default gw for the realservers is 192.168.1.254. For LVS-NAT, the default gw for the realservers is 192.168.1.9.

```
         ____________
        |            |192.168.1.254 (eth1)
        |  client    |----------------------
        |____________|                     |
      CIP=192.168.2.254 (eth0)             |
                |                          |
                |                          |
VIP=192.168.2.110 (eth0)              |
   ------------                       |
  |            |                      |
  |  director  |                      |
  |____________|                      |
DIP=192.168.1.9 (eth1, arps)          |
         |                            |
      (switch)------------------------
         |
RIP=192.168.1.2 (eth0)
VIP=192.168.2.110 (for LVS-DR, lo:0, no_arp)
   -------------
  |             |
  | realserver  |
  |_____________|
```

This setup works for both LVS-NAT and LVS-DR.

Here's the routing table for one of the realservers as in the LVS-NAT setup.

```
realserver:# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U        40 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U        40 0          0 lo
0.0.0.0         192.168.1.9     0.0.0.0         UG       40 0          0 eth0
```

Here's a traceroute from the realserver to the client showing 2 hops.

```
traceroute to client2.mack.net (192.168.2.254), 30 hops max, 40 byte packets
 1  director.mack.net (192.168.1.9)  1.089 ms  1.046 ms  0.799 ms
 2  client2.mack.net (192.168.2.254)  1.019 ms  1.16 ms  1.135 ms
```

Note the traceroute from the client box to the realserver only has one hop.

director icmp redirects are on, but the director doesn't issue a redirect (see 19.19 ()) because the packet RIP->CIP from the realserver emerges from a different NIC on the director than it arrived on (and with different source IP). The client machine doesn't send a redirect since it is not forwarding packets, it's the endpoint of the connection.

12.4 Run 10.1 (configure)

Use lvs_nat.conf as a template (sample here will setup LVS-NAT in the diagram above assuming the realservers are already on the network and using the DIP as the default gw).

```
#--------------lvs_nat.conf----------------------
LVS_TYPE=VS_NAT
INITIAL_STATE=on

#director setup:
VIP=eth0:110 192.168.1.110 255.255.255.0 192.168.1.255
```

```
DIP=eth0:10 10.1.1.10 10.1.1.0 255.255.255.0 10.1.1.255

#Services on realservers:
#telnet to 10.1.1.2
SERVICE=t telnet wlc 10.1.1.2:telnet
#http to a 10.1.1.2 (with weight 2) and to high port on 10.1.1.3
SERVICE=t 80 wlc 10.1.1.2:http,2 10.1.1.3:8080 10.1.1.4

#realserver setup (nothing to be done for LVS-NAT)

#----------end lvs_nat.conf------------------------------------
```

The output is a commented rc.lvs_nat file. Run the rc.lvs_nat file on the director and then the realservers (the script knows whether it is running on a director or realserver).

The 10.1 (configure script) will setup up masquerading, forwarding on the director and the default gw for the realservers.

12.5 Setting up demasquerading on the director; 2.4.x and 2.2.x

The packets coming in from the client are being demasqueraded by the director.

In 2.2.x you need to masquerade the replies. Here's the masquerading code in rc.lvs_nat, that runs on the director (produced by 10.1 (configure.pl)).

```
        echo "turning on masquerading "
        #setup masquerading
        echo "1" >/proc/sys/net/ipv4/ip_forward
        echo "installing ipchain rules"
        /sbin/ipchains -A forward -j MASQ -s 10.1.1.2 http -d 0.0.0.0/0
        #repeated for each realserver and service
        ..
        ..
        echo "ipchain rules "
        /sbin/ipchains -L
```

In this example, http is being masqueraded by the director, allowing the realserver to reply to the telnet requests from the director being demasqueraded by the director as part of the 2.2.x LVS code.

In 2.4.x masquerading of LVS'ed services is done explicitely by the LVS code and no extra masquerading (by iptables) commands need be run.

12.6 masquerading clients on realservers

You may want to allow clients on the realservers to connect to servers on the internet. Setting this up is independant of the LVS and the connections from clients on the realservers are unrelated to the functioning of the LVS. VS-NAT performs NAT only for its packets. You must define specific NAT rules for the non-LVS traffic (ipchains/iptables).

example: A client making a telnet request from a realserver, will be doing so from a high (>1024) port to 0.0.0.0:23. If the LVS is also forwarding requests for the same service, the connection is instead between 0.0.0.0:high_port and RIP:23 (VS-NAT) or VIP:23 (VS-DR).

In a normal LVS, connection requests from clients on the realserver will originate at the RIP and be sent through the realserver's default gw (the director in the case of LVS-NAT, a router for LVS-DR) without any masquerading. Since the realservers will usually have private IP's, the packets for the connection requests will not be routable. Instead you will need to NAT out the client requests using the director as the NAT box (and default gw for the client's requests). For LVS-NAT, the director is already the default gw. For LVS-DR, you have to 13.10 (route the packets from clients on the realservers through the director), while packets associated with the LVS, *i.e.* from the VIP on the realserver, is routed through a router.

If you want telnet clients on the realservers to be masqueraded by the director, to the outside world, then on the director you need to run a command like

```
director:/etc/lvs# /sbin/ipchains -p tcp -A forward -j MASQ -s $RIP -d 0.0.0.0 telnet
```

Telnet clients on the realservers will be now masqueraded out by the director, independently of the LVS setup. Masqueraded connections from the director will come from the primary IP on the outside of the director. If the VIP is an alias on the outside of the director (the usual situation), then the masqueraded connection will not come fom the VIP.

Here are the IP:port, seen by 'netstat -an' on each machine, for two cases

- client on the internet telnet'ing to an LVS-NAT
- the realserver connecting by masquerading through the director to the telnetd on the LVS client.

```
client                  director            realserver

connection from client to LVS
CIP:1041->VIP:23          -                 CIP:1041->RIP:23

telnet connection from realserver to telnetd on LVS client
CIP:23<-DIP:61000         -                 CIP:23<-RIP:1030
```

The masqueraded connection to the LVS client comes from the primary IP of the director (here the DIP) and not from the VIP, which in this setup is an alias (secondary IP) of the DIP.

The director doesn't have connections to any of its ports for either connection. It the case of LVS, the director is just forwarding packets. In the case of masquerading, the masqueraded ports can be seen on the director with

```
director:/etc/lvs# ipchains -M -L -n
IP masquerading entries
prot expire   source              destination         ports
TCP  14:53.91 RIP                 CIP                 1030 (61000) -> 23
```

Connections from clients start at high_port=1024. The masqueraded ports start at port=61000 (not 1024). The port number increments for each new connection in both cases. In the case where a machine is both connecting to the outside world (using ports starting at 1024) and masquerading connections from other machines (using port starting at 61000), there is 19.9 (no port collision detection). This can be a problem if the machine is masquerading a large number of connections and the port range has been increased.

12.7 re-mapping ports, rewriting in slow

With LVS-NAT, the ports can be re-mapped. A request to port 80 on the director can be sent to port 8000 on a realserver. This is possible because the source and destination of the packets are already being rewritten and no extra overhead is required to rewrite the port numbers. The rewriting is slow (60usec/packet on a pentium classic) and limits the throughput of LVS-NAT (for 536byte packets, this is 72Mbit/sec or about 100BaseT). While LVS-NAT throughput does not scale well with the number of realservers, the advantage of VS-NAT is that realservers can have any OS, no modifications are needed to the realserver to run it in an LVS, and the realserver can have services not found on Linux boxes.

12.8 masquerade timeouts

For the earlier versions of LVS-NAT (with 2.0.36 kernels) the timeouts were set by linux/include/net/ip_masq.h, the default values of masquerading timeouts are:

```
        #define MASQUERADE_EXPIRE_TCP 15*16*Hz
        #define MASQUERADE_EXPIRE_TCP_FIN 2*16*Hz
        #define MASQUERADE_EXPIRE_UDP 5*16*Hz
```

12.9 Julian's step-by-step check of a L4 LVS-NAT setup

Julian has his latest fool-proof setup doc at *http://www.linuxvirtualserver.org/~julian/L4-NAT-HOWTO.txt* `<http://www.linuxvirtualserver.org/~julian/L4-NAT-HOWTO.txt>`. I will try to keep the copy here as up-to-date as possible

```
Q.1 Can the real server ping client?

        rs# ping -n client

A.1 Yes => good
A.2 No => bad

        Some settings for the director:

        Linux 2.2/2.4:
        ipchains -A forward -s RIP -j MASQ

        Linux 2.4:
        iptables -t nat -A POSTROUTING -s RIP -j MASQUERADE

Q.2 Traceroute to client goes through LVS box and reaches the client?

        traceroute -n -s RIP CLIENT_IP

A.1 Yes => good
A.2 No => bad

        same ipchains command as in Q.1

        For client and server on same physical media use these
        in the director:

        echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
        echo 0 > /proc/sys/net/ipv4/conf/<DEV>/send_redirects


Q.3 Is the traffic forwarded from the LVS box, in both directions?

        For all interfaces on director:
        tcpdump -ln host CLIENT_IP

        The right sequence, i.e. the IP addresses and ports on each
        step (the reversed for the in->out direction are not shown):

        CLIENT
           | CIP:CPORT -> VIP:VPORT
           |             ||
           |             \/
 out       | CIP:CPORT -> VIP:VPORT
 ||     LVS box
 \/        | CIP:CPORT -> RIP:RPORT
 in        |             ||
           |             \/
           | CIP:CPORT -> RIP:RPORT
           +
        REAL SERVER

A.1 Yes, in both directions => good (for Layer 4, probably not for L7)
A.2 The packets from the real server are dropped => bad:

        - rp_filter protection on the incoming interface, probably
        hit from local client
        - firewall rules drop the replies

A.3 The packets from the real servers leave the director unchanged

        - missing -j MASQ ipchains rule in the LVS box

        For client and server on same physical media:

        The packets simply does not reach the director. The real
        server is ICMP redirected to the client. In director:

        echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
        echo 0 > /proc/sys/net/ipv4/conf/<DEV>/send_redirects

A.4 All packets from the client are dropped

        - the requests are received on wrong interface with rp_filter
        protection
        - firewall rules drop the requests

A.5 The client connections are refused or are served from service
in the LVS box

        - client and LVS are on same host => not valid
        - the packets are not marked from the firewall and don't hit
        firewall mark based virtual service

Q.4 Is the traffic replied from the real server?

        For the outgoing interface on real server:

        tcpdump -ln host CLIENT_IP

A.1 Yes, SYN+ACK => good
A.2 TCP RST => bad, No listening real service
A.3 ICMP message => bad, Blocked from Firewall/No listening service
A.4 The same request packet leaves the real server => missing accept
        rules or RIP is not defined
A.5 No reply => real server problem:

        - the rp_filter protection drops the packets
        - the firewall drops the request packets
        - the firewall drops the replies

A.6 Replies goes through another device or don't go to the LVS
box =? bad

        - the route to the client is direct and so don't pass the LVS
        box, for example:

                - client on the LAN
                - client and real server on same host

        - wrong route to the LVS box is used => use another

        Check the route:

        rs# ip route get CLIENT_IP from RIP


The result: start the following tests

rs# tcpdump -ln host CIP
rs# traceroute -n -s RIP CIP
lvs# tcpdump -ln host CIP
client# tcpdump -ln host CIP
```

```
For more deep problems use tcpdump -len, i.e. sometimes the link layer
addresses help a bit.


For FTP:

        VS-NAT in Linux 2.2 requires:

        - modprobe ip_masq_ftp (before 2.2.19)
        - modprobe ip_masq_ftp in_ports=21 (2.2.19+)

        VS-NAT in Linux 2.4 requires:

        - ip_vs_ftp

        VS-DR/TUN require persistent flag


        FTP reports with debug mode enabled are useful:

        # ftp
        ftp> debug
        ftp> open my.virtual.ftp.service
        ftp> ...
        ftp> dir
        ftp> passive
        ftp> dir

        There are reports that sometimes the status strings reported
        from the FTP real servers are not matched with the string
        constants encoded in the kernel FTP support. For example,
        Linux 2.2.19 matches
        "227 Entering Passive Mode (xxx,xxx,xxx,xxx,ppp,ppp)"


Julian Anastasov
```

12.10 How LVS-NAT works

ipvsadm does the following

```
        #setup connection for telnet, using round robin
        /sbin/ipvsadm -A -t 192.168.1.110:23 -s rr
        #connections to x.x.x.110:telnet are sent to
        #                 realserver 10.1.1.2:telnet
        #using LVS-NAT (the -m) with weight 1
        /sbin/ipvsadm -a -t 192.168.1.110:23 -r 10.1.1.2:23 -m -w 1
        #and to realserver 10.1.1.3
        #using LVS-NAT with weight 2
        /sbin/ipvsadm -a -t 192.168.1.110:23 -r 10.1.1.3:23 -m -w 2
```

(if the service was http, the webserver on the realhost could be listening on port 8000 instead of 80)

Example: client requests a connection to 192.168.1.110:23

director chooses real server 10.1.1.2:23, updates connection tables, then

```
packet                 source                         dest
incoming               CIP:3456                       VIP:23
inbound rewriting      CIP:3456                       RIP1:23
reply (routed to DIP)  RIP1:23                        CIP:3456
outbound rewriting     VIP:23                         CIP:3456
```

The client gets back a packet with the source_address = VIP.

For the verbally oriented...

The request packet is sent to the VIP. The director looks up its tables and sends the connection to realserver1. The packet is rewritten with a new destination (in this case with the same port, but the port could be changed too) and sent to RIP1. The realserver replies, sending back a packet to the client. The default gw for the realserver is the director. The director accepts the packet and rewrites the packet to have source=VIP and sends the rewritten packet to the client.

Why isn't the source of the incoming packet rewritten to be the DIP or VIP?

> Wensong ...changing the source of the packet to the VIP sounds good too, it doesn't require that default route rule, but requires additional code to handle it.

12.11 In LVS-NAT, how do packets get back to the client, or how does the director choose the VIP as the source_address for the outgoing packets?

Joe

In normal NAT, where a bunch of machines are sitting behind a NAT box, all outward going packets are given the IP on the outside of the NAT box. What if there are several IPs facing the outside world? For NAT it doesn't really matter as long as the same IP is used for all packets. The default value is usually the first interface address (eg eth0). With LVS-NAT you want the outgoing packets to have the source of the VIP (probably on eth0:1) rather than the IP on the main device on the director (eth0).

With a single realserver LVS-NAT LVS serving telnet, the incoming packet does this ,

```
CIP:high_port -> VIP:telnet     #client sends a packet
CIP:high_port -> RIP:telnet     #director demasquerades packet, forwards to realserver
RIP:telnet    -> CIP:high_port  #realserver replies
```

The reply arrives on the director (being sent there because the director is the default gw for the realserver). To get the packet from the director to the client, you have to reverse the masquerading done by the LVS. To do this (in 2.2 kernels), on the director you add an ipchains rule

```
director:# ipchains -A forward -p tcp -j MASQ -s realserver1 telnet -d 0.0.0.0/0
```

If the director has multiple IPs facing the outside world (eg eth0=192.168.2.1 the regular IP for the director and eth0:1=192.168.2.110 the VIP), the masquerading code has to choose the correct IP for the outgoing packet. Only the packet with src_addr=VIP will be accepted by the client. A packet with any other scr_addr will be dropped. The normal default for masquerading (eth0) should not be used in this case. The required m_addr (masquerade address) is the VIP.

Does LVS fiddle with the ipchains tables to do this?

Julian Anastasov `ja@ssi.bg` 01 May 2001 No, ipchains only delivers packets to the masquerading code. It doesn't matter how the packets are selected in the ipchains rule.

The m_addr (masqueraded_address) is assigned when the first packet is seen (the connect request from the client to the VIP). LVS sees the first packet in the LOCAL_IN chain when it comes from the client. LVS assigns the VIP as maddr.

The MASQ code sees the first packet in the FORWARD chain when there is a -j MASQ target in the ipchains rule. The routing selects the m_addr. If the connection already exists the packets are masqueraded.

The LVS can see packets in the FORWARD chain but they are for already created connections, so no m_addr is assigned and the packets are masqueraded with the address saved in the connections structure (the VIP) when it was created.

There are 3 common cases:

1. The connection is created as response to packet.
2. The connection is created as response to packet to another connection.
3. The connection is already created

Case (1) can happen in the plain masquerading case where the in->out packets hit the masquerading rule. In this case when nobody recommends the s_addr for the packets going to the external side of the MASQ, the masq code uses the routing to select the m_addr for this new connection. This address is not always the DIP, it can be the preferred source address for the used route, for example, address from another device.

Case (1) happens also for LVS but in this case we know:

- the client address/port (from the received datagram)
- the virtual server address/port (from the received datagram)
- the real server address/port (from the LVS scheduler)

But this is on out->in packet and we are talking about in->out packets

Case (2) happens for related connections where the new connection can be created when all addresses and ports are known or when the protocol requires some wildcard address/port matching, for example, ftp. In this case we expect the first packet for the connection after some period of time.

It seems you are interested how case (3) works. The answer is that the NAT code remembers all these addresses and ports in a connection structure with these components

- external address/port (LVS: client)
- masquerading address/port (LVS: virtual server)
- internal address/port (LVS: real server)
- protocol
- etc

LVS and the masquerading code simply hook in the packet path and they perform the header/data mangling. In this process they use the information from the connection table(s). The rule is simple: when a packet is already for established connection we must remember all addresses and ports and always to use same values when mangling the packet header. If we select each time different addresses or ports we simply break the connection. After the packet is mangled the routing is called to select the next hop. Of course, you can expect problems if there are fatal route changes.

So, the short answer is: the LVS knows what m_addr to use when a packet from the real server is received because the connection is already created and we know what addresses to use. Only in the masquerading case (where LVS os not involved) connections can be created and a masquerading address to be selected without using rule for this. In all other cases there is a rule that recommends what addresses to be used at creation time. After creation the same values are used.

12.12 One network LVS-NAT

The disadvantage of the 2 network LVS-NAT is that the realservers are not able to connect to machines in the network of the VIP. You couldn't make a LVS-NAT setup out of machines already on your LAN, which were also required for other purposes to stay on the LAN network.

Here's a one network LVS-NAT LVS.

```
                     --------
                    |        |
                    | client |
                    |________|
                    CIP=192.168.1.254
                        |
                        |
          ----------    |
         |          |   |   VIP=192.168.1.110 (eth0:110)
         | director |---|
         |__________|   |   DIP=192.168.1.9 (eth0:9)
                        |
                        |
       ------------------------------------
       |                |                 |
       |                |                 |
RIP1=192.168.1.2   RIP2=192.168.1.3  RIP3=192.168.1.4 (all eth0)
 -------------      -------------     -------------
|             |    |             |   |             |
| realserver  |    | realserver  |   | realserver  |
|_____________|    |_____________|   |_____________|
```

The problem:

A return packet from the realserver (with address RIP->CIP) will be sent to the realserver's default gw (the director). ICMP redirects will be sent from the director telling the realserver of the better route directly to the client. The realserver will then send the packet directly to the client and it will not be demasqueraded by the director. The client will get a reply from the RIP rather than the VIP and the connection will hang.

The cure:

In the previous HOWTO I said that initial attempts to handle this by turning off redirects had not worked. The problem appears now to be solved.

Thanks to `michael_e_brown@dell.com` and Julian `ja@ssi.bg` for help sorting this out.

To get a LVS-NAT LVS to work on one network -

1. On the director, turn off icmp redirects

```
director:/etc/lvs# echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
director:/etc/lvs# echo 0 > /proc/sys/net/ipv4/conf/default/send_redirects
director:/etc/lvs# echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

(Note: eth0 may be eth1 etc, on your machine).

2. Make the director the default and only route for outgoing packets.

You will probably have set the routing on the realserver up like this

```
realserver:/etc/lvs# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         director        0.0.0.0         UG        0 0          0 eth0
```

Note the route to 192.168.1.0/24. This allows the realserver to send packets to the client by just putting them out on eth0, where the client will pick them up directly (without being demasqueraded) and the LVS will not work.

Remove the route to 192.168.1.0/24.

realserver:/etc/lvs#route del -net 192.168.1.0 netmask 255.255.255.0 dev eth0

This will leave you with

```
realserver:/etc/lvs# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         director        0.0.0.0         UG        0 0          0 eth0
```

The LVS-NAT LVS now works. If LVS is forwarding telnet, you can telnet from the client to the VIP and connect to the realserver.

You can ping from the client to the realserver.

You can also connect _directly_ to services on the realserver _NOT_ being forwarded by LVS (in this case e.g. ftp).

You can no longer connect directly to the realserver for services being forwarded by the LVS. (In the example here, telnet ports are not being rewritten by the LVS, ie telnet->telnet).

```
client:~# telnet realserver
Trying 192.168.1.11...
^C

(i.e. connection hangs)
```

Here's tcpdump on the director. Since the network is switched the director can't see packets between the client and realserver. The client initiates telnet. ‘netstat -a‘ on the client shows a SYN_SENT from port 4121.

```
director:/etc/lvs# tcpdump
tcpdump: listening on eth0
16:37:04.655036 realserver.telnet > client.4121: S 354934654:354934654(0) ack 1183118745 win 32120 <mss 14
16:37:04.655284 director > realserver: icmp: client tcp port 4 121 unreachable [tos 0xc0]
```

(repeats every second until I kill telnet on client)

I don't see the connect request from client->realserver. The first packet I see is the ack from the realserver, which will be forwarded via the director. The director will rewrite the ack to be from the director. The client will not accept an ack to port 4121 from director:telnet.

12.12.1 One net LVS-NAT from the mailing list

Here's an untested solution from Julian for a one network LVS-NAT

put the client in the external logical network. By this way the client, the director and the real server(s) are on same physical network but the client can't be on the masqueraded logical network. So, change the client from 192.168.1.80 to 166.84.192.80 (or something else). Don't add through DIP (I don't see such IP for the Director). Why in your setup DIP==VIP ? If you add DIP (166.84.192.33 for example) in the director you can later add path for 192.168.1.0/24 through 166.84.192.33. There is no need to use masquerading with 2 NICs. Just remove the client from the internal logical network used by the LVS cluster.

A different working solution from Ray Bellis `rpb@community.net.uk`

the same *logical* subnet. I still have a dual-ethernet box acting as a director, and the VIP is installed as an alias interface on the external side of the director, even though the IP address it has is in fact assigned from the same subnet as the

Ray Bellis `rpb@community.net.uk` has used a 2 NIC director to have the RIPs on the same logical network as the VIP (ie RIP and VIP numbers are from the same subnet), although they are in different physical networks.

12.13 Performance of LVS-NAT, 2.0 and 2.2 kernels

The throughput of LVS-NAT is limited by the time taken by the director to rewrite a packet. The limit for a pentium classic 75MHz is about 80Mbit/sec (100baseT). Increasing the number of realservers does not increase the throughput.

The *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>` shows a slightly higher latency with LVS-NAT compared to LVS-DR or LVS-Tun, but the same maximum throughput. The load average on the director is high (>5) at maximum throughput, and the keyboard and mouse are quite sluggish. The same director box operating at the same throughput under VS_DR or VS-Tun has no perceptable load as measured by top or by mouse/keyboard responsiveness.

12.14 Performance of LVS-NAT, 2.4 kernels

Wayne NAT taks some CPU and memory copying. With a slower CPU, it will be slower.

the origial posting `<http://marc.theaimsgroup.com/?l=linux-virtual-server&m=99554689108488&w=2>`

Julian Anastasov `ja@ssi.bg` 19 Jul 2001

This is a myth from the 2.2 age. In 2.2 there are 2 input route calls for the out->in traffic and this reduces the performance. By default, in 2.2 (and 2.4 too) the data is not copied when the IP header is changed. Updating the checksum in the IP header does not cost too much time compared to the total packet handling time.

To check the difference between the NAT and DR forwarding method in out->in direction you can use 19.16.1 (testlvs) from http://www.linux-vs.org/ julian/ and to flood a 2.4 director in 2 setups: DR and NAT. My tests show that I can't see a visible difference. We are talking about 110,000 SYN packets/sec with 10 pseudo clients and same cpu idle during the tests (there is not enough client power in my setup for full test), 2 CPUx 866MHz, 2 100mbit internal i82557/i82558 NICs, switched hub:

3 testlvs client hosts -> NIC1-LVS-NIC2 -> packets/sec.

I use small number of clients because I don't want to spend time in routing cache or LVS table lookups.

Of course, the NAT involves in->out traffic and this can reduce twice the performance if the CPU or the PCI power is not enough to handle the traffic in both directions. This is the real reason the NAT method to look so slow in 2.4. IMO, the overhead from the TUN encapsulation or from the NAT process is negliable.

Here come the surprises:

The basic setup: 1 CPU PIII 866MHz, 2 NICs (1 IN and 1 OUT), LVS-NAT, SYN flood using testlvs with 10 pseudo clients, no ipchains rules. Kernels: 2.2.19 and 2.4.7pre7.

- Linux 2.2 (with ipchains support, with modified demasq path to use one input routing call, something like LVS uses in 2.4 but without dst cache usage):

```
In 80,000 SYNs/sec, Out 80,000 SYNs/sec, CPU idle: 99% (strange)
In 110,000 SYNs/sec, Out 88,000 SYNs/sec, CPU idle: 0%
```

- Linux 2.4 (with ipchains support): with 3-4 ipchains rules:

```
In 80,000 SYNs/sec, Out 55,000 SYNs/sec, CPU idle: 0%
In 80,000 SYNs/sec, Out 80,000 SYNs/sec, CPU idle: 0%
In 110,000 SYNs/sec, Out 63,000 SYNs/sec (strange), CPU idle: 0%
```

- Linux 2.4 (without ipchains support):

```
In 80,000 SYNs/sec, Out 80,000 SYNs/sec, CPU idle: 20%
In 110,000 SYNs/sec, Out 96,000 SYNs/sec, CPU idle: 2%
```

- Linux 2.4, 2 CPU (with ipchains support):

```
In 80,000 SYNs/sec, Out 80,000 SYNs/sec, CPU idle: 30%
In 110,000 SYNs/sec, Out 96,000 SYNs/sec, CPU idle: 0%
```

- Linux 2.4, 2 CPU (without ipchains support):

```
In 80,000 SYNs/sec, Out 80,000 SYNs/sec, CPU idle: 45%
In 110,000 SYNs/sec, Out 96,000 SYNs/sec, CPU idle: 15%, 30000 ctxswitches/sec
```

What I see is that:

- modified 2.2 and 2.4 UP look equal on 80,000P/s

  limits: 2.2=88,000P/s, 2.4=96,000P/s, i.e. 8% difference

- 1 and 2 CPU in 2.4 look equal 110,000->96,000 (100mbit or PCI bottleneck?), may be we can't send more that 96,000P/s through 100mbit NIC?
- the ipchains rules can dramatically reduce the performance - from 88,000 to 55,000 P/s
- 2.4.7pre7 SMP shows too many context switches
- DR and NAT show equal results for 2.4 UP 110,000->96,000P/s, 2-3% idle, so I can't claim that there is a NAT-specific overhead.

I performed other tests, testlvs with UDP flood. The packet rate is lower, the cpu idle time in the LVS box was increased dramatically but the client hosts show 0% cpu idle, may be more testlvs client hosts are needed.

Julian Anastasov `ja@ssi.bg` 16 Jan 2002

Many people think that the packet mangling is evil in the NAT processing. The picture is different: the NAT processing in 2.2 uses 2 input routing calls instead of 1 and this totally kills the forwarding of packets from/to many destinations. Such problems are mostly caused from the bad hash function used in the routing code and because the routing cache has hard limit for entries. Of course, the NAT setups handle more traffic than the other forwarding methods (both the forward and reply directions), a good reason to avoid LVS-NAT with a low power director. In 2.4 the difference between the DR and NAT processing in out->in direction can not be noticed (at least in my tests) because only one route call is used, for all methods.

Matthew S. Crocker Jul 26, 2001

DR is faster, less resource intensive but has issues with configuration because of the age old 'arp problem'

Horms `horms@vergenet.net`

VS-NAT is still fast enough for many aplications and is IMHO considerably easier to set up. While I think LVS-DR is great I don't think people should be under the impresion that LVS-NAT will intrisicly be a limiting factor to them.

Don Hinshaw `dwh@openrecording.com` 04 Aug 2001

Cisco, Alteon and F5 solutions are all NAT based. The real limiting factor as I understand it is the capacity of the netcard, which these three deal with by using gigabit interfaces.

Julian Anastasov `ja@ssi.bg` 05 Mar 2002 in discussion with Michael McConnell

Note that I used a modified demasq path which uses one input route for NAT but it is wrong. It only proves that 2.2 can reach the same speed as 2.4 if there was use_dst analog in 2.2. Without such feature the difference is 8%. OTOH, there is a right way to implement one input route call as in 2.4 but it includes rewriting of the 2.2 input processing.

> Michael McConnell From what I see here, it looks as though the 2.2 kernel handles a higher numberof SYN's better than the 2.4 kernel. Am I to asume, that the for the 110,000SYNs/sec in the 2.4 kernel, only 63,000 SYNs/sec were answers? The rest failed?

In this test 2.4 has firewall rules, while 2.2 has only ipchains enabled.

> Is the 2.2 kernel better at answer a higher number of requests?

No. Note also that the testlvs test was only in one direction, no replies, only client->director->realserver

> has anyone compared iptables/ipchains, via 2.2/2.4?

here are my *results* `<http://marc.theaimsgroup.com/?l=linux-virtual-server&m=100903333532449&w=2>` There is some magic in these tests, I don't know at one place why netfilter shows such bad results. Maybe someone can point to me to the problem.

12.15 Various debugging techniques for routes

This originally described how I debugged setting up a one-net LVS-NAT LVS using the output of route. Since it is more about networking tools than LVS-NAT it has been moved to the section on 24 (newer networking tools).

12.16 Connecting directly from the client to a service:port on a realserver with LVS-NAT

If you connect directly to the realserver in a LVS-NAT LVS, the reply packet will be routed through the director, which will attempt to masquerade it. This packet will not be part of an established connection and will be dropped by the director, which will issue an ICMP error.

> Paul Wouters `paul@xtdnet.nl` 30 Nov 2001 It would like to reach all LVS'ed services on the realservers directly, *i.e.* without going through the LVS-NAT director, say from a local client not on the internet.
>
> Connecting from client to a RIP should just completely bypass all the lvs code, but it seems that the lvs code is confused, and thinks a RIP->client answer should be part of its NAT structure.
>
> tcpdump running on internal interface of the director shows a packet from the client received on the RIP; the RIP replies (never reaches the client, the director drops it). The director then sends out a port unreachable:

Julian

The code that replies with an ICMP error can be removed but then you still have the problem of reusing connections. The local_client can select a port for direct connection with the RIP but if that port was used some seconds before for a CIP->VIP connection, it is possible that LVS to catch these replies as part of the previous connection. LVS does not inspect the TCP headers and does not accurately keep the TCP state. So, it is possible that LVS will not to detect that the local_client and the realserver have established a new connection with the same addresses and ports that are still known as NAT connection. Even stateful conntracking can't notice it because the local_clientIP->RIP packets are not subject to NAT processing. When LVS sees the replies from RIP to local_clientIP it will SNAT them and this will be fatal because the new connection is between the local_clientIP and RIP directly, not from CIP->VIP->RIP. The other thing is that CIP even does not know that it connects from same port to same server. It thinks there are 2 connections from same CPORT: to VIP and to RIP, so they can live even at the same time.

> But a proper TCP/IP stack on a client will not re-use the same port that quickly, unless it is REALLY loaded with connections right? And a client won't (can't?) use the same source port to different destinations (VIP and RIP) right? So, the problem becomes almost theoretical?

This setup is dangerous. As for the ICMP replies, they are only for anti-DoS purposes but may be are going to die soon. There is still no enough reason to remove that code (it was not first priority).

> Or make it switachable as #ifdef or /proc sysctl?

Wensong

Just comment out the whole block, for example,

```
#if 0
                if (ip_vs_lookup_real_service(iph->protocol,
                                              iph->saddr, h.portp[0])) {
                        /*
                         * Notify the real server: there is no existing
                         * entry if it is not RST packet or not TCP
packet.
                         */
                        if (!h.th->rst || iph->protocol != IPPROTO_TCP) {
                                icmp_send(skb, ICMP_DEST_UNREACH,
                                          ICMP_PORT_UNREACH, 0);
                                kfree_skb(skb);
                                return NF_STOLEN;
                        }
                }
#endif
```

> This works fine. Thanks

The same topic came up again. Here's another similar reply.

> I've set up a small LVS_NAT-based http load balancer but can't seem to connect to the realservers behind them via IP on port 80. Trying to connect directly to the real servers on port 80, though, translates everything correctly, but generates an ICMP port unreach.

Ben North `ben@antefacto.com` 06 Dec 2001

The problem is that LVS takes an interest in all packets with a source IP:port of a Real Service's IP:port, as they're passing through the FORWARD block. This is of course necessary — normally such packets would exist because of a connection between some client and the Virtual Service, mapped by LVS to some Real Service. The packets then have their source address altered so that they're addressed VIP:VPort -> CIP:CPort.

However, if some route exists for a client to make connections directly to the Real Service, then the packets from the Real Service to the client will not be matched with any existing LVS connection (because there isn't one). At this point, the LVS NAT code will steal the packet and send the "Port unreachable" message you've observed back to the Real Server. A fix to the problem is to #ifdef out this code — it's in ip_vs_out() in the file ip_vs_core.c.

12.17 Postings from the mailing list

`frederic.defferrard@ansf.alcatel.fr`

would be possible to use LVS-NAT to load-balance virtual-IPs to ssh-forwarded real-IPs?

Ssh can also be used to create a local access that is forwarded to a remote access throught the ssh protocol. For example you can use ssh to securely map a local acces to a remote POP server:

local:localport ==> local:ssh   ssh port forwarding   remote:ssh ==> remote:pop

And when you connect to local:localip you are transparently/securely connected to remote:pop

The main idea is to allow RS in differents LANs with RS that are non-Linux (precluding LVS-Tun).

Example:

```
                                - VS:81 ---- ssh ---- RS:80
                               /
INTERNET - - - - > VS:80 (NAT)-- VS:82 ---- ssh ---- RS:80
                               \
                                - VS:83 ---- ssh ---- RS:80
```

> Wensong you can use VPN (or CIPE) to map some external real servers into your private cluster network. If you use LVS-NAT, make sure the routing on the real server must be configuration properly so that the response packets will go through the load balancer to the clients.

I think that it isn't necessery to have the default router to the load balancer when using ssh because when the RS address is the same that the VS address (differents ports)

> With the NAT method, your example won't work because the LVS/NAT treats packets as local ones and forward to the upper layers without any change. However, your example give me an idea that we can dynamically redirect the port 80 to port 81, 82 and 83 respectively for different connections, then your example can work. However, the performance won't be good, because lots of works are done in the application level, and the overhead of copying from kernel to user-space is high.
>
> Another thought is that we might be able to setup LVS/DR with real server in different LANs by using of CIPE/VPN stuff. For example, we use CIPE to establish tunnels from the load balancer to real servers like

```
                    10.0.0.1================10.0.1.1 realserer1
                    10.0.0.2================10.0.1.2 realserer2
--- Load Balancer   10.0.0.3================10.0.1.3 realserer3
                    10.0.0.4================10.0.1.4 realserer4
                    10.0.0.5================10.0.1.5 realserer5
```

> Then, you can add LVS/DR configuration commands as:

```
ipvsadm -A -t VIP:www
ipvsadm -a -t VIP:www -r 10.0.1.1 -g
ipvsadm -a -t VIP:www -r 10.0.1.2 -g
ipvsadm -a -t VIP:www -r 10.0.1.3 -g
ipvsadm -a -t VIP:www -r 10.0.1.4 -g
ipvsadm -a -t VIP:www -r 10.0.1.5 -g
```

> I haven't tested it. Please let me know the result if anyone tests this configuration.

13 VS-DR

VS-DR is based on IBM's NetDispatcher. The NetDispatcher sits in front of a set of webservers, which appear as one webserver to the clients. The NetDispatcher served http for the Atlanta and the Sydney Olympic games and for the chess match between Kasparov and Deep Blue.

Here's an example set of IPs for a LVS-DR setup. Note that in this example the RIPs are on the same network as the VIP. If the RIPs were on a different network (eg 192.168.2.0/24) to the VIP && the router (here the client) was not allowed to route to RIP network (ie the realservers do not receive the arp requests from the router asking "who has VIP tell router"), then the arp problem is solved without requiring a patch on the director's kernel. In this example, for (my) convenience, the servers are on the same network as the client and you have to handle the arp problem (I used the arp -f /etc/ethers approach).

```
Host                         IP
client                       CIP=192.168.1.254
director                     DIP=192.168.1.1
virtual IP (VIP)             VIP=192.168.1.110 (arpable, IP clients connect to)
realserver1                  RIP1=192.168.1.2, VIP=192.168.1.110 (lo:0, not arpable)
realserver2                  RIP2=192.168.1.3, VIP=192.168.1.110 (lo:0, not arpable)
realserver3                  RIP3=192.168.1.4, VIP=192.168.1.110 (lo:0, not arpable)
.
.
realserver-n                 192.168.1.n+1

#lvs_dr.conf
LVS_TYPE=VS_DR
INITIAL_STATE=on
VIP=eth0:110 lvs 255.255.255.255 192.168.1.110
DIP=eth0 dip 192.168.1.0 255.255.255.0 192.168.1.255
DIRECTOR_DEFAULT_GW=client
SERVICE=t telnet rr realserver1 realserver2 realserver3
SERVER_VIP_DEVICE=lo:0
SERVER_NET_DEVICE=eth0
SERVER_DEFAULT_GW=client
#----------end lvs_dr.conf------------------------------------
```

```
                        --------
                       |        |
                       | client |
                       |________|
                       CIP=192.168.1.254
                           |
                        (router)
                           |
             ----------    |
            |          |   |    VIP=192.168.1.110 (eth0:1, arps)
            | director |---     DIP=192.168.1.1 (eth0)
            |__________|   |
                           |
                           |
          -------------------------------------
          |                |                  |
          |                |                  |
   RIP1=192.168.1.2  RIP2=192.168.1.3  RIP3=192.168.1.4 (eth0)
   VIP=192.168.1.110 VIP=192.168.1.110 VIP=192.168.1.110 (all lo:0, non-arping)
   -------------     -------------      -------------
  |             |   |             |    |             |
  | realserver  |    | realserver  |     | realserver  |
  |_____________|    |_____________|     |_____________|
         |                 |                   |
     (router)          (router)            (router)
         |                 |                   |
         ----------------------------------------------> to client
                                                         (or router
                                                          in front of
                                                          director)
```

Here's the lvs_dr.conf file

```
#--------------------lvs_dr.conf
LVS_TYPE=VS_DR
INITIAL_STATE=on

#director setup
VIP=eth0:12 192.168.1.110 255.255.255.255 192.168.1.110
DIP=eth0 192.168.1.10 192.168.1.0 255.255.255.0 192.168.1.255
#service setup, one service at a time
SERVICE=t telnet rr 192.168.1.1 192.168.1.8 127.0.0.1

#realserver setup
SERVER_LVS_DEVICE=lo0:1
SERVER_NET_DEVICE=eth0

#----------end lvs_dr.conf------------------------------------
```

VS-DR setup and testing is the same as LVS-Tun except that all machines within the LVS-DR (ie the director and realservers) must be able to arp each other. This means that they have to be on the same network without any forwarding devices between them. This means that they are using the same piece of transport layer hardware ("wire"), eg RJ-45, coax, fibre. There can be hub(s) or switch(es) in this mix. Communication within the LVS is by link-layer, using MAC addresses rather than IP's. All machines in the LVS have the VIP, only the VIP on the director replies to arp requests, the VIP on the realservers must be on a non-arping device (eg lo:0, dummy).

The restrictions for LVS-DR are

- The client must be able to connect to the VIP on the director
- Realservers and the director must be on the same piece of wire (they must be able to arp each other) as packets are sent by link-layer from the director to the realservers.
- The route from the realservers to the client _cannot_ go through the director, i.e. the director cannot be the default gw for the realservers. (Note: the client does not need to connect to the the realservers for the LVS to function. The realservers could be behind a firewall, but the realservers must be able to send packets to the client). The return packets, from the realservers to the client, go directly from the realservers to the client and _do_not_ go back through the director. For high throughput, each realserver can have its own router/connection to the client/internet and return packets need not go through the router feeding the director.

For more info see e-mail postings about LVS-DR topologies in the section 3.12 (More on the arp problem and topologies of LVS-DR and LVS-Tun LVS's).

To allow the director to be the default gw for the realservers (e.g. when the director is the firewall), see 13.6 (Julian's martian modification).

Note for LVS-DR (and LVS-Tun), the services on the realservers are listening to the VIP. You can have the service listening to the RIP as well, but the LVS needs the service to be listening to the VIP. This is not an issue with services like telnet which listen to all local IPs (ie 0.0.0.0), but httpd is set up to listen to only the IPs that you tell it.

Normally for LVS-DR, the client is on a different network to the director/server(s), and each realserver has its own route to the outside world. In the simple test case below, where all machines are on the 192.168.1.0 network, no routers are required, and the return packets, instead of going out (the router(s)) at the bottom of the diagram, would return to the client via the network device on 192.168.1.0 (presumably eth0).

13.1 How LVS-DR works

Here's part of the rc.lvs_dr script which configures the realserver with RIP=192.168.1.8

```
#setup servers for telnet, LVS-DR
/sbin/ipvsadm -A -t 192.168.1.110:23 -s rr
echo "adding service 23 to realserver 192.168.1.6 "
/sbin/ipvsadm -a -t 192.168.1.110:23 -R 192.168.1.6 -g -w 1
```

With LVS-DR, the target port numbers of incoming packets cannot be remapped (unlike LVS-NAT). A request to port 23 (telnet) on the VIP will be forwarded to port 23 on a realserver, thus the RIP entry has no accompanying port.

Here's the packet headers as the request is processed by the LVS.

```
packet                  source       dest         data
1. request from client  CIP:3456     VIP:23       -
2. ipvsadm table:
   director chooses server=RIP1, creates link-layer packet
                        MAC of DIP   MAC of RIP1  IP datagram
                                                  source=CIP:3456,
                                                  dest=VIP:23,
                                                  data= -
3. realserver recovers IP datagram
                        CIP:3456     VIP:23       -
4. realserver looks up routing table, finds VIP is local,
   processes request locally, generates reply
                        VIP:23       CIP:3456     "login:"

5. packet leaves realserver via its default gw, not via DIP.
```

For the verbally oriented...

A packet arrives from the client for the VIP (CIP:3456->VIP:23). The director looks up its tables and decides to send the connection to realserver_1. The director arps for the MAC address of RIP1 and sends a link-layer packet to that MAC containing an IP datagram with CIP:3456->VIP:23. This is the same src:dst as the incoming packet and the tcpip layer see this as a forwarded packet. To allow this packet to be sent to the realserver, it is not neccessary for forwarding must be on in the director (it is turned off by default in 2.2.x, 2.4.x kernels - turning it on is handled by the 10.1 (configure script)).

The packet arrives at realserver_1. The realserver recovers the IP datagram, looks up its routing table, finds that the VIP (on an otherwise unused, non-arping and nonfunctional device) is local.

I'm not sure what exactly happens next, but I believe the Linux tcpip stack then delivers the packet to the socket listeners, rather than to the device with the VIP, but I'm out of my depth now.

The realserver now has a packet CIP:3456->VIP:23, processes it locally, constructs a reply, VIP:23->CIP:3456. The realserver looks up its routing table and sends the reply out its default gw to the internet (or client). The reply does not go through the director.

The role of LVS-DR is to allow the director to deliver a packet with dst=VIP (the only arp'ing VIP being on the director), not to itself, but to some machine that (as far as the director knows) doesn't have the VIP address at all. The only difference between LVS-DR and LVS-Tun is that instead of putting the IP datagram inside a link-layer packet with dst=MAC of the RIP, for LVS-Tun the IPdatagram from the client CIP->VIP is put inside another IPdatagram DIP->RIP.

The use of the non-arping lo:0 and tunl0 to hold the VIP for VS-DR and LVS-Tun (respectively) is to allow the realserver's routing table to have an entry for a local device with IP=VIP _AND_ that so that other machines can't see this IP (ie it doesn't reply to arp requests). There is nothing particularly loopback about the lo:0 device that is required to make LVS-DR work anymore than there is anything tunnelling about a tunl0 device. For 2.0.x kernels, a tunnel packet is de-capsulated because it is marked type=IPIP, and will be decapsulated if delivered to an lo device just as well as if delivered to a tunl device. The 2.2.x kernels are more particular and need a tunl device (see "Properties of devices for VIP").

13.2 Handling the arp problem for LVS-DR

13.2.1 VIP on lo:0

The VIP on the realservers must not reply to arp requests from the client (or the router between the client and the director).

Realservers with Linux 2.2.x kernels The loopback device does not arp by default for all OS's except Linux 2.2.x,2.4.x kernels (even when you use -noarp with ifconfig). You may need to do something if you are running a realserver with a 2.2.x or 2.4.x kernel (see the 3 (arp problem)).

13.2.2 Lar's method

This requires hiding the VIP on the realservers, by putting them on a separate network.

Lars set this up first on LVS-Tun. Here it is for LVS-DR. The director has 2 NICs and the realservers are on a different network (10.1.1.0/24) to the VIP (192.168.1.0/24). All IPs reply to arps. The router/client cannot route to the realserver network and the RIPs do not need to be internet routable. Since the director has 2 NICs, in the lvs_dr.conf file, set the DIP to eth1.

```
                 ________
                |        |
                | client |
                |________|
                CIP=192.168.1.254
                    |
                 (router)
                    |
          VIP=192.168.1.110 (eth0, arps)
```

```
                    ----------
                   |          |
                   | director |
                   |__________|
                   DIP=10.1.1.1 (eth1, arps)
                         |
                         |
        -------------------------------------
        |                |                  |
        |                |                  |
 RIP1=10.1.1.2     RIP2=10.1.1.3     RIP3=10.1.1.4 (eth0)
 VIP=192.168.1.110 VIP=192.168.1.110 VIP=192.168.1.110 (all lo:0, can arp)
 -------------     -------------      -------------
|             |   |             |    |             |
| realserver  |   | realserver  |    | realserver  |
|_____________|   |_____________|    |_____________|
        |                |                  |
     (router)         (router)           (router)
        |                |                  |
        ------------------------------------------------> to client
```

13.2.3 Transparent Proxy (TP or Horms' method) - not having the VIP on the realserver at all.

This subject has it's 16 (own section).

13.3 VS-DR scales well

Performance tests (75MHz pentium classics, on 100Mbps network) with LVS-DR on the *performance page* <http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html> showed that the limit for LVS-DR is the rate at which the director can forward packets to the realservers. LVS doesn't add any detectable latency or change the throughput of forwarding. There is little load on the director operating at high throughput in LVS-DR mode. Apparently little computation is involved in forwarding.

13.4 VS-DR director is default gw for realservers

In the case where the director is the firewall for the realserver network, the director has to be the default gw for the realservers.

If the reply packet from the realserver to the client (VIP->CIP) goes through the director (which has a device with IP=VIP), the director is being asked to route a packet with a src address that is on the director.

> Horms horms@vergenet.net> The problem is that with Direct routing the reply from the real server has the vip as the source address. As this is an address of one of the interfaces on the director it will drop it if you try and forward it through the director. It appears from experimentation with /proc/sys/net/ipv4/conf/*/rp_filter that at least on 2.2.14, there is no way to turn this behaviour off.

This type of packet is called a "source martian" and is dropped by the director. martians can be logged with

```
# echo 1 >/proc/sys/net/ipv4/conf/all/log_martians.
```

There are 3 solutions to this; 2 by Julian and 1 by Horms.

13.5 Director has 1 NIC, accepts packets via transparent proxy.

If the director accepts packets for the VIP via transparent proxy, then the director doesn't have the VIP and the return packets are processed normally. (Note: transparent proxy only works on the director for 2.2.x kernels).

Here's Julian's posting

```
               Clients
                  |
                 ISP
                  |eth0/ppp0/...
               Router/Firewall/Director (LVS box)
                  |eth1
       +----------+-----------+
       |eth0                  |eth0
       Real 1                 Real2
```

Router: transparent proxy for VIP (or all served VIPs). The ISP must feed your Director with packets for your subnet 199.199.199.0/24 VS-DR mode (Yes, LVS-DR, this is not a mistake). eth1: 199.199.199.2. default gw is ISP.

Real server(s): nothing special. VIP on hidden device or via transparent proxy. eth0: 199.199.199.3. default gateway is 199.199.199.2 (the Director)

This is a minimum required config. You can add internal subnets yourself using the same physical network (one NIC) or by adding additional NICs, etc. They are not needed for this test.

Packets from the real servers with saddr=VIP will be forwarded from the director because VIP is not configured in the Director. We expect that this setup is faster than VS/NAT.

13.6 Julian's martian modification

also see *Julian's notes on using the director as a gateway for realservers in LVS-DR* `<http://www.linuxvirtualserver.org/~julian/##lvsgw>`.

(see 13.4 (earlier for an explanation of "source martians").)

The martian modification is currently (Aug 2001) implemented with the *hidden-forward_shared-xxx.diff patch* `<http://www.linuxvirtualserver.org/~julian/>`. This patch has the hidden (for realservers) and forward_shared (for directors) patch and can be applied to both realservers and directors. (Remember for the director you need the ipvs patch too).

This is a kernel patch, director has 2 NICs (doesn't work with one NIC), VIP is on outside NIC. Here are the 29.4 (original patches for the martian modification, 2.2 and 2.4 kernels).

The patch (below) has been tested against 2.2.15pre9 (Joe) and 2.2.13 (Stephen Zander `gibreel@pobox.com`). The kernel code is not changing very fast for these files. If patching other 2.2 kernels produces no rejects (*i.e.* no "HUNK FAILED" notices) then the patch is probably OK. The patch for 2.4.x kernels is at *Julian's patch page* `<http://www.linuxvirtualserver.org/~julian>` and in the text below.

After applying this patch, for a test, use the default values for */rp_filter(=0). This allows real servers to send packets with saddr=VIP and daddr=client through the Director.

If this patch is applied and external_eth/rp_filter is 0 (which is the default) the real servers can receive packets with saddr=any_director_ip and dst=any_RIP_or_VIP which is not very good. On the external net, set rp_filter=1 for better security.

Here's the test setup

```
             ------------
            |            |
            |  client    |
            |____________|
                  |
                  |  192.168.2.0/24
             -----|------
            |            |
            |  director  | LVS-DR director has 2 NICs
            |____________|
                  | eth0    192.168.1.9
                  | eth0:12 192.168.1.1
                  |
                  |  192.168.1.0/24
             -----|--------------------
            |
            |
       -----|----------
      |                |
      | realserver(s)  | default gw=192.168.1.1
      |________________|
```

192.168.1.1 is the normal router. For the test it was put on the director instead (as an alias). The director has 2 NICs, with forwarding=on (client and realservers can ping each other).

Director runs linux-0.9.8-2.2.15pre9 unpatched or with Julian's patch. LVS is setup using the 10.1 (configure script), redirecting telnet, with rr scheduling to 3 realservers. The realservers were running 2.0.36 (1) or 2.2.14 (2). The arp problem was handled for the 2.2.14 realservers by permanently installing in the client's arp table, the MAC address of the NIC on the outside of the director, using the command 'arp -f /etc/ethers'

The director was booted 4 times, into unpatched, patched, unpatched and patched. After each reboot the lvs scripts were run on the director and the realservers, then the functioning of the LVS tested by telnet'ing multiple times from the client to the VIP.

For the unpatched kernel, the client connection hung and inactive connections acccumulated for each realserver. For the patched kernel, the client telnet'ed to the VIP connecting with each realserver in turn.

The 10.1 (configure_script) will set up the modified LVS-DR (and will warn you that you need the patch for this to work). Setup details are in *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>`

13.6.1 Martian modification performance

Performance has similar latency to LVS-NAT but the load is low on the director at high throughput of LVS-DR (see the *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>`).

13.6.2 questions

> mstockda@logicworks.net Which interfaces need forward_shared? the interface on the real server lan _and_ the external side?

Julian Anastasov ja@ssi.bg 15 Mar 2002

No, you just enabled the feature which works only for the already selected interfaces. Check it with

```
ip route get from VIP to 1.2.3.4 iif CHECK_ALL_INTERFACES_HERE
```

You should enable forward_shared only for interfaces attached to internal mediums (hubs) and of course, only where is needed.

```
#define IN_DEV_FORWARD_SHARED(in_dev)   ((in_dev)->cnf.forward_shared && ipv4_devconf.forward_shared
```

13.7 Accepting packets on LVS-DR director by fwmarks

Horms 9 (fwmarks) allows the director to accept packets by fwmark. There is no VIP required on the director.

13.8 security concerns: default gw(s) and routing with LVS-DR/VS-Tun

The material here came from listening to a talk by Herbie Pearthree of IBM (posting 2000-10-10) and from a posting by TC Lewis (which I've lost).

In normal IP communication between two hosts, the routing is symmetrical: each end of the link has an ethernet device with an IP and a route to the other machine. Packets are transmitted in pairs (an outgoing packet and a reply, often just an ACK).

In LVS-DR or LVS-Tun the roles of the two machines are split between 3 machines. Here is a two network test setup, with the client in the position normally occupied by the router. In production, the client will have a public IP and connect via a router. (This is my test setup. A big trap is that services which make calls from the RIP, eg 17 (identd) and 11.26 (rshd) will work in my setup, but fail in a production setup as the RIP will not be a routable IP).

```
        ------------
       |            |192.168.1.254 (eth0)
       |  client    |----------------------
       |____________|          <-          |
     CIP=192.168.2.254 (eth1)              |
|             |                            |
V             |                            |
     VIP=192.168.2.110 (eth0)              |
        ------------                       |
       |            |                      |
       |  director  |                      |
       |____________|                      |
     DIP=192.168.1.1 (eth1, arps)          |
|             |                            |
V             |----------------------------
              |                 ->
```

```
RIP=192.168.1.2 (eth0)
VIP=192.168.2.110 (lo:0, no_arp)
   -------------
  |             |
  | realserver  |
  |_____________|
```

13.8.1 Director's default gw

The client sends a packet to the VIP on the director. In a normal exchange of packets between a pair of machines, the director would send a reply packet back to the client. With an LVS, the director's response instead is a packet to the MAC address of the RIP. In most normal operation, the VIP on the director never sends packets back to the client, it only sends packets to the realservers. A default gw for the director is not needed for the functioning of the LVS. Having a default gw would only allow the VIP director to reply to packets from the internet, such as port scans, creating a security hazard. The director doesn't need and shouldn't have a default gw.

There are pathological conditions when the VIP needs to reply to the client. If the realserver goes down, the director will issue ICMP "host unreachable" packets, till a new realserver is switched in by mon or ldirectord. (If you have a long lived tcp connection, eg with telnet or https, the new realserver will be getting packets for a connection which it doesn't know about, and it will issue a tcp reset. This reset will go out the default gw for the realserver and the client's session will hang or drop.)

Julian Anastasov `ja@ssi.bg`> 30 Aug 2001

> It may be these ICMPs are not fatal if they are not sent. This is true when LVS is used in transparent proxy setups and particulary in 2.4 where there is no real transparent proxy support. There icmp_send() does not send any packets when there is no running squid. But may be the original email sender wanted to use the LocalNode feature together with a DR setup, IIRC. I see that 10.1 (configure-lvs) does not have configs for such setups with mixed forwarding methods. So, as you said, the users with more knowledge can select another way to build their setup. And they will know when they need a default gateway :)
>
> If the mtu is not matched between the router and the director, the director will need to send ICMP "fragmentation needed" packets back to the router. This is a bad setup.

The 10.1 (configure script) doesn't add a default gw for packets from the VIP. If you want one, just put it in yourself.

If you need to talk to the outside world from the director, use another IP on the director (*e.g.* the primary IP on the outside of the director) and use iproute2 to send packets from this IP to 0/0.

13.8.2 Realserver's default gw, route to realserver from router

The realserver doesn't reply to the director, instead it sends its reply to the client. The realserver requires a default gw (here 192.168.1.154), but the client/router never replies to the realserver, the client/router sends its replies to the director. So the client/router doesn't need a route to the realserver network. To have one would be a security hazard. The realserver now can't ping its default gw (since there's no route for the reply packet), but the LVS still works.

The flow of packets around the LVS-DR LVS is shown by the ascii arrows.

When an attacker tries to access the nodes on the LVS, it can only connect to the LVS services on the director. It can't connect to the realserver network, as there is no routing to the realservers (even if they get access to the router). Presumably the realservers are not accessable from the outside as they'll be on private networks anyhow.

Note that for Julian's 13.6 (martian modification), the director will need a default gw.

13.9 routing to realserver from director

If you are only using the link between the director and realserver for VS-DR packets (*i.e.* you aren't telnet or ssh'ing from the realserver to the director for your admin, and you aren't copying logs from one machine to another), then you don't need an IP on the interface on the director which connects to the realserver(s).

> tc lewis `tcl@bunzy.net` 12 Jul 2000 (paraphrased) I would like to send packets from the LVS-DR director to the realservers by a separate interface (eth2), but not assign an IP to this interface. Normally I put a 192.168.100.x ip on eth2, but without it, route add -net 192.168.100.0 netmask 255.255.255.0 dev eth2 just gives me an error about eth2 not existing. I just want to save an extra IP.
>
> What i'm asking is: does the director's eth2 need an ip on 192.168.100.0/24, or can i just somehow add that route to that interface to tell the machine to send packets that way? With lvs, the real servers are never going to care about the director's interface ip, since there's no direct tcp/ip connections or anything there, but it looks like it still needs an ip anyway.
>
> If all that that interface is doing is forwarding outgoing packets from the director via the dr method, then i don't see why it needs an ip address.

Ted Pavlic `tpavlic@netwalk.com`

You basically want to do device routing. There's nothing special about this – many routers do it... NT even does it. So does Linux. Your original route command should work

```
route add -net 192.168.100.0 netmask 255.255.255.0 dev eth2
```

as long as you've brought up eth2. Now tricking Linux into bringing up eth2 without an address might be the hard part. Try this:

```
ifconfig eth2 0.0.0.0 up

or

ifconfig eth2 0 up
```

tc lewis `tcl@bunzy.net`

```
ifconfig eth0 0.0.0.0 up
```

then the route did work. I tried that before with a netmask but it didn't work.

Ted Pavlic `tpavlic@netwalk.com`

Remember that IP=0 actually is IP=0.0.0.0, which is another name for the default route.

The reason why IP=0 is 0.0.0.0 ... Remember that each IP address is simply a 4-byte unsigned integer, right? Well... the easiest way to envision this is to imagine that an IP is just like a base-256 number. For example:

```
216.69.192.12 (my mail server) would be:

12 +
192 * 256 +
69  * 256 * 256 +
216 * 256 * 256 * 256
```

Which is equal to 3628449804. So...

telnet 216.69.192.12 25

is the same as:

telnet 3628449804 25

0.0.0.0 is just a special system address which is the same as the default route. Making a route from 0.0.0.0 to some gateway will set your default route equal to that gateway. That's all "route add default gw ..." does. Don't believe me? Do a route -n.

So when I told TC to put 0 on his IP-less NIC, I was just choosing a system IP that I knew would not ever need to be transmitted on. Linux wanted an IP to create the interface... so I gave it one – the IP of the default gateway. Packets would never need to leave the system going to 0.0.0.0, and Linux has to listen to this address ANYWAY, so you might as well explicitly put it on an interface.

What would have also worked (and might have been a better idea) would be to put 127.0.0.1 on that interface. That is another system address that Linux will listen to anyway if loopback has been turned on... and it should never transmit anything away from itself with that as the destination address, so it's safe to put it on more than one interface.

The only reason I chose 0 over 127.0.0.1 is because 0 is easy... It's small... It's quick. Whenever I want to telnet to my localhost's port blah I just do a:

telnet 0 blah

because I'm lazy.. (Linux sees 0, interprets 0.0.0.0, sees an address it listens to, and basically treats 0 like a loopback)

Also you'll notice that if you give an interface 0.0.0.0 as an IP address and do an ifconfig to get stats on that interface, it will still retain no IP address. Another perquesite of using 0.0.0.0 in TC's particular situation. It may actually cause less confusion in the end.

13.10 Setting up NAT clients on LVS-DR realservers

The realserver in LVS-DR has two IPs, the RIP and the VIP. The LVS'ed services are running on the VIP. Packets from LVS'ed services, returning from the realserver, have src_addr=VIP. The RIP is not directly involved in the LVS. Services may be running on the RIP too, eg telnet which listens to 0.0.0.0, but services running on the RIP are of no interest to a LVS-DR. The director only needs the RIP to determine the target MAC address to forward packets from the clients destined for the VIP. Thus you are free to do whatever you like with the RIP without affecting the LVS. Usually the RIP is on a private IP (eg 192.168.x.x) so as to not require an extra IP, and to shield the realserver from the internet. It would be unusual to run non-LVS'ed services on the realservers, as the RIP would have to be a public IP and the realservers would have to be firewalled. However there it is reasonable to run clients on the realservers. A client session (*e.g.* telnet) initiated from the RIP would have to be NAT'ed out to the outside world. The NAT box could be the router or the director. Here's how to setup with the director doing the NAT'ing (the router setup would be the same).

13.10.1 Send client packets (src_addr=RIP) to the director and LVS packets (src_addr=VIP) to the router

This is not possible with the standard destination-based route command. You need the priority routing tools from 24.1 (iproute2).

Here's Julian's recipe (25 Sep 2000) for setting up NAT for clients on realservers in a LVS-DR LVS.

Settings for the real server(s), send all packets from the RIP network (RIPN) to the DIP (an IP on the director in the RIPN).

```
#create a rule with priority 100, which says that for any packet
#with src_addr in the RIP network, lookup the action in table 100.
realserver: #ip rule add prio 100 from RIPN/24 table 100

#route all packets in table 100 which go to 0/0
#(ie anywhere, the default route), via the DIP.
realserver: #ip route add table 100 0/0 via DIP dev eth0

#the result of this is that packets with src_addr=RIPnetwork
#and dst_addr=0/0 go via the DIP.
```

The director has to to listen on DIP (if it doesn't already), not send ICMP redirects from the DIP ethernet device and masquerade packets from the RIPN.

```
director: #ifconfig eth0:1 DIP netmask 255.255.255.0
director: #echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
director: #echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

13.10.2 masquerade the service(s) for the realservers

Here's how to masquerade all services, from all machines on the realserver network for kernel 2.2. In practice you will add masquerading by RIP:service, only masquerading those services needed.

```
director: #ipchains -A forward -s RIPN/24 -j MASQ
```

13.10.3 add a default route for packets from the primary IP on the outside of the director

For 13.8 (VS-DR, no default gw is needed) for packets from the primary IP on the outside of the director or from the VIP (which will be an alias/secondary IP). For security reasons then none is installed. To allow masquerading of clients on the realservers, a default route will be needed for packets from the primary IP on the outside of the director (but not for packets from the VIP).

If you want to test this out first, just put in a default route for the director using the route command. If you like it you can add the more restrictive routes with iproute2 later.

14 VS-Tun

(see also *Julian's LVS-Tun write up* <http://www.linuxvirtualserver.org/~julian/TUN-HOWTO.txt>).

VS-Tun is an LVS original. It is based on LVS-DR and has the same high scalability/throughput of LVS-DR.

VS-Tun can be used with realservers that can tunnel (==IPIP encapsulation). The director encapsulates the request packet inside an IPIP packet before sending it to the realserver. The realserver must be able to decapsulate the IPIP packet (currently linux only). (With LVS-DR, the realservers can have almost any OS.)

Unlike LVS-DR, with LVS-Tun the realservers can be on a network remote from the director, and can each be on separate networks. Thus the realservers could be in different countries (eg a set of ftp mirror sites for a project). If this is the case the realservers will be generating reply packets with VIP->CIP. Not being on the VIP network, the routers for the realservers will have to be programmed to accept outgoing packets with source=VIP. Routers normally drop these packets as an anti-spoofing measure.

If the realservers are on the same network as the director, then VS-DR and LVS-Tun are equivalent in terms of performance and ease of setup (see the *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>`)

Here's an example set of IPs for a LVS-Tun setup. For (my) convenience the servers are on the same network as the client. The only restrictions for LVS-Tun with remote hosts are that the client must be able to route to the director and that the realservers must be able to route to the client (the return packets to the client come directly from the realservers and do not go back through the director).

Normally for LVS-Tun,the client is on a different network to the director/server(s), and each server has its own route to the outside world. In the simple test case below where all machines are on the 192.168.1.0 network there would be no default route for the servers, and routing for packets from the servers to the client would use the device on the 192.168.1.0 network (presumably eth0). In reallife, the realservers would have their own router/connection to the internet and packets returning to the client would go through this router. In any case reply packets do not go back through the director.

```
Machine                      IP
client                       CIP=192.168.1.254
director                     DIP=192.168.1.1
                             VIP=192.168.1.110 (arps, IP clients connect to)
realserver-1                 RIP1=192.168.1.2, VIP (tunl0, non-arping, 192.168.1.110)
realserver-2                 RIP2=192.168.1.3, VIP (tunl0, non-arping, 192.168.1.110)
realserver-3                 RIP3=192.168.1.4, VIP (tunl0, non-arping, 192.168.1.110)
.
.
realserver-n                 RIPn=192.168.1.n+1, VIP (tunl0, non-arping, 192.168.1.110)
```

```
#lvs_tun.conf
LVS_TYPE=VS_TUN
INITIAL_STATE=on
VIP=eth0:110 192.168.1.110 255.255.255.255 192.168.1.110
DIP=eth0 192.168.1.9 192.168.1.0 255.255.255.0 192.168.1.255
DIRECTOR_DEFAULT_GW=client
SERVICE=t telnet rr realserver1 realserver2
SERVER_VIP_DEVICE=tunl0
SERVER_NET_DEVICE=eth0
SERVER_DEFAULT_GW=client
#----------end lvs_tun.conf------------------------------------
```

```
                        --------
                       |        |
                       | client |
                       |________|
                       CIP=192.168.1.254
                           |
                           |
                      VIP=192.168.1.110 (eth0:1, arps)
                       ----------
                      |          |
                      | director |
                      |__________|
                      DIP=192.168.1.1 (eth0)
                           |
                           |
          -------------------------------------
          |                |                  |
          |                |                  |
   RIP1=192.168.1.2  RIP2=192.168.1.3  RIP3=192.168.1.4 (eth0)
    -------------     -------------     -------------
   |             |   |             |   |             |
   | realserver  |   | realserver  |   | realserver  |
   |_____________|   |_____________|   |_____________|

   VIP=192.168.1.110 VIP=192.168.1.110 VIP=192.168.1.110 (all tunl0,non-arping)
          |                |                  |
      (router)          (router)           (router)
          |                |                  |
          ----------------------------------------------> to client
```

14.1 How LVS-Tun works

Here's part of the rc.lvs_dr script which configures the realserver with RIP=192.168.1.8

```
#setup servers for telnet
/sbin/ipvsadm -A -t 192.168.1.110:23 -s rr
/sbin/ipvsadm -a -t 192.168.1.110:23 -R 192.168.1.1 -i -w 1
```

With LVS-Tun, the target port numbers of incoming packets cannot be remapped. A request to port 23 on the VIP will be forwarded to port 23 on a realserver, thus no port number is used for setting up the IP of the realserver.

Here's the packet headers as the request is processed by LVS-Tun.

```
packet                  source       dest         data
1. request from client  CIP:3456     VIP:23       -
2. ipvsadm table:
   director chooses server=RIP1, encapsulates into IPIP packet
                        DIP          RIP1         IP datagram
                                                  source=CIP:3456,
                                                  dest=VIP:23,
                                                  data= -
3. realserver recovers IP datagram
```

```
                        CIP:3456      VIP:23        -
4. realserver looks up routing table, finds VIP is local,
   processes request locally, generates reply
                        VIP:23        CIP:3456     "login: "

5. packet leaves realserver via default gw, not via DIP.
```

For the verbally oriented...

A packet arrives for the VIP. The director looks up its tables and decides to send the connection to realserver_1. The director encapsulates the request packet in an IPIP datagram with DIP->RIP. The packet arrives at realserver1, the realserver recovers the original IP datagram, looks up its routing table, finds that the VIP (on an otherwise unused, non-arping and nonfunctional device) is local and processes the packet locally. A reply packet is generated with VIP:23->CIP:3456. The realserver looks up its routing table and finds that a packet to CIP goes out its default gw (not to the DIP).

The tunl0 device does not arp with 2.0.36 kernels, but does with 2.2.x kernels. Go look up the section on the 3 (arp problem) to see if you need to patch the kernel on the realserver.

14.2 Configure LVS-Tun

Edit the template lvs_tun.conf and run 10.1 (configure).

$ ./configure_lvs.pl lvs_nat.conf

Load the the parameters into the director and then the realservers with the command

$ . ./etc/rc.d/rc.lvs_tun

(the script knows whether it is running on a realserver or the director).

(later put in /etc/rc.d or /etc/init.d and put mon_xxx.cf in /etc/mon)

check the output from ipvsadm, ifconfig -a and netstat -rn, to see that the services/IP's are correct. If not re-edit and re-run the script(s).

14.3 Realservers on different network(s) to director

If the realserver is on a different network to the director, then it will be replying with packets having a source address of the VIP. In this case the router will have to be programmed to pass packets in the outward direction with the source address of the VIP. This may be a problem with some firewalls as they will be forwarding packets with a source address not in their network.

14.4 VS-Tun Questions

Joe How does a packet get to a tunl device from a remote machine without a MAC address?

Julian

tunl, lo and dummy are used just to configure the VIP. We don't send any packets through these devices. The requests are delivered to the real servers using their RIP. The director asks only about their RIP from ipvsadm. Only their gateway asks about VIP but only the director must reply. When the packet is received in the real server it is delivered locally (not forwarded or dropped) due to configured VIP. This is the only role of these "dummy" interfaces: the kernel to treat the received packet as it is destined to our host (the real server). Nothing more. No IPIP encapsulations (for tunl), no MAC address definitions, nothing more. When we answer the request we use eth0. The tunl/lo/dummy is not selected as device for the outgoing packets. We have routes for eth0 (default gateway) which we use for the outgoing traffic. This is for DROUTE and TUNNEL mode.

> If two linux boxes (not in an LVS) are joined by an IPIP tunnel and there is no MAC address associated with the tunl0 devices at each end of the link, then how do the packets get from one machine to the other?

Julian

The packets are encapsulated via IPIP and sent to the tunnel ends real IP where they are decapsulated again and appear on the tunl interface. You don't need a MAC address for point-to-point links, or logical interfaces like tunnels.

15 localnode

(We rarely hear of anyone using this.)

The director machine can be a realserver too. This is convenient when only a small number of machines are available as servers.

To use this feature, add a realserver with IP 127.0.0.1 (or a local IP on your director) to your lvs_xxx.conf file, rerun the 10.1 (configure script), and reinstall the rc.lvs_xxx files.

You will need to setup the service to listen to the VIP on the director. Some services, eg telnet listen on all IP's on the machine and you won't have to do anything special for them. Other services, eg http have to be specifically configured to listen to each IP. You are _not_ connecting to a service on 127.0.0.1 despite what this instruction might look like.

LocalNode operates independantly of NAT,TUN or DR modules.

15.1 You can't rewrite ports with localnode

Paul Monaghan wrote:

Okay, perhaps what I am trying to do can't work for what ever reason but here it is. I've setup ipvs rules as follows:

```
TCP 209.226.95.146:8080 wlc
       -> 10.0.0.1:8001         Local   1      1          0
       -> 10.0.0.1:8000         Local   1      0          0
```

> Wensong For the LocalNode feature, the load balancer just lets the packets pass to the upper layer (up to service daemon) to process the request. So, the port number of the local service must be equal to that of the virtual service, otherwise it won't work. Maybe I should add some code to check whether the port number of local service is equal to that of virtual service, if not, reject it to avoid such an error.

i.e. you can't use localnode with LVS-NAT and have it rewrite ports. You can't have a request coming to port VIP:80 and have it serviced by a demon listening on 127.0.0.1:81.

If you want to do this, you could try instead

```
$ipchains -A input -j REDIRECT 81 -d 192.168.1.12 80 -p tcp
```

Requests to 192.168.1.12:80 will go to 192.168.1.12:81 (more 16.2 (info is available)).

15.2 Testing LocalNode

If you want to explore installing localnode by hand, try this. First make sure scheduling is turned on at the director (this command adds round robin scheduling and direct routing)

```
#ipvsadm -A -t 192.168.1.110:23 -s rr
```

With an httpd listening on the VIP (192.168.1.110:80) of the director (192.168.1.1) AND with _no_ entries in the ipvsadm table, the director appears as a normal non-LVS node and you can connect to this service at 192.168.1.110:80 from an outside client. If you then add an external realserver to the ipvsadm table in the normal manner with

```
#/sbin/ipvsadm -a -t 192.168.1.110:80 -r 192.168.1.2
```

then connecting to 192.168.1.110:80 will display the webpage at the realserver 192.168.1.2:80 and not the director. This is easier to see if the pages are different (eg put the real IP of each machine at the top of the webpage).

Now comes the LocalNode part -

You can now add the director back into the ipvsadm table with

```
/sbin/ipvsadm -a -t 192.168.1.110:80 -r 127.0.0.1
```

(or replace 127.0.0.1 by another IP on the director)

Note, the port is the same for LocalNode. LocalNode is independant of the LVS mode (VS-NAT/Tun/DR) that you are using for the other IP:ports.

Shift-reloading the webpage at 192.168.1.110:80 will alternately display the wepages at the server 192.168.1.2 and the director at 192.168.1.1 (if the scheduling is unweighted round robin). If you remove the (external) server with

```
/sbin/ipvsadm -d -t 192.168.1.110:80 -r 192.168.1.2
```

you will connect to the LVS only at the directors port. The ipvsadm table will then look like

```
Protocol Local Addr:Port ==>
                        Remote Addr           Weight ActiveConns TotalConns
                        ...
TCP      192.168.1.110:80 ==>
                        127.0.0.1             2      3           3
```

From the client, you cannot tell whether you are connecting directly to the 192.168.1.110:80 socket or through the LVS code.

16 Transparent proxy (TP or Horms' method)

Transparent proxy is a piece of Linux kernel code which allows a packet destined for an IP _not_ on the host, to be accepted locally, as if the IP was on the host.

Note: Oct 2001. The TP kernel code for 2.4.x kernels behaves differently than the 2.2 code - see elsewhere in this section. 2.4 TP is unusable for LVS, although it works fine for web-caches (*i.e.* squids), it's original purpose. On talking to Harald Welte at the 2001 Ottawa Linux Symposium, there had been much discussion on the netfilter mailing lists as to whether to preserve the original behaviour. Since no-one that they knew about, needed the original behaviour, that functionality was dropped. It seems too late to restore the functionality to netfilter now. It's possible to patch the code for LVS, but this would require someone to keep track of the netfilter code for each version of the kernel. All the functionality that LVS wants out of TP is available via 9 (fwmark) and so the issue is probably moot now, and we're not going to ask the netfilter people to restore the original TP functionality for 2.4 kernels.

Take-home lesson: TP only works for LVS on 2.0 and 2.2 kernels.

Note: web caches (proxies) can operate in transparent mode, when they cache all IP's on the internet. In this mode, requests are received and transmitted without changing the port numbers (ie port 80 in and port 80 out). In a normal web cache, the clients are asked to reconfigure their browsers to use the proxy, some_IP:3128. It is difficult to get clients to do this, and the solution is transparent caching. This is more difficult to setup, but all clients will then use the cache.

In the web caching world, transparent caching is often called "transparent proxy" because it is implemented with transparent proxy. In the future, it is conceivable that transparent web caching will be implemented by another feature of the tcpip layer and it would be nice if functionality of transparent web caching had a name separate from the command that is used to implement it.

16.1 General

This is Horms' (`horms@vergenet.net`) method (now called the transparent proxy or TP method). It uses the transparent proxy feature of ipchains to accept packets with dst=VIP by the host (director or realservers) when it doesn't have the IP (eg the VIP) on a device. It can be used on the realservers (where it handles the 3 (arp problem)) or the director to accept packets for the VIP. When used on the director, TP allows the director to be the default gw for LVS-DR (see 13.6 (Julian's martian modification)).

Unfortunately the 2.2 and 2.4 versions of transparent proxy are as different as chalk and cheese in an LVS. Presumably the functionality has been maintained for for transparent web caching but the effect on LVS has not been considered.

You can use transparent proxy for

- 2.2.x, director and realservers
- 2.4.x, realservers only

(Historical note from Horms:) From memory I was getting a cluster ready for a demo at Inetnet World, New York which was held in October 1999. The cluster was to demo all sorts of services that Linux could run that were relevant to ISPs. Apache, Sendmail, Squid, Bind and Radius I believe. As part of this I was playing with LVS-DR and spotted that the real servers coulnd't accept traffic for the VIP. I had used Transparent Proxying in the past so I tried it and it worked. That cluster was pretty cool, it took me a week to put it together and it was an ISP in an albeit very large box.

Transparent proxy is only implemented in Linux.

- 2.2.x you need IP masquerading, transparent proxing and IP firewalls turned on.
- 2.4.x, TP is a standard part of the kernel build, there is no separate TP option. In the netfilter options, there are the options under "Full NAT (NEW)" MASQUERADE, REDIRECT. I suspect you need all these.

Julian

Transparent proxy support calls ip_local_deliver from where the LVS code is reached. One of the advantages of this method is that it is easy for a director and realserver to 3.12.3 (exchange roles) in a failover setup.

16.2 How you use TP

This is a demonstration of TP using 2 machines: a server (which will accept packets by TP) and a client.

On the server: ipv4 forwarding must be on.

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

You want your server to accept telnet requests on an IP that is not on the network (say 192.168.1.111). Here's the result of commands run at the server console before running the TP code, confirming that you can't ping or telnet to the IP.

```
server:# ping 192.168.1.111
PING 192.168.1.111 (192.168.1.111) from 192.168.1.11 : 56(84) bytes of data.
From server.mack.net (192.168.1.11): Destination Host Unreachable

server:# telnet 192.168.1.111
Trying 192.168.1.111...
telnet: Unable to connect to remote host: No route to host
```

so add a route and try again (lo works here, eth0 doesn't)

```
server:# route add -host 192.168.1.111 lo
server:# telnet 192.168.1.111
Trying 192.168.1.111...
Connected to 192.168.1.111.
Escape character is '^]'.

Welcome to Linux 2.2.16.
server login:
```

This shows that you can connect to the new IP from the localhost. No transparent proxy involved yet.

If you go to another machine on the same network and add a route to the new IP.

```
client:# route add -host 192.168.1.111 gw 192.168.1.11
client:# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.111   192.168.1.11    255.255.255.255 UGH       0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
```

raw sockets work between the client and server -

```
client:# traceroute 192.168.1.111
traceroute to 192.168.1.111 (192.168.1.111), 30 hops max, 40 byte packets
 1  server.mack.net (192.168.1.11)  0.634 ms  0.433 ms  0.561 ms
```

however you can't ping (i.e. icmp doesn't work) or telnet to that IP from the other machine.

```
client:# ping 192.168.1.111
PING 192.168.1.111 (192.168.1.111) from 192.168.1.9 : 56(84) bytes of data.
From server.mack.net (192.168.1.11): Time to live exceeded

client:# telnet 192.168.1.111
Trying 192.168.1.111...
telnet: Unable to connect to remote host: No route to host
```

Here's the output of tcpdump running on the target host

```
14:09:09.789132 client.mack.net.1101 > tip.mack.net.telnet: S 1088013012:1088013012(0) win 32120 <ms
14:09:09.791205 server.mack.net > client.mack.net: icmp: time exceeded in-transit [tos 0xd0]
```

(Anyone have an explanation for this, apart from the fact that icmp is not working? Is the lack of icmp the only thing stopping the telnet connect?)

The route to 192.168.1.111 is not needed for the next part.

```
server:# route del -host 192.168.1.111
```

Now add transparent proxy to the server to allow the server to accept connects to 192.168.1.111:telnet

This is the command for 2.2.x kernels

```
server:# ipchains -A input -j REDIRECT telnet -d 192.168.1.111 telnet -p tcp
server:# ipchains -L
Chain input (policy ACCEPT):
target     prot opt     source                destination          ports
REDIRECT   tcp  ------  anywhere             192.168.1.111         any ->   telnet => telnet
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

16.2.1 redirecting any port at all

In the normal functioning of an LVS, once the packet has been redirected, the director steps in and sends it to the realservers and the reply comes from the realservers. However you can use the REDIRECT to connect with a socket on a different port independantly of the LVS function.

Joe, 4 Jun 2001

If I have 2 boxes (not part of an LVS) and on the server box I run

```
$ipchains -A input -j REDIRECT telnet serverIP 81 -p tcp
```

then I can telnet to port 81 on the server box and have a normal telnet session. I watched with tcpdump on the server and all I see is a normal exchange of packets with dest-port=81.

I thought with REDIRECT that the packet with dest-port=81 was delivered to the listener on serverIP:telnet. How does the telnetd know to return a packet with source-port=telnet?

> Julian This is handled from the protocol, TCP in this case:
>
> ```
> grep redirport net/ipv4/*.c
> ```
>
> The higher layer (telnet in this case) can obtain the two dest addr/ports by using getsockname(). In 2.4 this is handled additionally by using getsockopt(...SO_ORIGINAL_DST...)
>
> The *netfilter mailing list* `<http://marc.theaimsgroup.com/?l=netfilter&r=1&w=2>` contains examples on this issue. You can search for "getsockname"

16.2.2 For 2.4.x kernels

```
server:# iptables -t nat -A PREROUTING -p tcp -d 192.168.1.111 --dport telnet -j REDIRECT
server:# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             192.168.1.111      tcp dpt:telnet

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

You still can't ping the transparent proxy IP on the server from the client

```
client:# ping 192.168.1.111
PING 192.168.1.111 (192.168.1.111) from 192.168.1.9 : 56(84) bytes of data.
From server.mack.net (192.168.1.11): Time to live exceeded
```

The transparent proxy IP on the server will accept telnet connects

```
client:# telnet 192.168.1.111
Trying 192.168.1.111...
Connected to 192.168.1.111.
Escape character is '^]'.

Welcome to Linux 2.2.16.
server login:
```

but not requests to other services

```
client:# ftp 192.168.1.111
ftp: connect: No route to host
ftp>
```

Conclusion: The new IP will only accept packets for the specified service. It won't ping and it won't accept packets for other services.

16.3 The original 2.2 TP setup method

```
                        --------
                       |        |
                       | client |
                       |________|
                       CIP=192.168.1.254
                           |
                        (router)
                           |
                 VIP=192.168.1.110 (eth0, arps)
                      ----------
                     |          |
                     | director |
                     |__________|
                     DIP=192.168.1.1 (eth1, arps)
                           |
                           |
          -------------------------------------
          |                |                  |
  RIP1=192.168.1.2  RIP2=192.168.1.3   RIP3=192.168.1.4 (eth0)
   -------------     -------------      -------------
  |             |   |             |    |             |
  | realserver  |   | realserver  |    | realserver  |
  |_____________|   |_____________|    |_____________|
          |                |                  |
      (router)          (router)           (router)
          |                |                  |
          ----------------------------------------------> to client
```

Here's a script to run on 2.2.x realservers/directors to setup Horms' method. This is incorporated into the 10.1 (configure script).

```
#!/bin/sh
#rc.horms
#script by Joseph Mack and Horms (C) 1999, released under GPL.
#Joseph Mack jmack@wm7d.net, Horms horms@vergenet.net
#This code is part of the Linux Virtual Server project
#http://www.linuxvirtualserver.org
#
#
#Horm's method for solving the LVS arp problem for a LVS-DR LVS.
#Uses ipchains to redirect a packet destined for an external
#machine (in this case the VIP) to the local device.

#-----------------------------------------------------
#Instructions:
#
#1. Director: Setup normally (eg turn on LVS services there with ipvsadm).
#2. Realservers: Must be running 2.2.x kernel.
```

```
# 2.1 recompile the kernel (and reboot) after turning on the following under "Networking options"
#       Network firewalls
#       IP: firewalling
#       IP: transparent proxy support
#       IP: masquerading
# 2.2 Setup the realserver as if it were a regular leaf node on the network,
#      ie with the same gateway and IP as if it were in the LVS, but DO NOT
#      put the VIP on the realserver. The realserver will only have its regular IP
#      (called the RIP in the HOWTO).
#3. Edit "user configurable" stuff below"
#4. Run this script
#-----------------------------------------------------
#user configurable stuff

IPCHAINS="/sbin/ipchains"
VIP="192.168.1.110"

#services can be represented by their name (in /etc/services) or a number
#SERVICES is a quote list of space separated strings
# eg SERVICES="telnet"
#    SERVICES="telnet 80"
#    SERVICES="telnet http"
#Since the service is redirected to the local device,
#make sure you have SERVICE listening on 127.0.0.1
#
SERVICES="telnet http"
#
#----------------------------------------------------
#main:

#turn on IP forwarding (off by default in 2.2.x kernels)
echo "1" > /proc/sys/net/ipv4/ip_forward

#flush ipchains table
$IPCHAINS -F input

#install SERVICES
for SERVICE in $SERVICES
do
        {
        echo "redirecting ${VIP}:${SERVICE} to local:${SERVICE}"
        $IPCHAINS -A input -j REDIRECT $SERVICE -d $VIP $SERVICE -p tcp
        }
done

#list ipchain rules
$IPCHAINS -L input

#rc.horms---------------------------------------------
```

Here's the conf file for a LVS-DR LVS using TP on both the director and the realservers. This is for a 2.2.x kernel director. (For a 2.4.x director, the VIP device can't be TP - TP doesn't work on a 2.4.x director).

```
#-------------------------------------
#lvs_dr.conf for TP on director and realserver
#you will have to add a host route or equivelent on the client/router
#so that packets for the VIP are routed to the director
LVS_TYPE=VS_DR
INITIAL_STATE=on
#note director VIP device is TP
VIP=TP lvs 255.255.255.255 lvs
DIP=eth0 dip 192.168.1.0 255.255.255.0 192.168.1.255
DIRECTOR_DEFAULT_GW=client
SERVICE=t telnet rr realserver1 realserver2
#note realserver VIP device is TP
SERVER_VIP_DEVICE=TP
SERVER_NET_DEVICE=eth0
SERVER_DEFAULT_GW=client
#----------end lvs_dr.conf------------------------------------
```

Here's the output from ipchains -L showing the redirects for just the 2.2.x director

```
Chain input (policy ACCEPT):
target     prot opt     source               destination          ports
REDIRECT   tcp  ------  anywhere            lvs2.mack.net        any ->   telnet => telnet
REDIRECT   tcp  ------  anywhere            lvs2.mack.net        any ->   telnet => telnet
REDIRECT   tcp  ------  anywhere            lvs2.mack.net        any ->   www => www
REDIRECT   tcp  ------  anywhere            lvs2.mack.net        any ->   www => www
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

16.4 Transparent proxy for 2.4.x

Transparent proxy for 2.2.x kernels is installed with ipchains.

For 2.4.x kernels transparent proxy is built on netfilter and is installed with iptables (you need iptables support in the kernel and the ip_tables module must be loaded). The ip_tables module is incompatible with the ipchains module (which in 2.4.x is available for compatibility with scripts written for 2.2.x kernels). If present, the ipchains module must be unloaded.

The command for installing transparent proxy with iptables for 2.4.x came from looking in Daniel Kiracofe's drk@unxsoft.com *Transparent Proxy with Squid mini-HOWTO* <http://www.unxsoft.com/TransparentProxy.html> and guessing the likely command. It turns out to be

```
director:# iptables -t nat -A PREROUTING [-i $SERVER_NET_DEVICE] \
         -d $VIP -p tcp --dport $SERVICE -j REDIRECT
```

(where $SERVICE = telnet, $SERVER_NET_DEVICE = eth0).

Here's the result of installing the VIP by transparent proxy on one of the realservers.

```
realserver:~# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
```

```
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             lvs2.mack.net      tcp dpt:telnet
REDIRECT   tcp  --  anywhere             lvs2.mack.net      tcp dpt:http

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

This works fine for the realserver allowing it to accept packets for the VIP, without having the VIP on an ethernet device (eg lo, eth0). If you do the same on the director, setup for an LVS with (say) telnet forwarded in the ipvsadm tables, then the telnet connect request from the client is accepted by the director, rather than forwarded by ipvs to the realservers (tcpdump sees a normal telnet login to the director). Apparently ipchains is sending the packets to a place that ipvs can't get at them.

> Joe I have got TP to work on a LVS-DR telnet 2.4 realserver with the command
>
> ```
> #iptables -t nat -A PREROUTING -p tcp -d $VIP --dport telnet -j REDIRECT
> ```
>
> When I put the VIP onto the director this way, the LVS doesn't work. I connect to the director instead of the realservers. ipvsadm doesn't show any connections (active or otherwise)
> If I run the same command on the director, with ipvsadm blank (ie no LVS configured), then I connect to the director from the client (as expected) getting the director's telnet login.
> I presume that I'm coming in at the wrong place in the input chain of the director and ipvsadm is not seeing the packets?

Julian

I haven't tried tproxy in 2.4 but in theory it can't work. The problem is that netfilter implements tproxy by mangling the destination address in the prerouting. LVS requires from the tproxy implementation only to deliver the packet locally and not to alter the header. So, I assume LVS detects the packets with daddr=local_addr and refuses to work.

Netfilter maintains a sockopt SO_ORIGINAL_DST that can be used from the user processes to obtain the original dest addr/port before they are mangled in the pre routing nat place. This can be used from the squids, for example, to obtain these original values.

If LVS wants to support this broken tproxy in netfilter we must make a lookup in netfilter to receive the original dst and then again to mangle (for 2nd time) the dst addr/port. IMO, this is very bad and requires LVS always to require netfilter nat because it will always depend on netfilter: LVS will be compiled to call netfilter functions from its modules.

So, the only alternative remains to receive packets with advanced routing with fwmark rules. There is one problem in 2.2 and 2.4 when the tproxy setups must return ICMP to the clients (they are internal in such setup), for example, when there is no real server LVS returns ICMP DEST_UNREACH:PORT_UNREACH. In this case both kernels mute and don't return the ICMP. icmp_send() drops it. I contacted Alexey Kuznetsov, the net maintainer, but he claims there are more such places that must be fixed and "ip route add table 100 local 0/0 dev lo" is not a good command to use. But in my tests I don't have any problems, only the problem with dropped ICMP replies from the director.

So, for TP, I'm not sure if we can support it in the director. May be it can work for the real servers and even when the packet is mangled I don't expect peformance problems but who knows.

16.5 Transparent proxy Q&A

Q. How does a packet get to the host which is accepting packets for the VIP if the host doesn't have the VIP on an eth device to respond to arp requests?

A. You have to send it there.

On the realservers in a LVS-DR LVS this is already handled for you. In LVS-DR when the director wants to send a packet to a realserver, it looks up the MAC address for the RIP and sends the packet with addresses CIP->VIP by linklayer to the MAC address of the realserver. Once the packet arrives at the realserver, processes listening to VIP:port pick up the packet.

As pointed out by Julian, you can put a transparent proxy IP onto any host, so you can use transparent proxy to put the VIP onto the director if you like (for 2.2.x kernels). To receive the packet for the VIP, the director has to be the default gw for the network in the incoming router's table (or the router needs a host route for the VIP to the director).

The same problem exists when 9.10 (accepting packets with fwmark).

Q. Some demons listen only to specific IPs. What IP is the telnet/httpd listening to when it accepts a connection by transparent proxy?

A. It depends where you are when you make the connect request (this is for 2.2.x kernels).

example:

You are on the console of a host and add x.x.x.111:http by transparent proxy and setup the httpd to listen to x.x.x.111:80. You cannot ping x.x.x.111. To connect to x.x.x.111:http you need to

route add -host 192.168.1.111 lo

(adding a route to eth0 does not work).

If you go to an outside machine, you still cannot ping x.x.x.111 and you cannot connect to x.x.x.111:http unless you make the target box the default gw or add a host route to x.x.x.111.

If you now go back to the console of the transparent proxy machine and change the httpd to listen to 127.0.0.1:http (and not to x.x.x.111:http) you can still connect to x.x.x.111:http even though nothing is listening to that IP:port (linux tcpip does local delivery to local IPs). (You can also connect to 127.0.0.1:http, but this is not concerned with transparent proxy.)

Returning to the outside machine, you cannot connect to x.x.x.111:http.

The connections from the outside machine model connections to the director with the VIP by transparent proxy, while the connections from the console model the realserver which has a packet delivered from the director. On the realserver you could have your services listening to 127.0.0.1 rather than the VIP. You may run into DNS problems (see 11.18.7 (Running indexing programs)) if the process listening to 127.0.0.1 doesn't know that it's answering to lvs.domain.org.

16.6 Experiments showing that 2.4TP is different to 2.2TP

These experiments were conducted with 2.2 or 2.4 kernel realservers accepting packets for the VIP by TP. I initially noticed that the connection to 2.4 realservers was not delayed by identd (which is running on my realservers). What was happening was that the realserver was accepting the packet at the RIP and generating the reply from the RIP, rather than the VIP. On my setup, the RIP is routable to the client and the client probably received the identd request directly from the realserver (I didn't figure out what was going on for a while after I did this. I originally thought this had something to do with identd).

Here's the data showing that TP behaves differently for 2.2 and 2.4 kernels. If you want to 16.7 (skip ahead), the piece of information you need is that the IP of the packet when it arrives on the target machine by TP, is different for 2.2 and 2.4 TP.

As we shall see, for 2.2.x the TP'ed packets arrive on the VIP, while for 2.4.x, the TP'ed packets arrive on the RIP.

16.6.1 Realserver, Linux 2.4.2 kernel accepting packets for VIP on lo:110, is delayed

Here's the tcpdump on the realserver (bashfull) for a telnet request delayed by authd (the normal result for LVS). Realserver 2.4.2 with Julian's hidden patch, director 0.2.5-2.4.1. The VIP on the realserver is on lo:110.

Note: all packets on the realserver are originating and arriving on the VIP (lvs2) as expected for a LVS-DR LVS.

```
initial telnet request

21:04:46.602568 client2.1174 > lvs2.telnet: S 461063207:461063207(0) win 32120 <mss 1460,sackOK,time
21:04:46.611841 lvs2.telnet > client2.1174: S 3724125196:3724125196(0) ack 461063208 win 5792 <mss 1
21:04:46.612272 client2.1174 > lvs2.telnet: . ack 1 win 32120 <nop,nop,timestamp 17832676 514409> (D
21:04:46.613965 client2.1174 > lvs2.telnet: P 1:28(27) ack 1 win 32120 <nop,nop,timestamp 17832676 5
21:04:46.614225 lvs2.telnet > client2.1174: . ack 28 win 5792 <nop,nop,timestamp 514409 17832676> (D

realserver makes authd request to client

21:04:46.651500 lvs2.1061 > client2.auth: S 3738365114:3738365114(0) win 5840 <mss 1460,sackOK,times
21:04:49.651162 lvs2.1061 > client2.auth: S 3738365114:3738365114(0) win 5840 <mss 1460,sackOK,times
21:04:55.651924 lvs2.1061 > client2.auth: S 3738365114:3738365114(0) win 5840 <mss 1460,sackOK,times

after delay of 10secs, telnet request continues

21:04:56.687334 lvs2.telnet > client2.1174: P 1:13(12) ack 28 win 5792 <nop,nop,timestamp 515416 178
21:04:56.687796 client2.1174 > lvs2.telnet: . ack 13 win 32120 <nop,nop,timestamp 17833684 515416> (
```

16.6.2 Realserver, Linux 2.4.2, accepting packets for VIP by TP, is not delayed

Here's the tcpdump on the realserver (bashfull) for a telnet request which connects immediately. This is not the normal result for LVS. Realserver 2.4.2 with Julian's hidden patch (not used), director 0.2.5-2.4.1. Packets on the VIP are being accepted by TP rather than on lo:0 (the only difference).

Note: some packets on the realserver (bashfull) are arriving and originating on the VIP (lvs2) and some on the RIP (bashfull). In particular all telnet packets from the CIP are arriving on the RIP, while all telnet packets from the realserver are originating on the VIP. For authd, all packets to and from the realserver are using the RIP.

```
initial telnet request

20:56:43.638602 client2.1169 > bashfull.telnet: S 4245054245:4245054245(0) win 32120 <mss 1460,sackO
20:56:43.639209 lvs2.telnet > client2.1169: S 3234171121:3234171121(0) ack 4245054246 win 5792 <mss
20:56:43.639654 client2.1169 > bashfull.telnet: . ack 3234171122 win 32120 <nop,nop,timestamp 177843
20:56:43.641370 client2.1169 > bashfull.telnet: P 0:27(27) ack 1 win 32120 <nop,nop,timestamp 177843
20:56:43.641740 lvs2.telnet > client2.1169: . ack 28 win 5792 <nop,nop,timestamp 466118 17784380> (D
realserver makes authd request to client

20:56:43.690523 bashfull.1057 > client2.auth: S 3231319041:3231319041(0) win 5840 <mss 1460,sackOK,t
20:56:43.690785 client2.auth > bashfull.1057: S 4243940839:4243940839(0) ack 3231319042 win 32120 <m
20:56:43.691125 bashfull.1057 > client2.auth: . ack 1 win 5840 <nop,nop,timestamp 466123 17784385> (
20:56:43.692638 bashfull.1057 > client2.auth: P 1:10(9) ack 1 win 5840 <nop,nop,timestamp 466123 177
20:56:43.692904 client2.auth > bashfull.1057: . ack 10 win 32120 <nop,nop,timestamp 17784385 466123>
20:56:43.797085 client2.auth > bashfull.1057: P 1:30(29) ack 10 win 32120 <nop,nop,timestamp 1778439
20:56:43.797453 client2.auth > bashfull.1057: F 30:30(0) ack 10 win 32120 <nop,nop,timestamp 1778439
20:56:43.798336 bashfull.1057 > client2.auth: . ack 30 win 5840 <nop,nop,timestamp 466134 17784395>
20:56:43.799519 bashfull.1057 > client2.auth: F 10:10(0) ack 31 win 5840 <nop,nop,timestamp 466134 1
20:56:43.799738 client2.auth > bashfull.1057: . ack 11 win 32120 <nop,nop,timestamp 17784396 466134>

telnet connect continues, no delay

20:56:43.835153 lvs2.telnet > client2.1169: P 1:13(12) ack 28 win 5792 <nop,nop,timestamp 466137 177
20:56:43.835587 client2.1169 > bashfull.telnet: . ack 13 win 32120 <nop,nop,timestamp 17784399 46613
```

Evidently TP on the realserver is making the realserver think that the packets arrived on the RIP, hence the authd call is made from the RIP.

As it happens in my test setup, the client can connect directly to the RIP. (In a LVS-DR LVS, the client doesn't exchange packets with the RIP, so I haven't blocked this connection. In production, the router would not allow these packets to pass). Since the authd packets are between the RIP and CIP, the authd exchange can proceed to completion.

16.6.3 Realserver, Linux 2.2.14, accepting packets for VIP by TP, is delayed

Here's the tcpdump on the realserver (bashfull) for a telnet request which connects immediately. This is not the normal result for LVS. Realserver 2.2.14, director 0.2.5-2.4.1. Packets on the VIP are being accepted by TP rather than on lo:0.

Note: TP is different in 2.2 and 2.4 kernels. Unlike the case for the 2.4.2 realserver, the packets all arrive at the RIP.

```
initial telnet request

22:16:23.407607 client2.1177 > lvs2.telnet: S 707028448:707028448(0) win 32120 <mss 1460,sackOK,time
22:16:23.407955 lvs2.telnet > client2.1177: S 3961823491:3961823491(0) ack 707028449 win 32120 <mss
22:16:23.408385 client2.1177 > lvs2.telnet: . ack 1 win 32120 <nop,nop,timestamp 18262396 21648> (DF
22:16:23.410096 client2.1177 > lvs2.telnet: P 1:28(27) ack 1 win 32120 <nop,nop,timestamp 18262396 2
22:16:23.410343 lvs2.telnet > client2.1177: . ack 28 win 32120 <nop,nop,timestamp 21648 18262396> (D

authd request from realserver

22:16:23.446286 lvs2.1028 > client2.auth: S 3966896438:3966896438(0) win 32120 <mss 1460,sackOK,time
22:16:26.445701 lvs2.1028 > client2.auth: S 3966896438:3966896438(0) win 32120 <mss 1460,sackOK,time
22:16:32.446212 lvs2.1028 > client2.auth: S 3966896438:3966896438(0) win 32120 <mss 1460,sackOK,time

after delay of 10secs, telnet proceeds
```

```
22:16:33.481936 lvs2.telnet > client2.1177: P 1:13(12) ack 28 win 32120 <nop,nop,timestamp 22655 182
22:16:33.482414 client2.1177 > lvs2.telnet: . ack 13 win 32120 <nop,nop,timestamp 18263404 22655> (D
```

16.7 What IP TP packets arriving on?

Note: for TP, there is no VIP on the realservers as seen by ifconfig.

Since telnetd on the realservers listens on 0.0.0.0, we can't tell which IP the packets have on the realserver after being TP'ed. tcpdump only tells you the src_addr after the packets have left the sending host.

Here's the setup for the test.

The IP of the packets after arriving by TP was tested by varying the IP (localhost, RIP or VIP) that the httpd listens to on the realservers. At the same time the base address of the web page was changed to be the same as the IP that the httpd was listening to. The nodes on each network link can route to and ping each other (eg 192.168.1.254 and 192.168.1.12).

```
        ____________
       |            |192.168.1.254 (eth1)
       |  client    |----------------------
       |____________|                     |
     CIP=192.168.2.254 (eth0)             |
              |                           |
              |                           |
     VIP=192.168.2.110 (eth0)             |
        ____________                      |
       |            |                     |
       |  director  |                     |
       |____________|                     |
     DIP=192.168.1.9 (eth1, arps)         |
              |                           |
           (switch)------------------------
              |
     RIP=192.168.1.12 (eth0)
     VIP=192.168.2.110 (VS-DR, lo:0, hidden)
        _____________
       |             |
       | realserver  |
       |_____________|
```

The results (VS-DR LVS) are

For 2.2.x realservers

- the httpd can bind to the VIP, RIP and localhost.
- LVS client gets webpage if realserver is listening to RIP or VIP.
- LVS client does not get webpage if realserver is listening to localhost.

For 2.4.x realservers

- httpd can bind to the RIP and localhost.

- httpd cannot bind to the VIP.
- LVS client gets webpage if realserver is listening to RIP.
- LVS client does not get webpage if realserver is listening to localhost.

During tests, the browser says "connecting to VIP", then says "transferring from..."

- VS-DR, VIP on TP, kernel 2.4.2, "transferring data from RIP"
- VS-DR, VIP on TP, kernel 2.2.14, "transferring data from VIP" (or RIP)
- VS-DR VIP on lo:0, httpd listening to VIP, "transferring data from VIP"
- VS-Tun VIP on tunl0:0, httpd listening on VIP, "transferring from VIP"
- VS-NAT, httpd listening on RIP, "transferring data from realserver1" (or realserver2)

Some of these connections are problematic. The client in a LVS-DR LVS isn't supposed to be getting packets from the RIP. What is happening is

- the httpd on the realserver is listening on the RIP
- the base address of the webpage is the RIP
- an incoming request from the client to the VIP will retrieve a webpage with references to gif etc that are at the RIP
- the client will then ask for the gifs from the RIP.
- in the above setup that I use for testing, the client does not request packets from the RIP.
- in the above setup, the client can connect to the RIP directly (this will not be allowed in a production server, either the router will prevent the connection, or the RIP will be a non-routable IP).
- the client retrieves the gifs, and the rest of the page, directly from the realserver

The way to prevent this is to remove the route on the client to the RIP network (eg see 13.8 (removing routes not needed for LVS-DR)). Doing so when the httpd is listening to the RIP and the base address is the RIP causes the browser on the client to hang. This shows that the client is really retrieving packets directly from the RIP. Changing the base address of the webpage back to the VIP allows the webpage to be delivered to the client, showing that the client is now retrieving packets by making requests to the VIP via the director.

It would seem then that with 2.4 TP, the realserver is receiving packets on the RIP, rather than the VIP as it does with 2.2 TP. With a service listening to only 1 port (eg httpd) then the httpd has to

- listen on the RIP
- the addresses on the webpage have to be for the VIP

The client will then ask for the webpage at the VIP. The realserver will accept this request on the RIP and return a webpage full of references to the VIP (eg gifs). The client will then ask for the gifs from the VIP. The realserver will accept the requests on the RIP and return the gifs.

16.8 Take home lesson for setting up TP on realservers

16.8.1 2.2.x

Let httpd listen on VIP or RIP, return pages with references to VIP

16.8.2 2.4.x

Let httpd listen on RIP, return pages with references to VIP.

16.9 Handling identd requests from 2.4.x LVS-DR realservers using TP

Since the identd request is coming from the RIP (rather than the VIP) on the realserver, you can use Julian's 13.10 (method for NAT'ing client requests from realservers).

16.10 Performance of Transparent Proxy

Using transparent proxy instead of a regular ethernet device has slightly higher latency, but the same maximum throughput.

For performance of transparent proxy compared to accepting packets on an ethernet device see the *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>`.

Transparent proxy requires reprocessing of incoming packets, and could have a similar speed penalty as LVS-NAT. However only the incoming packets are reprocessed. Initial results (before the performance tests above) were initially not encouraging.

> Doug Bagley `doug@deja.com` Subject: [lvs-users] chosen arp problem solution can apparently affect performance
>
> I was interested in seeing if the linux/ipchains workaround for the arp problem would perform just as well as the arp_invisible kernel patch. It is apparently much worse.
>
> I ran a test with one client running ab ("apache benchmark"), one director, and one realserver running Apache. They are all various levels of pentium desktop machines running 2.2.13.
>
> Using the arp_invisible patch/dummy0 interface, I get 226 HTTP requests/second. Using the ipchains redirect method, I get 70 requests per second. All other things remained the same during the test.

See the *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>` for discussion and sample graphs of hits/sec for http servers. Hits/sec can increase to high levels as the payload decreases in size. While large numbers for hits/sec may be impressive, they only indicate one aspect of a web server's performance. If large (> 1 packet) files are transferred/hit or computation is involved, then hits/sec is not a useful measure of web performance.

Here's the current explanation for decreased latency of transparent proxy.

> Kyle Sparger `ksparger@dialtoneinternet.net` Logically, it's just a function of the way the redirect code operates.

```
Without redirect:
Ethernet -> TCP/IP -> Application -> TCP/IP -> Ethernet
```

```
With redirect:
Ethernet -> TCP/IP -> Firewall/Redirect Code -> TCP/IP -> Application -> TCP/IP -> Ethernet
```

> That would definitely explain the slowdown, since _every single packet_ received is going to go through these extra steps.

Other people are happy with TP

> Jerry Glomph Black `black@real.com` Nov 99 (or thereabouts) The revival of Horms' posting, which I overlooked a month ago, was a lifesaver for us. We had a monster load distribution problem, and spread 4 virtual IP numbers across 10 'real' boxes (running Roxen, a fantastic web platform). The ipchains-REDIRECT feature works perfectly, without any of that arp aggravation! A PII_450 held up just fine at 20 megabits/s of HTTP -REQUEST- TRAFFIC!

Here's Jerry 18 months later.

> Jerry Glomph Black `black@prognet.com` 06 Jul 2001 The ipchains/iptables REDIRECT method (introduced to this list by Mr Horms a long time ago) works fine, we've used it in production in the past.
>
> However, at -very- high packet loads it is far less CPU-efficient than getting the ARP settings correctly working. The REDIRECT method was bogging down our LVS boxes during peak traffic, something which does not happen with doing it the 'right way' with LVS-DR and silent arp-less interfaces on the real servers.

17 Authd/Identd

If initial connection to your service (telnet, ftp, sendmail...) is delayed by 10secs..5mins, but after you connect everything is fine, then you have problems with identd.

17.1 What is authd/identd?

identd is a demon run under inetd. Other services running on the server can use identd to ask the client machine for the identity of the user making the request. When a request arrives at a server for such a service (e.g. telnet, sendmail), the auth client will connect from a high port to client:auth asking "who is the owner of the process requesting this service". If the client's authd replies with a `username@nodename`, the reply will be optionally logged on the server (eg to syslog) and the connection request will be handed back to telnetd (or whichever service). If the reply is `"root@nodename"`, or some null reply, or there is no authd on the client, then the server's authd will wait till a timeout before allowing connection. The delay is about 10secs for Slackware and 2mins for RedHat7.0. There is no checking of the validity of the reply and since the reply is under control of the client machine, the reply `username@nodename` could be bogus.

The authd is a security feature. However it doesn't get the server very much (you don't know who has made the connect request, only what they told you), while clients that fail are delayed. This may only be a nuisance for people telnet'ing in (provided they understand what's happening), but will bring mail delivery to a crawl.

If you setup an LVS with realservers that have services running inside identd, you will have to deal with identd. Any service in inetd running under tcpwrappers (probably just about every service, if tcpwrappers is installed) and sendmail (see section on sendmail) use it.

Since problems with identd affect many aspects of an LVS, there are references to identd in several places in this document.

17.2 symptoms of the ident problem

There are two parts to identd on your servers

- Identd runs on your realservers. This isn't a problem for LVS. Identd on the realservers is for clients on your realservers connecting to services on remote machines. These clients will be connecting from the RIP and not the VIP. You aren't using this identd when setting up an LVS. However if you telnet from your realservers for some other reason, you'll need to think about what this identd is doing.
- your LVS'ed services *may* (*e.g.* sendmail or services running inside tcpwrappers), ask the identd client on your server to connect to the identd on the client machine and ask for the identity of the person connecting to the service on the realserver. You don't want this. In general there is no way in an LVS, for the reply from the client to return to the realserver.

The problem is in the second part, *i.e.* if the LVS'ed service on the realserver asks for the ident client to connect to the identd on the client. (If this is confusing, remember machines can be clients and servers at the same time.)

Here's a example telnet connection through a director to a realserver where telnetd is running inside tcpwrappers. tcpwrappers uses the ident client on the remote host (the one with the telnetd) to connect to the identd on the local (telnet client) host.

```
client:/director/usr/src/arch# telnet lvs2
Trying 192.168.2.110...
Connected to lvs2.mack.net.
Escape character is '^]'.

(delay)
Welcome to Linux 2.2.19.


bashfull login:
(successful login)
```

17.3 comp.os.linux.security FAQ on identd

```
comp.os.linux.security FAQ
Daniel Swan <tt/swan_daniel@hotmail.com/
v0.1 - Last updated:  April 20, 2000
```

4.5) What is Identd? Can I disable it?

Identd identifies the username of a process owning a specific TCP/IP connection. It is usually run via inetd and listens on port 113. Identd should not be used as a method of authentication - anyone with root access can alter their identd response. Indeed, on many systems (such as FreeBSD and Windows) even a non-privledged user can specify whatever identd response they want. The protocol is most useful on multiuser systems as a method of tracking down problem users. If one of your users is causing problems on another system, that system's admin can inform you of the username of the specific user causing problems, saving you a lot of legwork. Should you run identd? That's really a judgement call. On systems with many users, the benefits could be great, but it doesn't serve any particular purpose on a single user box. Not running identd may limit your ability to connect to certain servers - many IRC and some FTP servers don't allow, or severly restrict, non-identd'd connections, for example. However, running it means leaving a service open to the outside world, with all the security risks that entails. Another thing to consider is that identd can allow attackers to find out valuable information about your system, such as whether a certain service is running as root, the operating system you are running, and the usernames of your users. Consider running identd with the -n flag, which sends userid numbers instead of usernames. See the identd manpage and /etc/identd.conf for more information about the available options. You can block access to identd by shutting it off entirely (usually done via inetd, see section on disabling services), or by using tcpwrappers and/or firewalling software to disable/restrict access. If you need identd enabled in order to connect to a certain server, you might want to consider allowing access to it only from that server. If you do choose to firewall the identd port, strongly consider using a reject policy rather than deny. Using deny may greatly increase the time it takes you to connect to servers that utilize identd, as they will wait for a response of some type before allowing you to connect.

17.4 Why identd is a problem for LVS

The problem is that the identd/authd client makes a callback from the RIP (for LVS-NAT) or the VIP (VS-DR, LVS-Tun) and LVS doesn't handle clients on realservers. For the simple case where clients call from the RIP on NAT'ed realserver see the section on 25.19 (running clients on realservers). There the client is independant of the LVS.

The case of clients on the realservers making call backs triggered by an LVS client's requests to an LVS'ed service is more difficult as the result has to get back to the LVS'ed service.

Normally in an LVS, the director in an LVS responds to connect requests by handing them to an arbitrary realserver. The corrollary of this is that replies to a client request initiated on a realserver, to the outside world, will not return to the realserver unless something is done to handle it. (The only solutions we have are those in the section on 25.19 (running clients on realservers).)

- replies from the client which is connecting to the LVS, arriving at the director are not connect requests, and will not belong to an established connection. They will be dropped.
- even if the director could forward these replies to a realserver, they could go to any realserver, and not neccessarily to the realserver which originated the request.

The result is that the client request will hang or timeout.

17.5 tcpdumps of connections delayed by identd

Here's the tcpdump of the client telnet'ing to a LVS-DR LVS. Telnet on the realserver is running inside tcpwrappers, client and realservers cannot connect directly *i.e.* they have no routing to each other.

seen from client:

```
telnet connect request

12:56:05.427252 client2.1038 > lvs.telnet: S 1170880662:1170880662(0) win 32120 <mss 1460,sackOK,tim
12:56:05.427949 client2.1038 > lvs.telnet: . ack 416490630 win 32120 <nop,nop,timestamp 6539901 1618
12:56:05.431752 client2.1038 > lvs.telnet: P 0:27(27) ack 1 win 32120 <nop,nop,timestamp 6539902 161

client replying to realserver's auth request

12:56:05.465152 client2.auth > lvs.1377: S 1159930752:1159930752(0) ack 417813448 win 32120 <mss 146
12:56:05.465405 lvs.1377 > client2.auth: R 417813448:417813448(0) win 0
```

```
12:56:08.464671 client2.auth > lvs.1377: S 1162930275:1162930275(0) ack 417813448 win 32120 <mss 146
12:56:08.464901 lvs.1377 > client2.auth: R 417813448:417813448(0) win 0

6 second delay then trying again

12:56:14.466048 client2.auth > lvs.1377: S 1168931649:1168931649(0) ack 417813448 win 32120 <mss 146
12:56:14.466275 lvs.1377 > client2.auth: R 417813448:417813448(0) win 0

client login to LVS

12:56:15.501272 client2.1038 > lvs.telnet: . ack 13 win 32120 <nop,nop,timestamp 6540908 161875546>
12:56:15.503946 client2.1038 > lvs.telnet: P 27:125(98) ack 52 win 32120 <nop,nop,timestamp 6540909
12:56:15.509024 client2.1038 > lvs.telnet: P 125:128(3) ack 55 win 32120 <nop,nop,timestamp 6540909
12:56:15.538816 client2.1038 > lvs.telnet: P 128:131(3) ack 88 win 32120 <nop,nop,timestamp 6540912
12:56:15.551836 client2.1038 > lvs.telnet: . ack 90 win 32120 <nop,nop,timestamp 6540914 161875550>
12:56:15.571837 client2.1038 > lvs.telnet: . ack 106 win 32120 <nop,nop,timestamp 6540916 161875551>
```

Here's what it looks like on the realserver (this is a different connection from the above sample, so the times are not the same).

```
realserver receives telnet request on VIP
12:50:58.049909 client2.1040 > lvs.telnet: S 1605709966:1605709966(0) win 32120 <mss 1460,sackOK,tim
12:50:58.051263 lvs.telnet > client2.1040: S 862075007:862075007(0) ack 1605709967 win 32120 <mss 14
12:50:58.051661 client2.1040 > lvs.telnet: . ack 1 win 32120 <nop,nop,timestamp 6580274 161914907> (
12:50:58.052819 client2.1040 > lvs.telnet: P 1:28(27) ack 1 win 32120 <nop,nop,timestamp 6580274 161
12:50:58.053036 lvs.telnet > client2.1040: . ack 28 win 32120 <nop,nop,timestamp 161914907 6580274>

realserver initiates auth request from VIP to client:auth

12:50:58.088510 lvs.1379 > client2.auth: S 852509908:852509908(0) win 32120 <mss 1460,sackOK,timesta
12:51:01.083659 lvs.1379 > client2.auth: S 852509908:852509908(0) win 32120 <mss 1460,sackOK,timesta

realserver waits for timeout (about 8secs), sends final request to client:auth

12:51:07.083164 lvs.1379 > client2.auth: S 852509908:852509908(0) win 32120 <mss 1460,sackOK,timesta

telnet replies from realserver continue, login occurs

12:51:08.117727 lvs.telnet > client2.1040: P 1:13(12) ack 28 win 32120 <nop,nop,timestamp 161915914
12:51:08.118142 client2.1040 > lvs.telnet: . ack 13 win 32120 <nop,nop,timestamp 6581281 161915914>
```

17.6 There are solutions to identd problem in some cases

17.6.1 Director is LVS-NAT

In an LVS, authd on the realserver will be able to connect to the client if -

VS-NAT, the realservers are on public IPs (not likely, since you usually hide the realservers from public view and they'll be on 192.168.x.x or 10.x.x.x networks)

VS-NAT, and high ports are nat'ed out with a command like

```
director:/etc/lvs# ipchains -A forward -j MASQ -s 192.168.1.0/24 -d 0.0.0.0/0
```

You usually don't want to blanket masquerade all ports. You really only want to masquerade ports that are being LVS'ed (so you can still get to the other services) in which case, for each service being LVS'ed, you to use ipchains rules like

```
director:# ipchains -A forward -p tcp -j MASQ -s realserver1 telnet -d 0.0.0.0/0
```

Since the auth client (on your telnet server) is connecting from a high port on the server, a better ipchains rule which will allow auth to work when the realservers are on private IPs.

```
director:# ipchains -A forward -p tcp -j MASQ -s realserver1 1024:65535 -d 0.0.0.0/0
```

17.6.2 VS-DR, LVS-Tun, 2.2.x kernel directors

There is no solution for LVS-DR for 2.2.x directors. The auth client on the realserver initiates the connection from the VIP. There is no way for a packet from VIP:high port to get a reply through the LVS because

- the incoming packet from the client on the internet is destined for a non-LVS'ed high port
- the incoming packet is not a connect request.
- the incoming packet is not associated with an established connection.

The reply from the LVS client will be dropped.

17.6.3 VS-DR, LVS-Tun, 2.4.x kernel directors

Transparent proxy in 2.4 is different to 2.2 (see section on 16.6 (identd with 2.4 TP)). You should be able to 16.9 (masquerade the identd client's request on the realserver).

17.7 Turn off tcpwrappers

The best cure is to turn off tcpwrappers. inetd.conf will have a line like

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

change this to

```
telnet stream tcp nowait root /usr/sbin/in.telnetd in.telnetd
```

and re-HUP inetd.

17.8 Identd and smtp/pop/qmail

(This is older stuff from the mailing list when the problem first came up)

Problem: In the case of identd, the smtpd on a realserver says to identd "give me the name of the owner of the process on IP:port that is asking me to accept mail". If identd thinks it is running on the RIP rather than the VIP and, as is most likely, RIP is not routable from the outside world, mail on the realserver will hang. If identd is running on the VIP, then replies will probably return to another realserver and mail will still hang.

The converse case, of sending mail from the LVS, has the smtp server out in internetland asking the LVS for the name of the owner of the process running on the VIP sending him mail. If identd is clustered, then the request will in all probability go to another realserver. This seems equally intractable at the moment.

Originally the problem was raised by

```
Chris Kennedy <tt/ckennedy@iland.net/
Subject: SMTP, POP3 using Qmail and Ident, also using Solaris as real servers
```

I have setup a virtual server using the Linux 2.2 patch and 3 Sun Ultras as the actual servers. It has crashed twice, though possibly from running bind on the Virtual Server, since it was right when I started it up (bind) that the virtual server would crash.

The major problem I am having is a timeout for Ident requests on POP3 and SMTP ports which seem to be confused. When looking at the problem with tcpdump on the virtual server and the real servers the vserver seems to do the following:

```
13:41:48.635985 10.0.0.1.4658 > vserver.net.smtp: . ack 2764990963 win 8760
13:41:48.636030 10.0.0.1.4658 > vserver.net.smtp: . ack 1 win 8760
13:41:48.658875 10.0.0.1.auth > vserver.net.48981: R 0:0(0) ack 2765099549
win 0  <<<<<<
13:41:52.143790 10.0.0.1.auth > vserver.net.48981: R 0:0(0) ack 1 win 0
<<<<<<
13:41:58.144210 10.0.0.1.auth > vserver.net.48981: R 0:0(0) ack 1 win 0
<<<<<<
```

The Ident, or auth port on the client machine trying to connect back to the vserver is where it will pause for about 10-15 seconds then connect just fine. I believe this may be qmail specific since a server funning sendmail will not have this problem and ident seems to be used by qmail more than it or something.

No-one answered, then months later...

> Ted I currently have HTTP loadbalanced just fine with the LinuxDirector. I've setup SMTP in the same fashion, and I don't have as much luck.

Lars

Check if your system (tcpwrapper or sendmail) is doing a NS lookup before accepting the connection or trying to connect to ident.

```
Chris Kennedy <tt/ckennedy@iland.net/
Subject: Re: SMTP -- very slow connection
```

I had the same problem with the Direct Routing and SMTP and POP3. It looked like a problem with the Ident lookup to the server by the client, it was what always was occurring during that time out period. I saw this while doing tcpdumps on the virtual server where the client would just keep asking for Ident lookups to the Virtual IP address which are from the client port 113 to a random port above 1023 on the virtual server. I can see how this is tricky with the direct routing method since this traffic should be sent on to the real server but is not. I sort of gave up on Direct Routing for now since this looks pretty hard to fix if it really is Ident and the client requirements getting in the way.

> Ted I'm connecting directly to the IP. But just to be sure, I'll add an entry to the nameserver for that particular IP – in both forward and reverse lookups...........done.....
>
> And it still does the same thing. :(
>
> Understand that the thirty seconds are *AFTER* the connection... Telnet connects, gives me the escape character, and sits. If it was a nameservice thing, I'd imagine it'd sit before it connected. I'd actually be happier if it wasn't connecting. :) Then I'd know there was definitely something I needed to fix between me and the real machine. But when it connects and THEN has trouble... I'm lost. :(

Michael Baird `mike@tc3net.com`

Sound's like an issue with ident lookup's, you probably aren't clustering IDENT, you can 1) cluster identd, or edit your sendmail.cf file and set the value

```
0 Timeout.ident=0
```

> Ted Great idea. That was it. I turned off ident in sendmail and things worked fine. However, I don't want to turn of ident in sendmail, and I figure other things might want ident too... so I want to cluster ident. Clustering ident didn't help. I clustered tcp port 113 to both servers. (I even tried "loadbalancing" 25 and 113 just to ONE server – that way it'd always hit the same server)... And that didn't work. I got the same results – telnet to port 25... connect... thirty seconds... and then sendmail would enter command mode.
>
> Any ideas? Do I have to loadbalance anything else besides tcp 113 for identd to work?
>
> Why is identd run with smtp (any other reason other than wanting to know who is sending me the mail?) Do you have to turn identd off in smtp to get LVS smtp to work? Has anyone LVS'ed identd? (I'd imagine you wouldnt neccessarily get the ident from the same machine running the process for which you want the ident)

18 Squid Real-Servers (poor man's L7 switch)

One of the first uses for LVS was to increase throughput of webcaches. A 5.2 (550MHz PIII director can handle 120Mbps throughput).

A scheduler (-dh = destination hash) specially 7 (designed for webcaches) is in LVS derived from code posted to the LVS mailing list by Thomas Proell (about Oct 2000).

This section was written by `andreas.koenig@anima.de` Andreas J. Koenig and was posted to the mailing list.

An often lamented shortcoming of LVS clusters is that the realservers have to be configured to work identically. Thus if you want to build up a service with many servers that need to be configured differently for some reason, you cannot take advantage of the powerful LVS.

The following describes an LVS topology where not all servers in the pool of available servers are configured identical and where loadbalancing is content-based.

The goal is achieved by combining the features of Squid and LVS. The workhorses are running Apache, but any HTTP server would do.

18.1 Terminology

Before we start we need to introduce a bit of Squid terminology. A redirector (http://www.squid-cache.org/Doc/FAQ/FAQ-15.html) is a director that examines the URL and request method of an

HTTP and is enabled to change the URL in any way it needs. An accelerator (http://www.squid-cache.org/Doc/FAQ/FAQ-20.html) plays the role os a buffer and cache. The accelerator handles a relatively big amount of slow connections to the clients on the internet with a relativly small amount of memory. It passes requests through to any number of back-end servers. It can be configured to cache the results of the back-end servers according to the HTTP headers.

18.2 Preview

In the following example installation, we will realize this configuration (real IP addresses anonymized):

```
VIP               10.0.0.62

LVSdirector       10.0.0.2
LVS_hot_standby   10.0.0.3

Squid1            10.0.0.5:80
Squid2            10.0.0.7:80   # Same box as Webserver2

Webserver1        10.0.0.6:81
Webserver2        10.0.0.7:81   # Same box as Squid2
Webserver3        10.0.0.8:81
Webserver4        10.0.0.9:81
```

Note that a squid and a webserver can coexist in a single box, that's why we have put Squid2 and Webserver7 into a single machine.

Note also that squids can cache webserver's output and thus reduce the work for them. We dedicate 24 GB disk to caching in Squid1 and 6 GB disk in Squid2.

And finally note that several squids can exchange digest information about cached data if they want. We haven't yet configured for this.

Strictly speaking, a single squid can take the role of an LVSdirector, but only for HTTP. It's slower, but it works. By accessing one of the squids in our setup directly, this can be easily demonstrated.

18.3 Let's start assembling

I'd suggest, the first thing to do is to setup the four apache on Webserver1..4. These servers are the working horses for the whole cluster. They are not what LVS terminology calls realservers though. The realservers according to LVS are the Squids.

We configure the apaches completely stardard. The only deviation from a standard installation here is that we specify

```
    Port 81
```

in the httpd.conf. Everything else is the default configuration file that comes with apache. In the choice of the port we are, of course, free to choose any port we like. It's an old habit of mine to select 81 if a squid is around to act as accelerator.

We finish this round of assembling with tests that only try to access Webserver1..4 on port 81 directly. For later testing, I recommend to activate the printenv CGI program that comes with Apache:

```
chmod 755 /usr/local/apache/cgi-bin/printenv
```

This program shows us, on which server the script is running (SERVER_ADDR) and which server appears as the requesting site (REMOTE_ADDR).

18.4 One squid

Next we should configure one Squid box. The second one will mostly be a replication of the first, so let's first nail that first one down.

When we compile the squid 2.3-STABLE4, we need already decide about compilation options. Personally I like the features associated with this configuration:

```
./configure --enable-heap-replacement --disable-http-violations \
            --enable-cache-digests    --enable-delay-pools
```

We can build and install squid with these settings. But before we start squid, we must go through a 2700 lines configuration file and set lots of options. The following is a collection of diffs between the squid.conf.default and my squid.conf with comments in between.

```
--- squid.conf.default  Mon Aug 14 12:04:33 2000
+++ squid.conf  Mon Aug 14 14:34:35 2000
@@ -47 +47 @@
-#http_port 3128
+http_port 80
```

Yes, we want this squid on port 80 because from outside it looks like a normal HTTP server.

```
@@ -54 +54 @@
-#icp_port 3130
+icp_port 0
```

In the demo installation I turned ICP off, but I'll turn it on again later. ICP is the protocol that the squids can use to exchange sibling information about what they have on their disks.

```
@@ -373 +373 @@
-#cache_mem  8 MB
+cache_mem 700 MB
```

This is the memory reserved for holding cache data. We have 1 GB total physical memory and 24 GB disk cache. To manage the disk cache, squid needs about 150 MB of memory (estimate 6 MB per GB for an average object size of 13kB). Once you're running, you can use squid's statistics to find out *your* average object size. I usually leave 1/6 of the memory for the operating system, but at least 100 MB.

```
@@ -389,2 +389,2 @@
-#cache_swap_low  90
-#cache_swap_high 95
+#cache_swap_low  94
+#cache_swap_high 96
@@ -404 +404 @@
-#maximum_object_size 4096 KB
+maximum_object_size 8192 KB
```

Please refer to squid's docs for these values.

```
@@ -463,0 +464,5 @@
+cache_dir ufs /var/squid01 5600 16 256
+cache_dir ufs /var/squid02 5600 16 256
+cache_dir ufs /var/squid03 5600 16 256
+cache_dir ufs /var/squid04 5600 16 256
+
```

You do not need bigger disks, you need many disks to speed up squid. Join the squid mailing list to find out about the efficiency of filesystem tuning like "noatime" or Reiser FS.

```
@@ -660 +665 @@
-#redirect_program none
+redirect_program /usr/local/squid/etc/redirector.pl
```

This is the meat of our usage of squid. This program can be as simple as you want or as powerful as you want. It can be implemented in any language and it will be run within a pool of daemons. My program is written in perl and looks something like the following:

```
    $|=1;
    while (<>) {
      chomp;
      my($url,$host,$ident,$method) = split;
      my @redir = $url =~ /\bh=([\d,]+);?/ ?
                  split(/,/,$1) : (6,7,8,9); # last components of our IP numbers
      my $redir = $redir[int rand scalar @redir];
      $url =~ s/PLACEHOLDER:81/10.0.0.$redir\:81/i;
      print STDOUT "$url\n";
    }
```

This is ideal for testing, because it allows me to request a single backend server or a set of backend servers to choose from via the CGI querystring. A request like

http://10.0.0.62/cgi-bin/printenv?h=6

will then be served by backend apache 10.0.0.6.

```
@@ -668 +673 @@
-#redirect_children 5
+redirect_children 10
```

The more complex the redirector program is, the more processes should be allocated to run it.

```
@@ -674 +679 @@
-#redirect_rewrites_host_header on
+redirect_rewrites_host_header off
@@ -879 +884 @@
-#replacement_policy LFUDA
+replacement_policy LFUDA
@@ -1168 +1173 @@
```

```
-acl Safe_ports port 80 21 443 563 70 210 1025-65535
+acl Safe_ports port 80 81 21 443 563 70 210 1025-65535
@@ -1204 +1209 @@
-http_access deny all
+http_access allow all
```

For all of the above changes, please refer to the squid.conf.default.

```
@@ -1370,2 +1375,3 @@
-#httpd_accel_host hostname
-#httpd_accel_port port
+# we will replace www.meta-list.net:81 with our host of choice
+httpd_accel_host PLACEHOLDER
+httpd_accel_port 81
```

As we are redirecting everything through the redirector, we can fill in anything we want. No real hostname, no real port is needed. The redirector program will have to know what we chose here.

```
@@ -1377 +1383 @@
-#httpd_accel_with_proxy off
+httpd_accel_with_proxy on
```

If we want ICP working (and we said, we would like to get it working), we need this turned on.

We're done with our first squid, we can start it and test it. If you send a request to this squid, one of the backend servers will answer according to the redirect policy of the redirector program.

Basically, at this point in time we have a fully working content based redirector. As already mentioned, we do not really need LVS to accomplish this. But the downside of this approach is:

- we are comparatively slow: squid is not famous for speed.

- we do not scale well: if the bottleneck is a the squid, we want LVS to scale up.

18.5 Another squid

So the next step in our demo is to build another squid. This is very trivial given that we have already one. We just copy the whole configuration and adjust a few parameters if there are any differences in the hardware.

18.6 Combining pieces with LVS

The rest of the story is to read the appropriate docs for LVS. I have used Horms's Ultra Monkey docs and there's nothing to be added for this kind of setup. Keep in mind that only the squids are to be known by the LVS box. They are the "realservers" in LVS terminology. The apache back end servers are only known to the squids' redirector program.

18.7 Problems

It has been said that LVS is fast and squid is slow, so people believe, they must implement a level 7 switch in LVS to have it faster. This remains to be proofed.

Squid is really slow compared to some of the HTTP servers that are tuned for speed. If you're serving static content with a hernel HTTP daemon, you definitely do not want to lose the speed by running it through a squid.

If you want persistent connections, you need to implemented them in your redirector. If you want to take dead servers out of the pool, you must implement it in your redirector. If you have a complicated redirector, you need more of them and thus need more ressources.

In the above setup, ldirectord monitors just the two squids. A failure of one of the apaches might go by unnoticed, so you need to do something about this.

If you have not many cacheable data like SSL or things that need to expire immediately or a high fraction of POST requests, the squid seems like a waste of resources. I'd say, in that case you just give it less disk space and memory.

Sites that prove unviewable through Squid are a real problem (Joe Cooper reports there is a stock ticker that doesn't work through squid). If you have contents that cannot be served through a squid, you're in big trouble–and as it seems, on your own.

19 Details of LVS operation (including Security, DoS)

19.1 Director Connection Hash Table

The director maintains a hash table of connections marked with

```
<CIP, CPort, VIP, VPort, RIP, RPORT>
```

where

- CIP: Client IP address
- CPort: Client Port number
- VIP: Virtual IP address
- VPort: Virtual Port number
- RIP: RealServer IP address
- RPort: RealServer Port number.

The hash table speeds up the connection lookup and keeps state so that packets belonging to a connection from the client will be sent to the allocated realserver. If you are editing the .config by hand look for CONFIG_IP_MASQUERADE_VS_TAB_BITS.

19.2 Hash table size

The default LVS 19.1 (hash table) size (2^12 entries) originally meant 2^12 simultanous connections. These early versions of ipvs would crash your machine if you alloted too much memory to this table.

> Julian 7 Jun 2001 This was because the resulting bzImage was too big. Users selected a value too big for the hash table and even the empty table (without linked connections) couldn't fit in the available memory.

This problem has been fixed in kernels>0.9.9 with the connection table being a linked list.

(Note: If you're looking for memory use with "top", it reports memory allocated, not memory you are using. No matter how much memory you have, Linux will eventually allocate all of it as you continue to run the machine and load programs.)

Each connection entry takes 128 bytes, 2^12 connections requires 512kbytes. (Note: not all connections are active - some are waiting to timeout).

As of ipvs-0.9.9 the hash table is different.

> Julian Anastasov `uli@linux.tu-varna.acad.bg` With CONFIG_IP_MASQUERADE_VS_TAB_BITS we specify not the max number of the entries (connections in your case) but the number of the rows in a hash table. This table has columns which are unlimited. You can set your table to 256 rows and to have 1,800,000 connections in 7000 columns average. But the lookup is slower. The lookup function chooses one row using hash function and starts to search all these 7000 entries for match. So, by increasing the number of rows we want to speedup the lookup. There is _no_ connection limit. It depends on the free memory. Try to tune the number of rows in this way that the columns will not exceed 16 (average), for example. It is not fatal if the columns are more (average) but if your CPU is fast enough this is not a problem.
>
> All entries are included in a table with (1 << IP_VS_TAB_BITS) rows and unlimited number of columns. 2^16 rows is enough. Currently, LVS 0.9.7 can eat all your memory for entries (using any number of rows). The memory checks are planned in the next LVS versions (are in 0.9.9?).

Julian 7 Jun 2001

> Here is the picture: the hash table is an array of double-linked list heads, i.e.
> struct list_head *ip_vs_conn_tab;
> In some versions ago (< 0.9.9?) it was a static array, i.e.
> struct list_head ip_vs_table[IP_VS_TAB_SIZE];
> struct list_head is 8 bytes (d-linked list), the next and prev pointers
>
> In the second variant when IP_VS_TAB_SIZE is selected too high the kernel crashes on boot. Currently (the first variant), vmalloc(IP_VS_TAB_SIZE*sizeof(struct list_head)) is used to allocate the space for the empty hash table for connections. Once the table is created, more memory is allocated only for connections, not for the table itself.
>
> In any case, after boot, before any connections are created, the occupied memory for this empty table is IP_VS_TAB_SIZE*8 bytes. For 20 bits this is (2^20)*8 bytes=8MB. When we start to create connection they are enqueued in one of these 2^20 double-linked lists after evaluating a hash function. In the ideal case you can have one connection per row (a dream), so 2^20 connections. When I'm talking about columns, in this example we have 2^20 rows and average 1 column used.
>
> The *TAB_BITS define only the number of rows (the power of 2 is useful to mask the hash function result with the IP_VS_TAB_SIZE-1 instead of using '%' module operation). But this is not a limit for the number of connections. When the value is selected from the user, the real number of connections must be considered. For example, if you think your site can accept 1,000,000 simultaneous connections, you have to select such number of hash rows that will spread all connections in short rows. You can create these 1,000,000 conns with TAB_BITS=1 too but then all these connections will be linked in two rows and the lookup process will take too much time to walk 500,000 entries. This lookup is performed on each received packet.
>
> The selection of *TAB_BITS is entirely based on the recommendation to keep the d-linked lists short (less than 20, not 500,000). This will speedup the lookup dramatically.

So, for our example of 1,000,000 we must select table with 1,000,000/20 rows, i.e. 50,000 rows. In our case the min TAB_BITS value is 16 (2^16=65536 >= 50000). If we select 15 bits (32768 rows) we can expect 30 entries in one row (d-linked list) which increases the average time to access these connections.

So, the TAB_BITS selection is a compromise between the memory that will use the empty table and the lookup speed in one table row. They are orthogonal. More rows => More memory => faster access. So, for 1,000,000 entries (which is an real limit for 128MB directors) you don't need more than 16 bits for the conn hash table. And the space occupied by such empty table is 65536*8=512KBytes. Bits greater than 16 can speedup the lookup more but we waste too much memory. And usually we don't achieve 1,000,000 conns with 128MB directors, some memory is occupied for other things.

The reason to move to vmalloc-ed buffer is because an 65536-row table occupies 512KB and if the table is statically defined in the kernel the boot image is with 512KB longer which is obviously very bad. So, the new definition is a pointer (4 bytes instead of 512KB in the bzImage) to the vmalloc'ed area.

Ratz's code adds limits per service while this sysctl can limit everything. Or it can be additional strategy (oh, another one) vs/lowmem. The semantic can be "Don't allocate memory for new connections when the low memory threshold is reached". It can work for the masquerading connections too (2.2). By this way we will reserve memory for the user space. Very dangerous option, though.

Joe

what's dangerous about it?

One user process can allocate too much memory and to cause the LVS to drop new connections because the lowmem threshold is reached.

May be conn_limit is better or something like this:

```
if (conn_number > min_conn_limit && free_memory < lowmem_thresh)
        DROP_THIS_PACKET_FOR_NEW_CONN
```

why have a min_conn_limit in here? If you put more memory into the director, hen you'll have to recompile your kernel. Is it because finding conn_number is cheaper than finding free_memory?

:) The above example with real numbers:

```
if (conn_number > 500000 && free_memory < 10MB) DROP
```

I.e. don't allow the user processes to use memory that LVS can use. But when there are "enough" LVS connections created we can consider reserving 10MB for the user space and to start dropping new connections early, i.e. when there are less than 10MB free memory. If conn_number < 500000 LVS simply will hit the 0MB free memory point and the user space will be hurted because these processes allocated too much memory in this case.

But obtaining the "free_memory" may be costs CPU cycles. May be we can stick with a snapshot on each second.

The number of valid connections shouldn't change dramatically in 1 sec. However a DoS might still cause problems.

Yes, the problem is on SYN attack.

> Ratz
> max amount of concurrent connections: 3495. We assume having 4 realservers equally load balance, thus we have to limit the upper threshold per realserver to 873. Like this you would never have a memory problem but a security problem.

what's the security problem?

> SYN/RST flood. My patch will set the weight of the realserver to 0 in case the upper threshold is reached. But I do not test if the requesting traffic is malicious or not, so in case of SYN-flood it may be 99% of the packets causing the server to be taken out of service. In the end we have set all server to weight 0 and the load balancer is non-functional either. But you don't have the memory problem :)

And it hasn't crashed either.

> Ratz I kinda like it but as you said, there is the amem_thresh, my approach (which was not actually done because of this problem :) and now having a lowmem_thresh. I think this will end up in a orthogonal semantic for memory allocation. For example if you enable the amem_thresh the conn_number > min_conn_limit && free_memory < lowmem_thresh would never be the case. OTOH if you set the lowmem_thresh to low the amem_thresh is ineffective. My patch would suffer from this too.

> Julian Anastasov `ja@ssi.bg` 08 Jun 2001 lowmem_thresh is not related to amemthresh but when amemthresh < lowmem_thresh the strategies will never be activated. lowmem_thresh should be less than amemthresh. Then the strategies will try to keep the free memory in the lowmem_thresh:amemthresh range instead of the current range 0:amemthresh
>
> Example (I hope you have memory to waste):
>
> lowmem_thresh=16MB (think of it as reserved for user processes and kernel) amemthresh=32MB (when the defense strategies trigger) min_conn_limit=500000 (think of it as 60MB reserved for LVS connections)
>
> So, the conn_number can grow far away after min_conn_limit but only while lowmem_thresh is not reached. If conn_number < 500000 and free_memory < lowmem_thresh we will wait the OOM killer to help us. So, we have 2 tuning parameters: the desired number of connections and some space reserved for user processes. And may be this is difficult to tune, we don't know how the kernel prevents problems in VM before activating the killer, i.e. swapping, etc. And the cluster software can take some care when allocating memory.

19.3 Hash table timeouts

How long are the connection entries held for ? (Column 8 of /proc/net/ip_masquerade ?)

> Julian The default timeout value for TCP session is 15 minutes, TCP session after receiving FIN is 2 miniutes, and UDP session 5 minutes. You can use "ipchains -M -S tcp tcpfin udp" to set your own time values.

If we assume a clunky set of web servers being balanced that take 3s to serve an object, then if the connection entries are dropped immediately then we can balance about 20 million web requests per minute with 128M RAM. If however the connection entries are kept for a longer time period this puts a limit on the balancer.

> Yeah, it is true.

Eg (assuming column 8 is the thing I'm after!)

> Actually, the column 8 is the delta value in sequence numbers. The timeout value is in column 10.
>
> ```
> [zathras@consus /]$ head -n 1000 /proc/net/ip_masquerade |sed
> -e "s/  */ /g"|cut -d" " -f8|sort -nr|tail -n500|head -n1
> 8398
> ```
>
> *i.e.* Held for about 2.3 hours, which would limit a 128Mb machine to balance about 10.4 million requests per day. (Which is definitely on the low side knowing our throughput...)

Horms horms@vergenet.net

When a connection is recieved by an IPVS server and forwarded (by whatever means) to a back-end server at what stage is this connection entered into the IPVS table. It is before or as the packet is sent to the back-end server or delayed until after the 3 way handshake is complete.

> Lars The first packet is when the connection is assigned to a real server, thus it must be entered into the table then, otherwise the 3 way handshake would likely hit 3 different real servers.

It has been alleged that IBMs Net Director waits until the completion of the three way handshake to avoid the table being filled up in the case of a SYN flood. To my mind the existing SYN flood protection in Linux should protect the IPVS table in any case and the connection needs to be in the IPVS table to enable the 3 way handshake to be completed.

> Wensong There is state management in connection entries in the IPVS table. The connection in different states has different timeout value, for example, the timeout of the SYN_RECV state is 1 minute, the timeout of the ESTABLISHED state is 15 minutes (the default). Each connection entry occupy 128 bytes effective memory. Supposing that there is 128 Mbytes free memory, the box can have 1 million connection entries. The over 16,667 packet/second rate SYN flood can make the box run out of memory, and the syn-flooding attacker probably need to allocate T3 link or more to perform the attack. It is difficult to syn-flood a IPVS box. It would be much more difficult to attach a box with more memory.

I assume that the timeout is tunable, though reducing the timeout could have implications for prematurely dropping connections. Is there a possibility of implementing random SYN drops if too many SYN are received as I believe is implemented in the kernel TCP stack.

> Yup, I should implement random early drop of SYN entries long time ago as Alan Cox suggested. Actually, it would be simple to add this feature into the existing IPVS code, because the slow timer handler is activated every second to collect stale entries. I just need to some code to that handler, if over 90% (or 95%) memory is used, run drop_random_entry to randomly tranverse 10% (or 5%) entries and drop the SYN-RCV entries in them.

A second, related question is if a packet is forwarded to a server, and this server has failed and is sunsequently removed from the available pool using something like ldirectord. Is there a window where the packet can be retransmitted to a second server. This would only really work if the packet was a new connection.

> Yes, it is true. If the primary load balaner fails over, all the established connections will be lost after the backup takes over. We probably need to investigate how to exchange the state (connection entries) periodically between the primary and the backup without too much performance degradation.

If persistent connections are being used and a client is cached but doesn't have any active connections does this count as a connection as far as load balancing, particularly lc and wlc is concerned. I am thinking no. This being the case, is the memory requirement for each client that is cached but has no connections 128bytes as per the memory required for a connection.

> The reason that the existing code uses one template and creates different entries for different connections from the same client is to manage the state of different connections from the same client, and it is easy to seemlessly add into existing IP Masquerading code. If only one template is used for all the connections from the same client, the box receives a RST packet and it is impossible to identify from which connection.

We using Hash Table to record an established network connection. How do we know the data transmission by one conection is over and when should we delete it from the Hash Table?

> Julian Anastasov `ja@ssi.bg` 24 Dec 2000 OK, here we'll analyze the LVS and mostly the MASQ transition tables from net/ipv4/ip_masq.c. LVS support adds some extensions to the original MASQ code but the handling is same.
>
> First, we have three protocols handled: TCP, UDP and ICMP. The first one (TCP) has many states and with different timeout values, most of them set to reasonable values corresponding to the recommendations from some TCP related rfc* documents. For UDP and ICMP there are other timeout values that try to keep the both ends connected for reasonable time without creating many connection entries for each packet.
>
> There are some rules that keep the things working:
>
> - when a packet is received for an existing connection or when a new connection is created a timer is started/restarted for this connection. The timeout used is selected according to the connection state. If a packet is received for this connection (from one of the both ends) the timer is restarted again (and may be after a state change). If no packet is received during the selected period, the masq_expire() function is called to try to release the connection entry. It is possible masq_expire() to restart the timer again for this connection if it is used from other entries. This is the case for the templates used to implement the persistent timeout. They occupy one entry with timer set to the value of the persistent time interval. There are other cases, mostly used from the MASQ code, where helper connections are used and masq_expire() can't release the expired connection because it is used from others.
>
> - according to the direction of the packet we distinguish two cases: INPUT where the packet comes in demasq direction (from the world) and OUTPUT where the packet comes from internal host in masq direction.
>
> masq. What does "masq direction" mean for packets that are not translated using NAT (masquerading), for example, for Direct Routing or Tunneling? The short answer is: there is no masq direction for these two forwarding methods. It is explained in the LVS docs. In short, we have packets in both directions when NAT is used and packets only in one direction (INPUT) when DR or TUN are used. The packets are not demasqueraded for DR and TUN method. LVS just hooks the LOCAL_IN chain as the MASQ code is privileged in Linux 2.2 to inspect the incoming traffic when the routing decides that the traffic must be delivered locally. After some hacking, the demasquerading is avoided for these two methods, of course, after some changes in the packet and in its next destination - the real servers. Don't forget that without LVS or MASQ rules, these packets hit the local socket listeners.
>
> How are the connection states changed? Let's analyze for example the masq_tcp_states table (we analyze the TCP states here, UDP and ICMP are trivial). The columns specify the current state. The rows explain the TCP flag used to select the next TCP state and its timeout. The TCP flag is selected from masq_tcp_state_idx(). This function analyzes the TCP header and

decides which flag (if many are set) is meaningful for the transition. The row (flag index) in the state table is returned. masq_tcp_state() is called to change ms->state according to the current ms->state and the TCP flag looking in the transition table. The transition table is selected according to the packet direction: INPUT, OUTPUT. This helps us to react differently when the packets come from different directions. This is explained later, but in short the transitions are separated in such way (between INPUT and OUTPUT) that transitions to states with longer timeouts are avoided, when they are caused from packets coming from the world. Everyone understands the reason for this: the world can flood us with many packets that can eat all the memory in our box. This is the reason for this complex scheme of states and transitions. The ideal case is when there is no different timeouts for the different states and when we use one timeout value for all TCP states as in UDP and ICMP. Why not one for all these protocols? The world is not ideal. We try to give more time for the established connections and if they are active (i.e. they don't expire in the 15 mins we give them by default) they can live forever (at least to the next kernel crash^H^H^H^Hupgrade).

How does LVS extend this scheme? For the DR and TUN method we have packets coming from the world only. We don't use the OUTPUT table to select the next state (the director doesn't see packets returning from the internal hosts). We need to relax our INPUT rules and to switch to the state required by the external hosts :(We can't derive our transitions from the trusted internal hosts. We change the state only based on the packets coming from the the clients. When we use the INPUT_ONLY table (for DR and TUN) the LVS expects a SYN packet and then an ACK packet from the client to enter the established state. The director enters the established state after a two packet sequence from the client without knowing what happens in the real server, which can drop the packets (if they are invalid) or establish a connection. When an attacket sends SYN and ACK packets to flood a LVS-DR or LVS-Tun director, many connections are established state. Each each established connection will allocate resources (memory) for 15 mins by default. If the attacker uses many different source addresses for this attack the director will run out of memory.

For these two methods LVS introduces one more transition table: the INPUT_ONLY table which is used for the connections created for the DR and TUN forwarding methods. The main goal: don't enter established state too easily - make it harder.

Oh, maybe you're just reading the TCP specifications. There are sequence numbers that the both ends attach to each TCP packet. And you don't see the masq or LVS code to try to filter the packets according to the sequence numbers. This can be fatal for some connections as the attacker can cause state change by hitting a connection with RST packet, for example (ES->CL). The only info needed for this kind of attack is the source and destination IP addresses and ports. Such kind of attacks are possible but not always fatal for the active connections. The MASQ code tries to avoid such attacks by selecting minimal timeouts that are enough for the active connections to resurrect. For example, if the connection is hit by TCP RST packet from attacker, this connection has 10 seconds to give an evidence for its existance by passing an ACK packet through the masq box.

To make the things complex and harder for the attacker to block a masq box with many established connections, LVS extends the NAT mode (INPUT and OUTPUT tables) by introducing internal server driven state transitions: the secure_tcp defense strategy. When enabled, the TCP flags in the client's packet can't trigger switching to established state without acknowledgement from the internal end of this connection. secure_tcp changes the transition tables and the state timeouts to achieve this goal. The mechanism is simple: keep the connection is SR state with timeout 10 seconds instead of the default 60 seconds when the secure_tcp is not enabled.

This trick depends on the different defense power in the real servers. If they don't implement SYN cookies and so sometimes don't send SYN+ACK (because the incoming SYN is dropped from their full backlog queue), the connection expires in LVS after 10 seconds. This action assumes that this is a connection created from attacker, since one SYN packet is not followed by another, as part from the retransmissions provided from the client's TCP stack.

We give 10 seconds to the real server to reply with SYN+ACK (even 2 are enough). If the real server implements SYN cookies the SYN+ACK reply follows the SYN request immediatelly. But if there are no SYN cookies implemented the SYN requests are dropped when the backlog queue length is exceeded. So secure_tcp is by default useful for real servers that don't implement SYN cookies. In this case the LVS expires the connections in SYN state in a short time, releasing the memory resources allocated from them. In any case, secure_tcp does not allow switching to established state by looking in the clients packets. We expect ACK from the realserver to allow this transition to EST state.

The main goal of the defense strategies is to keep the LVS box with more free memory for other connections. The defense for the real servers can be build in the real servers. But may be I'll propose to Wensong to add a per-connection packet rate limit. This will help against attacks that create small number of connections but send many packets and by this way load the real servers dramatically. May be two values: rate limit for all incoming packets and rate limit per connection.

The good news is that all these timeout values can be changed in the LVS setup, but only when the secure_tcp strategy is enabled. An SR timeout of 2 seconds is a good value for LVS clusters when realservers don't implement SYN cookies: if there is no SYN+ACK from the realserver then drop the entry at the director.

The bad news is of course, for the DR and TUN methods. The director doesn't see the packets returning from the realservers and LVS-DR and LVS-Tun forwarding can't use the internal server driven mechanism. There are other defense strategies that help when using these methods. All these defense strategies keep the director with memory free for more new connections. There is no known way to pass only valid requests to the internal servers. This is because the realservers don't provide information to the director and we don't know which packet is dropped or accepted from the socket listener. We can know this only by receiving an ACK packet from the internal server when the three-way handshake is completed and the client is identified from the internal server as a valid client, not as spoofed one. This is possible only for the NAT method.

kspargerdialtoneinternet.net (29 Jan 2001) rephrases this by saying the LVS-NAT is layer-3 aware. For example, NAT can 'see' if a real server responds to a packet it's been sent or not, since it's watching all of the traffic anyway. If the server doesn't respond within a certain period of time, the director can automatically route that packet to another server. LVS doesn't support this right now, but, NAT would be the more likely candidate to support it in the future, as NAT understands all of the IP layer concepts, and DR doesn't necessarily.

Julian Someone must put back the real server when it is alive. This sounds like a user space job. The traffic will not start until we send requests. We have to send L4 probes to the real server (from the user space) or to probe it with requests (LVS from kernel space)?

19.4 timeouts the same for all services

Alois Treindl alois@astro.ch I have LVS-NAT configured so that ssh VIP connects me to one particular realserver. I would like to keep this ssh connection permanent, (to observe the cluster during its operation). This ssh connection times out with inactivity as expected. Can this be changed, without affecting the timeout values of other LVS services? Alternately I can connect by ssh to each machine without using LVS.

Julian Anastasov ja@ssi.bg 12 May 2001

Currently, there are only global timeout values which are not very useful for some boxes with mixed functions. The masquerading, the LVS and its virtual services use same timeout values. The problem is that there are too many timeouts.

The solution would be to separate these timeouts, i.e. per virtual service timeouts, separated from the masquerading. According to the virtual service protocol it can serve the TCP EST and the UDP timeout role. So this can be one value that will be specified from the users. By this way the in->out ssh/telnet/whatever connections can use their own timeout (1/10/100 days) and the external clients will use the standard credit of 15 minutes. But may be it is too late for 2.2 to change this model. Is one user specified timeout value enough?

19.5 tcp timeout values, don't change them

The tcp timeout values have their values for good reason (even if you don't know them they are), and realservers operating as an LVS must appear as normal tcp servers to the clients.

> Wayne, 19 Oct 2001 I have a question about the 'IP_MASQ_S_FIN_TIMEOUT" values in "net/ipv4/ip_masq.c" for the 2.2.x kernel. What purpose is served by having the terminated masqueraded TCP connection entries remain in memory for the default timeout of 2 minutes? Why isn't the entry freed immediately?

Julian Anastasov `ja@ssi.bg 20 Oct 2001>`

Because the TCP connection is full-duplex. The internal-end sends FIN and waits for the FIN from external host. Then TIME_WAIT is entered.

> Perhaps what I'm really asking is why there is an mFW state at all.

```
[IP_MASQ_S_FIN_WAIT] = 2*60*HZ,
/* OUTPUT */
/* mNO, mES, mSS, mSR, mFW, mTW, mCL, mCW, mLA, mLI */
/*syn*/ {{mSS, mES, mSS, mSR, mSS, mSS, mSS, mSS, mSS, mLI }},
/*fin*/ {{mTW, mFW, mSS, mTW, mFW, mTW, mCL, mTW, mLA, mLI }},
/*ack*/ {{mES, mES, mSS, mES, mFW, mTW, mCL, mCW, mLA, mES }},
/*rst*/ {{mCL, mCL, mSS, mCL, mCL, mTW, mCL, mCL, mCL, mCL }},
};
/mFW
```

This state has timeout corresponding to the similar state in the internal end. The remote end is still sending data while the internal side is in FIN_WAIT state (after a shutdown). The remote end can claim that it is still in established state not seeing the FIN packet from internal side. In any case, the remote end has 2 minutes to reply. It can even work for longer time if each packet follows in these two minutes not allowing the timer to expire. It depends in what state is the internal end, FIN_WAIT1 or FIN_WAIT2. May be the socket in the internal end is already closed.

> The only thing I can think of is if the other end of the TCP connection spontaneously issues a half close before the initiator sends his half close. Then it might be desirable to wait a while for the initiator to send his half close prior to disposing of the connection totally. What would be the consequences of using "ipchains -M -S" to set this value to, say, 1 second?

In any case, timeout values lower than those in the internal hosts are not recommended. If we drop the entry in LVS, the internal end still can retransmit its FIN packet. And the remote end has two minutes to flush its sending queue and to ack the FIN. IMO, you should claim that the timer in FIN_WAIT state should not be restarted on packets coming from remote end. Anything else is not good because it can drop valid packets that fit into the normal FIN_WAIT time range.

19.6 Hash table size, director will crash when it runs out of memory.

> Yasser Nabi IP Virtual Server version 0.9.0 (size=16777216)

Julian Anastasov `ja@ssi.bg` 25 May 2001

Too much, it takes 128MB for the table only. Use 16 bits for example.

> Is this a hidden/undocumented problem with IPVS or it's just an observation of memory waste ? (we use 18 bits in production)

```
empty hash tables:

18 bits occupy 2MB RAM
24 bits occupy 128MB RAM
```

If the box has 128MB and the bits are 24 the kernel crash is mandatory, soon or later. And this is a good reason the virtual service not to be hit. Expect funny things to happen on box with low memory

> I forgot that not anyone uses 256Mb or more RAM on directors :)

Yes, 256MB in real situation is 1,500,000 connections, 128 bytes each, 64MB for other things ... until someone experiments with SYN attack

> However, for me it makes sense to use up to 66% of total memory for LVS, especially on high-traffic directors (in the idea that the directors doesn't run all the desktop garbage that comes with most distros).

If the used bits are 24, an empty hash table is 128MB. For the rest 128MB you can allocate 1048576 entries, 128 bytes each ... after the kernel killed all processes.

Some calcs considering the magic value 16 as average bucket length and for 256MB memory:

For 17 bits:

2^17=131072 => 1MB for empty hash table

131072*16=2097152 entries=256MB for connections

For 18 bits:

2^18=262144 => 2MB for empty hash table

for each MB for hash table we lose space for 8192 entries but we speedup the lookup.

So, even for 1GB directors, 19 or 20 is the recommended value. Anything above is a waste of memory for hash table. In 128MB we can put 1048576 entries. In the 24-bit case they are allocated for d-linked list heads.

Joe 6 Jun 2001

what happens after the table fills up? Does ipvs handle new connect requests gracefully (ie drops them and doesn't crash)?

Julian

The table has fixed number of rows and unlimited number of columns (d-lists where the connection structures are enqueued). The number of connections allocated depends on the free memory.

Once there is no memory to allocate connection structure, the connection requests will be dropped. Expect crashes maybe at another place (usually user space) :)

I'm not sure what the kernel will decide in this situation but don't rely on the fact some processes will not be killed. There is a constant network activity and a need for memory for packets (floods/bursts).

And the reason the defense strategies to exist is to free memory for new connections by removing the stalled ones. The defense strategy can be automatically activated on memory threshold. Killing the cluster software on memory presure is not good.

So, the memory can be controlled, for example, by setting drop_entry to 1 and tuning amemthresh. On floods it can be increased. It depends on the network speed too: 100/1000mbit. Thresholds of 16 or 32 megabytes can be used in such situations, of course, when there are more memory chips.

Roberto Nibali `ratz@tac.ch` The director never crashes because of exhaustion of memory. If he tries to allocate memory for a new entry into the table and kmalloc returns NULL, we return, or better drop the packet in processing and generate a page fault.

You could use my 29.3 (treshhold limitation patch). You calculate how many connections you can sustain with your memory by multiplying each connection entry with 128bytes and divide by the amount of realserver and set the limitation alike. Example:

128MByte, persistency 300s: max amount of concurrent connections: 3495. We assume having 4 realservers equally load balance, thus we have to limit the upper threshhold per realserver to 873. Like this you would never have a memory problem but a security problem.

Joe

It would seem that we need a method of stopping the director hash table from using all memory whether as a result of a DoS attack or in normal service. Let's say you fill up RAM with the hash table and all user processes go to swap, then there will be problems - I don't know what, but it doesn't sound great - at a high number of connections I expect the user space processes will be needed too. I expect we need to leave a certain amount for user space processes and not allow the director to take more than a certain amount of memory.

It would be nice if the director didn't crash when the number of connections got large. Presumably a director would be functioning only as a director and the amount of memory allocated to user space processes wouldn't change a whole lot (ie you'd know how much memory it needed).

19.7 The LVS code does not swap

Joe Feb 2001

With sufficient number of connections, a director could start to swap out its tables (is this true?) In this case, throughput could slow to a crawl. I presume the kernel would have to retrieve parts of the table to find the realserver associated with incoming packets. I would think in this case it would be better to drop connect requests than to accept them.

Julian IMO, this is not true. LVS uses GFP_ATOMIC kind of allocations and as I know such allocations can't be swapped out.

If it's possible for LVS to start the director to swap, is there some way to stop this?

> You can try with 19.16.1 (testlvs) whether the LVS uses swap. Start the kernel with LILO option mem=8M and with large swap area. Then check whether more than 8MB swap is used.

19.8 Other factors determining the number of connections

In earlier verions of LVS, you set the amount of memory for the tables (in bytes). Now you allocate a number of hashes, whose size can grow without limit, allowing an unlimited number of connections. Once the number of connections becomes sufficiently large, then other resources will become limiting.

- out of memory.

  The ipvs code doesn't handle this, presumably the director will crash (also see the 29.3 (threshhold patch)). Instead you handle this by brute force, adding enough memory to accept the maximum number of connections your setup will ever be asked to handle (*e.g.* under a DoS attack). This memory size can be determined by the multiplying the rate at which your network connection can push connect requests to the director, by the timeout values, which are set by FIN_WAIT or the persistence.

- out of ports.

  You can 19.9 (expand the number of ports) to 65k, but eventually you'll 19.16.10 (reach the 65k port limit).

19.9 Port range: limitations, expanding port range on directors

Sometimes client processes on the realservers need to connect with machines on the internet (see 12.6 (clients on LVS-NAT realservers) and 13.10 (clients on LVS-DR realservers)).

> Wayne `wayne@compute-aid.com` Nov 5 2001 Say you have a web page that has to retrieve on-line ads from one of your advertiser (people who pay you for showing their ads). If you have 50,000 visitors on your site, you will open 50,000 connections between your web server and the ad server out there somewhere. The masquerade limit is 4,096 per pair of IP addresses, and 40,960 per LVS box. In our case, the realserver is behind the LVS-NAT director, which also functions as the firewall, so the realserver MUST use the director to reach the ad servers.

Usually the RIP is private (*e.g.*192.168/16) and will have to be NAT'ed to the outside world. This can be done with LVS-NAT or LVS-DR by adding masquerading rules to the director's iptables/ipchains rules. (With LVS-DR, you also have to 13.10 (route the packets from the RIP) - this routing is setup by default with the 10.1 (configure script).)

Less often you want to use more ports on your LVS client machines.

> Wang Haiguang My client machine it uses port numbers between 1024 - 4096. After reaching 4096, it will loop back to 1024 and reuse the ports. I want to use more port nubmers

`michael_e_brown@dell.com` 06 Feb 2001

```
echo 1024 65000 > /proc/sys/net/ipv4/ip_local_port_range
/usr/src/linux/Documentation/networking/ip-sysctl.txt
```

While normal client processes start using ports at 1024, masqueraded ports start at 61000 (see 12.6 (clients on LVS-NAT realservers)). The masquerading code does not check if other processes are requesting ports and thus port collisions could occur. It is assumed on a NAT box that no other processes are initiating connections (*i.e.* you aren't running a passive ftp server).

> Wayne `wayne@compute-aid.com` 14 May 2000, If running a load balancer tester, say the one from IXIA to issue connections to 100 powerful web servers, would all the parameters in Julian's description need to be changed, or it should not be a problem for having many many connections from a single tester?

Julian

There is no limit for the connections from the internal hosts. Currently, the masquerading allows one internal host to create 40960 TCP connections. But the limit of 4096 connections to one external service is still valid.

If 10 internal hosts try to connect to one external service, each internal host can create 4096/10 => 409 connections.

For UDP the problem is sometimes worse. It depends on the /proc/sys/net/ipv4/ip_masq_udp_dloose value.

> Joewhich is internal and which is external here? The client, the realservers?

This is a plain masquerading so internal and external refer to masquerading. These limits are not for the LVS connections, they are only for the 2.2 masquerading.

```
                / 65095        Internal Servers
External Server:PORT    -  ...   MADDR --------------------
                \ 61000
```

When many internal clients try to connect to same external real service, the total number of TCP connections from one MADDR to this remote service can be 4096 because the masq uses only 4096 masq ports by default. This is a normal TCP limit, we distinguish the TCP connections by the fact they use different ports, nothing more. And the masq code is restricted by default to use the above range of 4096 ports.

In the whole masquerading table there is a space only for 40960 TCP, 40960 UDP and 40960 ICMP connections. These values can be tuned by changing ip_masq.c:PORT_MASQ_MUL.

> 1 Nov 2001 Wayne`wayne@compute-aid.com` PORT_MASQ_MUL appears to serve only as a check to make sure the masquerading facility does not hog all the memory, and that actually things would still work no matter how large PORT_MASQ_MUL is, or even if the checks using it are disabled. Is this true?
>
>> Julian By multiplying this constant with the masq port range, you define the connection limit for each protocol. This is related to the memory used for masquerading. This is a real limit, but not for LVS connections, because they are usually not limited by port collisions, and LVS does not check this limit.
>
> What about using more than the 32k range? What is the maximum I could select?
>
>> Peter Mueller`pmueller@sidestep.com` You should be able to use about 60k, *i.e.* 1024-6100. I hope you have lots of RAM :-)

Julian continuing

The PORT_MASQ_MUL value simply determines the recommended length of one row in the masq hash table for connections, but in fact it is involved in the above connection limits. It is recommended that the busy masq routers must increase this value. May be the 4096 masq port range too. This involves squid servers behind masq router.

LVS uses another table without limits. For LVS setups the same TCP restrictions apply but for the external clients:

```
        4999 \
Client       - VIP:VPORT LVS Director
        1024 /
```

The limit of client connections to one VIP:VPORT is limited to the number of used client ports from same Client IP.

The same restrictions apply to UDP. UDP has the same port ranges. But for UDP the 2.2 kernel can apply different restrictions. They are caused from some optimizations that try to create one UDP entry for many connections. The reason for this is the fact that one UDP client can connect to many UDP servers while this is not common for TCP.

> Joe when you increase the port range, you need more memory. Is this only because you can have more connections and hence will need a bigger ipvsadm table?

Yes, the first need is for more masqueraded connections and they allocate memory. LVS uses separate table and it is not limited. We distinguish LVS-NAT from Masquerade. LVS-NAT (and any other method) does not allocate extra ports, even for other ranges. It shadows only the defined port. No other ports are involved until masquerading is used.

> ipvs doesn't check port ranges and so collisions can occur with regular services (ftp was mentioned). I would have thought that a process needing to open a IP connnection would ask the tcp code in the kernel for a connection and let that code handle the assignment of the port.

LVS does not allocate local ports. When the masquerade is used to help with some protocol, the masquerade performs the check (ftp for example).

The port range has nothing to do with LVS. It helps the masquerading to create more connections because there is fixed limit for each protocol. But sometimes LVS for 2.2 uses ip_masq_ftp, so may be only then this mport range is used.

> X-window connections are at 6000.. Will you be able to start an X-session if these ports are in use by a director masquerading out connections from the realservers?

If we put LVS (ipvsadm -A) in front of this port 6000 then X sessions will be stopped. OTOH, masquerade does not select ports in this range, the default is above 61000. So, any FTP sessions will not disturb local ports, of course, if you don't increase the mport range to cover the well defined server ports such as X.

19.10 apps starved for ports

> LVS Account, 27 Feb 2001 I'm trying to do some load testing of LVS using a reverse proxy cache server as the load balanced app. The error I get is from a load generating app.. Here is the error:

```
byte count wrong 166/151
```

Julian Anastasov ja@ssi.bg>

Broken app.

> this goes on for a few hundred requests then I start getting:
>
> ```
> Address already in use
> ```

App uses too many local ports.

> This is when I can't telnet to port 80 any more... If I try to telnet to 10.0.0.80 80 I get this:
>
> ```
> $ telnet 10.0.0.80 80
> Trying 10.0.0.80...
> telnet: Unable to connect to remote host: Resource temporarily unavailable
> ```

No more free local ports.

> If I go directly to the web server OR if I go directly to the IP of the reverse proxy cache server, I don't get these errors.

Hm, there are free local ports now.

> I'm using a load balancing app that I call this way:
>
> ```
> /home/httpload/load -sequential -proxyaddr 10.0.0.80 -proxyport
> 0  -parallel 120 -seconds 6000000 /home/httpload/url
> ```
>
> upping the local port range has helped tremendously

19.11 realserver running out of ports

Here's a case where a realserver ran out of udp ports doing DNS looksup while serving http.

> Hendrik Thiel thiel@falkag.de I am using IP Virtual Server version 0.9.14 (size=4096). We have 6 Realservers each.
>
> ```
> -> RemoteAddress:Port   Forward Weight ActiveConn InActConn
> -> server1:www          Masq    1      68         12391
> ```
>
> Today we reached a new peak (very fast, few minutes) 30Mbps, up from the normal 15Mbit/s. Afterwards the following kernel messages (dmesg) showed up...
>
> ```
> IP_MASQ:ip_masq_new(proto=UDP): could not get free masq entry (free=31894).
> IP_MASQ:ip_masq_new(proto=UDP): could not get free masq entry (free=31894).
> IP_MASQ:ip_masq_new(proto=UDP): could not get free masq entry (free=31888).
> ```

Julian Anastasov ja@ssi.bg 20 Nov 2001 (heavily edited by Joe)

It seems you are flooding a single remote host with UDP requests from a realserver. Your service, www, is TCP and is not directly connected to these messages. You've reached the UDP limit per destination (4096), there are still free UDP ports on the realserver for other destinations.

> Hendrik yes it's DNS, each realserver is a caching DNS.

```
resolv.conf
nameserver 127.0.0.1
nameserver external IP
```

19.12 DoS

LVS is vunerable to DoS by an attacker making repeated connection requests. Each connection requires 128bytes of memory - eventually the director will run out of memory. This will take a while but an attacker has plenty of time if you're asleep. As well with LVS-DR and LVS-Tun, the director doesn't have access to the TCPIP tables in the realserver(s) which show whether a connection has closed (see 19.1 (director hash table)). The director can only guess that the connection has really closed, and does so using timeouts.

For information on DoS strategies for LVS see *DoS page* <http://www.linuxvirtualserver.org/defense.html>.

> Laurent Lefoll Laurent.Lefoll@mobileway.com 14 Feb 2001 If I am not misunderstanding something, the variable /proc/sys/net/ipv4/vs/timeout_established gives the time a TCP connection can be idle and after that the entry corresponding to this connection is cleared. My problem is that it seems that sometimes it's not the case. For example I have a system (2.2.16 and ipvs 0.9.15) with /proc/sys/net/ipv4/vs/timeout_established = 480 but the entries are created with a real timeout of 120 ! On another system

Julian Anastasov ja@ssi.bg

Read *The secure_tcp defense strategy* <http://www.linuxvirtualserver.org/defense.html> where the timeouts are explained. They are valid for the defense strategies only. For TCP EST state you need to read the ipchains man page.

For more explanation of the secure_tcp strategy also see the 19.1 (explanation of the director's hash table).

> when I play with "ipchains -M -S > [value] 0 0" the variable /proc/sys/net/ipv4/vs/timeout_established is modified even when /proc/sys/net/ipv4/vs/secure_tcp is set to 0, so I'm not using the secure TCP defense. The "real" timeout is of course set to [value] when a new TCP connection appears. So should I understand that timeout_established, timeout_udp,... are always modified by "ipchains -M -S ...." whatever I use or not secure TCP defense but if secure-tcp is set to 0, other variables give the timeouts to use ? If so, are these variable accessible or how to check their value ?

ipchains -M -S modifies the two TCP and the UDP timeouts in the two secure_tcp modes: off and on. So, ipchains changes the three timeout_XXX vars. When you change the timeout_* vars you change them for secure_tcp=on only. Think for the timeouts as you have two sets: for the two secure_tcp modes: on and off. ipchains changes the 3 vars in the both sets. While secure_tcp is off changing timeout_* does not affect the connection timeouts. They are used when secure_tcp is on.

(Joe: 'ipchains 0 value 0', where value=10 does not change the timeout values or number of entries seen in InActConn or seen with netstat -M, or ipchains -M -L -n).

LVS has its own tcpip state table, when in secure_tcp mode.

> carl.huang what are the vs_tcp_states[ ] and vs_tcp_states_dos[ ] elements in the in ip_vs_conn structure for?

Roberto Nibali ratz@tac.ch 16 Apr 2001

The vs_tcp_states[] table is the modified state transition table for the TCP state machine. The vs_tcp_states_dos[] is a yet again modified state table in case we are under attack and secure_tcp is enabled. It is tigher but not conforming to the RFC anymore. Let's take an example how you can read it:

```
static struct vs_tcp_states_t vs_tcp_states [] = {
/*      INPUT */
/*        sNO, sES, sSS, sSR, sFW, sTW, sCL, sCW, sLA, sLI, sSA */
/*syn*/ {{sSR, sES, sES, sSR, sSR, sSR, sSR, sSR, sSR, sSR, sSR }},
/*fin*/ {{sCL, sCW, sSS, sTW, sTW, sTW, sCL, sCW, sLA, sLI, sTW }},
/*ack*/ {{sCL, sES, sSS, sES, sFW, sTW, sCL, sCW, sCL, sLI, sES }},
/*rst*/ {{sCL, sCL, sCL, sSR, sCL, sCL, sCL, sCL, sLA, sLI, sSR }},
```

The elements 'sXX' mean state XX, so for example, sFW means TCP state FIN_WAIT, sSR means TCP state SYN_RECV and so on. Now the table describes the state transition of the TCP state machine from one TCP state to another one after a state event occured. For example: Take row 2 starting with sES and ending with sCL. At the first, commentary row, you see the incoming TCP flags (syn,fin,ack,rst) which are important for the state transition. So the rest is easy. Let's say, you're in row 2 and get a fin so you go from sES to sCW, which should by conforming to RFC and Stevens.

Short illustration:

```
/*           , sES,
/*syn*/ {{   ,    ,
/*fin*/ {{   , sCW,
```

It was some months ago last year when Wensong, Julian and me discussed about a security enhancement for the TCP state transition and after some heavy discussion they implemented it. So the second table vs_tcp_states_dos[] was born. (look in the mailing list in early 2000).

19.13 DoS, from the mailing list

19.13.1 testing DoS

joern maier 22 Nov 2000

I've got some Problem protecting my VS from SYN - flood attacks. Somehow the drop_entry mechanism seems not to work. Doing a SYN-Flood with 3 clients to my VS (1 director + 3 RS) the System gets unreachable. A single realserver under the same attack by those clients stays alive.

> Julian You can't SYN flood the director with only 3 clients. You need more clients (or as an alternative, you can download 19.16.1 (testlvs) from the LVS web site). What does ipvsadm -Ln show under attack? How you activate drop_entry? What does "cat drop_entry" show?

all realserver have tcp_syncookies enabled (1), tcp_max_syn_backlog=128, the director is set drop_entry var to 1 (echo 1 > drop_entry). Before compiling the Kernel I set the table size to 2^20. My Director has 256 MB and no other applications running.

> You don't need such large table, really.

With testlvs and two clients, my LVS gets to a denial of service, although "cat drop_entry" shows me a "1".

ipvsadm -Ln:

```
192.168.10.1:80 lc
192.168.1.4:80  Tunnel  1       0       33246
192.168.1.3:80  Tunnel  1       0       33244
192.168.1.2:80  Tunnel  1       0       33246
```

> run testlvs with 100,000 source addresses.

during the flooding attack the connection values stay around this size. Using the SYN-Flood tool with which I tried it before, ivsadm shows me this:

```
192.168.10.1:80 lc
192.168.1.4:80  Tunnel  1       0       356046
192.168.1.3:80  Tunnel  1       0       355981
192.168.1.2:80  Tunnel  1       0       356013
```

so it shows me about ten times as many connections as your tool. I took a look at the packets, both are quiet similar, they only differ in the Windowsize (testlvs has 0, the other tool uses a size of 65534) and sequence numbers (o.k. checksum as well)

I am activating drop entry like this:

I switch on my computer (director) and start linux with the LVS Kernel

```
cd /proc/sys/net/ipv4/vs
echo 1 > drop_entry
```

> Julian Maybe you need to tune amemthresh. 1024 pages (4MB) are too low value. How much memory does "free" show under attack? You can try with 1/8 RAM size for example. The main goal of these defense strategies is to keep free memory in the director, nothing more. The defense strategies are activated according to the free memory size. The packet rate is not considered.

joern maier

That sounds all good to me, but what I'm really wondering about is, why has the drop_entry variable still a value of 1. I thought it has to be 2 when my System is under attack? To me it looks like LVS does not even think it's under attack and therefore does not use the drop_entry mechanism.

> You are right. You forgot to specify when the LVS to think it is under attack. drop_entry switches automatically from 1 to 2 when the free memory reaches amemthresh. Do you know that your free memory is below 4MB. See <http://www.linuxvirtualserver.org/defense.html>. So, 1,000,000 entries created from the other tool occupy 128MB memory. You have 256MB :) Boot with mem=128MB (in lilo) or set amemthresh to 32768 or run 19.16.1 (testlvs) with more source addresses (2,000,000). I'm not sure if the last will help if the other tool you use does not limit the number of spoofed addresses. But don't run testlvs with less than -srcnum 2000000. If the setup allows rate > 33,333 packets/sec LVS can create 2,000,000 entries that expire for 60 seconds (the SYN_RECV timeout). Better not to use the -random option in testlvs for this test.
>
> So, you can test with such large values but make sure you tune amemthresh in production with the best value for your director. The default value is not very useful. You can test whether 1/8 is a good value (8192 for 4K page size).

19.13.2 on the design of the DoS preventer

Alan Cox `alan@lxorguk.ukuu.org.uk`>

The biggest problem with load balancing, when you need to do this sort of trickery (and its one the existing load balancing patches seem to have is that if you store per connection state then a synflood will take out your box (if you run out of ram) or run a delightfully efficient DoS attack if you don't. The moment you factor time into your state tables you are basically well and truely screwed.

> Lars Marowsky-Bree `lmb@teuto.net`> 8 Jun 1999 This can be solved with a hashtable, where you take the source ip as the key and look up the server to direct the request. Since the hash table is fixed size, we can do with fixed resources.
>
> Given a proper hash function, this scheme is _ideal_ for basic round-robin and weighted round-robin with fixed weights and we should look at implementing this. Keeping state if not necessary _is_ a bug.
>
> We are screwed however and can't do this if we want to do least-connections, dynamic load-based balancing, add servers at a later time etc and still deliver sticky connections (ie that connections from client A will stay on server B until a timeout happens or server B dies).
>
> Basically, since we _need_ to keep state on a per-client basis for this we can be screwed easily by bombarding us with a randomized source IP.
>
> Now - for all but the most simple load balancing, we NEED to keep state. So, we need to weasle our way out of this mess somehow.
>
> One approach would be to integrate SYN cookies into the load-balancer itself and only pass on the connection if the TCP handshake succeeded. Now, there are a few obvious problems with this: It is a very complex task. And, it still screws us in the case of an UDP flood.
>
> "The easy way out" for TCP connections is to do this stuff in user space - a load-balancing proxy, which connects to the backend servers. Problems with this are that it isn't transparent to the backend servers anymore (all connections come from the IP of the loadbalancer), it does not scale as well (no direct routing approach etc possible), and we still did not solve UDP.
>
> I propose the following: We continue to maintain state like we always did. But when we hit, lets say, 32,000 simulteanous connections, we go into "overload" mode - that is, all new connections are mapped using the hash table like Alan proposed, but we still search the stateful database first.
>
> There are a few problems with this too: It is not as fast as the pure hash table, since we need to look into the stateful database before consulting the hashtable. If weights change during overload mode, sticky connections can't be easily guaranteed (I thus propose to NOT change weights during overload mode, or at least ignore the changes with regard to the hashing).
>
> However, these are disadvantages which only happen under attack. At the moment, we would simply crash, so it IS an improvement. It is a fully transparent approach and works with UDP too. The effort to implement this is acceptable. (if it were userspace I would give it a try sometime;)
>
> And if we implement this scheme for fixed loadbalancing, which someone else definetely should, reusing the code here might not be that much of a problem.

19.13.3 Timeout in MASQ tables

Michael McConnell `michaelm@eyeball.com` 08 Oct 2001

the command

```
#ipchains -L -M
```

returns a list of masqueraded connections, *i.e.*

```
TCP  01:38.01 10.1.1.41          21.1.112.43        80 (80) -> 4052
TCP  01:38.08 10.1.1.41          21.1.112.43        80 (80) -> 4053
TCP  00:25.09 10.1.1.11          20.170.180.17      80 (80) -> 4430
```

If ipchains (kernel 2.2) has been set with a 10hr TCP timeout

```
ipchains -M -S 7200 0 0 (10 hour TCP timeout)
```

Now these connections remain (will populate the ipvsadm table) for 10 hours. Does anyone have any suggestions as to how to purge this table manually? If I run out of ports, I get a DoS (2 hr timeout, 30,000 TCP connections...DoS)

> Peter Mueller If you alter /proc/net/ip_masquerade, it will break the established connection. Isn't that what you want to do?

No matter what I do I can not seem to reset, clear or modify this manually.

> if you do not like the prospect of altering directly perhaps try a shell script:
>
> ```
> #!/bin/sh
> #hopefully this works and you won't shoot yourself in the foot...
> ipchains -M -S 1 0 0
> sleep 5
> ipchains -M -S 7200 0 0
> ```

Setting this Value only effects *NEW* connections, connections already set are unaffected.

Julian Anastasov ja@ssi.bg>

Without a timeout values specific for each LVS virtual service and another for the masqueraded connections it is difficult to play such games. It seems only one timeout needs to be separated, the TCP EST timeout. The reason that such support is not in 2.2 is because nobody wants to touch the user structures. IMO, it can be added for 2.4 if there are enough free options in ipvsadm but it also depends on some implementation details.

If you worry for the free memory you can use some defense *the LVS DoS defense strategies* <http://www.linuxvirtualserver.org/docs/defense.html>

```
echo 1 > drop_entry
```

19.14 Writing Filter Rules

iptables (2.4 kernels) is different to ipchains (2.2 kernels). For one thing there is no "iptables -C" to check your rules (at least yet - one is promised).

> Ratz If you're dealing with netfilter, packets don't travel through all chains anymore. Julian once wrote something about it:
>
> packets coming from outside to the LVS do:
>
> ```
> PRE_ROUTING -> LOCAL_IN(LVS in) -> POST_ROUTING
> ```

packets leaving the LVS travel:

```
PRE_ROUTING -> FORWARD(LVS out) -> POST_ROUTING
```

> From the iptables howto: COMPATIBILITY WITH IPCHAINS This iptables is very similar to ipchains by Rusty Russell. The main difference is that the chains INPUT and OUTPUT are only traversed for packets coming into the local host and originating from the local host respectively. Hence every packet only passes through one of the three chains; previously a forwarded packet would pass through all three.

I don't yet understand all of iptables yet, but it helped making the whole filtering and NAPT stuff more smooth and more flexible. You have more possibilies to manage the traffic. If it is better has to be proven in reality, so far there are not many setups with complex firewall settings, like merging different advanced routing aspects with QoS and own Targets over different networks with all kind of non-TCP/UDP traffic and an IPV6 connection.

Julian

2.4 director:

Packets coming into the director (out->in):

- NAT: INPUT -> input routing -> local: LVS/DEMASQ -> input routing -> forwarding -> OUTPUT
- DR/TUN: INPUT -> input routing -> local: LVS -> output routing -> OUTPUT

packets leaving the LVS travel (in->out):

- NAT only: INPUT -> input routing -> FORWARD (-j MASQ) -> LVS/MASQ -> OUTPUT

2.2 director:

INPUT in 2.2 is similar as PRE_ROUTING in 2.4, i.e. INPUT, OUTPUT and FORWARD are the 2.2 firewall chains

```
input routing: ip_route_input()
output routing: ip_route_output()
forwarding: ip_forward()
local: ip_local_deliver()
```

Matthew S. Crocker `matthew@crocker.com` 31 Aug 2001 How do I filter LVS? Does LVS grab the packets before or after iptables?

Julian

LVS is designed to work after any kind of firewall rules. So, you can put your ipchains/iptables rules safely. If you are using iptables put them on LOCAL_IN, not on FORWARD. The LVS packets do not go through FORWARD.

Joe

If you are being attacked, it might be better to filter upstream (*e.g.* the router or your ISP), to prevent the LAN from being flooded.

19.15 Active/Inactive connnection

The output of ipsvadm lists connections, either as

- ActConn - in ESTABLISHED state
- InActConn - any other state

Entries in the ActConn column come from

- service with an established connection. Examples of services which hold connections in the ESTABLISHED state for long enough to see with ipvsadm are telnet and ftp (port 21).

Entries in the InActConn column come from

- Normal operation
  - Services like http (in non-persistent *i.e.* HTTP /1.0 mode) or ftp-data(port 20) which close the connections as soon as the hit/data (html page, or gif etc) has been retrieved (<1sec). You're unlikely to see anything in the ActConn column with these LVS'ed services. You'll see an entry in the InActConn column untill the connection times out. If you're getting 1000connections/sec and it takes 60secs for the connection to time out (the normal timeout), then you'll have 60,000 InActConns. This number of InActConn is quite normal. If you are running an e-commerce site with 300secs of persistence, you'll have 300,000 InActConn entries. Each entry takes 128bytes (300,000 entries is about 40M of memory, make sure you have enough RAM for your application). The number of ActConn might be very small.
- Pathological Conditions (*i.e.* your LVS is not setup properly)
  - identd delayed connections: The 3 way handshake to establish a connection takes only 3 exchanges of packets (*i.e.* it's quick on any normal network) and you won't be quick enough with ipvsadm to see the connection in the states before it becomes ESTABLISHED. However if the service on the realserver is under 17 (identd), you'll see an InActConn entry during the delay period.
  - Incorrect routing (usually the wrong default gw for the realservers):

    In this case the 3 way handshake will never complete, the connection will hang, and there'll be an entry in the InActConn column.

Usually the number of InActConn will be larger or very much larger than the number of ActConn.

Here's a LVS-DR LVS, setup for ftp, telnet and http, after telnetting from the client (the client command line is at the telnet prompt).

```
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port              Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:www rr
-> bashfull.mack.net:www           Route   1      0          0
-> sneezy.mack.net:www             Route   1      0          0
TCP  lvs2.mack.net:0 rr persistent 360
-> sneezy.mack.net:0               Route   1      0          0
TCP  lvs2.mack.net:telnet rr
-> bashfull.mack.net:telnet        Route   1      1          0
-> sneezy.mack.net:telnet          Route   1      0          0
```

showing the ESTABLISHED telnet connection (here to realserver bashfull).

Here's the output of netstat -an | grep (appropriate IP) for the client and the realserver, showing that the connection is in the ESTABLISHED state.

```
client:# netstat -an | grep VIP
tcp        0      0 client:1229      VIP:23           ESTABLISHED

realserver:# netstat -an | grep CIP
tcp        0      0 VIP:23           client:1229      ESTABLISHED
<verb>

Here's immediately after the client logs out from the telnet session.

<verb>
director# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port              Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:www rr
-> bashfull.mack.net:www           Route   1      0          0
-> sneezy.mack.net:www             Route   1      0          0
TCP  lvs2.mack.net:0 rr persistent 360
-> sneezy.mack.net:0               Route   1      0          0
TCP  lvs2.mack.net:telnet rr
-> bashfull.mack.net:telnet        Route   1      0          0
-> sneezy.mack.net:telnet          Route   1      0          0

client:# netstat -an | grep VIP
#ie nothing, the client has closed the connection

#the realserver has closed the session in response
#to the client's request to close out the session.
#The telnet server has entered the TIME_WAIT state.
realserver:/home/ftp/pub# netstat -an | grep 254
tcp        0      0 VIP:23        CIP:1236      TIME_WAIT

#a minute later, the entry for the connection at the realserver is gone.
```

Here's the output after ftp'ing from the client and logging in, but before running any commands (like 'dir' or 'get filename').

```
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port              Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:www rr
-> bashfull.mack.net:www           Route   1      0          0
-> sneezy.mack.net:www             Route   1      0          0
TCP  lvs2.mack.net:0 rr persistent 360
-> sneezy.mack.net:0               Route   1      1          1
```

```
TCP  lvs2.mack.net:telnet rr
-> bashfull.mack.net:telnet         Route   1      0          0
-> sneezy.mack.net:telnet           Route   1      0          0

client:# netstat -an | grep VIP
tcp        0      0 CIP:1230      VIP:21        TIME_WAIT
tcp        0      0 CIP:1233      VIP:21        ESTABLISHED

realserver:# netstat -an | grep 254
tcp        0      0 VIP:21        CIP:1233      ESTABLISHED
```

The client opens 2 connections to the ftpd and leaves one open (the ftp prompt). The other connection, used to transfer the user/passwd information, is closed down after the login. The entry in the ipvsadm table corresponding to the TIME_WAIT state at the realserver is listed as InActConn. If nothing else is done at the client's ftp prompt, the connection will expire in 900 secs. Here's the realserver during this 900 secs.

```
realserver:# netstat -an | grep CIP
tcp        0      0 VIP:21        CIP:1233      ESTABLISHED
realserver:# netstat -an | grep CIP
tcp        0     57 VIP:21        CIP:1233      FIN_WAIT1
realserver:# netstat -an | grep CIP
#ie nothing, the connection has dropped

#if you then go to the client, you'll find it has timed out.
ftp> dir
421 Timeout (900 seconds): closing control connection.
```

http 1.0 connections are closed immediately after retrieving the URL (*i.e.* you won't see any ActConn in the ipvsadm table immediately after the URL has been fetched). Here's the outputs after retreiving a webpage from the LVS.

```
director:# ipvsadm
IP Virtual Server version 0.2.8 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port               Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:www rr
-> bashfull.mack.net:www            Route   1      0          1
-> sneezy.mack.net:www              Route   1      0          0

client:~# netstat -an | grep VIP

bashfull:/home/ftp/pub# netstat -an | grep CIP
tcp        0      0 VIP:80        CIP:1238      TIME_WAIT
```

19.15.1 Q&A from the mailing list

> Ty Beede wrote: I am curious about the implementation of the inactconns and activeconns variables in the lvs source code.

Julian

```
        Info about LVS <= 0.9.7
TCP
        active:         all connections in ESTABLISHED state
        inactive:       all connections not in ESTABLISHED state
UDP
        active:         0 (none) (LVS <= 0.9.7)
        inactive:       all (LVS <= 0.9.7)

        active + inactive = all
```

Look in this table for the used timeouts for each protocol/state:

/usr/src/linux/net/ipv4/ip_masq.c, masq_timeout_table

For VS/TUNNEL and VS/DR the TCP states are changed checking only the TCP flags from the incoming packets. For these methods UDP entries can expire (5 minutes?) if only the real servers sends packets and there are no packets from the client.

For info about the TCP states:

- /usr/src/linux/net/ipv4/tcp.c

- rfc793.txt

From: Jean-francois Nadeau `jf.nadeau@videotron.ca`

Done some testing (netmon) on this and here's my observations :

1. A connection becomes active when LVS sees the ACK flag in the TCP header incoming in the cluster : i.e when the socket gets established on the real server.

2. A connection becomes inactive when LVS sees the ACK-FIN flag in the TCP header incoming in the cluster. This does NOT corespond to the socket closing on the real server.

Example with my Apache Web server.

```
Client   <--> Server

A client request an object on the web server on port 80 :

SYN REQUEST     ---->
SYN ACK               <----
ACK                        -----> *** ActiveConn=1 and 1 ESTABLISHED socket on real server.
HTTP get                -----> *** The client request the object
HTTP response     <----- *** The server sends the object
APACHE closes the socket : *** ActiveConn=1 and 0 ESTABLISHED socket on real server
The CLIENT receives the object. (took 15 seconds in my test)
ACK-FIN                -----> *** ActiveConn=0 and 0 ESTABLISHED socket on real server
```

Conclusion : ActiveConn is the active number of CLIENT connections..... not on the server in the case of short transmissions (like objects on a web page). Its hard to calculate a server's capacity based on this because slower clients makes ActiveConn greater than what the server is really processing. You wont be able to reproduce that effect on a LAN, because the client receives the segment too fast.

In the LVS mailing list, many people explained that the correct way to balance the connections is to use monitoring software. The weights must be evaluated using values from the real server. In VS/DR and VS/TUNi, the Director can be easily fooled with invalid packets for some period and this can be enough to inbalance the cluster when using "*lc" schedulers.

I reproduce the effect connecting at 9600 bps and getting a 100k gif from Apache, while monitoring established sockets on port 80 on the real server and ipvsadm on the cluster.

> Julian You are probably using VS/DR or VS/TUN in your test. Right? Using these methods, the LVS is changing the TCP state based on the incoming packets, *i.e.* from the clients. This is the reason that the Director can't see the FIN packet from the real server. This is the reason that LVS can be easily SYN flooded, even flooded with ACK following the SYN packet. The LVS can't change the TCP state according to the state in the real server. This is possible only for VS/NAT mode. So, in some situations you can have invalid entries in ESTABLISHED state which do not correspond to the connections in the real server, which effectively ignores these SYN packets using cookies. The VS/NAT looks the better solution against the SYN flood attacks. Of course, the ESTABLISHED timeout can be changed to 5 minutes for example. Currently, the max timeout interval (excluding the ESTABLISHED state) is 2 minutes. If you think that you can serve the clients using a smaller timeout for the ESTABLISHED state, when under "ACK after SYN" attack, you can change it with ipchains. You don't need to change it under 2 minutes in LVS 0.9.7. In the last LVS version SYN+FIN switches the state to TIME_WAIT, which can't be controlled using ipchains. In other cases, you can change the timeout for the ESTABLISHED and FIN-WAIT states. But you can change it only down to 1 minute. If this doesn't help, buy 2GB RAM or more for the Director.
>
> One thing that can be done, but this is may be paranoia:
>
> change the INPUT_ONLY table:

```
from:
              FIN
        SR ---> TW
to:
              FIN
        SR ---> FW
```

> OK, this is incorrect interpretation of the TCP states but this is a hack which allows the min state timeout to be 1 minute. Now using ipchains we can set the timeout to all TCP states to 1 minute.
>
> If this is changed you can now set ESTABLISHED and FIN-WAIT timeouts down to 1 minute. In current LVS version the min effective timeout for ESTABLISHED and FINWAIT state is 2 minutes.

Jean-Francois Nadeau `jf.nadeau@videotron.ca`

I'm using DR on the cluster with 2 real servers. I'm trying to control the number of connections to acheive this :

The cluster in normal mode balances requests on the 2 real servers. If the real servers reaches a point where they can't serve clients fast enough, a new entry with a weight of 10000 is entered in LVS to send the ovry with a weight of 10000 is entered in LVS to send the ovry with a weight of 10000 is entered in LVS to send the overflow locally on a web server with a static web page saying "we're too busy". It's a cgi that intercept 'deep links' in our site and return a predefined page. A 600 seconds persistency ensure that already connected clients stays on the server they began to browse. The client only have to hit refresh until the number of AciveConns (I hoped) on the real servers gets lower and the overflow entry gets deleted.

Got the idea... Load balancing with overflow control.

> Julian Good idea. But the LVS can't help you. When the clients are redirected to the Director they stay there for 600 seconds.

But when we activate the local redirection of requests due to overflow, ActiveConn continues to grow in LVS, while Inactconn decreases as expected. So the load on the real server gets OK... but LVS doesnt sees it and doesnt let new clients in. (it takes 12 minutes before ActiveConns decreases enough to reopen the site)

I need a way, a value to check at that says the server is overloaded, begin redirecing locally and the opposite.

I know that seems a little complicated....

> Julian
> What about trying to:
>
> - use persistent timeout 1 second for the virtual service. If you have one entry for this client you have all entries from this client to the same real server. I didn't tested it but may be a client will load the whole web page. If the server is overloaded the next web page will be "we're too busy".
> - switch the weight for the Director between 0 and 10000. Don't delete the Director as real server. Weight 0 means "No new connections to the server". You have to play with the weight for the Director, for example:
> - if your real servers are loaded near 99% set the weight to 10000
> - if you real servers are loaded before 95% set the weight to 0

From: Jean-Francois Nadeau `jf.nadeau@videotron.ca`

Will a weight of 0 redirect traffic to the other real servers (persistency remains ?)

> Julian If the persistent timeout is small, I think.

I can't get rid of the 600 seconds persistency because we run a transactionnal engine. i.e. if a client begins on a real server, he must complete the transaction on that server or get an error (transactionnal contexts are stored locally).

> Such timeout can't help to redirect the clients back to the real servers. You can check the free ram or the cpu idle time for the real servers. By this way you can correctly set the weights for the real servers and to switch the weight for the Director.
>
> These recommendations can be completely wrong. I've never tested them. If they can't help try to set httpd.conf:MaxClients to some reasonable value. Why not to put the Director as real server permanently. With 3 real servers is better.

Jean

Those are already optimized, bottleneck is when 1500 clients tries our site in less than 5 minutes.....

One of ours has suggested that the real servers check their own state (via TCP in use given by sockstat) and command the director to redirect traffic when needed.

Can you explain more in details why the number of ActiveConn on real server continue to grow while redirecting traffic locally with a weight of 10000 (and Inactonn on that real server decreasing normally).

> Julian Only the new clients are redirected to the Director at this moment. Where the active connections continue to grow, in the real servers or in the Director (weight=10000)?

19.16 Creating large numbers of InActConn: testlvs; testing DoS strategies

19.16.1 testlvs

testlvs (by Julian `ja@ssi.bg`) is available on *Julian's software page* `<http://www.linuxvirtualserver.org/~julian>`.

It sends a stream of SYN packets (SYN flood) from a range of addresses (default starting at 10.0.0.1) simulating connect requests from many clients. Running testlvs from a client will occupy most of the resources of your director and the director's screen/mouse/keyboard will/may lock up for the period of the test. To run testlvs, I export the testlvs directory (from my director) to the realservers and the client and run everything off this exported directory.

> Fabrice `fabrice@urbanet.ch` 11 Dec 2001 I can reach 60K SYN/s with a mean of about 54K using a PIII 500MHz client.
>
> The load on the LVS-NAT director was a high (always 100% system usage, and a swap between the ttys takes about 3-5 seconds). That poor box couldn't handle the load and wasn't able to send back packets (maybe only 10 per seconds). This means that the DoS was successfull but it's only working during the flood, it won't brake any services (thanks to Syn_Cookies).

Julian Anastasov `ja@ssi.bg`

If you want to measure a maximal possible rate use -srcnum 10 or another small number to avoid beating the routing cache in the director. If you need to test the defense strategies you need large value in -srcnum. The default is too small for this, it avoids errors.

> I think the only way to prevent the DoS in this case is to upgrade the LVS box hardware :)

Not always. LVS does not protect the real servers. The result can be the output pipe loaded from replies on DoS attack. You should try some ingress rate limiting, independent from LVS. Of course, your hardware should not be blocked from such attacks, you need faster MB+CPU if you care for such problems.

> I looked with the vmstat 1 and 10, as Julian recommanded. Shouldn't the values of the number of interruptions with "vmstat 10" be 10 times more than "vmstat 1"'s?

No, they should be equal, up to 5% are good, they show that the process scheduling really works. If you are under attack and you can't handle it then the snapshots from vmstat 1 are delayed and the results differ too much from the results provided for longer time interval.

> I got with vmstat 1: interrupts = ca. 400'000, cpu sys = 100 and with vmstat 10: interrupts = ca. 60'000, cpu sys = 100

Your director reached its limits. You should try to flood it with slower client(s). When you see that the input packet rate is equal to the successfully forwarded packets (received on the real server) then stop to slow down the attack. You reach the maximal packet rate possible to deliver to the real servers. On NAT you should consider the replies, they are not measured with testlvs tests. They will need may be the same CPU power.

19.16.2 configure realserver

The realserver is configured to reject packets with src_address 10.0.0.0/8.

Here's my modified version of Julian's show_traffic.sh , which is run on the realserver to measure throughput. Start this on the realserver before running testlvs on the client. For your interest you can look on the realserver terminal to see what's happening during a test.

```
#!/bin/sh
#show_traffic.sh
#by Julian Anastasov ja@ssi.bg
#modified by Joseph Mack jmack@wm7d.net
#
#run this on the realserver before starting testlvs on the client
#when finished, exit with ^C.
#
#suggested parameters for testlvs
#testlvs VIP:port -tcp -packets 20000
#where
#       VIP:port - target IP:port for test
#
#packets are sent at about 10000 packets/sec on my
#100Mbps setup using 75 and 133MHz pentium classics.
#
#-----------------------------------------
# setup a few things
to=10           #sleep time
trap return INT #trap ^C from the keyboard (used to exit the program)
iface="$1"      #NIC to listen on

#-----------------------------------------
#user defined variables

#network has to be the network of the -srcnet IP
#that is used by the copy of testlvs being run on the client
#(default for testlvs is 10.0.0.0)

network="10.0.0.0"
netmask="255.0.0.0"
#------------------------------------------
function get_packets() {
        cat /proc/net/dev | sed -n "s/.*${iface}:\(.*\)/\1/p" | \
        awk '{ packets += $2} ; END { print packets }'
}

function call_get_packets() {
        while true
        do
                sleep $to
                p1="`get_packets "$iface"`"
                echo "$((($p1-$p0)/$to)) packets/sec"
                p0="$p1"
        done
}
#------------------------------------------
```

```
echo "Hit control-C to exit"

#initialise packets at $iface
p0="`get_packets "$iface"`"

#reject packets from $network
route add -net $network netmask $netmask reject

call_get_packets

#restore routing table on exit
route del -net $network netmask $netmask reject
#------------------------------------------
```

19.16.3 configure director

I used LVS-NAT on a 2.4.2 director, with netpipe (port 5002) as the service on two realservers. You won't be using netpipe for this test, ie you won't need a netpipe server on the realserver You just need a port that you can set up an LVS on and netpipe is in my /etc/services, so the port shows up as a name rather than a number.

Here's my director

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.6 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:netpipe rr
  -> bashfull.mack.net:netpipe       Masq    1      0          0
  -> sneezy.mack.net:netpipe         Masq    1      0          0
```

19.16.4 run testlvs from client

run testlvs (I used v0.1) on the client. Here testlvs is sending 256 packets from 254 addresses (the default) in the 10.0.0.0 network. (My setup handles 10,000 packets/sec. 256 packets appears to be instantaneous.)

```
client: #./testlvs 192.168.2.110:5002 -tcp -packets 256
```

when the run has finished, go to the director

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.6 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:netpipe rr
  -> bashfull.mack.net:netpipe       Masq    1      0          127
  -> sneezy.mack.net:netpipe         Masq    1      0          127
```

(If you are running a 2.2.x director, you can get more information from ipchains -M -L -n, or netstat -M. For 2.4.x use cat /proc/net/ip_conntrack.)

This output shows 254 connections that have closed are are waiting to timeout. A minute or so later, the InActConn will have cleared (on my machine, it's 50secs).

If you send the same number of packets (256), from 1000 different addresses, (or 1000 packets to 256 addresses), you'll get the same result in the output of ipvsadm (not shown)

```
client: #./testlvs 192.168.2.110:5002 -tcp -srcnum 1000 -packets 256
```

In all cases, you've made 254 connections.

If you send 1000 packets from 1000 addresses, you'd expect 1000 connections.

```
./testlvs 192.168.2.110:5002 -tcp -srcnum 1000 -packets 1000
```

Here's the total number of InActConn as a function of the number of packets (connection attempts). Results are for 3 consecutive runs, allowing the connections to timeout in between.

The numbers are not particularly consistent between runs (aren't computers deterministic?). Sometimes the blinking lights on the switch stopped during a test, possibly a result of the tcp race condition (see the *performance page* <http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>)

```
packets         InActConn
 1000            356, 368, 377
 2000            420, 391, 529
 4000            639, 770, 547
 8000            704, 903,1000
16000           1000,1000,1000
```

You don't get 1000 InActConn with 1000 packets (connection attempts). We don't know why this is.

> Julian I'm not sure what's going on. In my tests there are dropped packets too. They are dropped before reaching the director, maybe from the input device queue or from the routing cache. We have to check it.

19.16.5 InActConn with drop_entry defense strategy

repeating the control experiment above, but using the drop_entry strategy (see 19.12 (the DoS strategies) for more information).

director:/etc/lvs# echo "3" >/proc/sys/net/ipv4/vs/drop_entry

```
packets         InActConn, drop_entry=3
 1000           369,368,371
 2000           371,380,409
 4000           467,578,458
 8000           988,725,790
16000           999,994,990
```

The drop_entry strategy drops 1/32 of the entries every second, so the number of InActConn decreases linearly during the timeout period, rather than dropping suddenly at the end of the timeout period.

19.16.6 InActConn with drop_packet defense strategy

repeating the control experiment above, but using the drop_packet strategy (see 19.12 (the DoS strategies) for more information).

director:/etc/lvs# echo "3" >/proc/sys/net/ipv4/vs/drop_packet

```
packets         InActConn, drop_packet=3
 1000           338,339,336
 2000           331,421,382
 4000           554,684,629
 8000           922,897,480,662
16000           978,998,996
```

The drop_packet=3 strategy will drop 1/10 of the packets before sending them to the realserver. The connections will all timeout at the same time (as for the control experiment, about 1min), unlike for the drop_entry strategy. With the variability of the InActConn number, it is hard to see the drop_packet defense working here.

19.16.7 InActConn with secure_tcp defense strategy

repeating the control experiment above, but using the secure_tcp strategy (see 19.12 (the DoS strategies) for more information). The SYN_RECV value is the suggested value for LVS-NAT.

```
director:/etc/lvs# echo "3" >/proc/sys/net/ipv4/vs/secure_tcp
director:/etc/lvs# echo "10" >/proc/sys/net/ipv4/vs/timeout_synrecv
```

```
packets         InActConn, drop_packet=3
 1000            338, 372, 359
 2000            405, 367, 362,
 4000            628, 507, 584
 8000            642,1000, 886
16000           1000,1000,1000
```

This strategy drops the InActConn from the ipvsadm table after 10secs.

19.16.8 maximum number of InActConn

If you want to get the maximum number of InActConn, you need to run the test for longer than the FIN timeout period (here 50secs). 2M packets is enough here. As well you want as many different addresses used as possible. Since testlvs is connecting from the 10.0.0.0/8 network, you could have 254ˆ3=16M connections. Since only 2M packets can be passed before connections start to timeout and the director connection table reaches a steady state with new connections arriving and old connections timing out, there is no point in sending packets from more that 2M source addresses.

Note: you can view the contents of the connection table with

2.2

- netstat -Mn
- cat /proc/net/ip_masquerade

2.4

- cat /proc/net/ip_vs_conn

Here's the InActConn with various defense strategies. The InActConn is the maximum number reachable, the scrnum and packets are the numbers needed to saturate the director. The time of the test must exceed the timeouts. InActConn was determined by running a command like this

```
client: #./testlvs 192.168.2.110:5002 -tcp -srcnum 1000000 -packets 2000000
```

and then adding the (two) entries in the InActConn column from the output of ipvsadm.

```
kernel          DoS strategy    InActConn       -srcnum -packets (10k/sec)
SYN cookie
no              secure_tcp      13,400          200,000 200,000
                syn_recv=10
no              none            99,400          500,000 1,000,000
yes             non             70,400          1,000,000 2,000,000
```

19.16.9 Is the number of InActConn a problem?

> edited from Julian The memory used is 128 bytes/connection and 60k connections will tie up 7M of memory. LVS does not use system sockets. LVS has its own connection table. The limit is the amount of memory you have - virtually unlimited. The masq table (by default 40960 connections per protocol). is a separate table and is used only for LVS/NAT FTP or for normal MASQ connections.

However the director was quite busy during the testlvs test. Attempts to connect to other LVS'ed services (not shown in the above ipvsadm table) failed. Netpipe tests run at the same time from the client's IP (in the 192.168.1.0/24 network) stopped, but resumed at the expected rate after the testlvs run completed (i.e. but before the InActConn count dropped to 0).

19.16.10 port starved machines

Matthijs van der Klip `matthijs.van.der.klip@nos.nl` 10 Nov 2001

used a fast (Origin 200) single client to generate generate between 3000 and 3500 hits/connections per second to his LVS'ed web cluster. No matter how many/few realservers in the cluster, he could only get 65k connections.

> Julian
> You are missing one reason for this problem: the fact that your client(s) create connections from limited number of addresses and ports. Try to answer yourself from how many different client saddr/sport pairs you hit the LVS cluster. IMO, you reach this limit. I'm not sure how many test client hosts you are using. If the client host is only one then there is a limit of 65536 TCP ports per src IP addr. Each connection has expiration time according to its proto state. When the rate is high enough not to allow the old entries to expire, you reach a situation where the connections are reused, i.e. the connection number showed from ipvsadm -L does not increase.

19.17 Debugging LVS

19.17.1 new way

```
echo x > /proc/sys/net/ipv4/debug_level
where 0<x<9
```

19.17.2 old way (may still work - haven't tested it)

Is there any way to debug/watch the path between the director and the realserver?

> Wensong below the entry
> CONFIG_IP_MASQUERADE_VS_WLC=m
> in /usr/src/linux/.config, add the line
> CONFIG_IP_VS_DEBUG=y
> This switch affects ip_vs.h and ip_vs.c. make clean in /usr/src/linux/net/ipv4 and rebuild the kernel and modules.

(other switches you will find in the code are IP_VS_ERR IP_VS_DBG IP_VS_INFO)

Look in syslog/messages for the output. The actual location of the output is determined by /etc/syslog.conf. For instance

```
kern.*                                         /usr/adm/kern
```

sends kernel messages to /usr/adm/kern (re-HUP syslogd if you change /etc/syslog.conf). Here's the output when LVS is first setup with ipvsadm

```
$ tail /var/adm/kern

Nov 13 17:26:52 grumpy kernel: IP_VS: RR scheduling module loaded.
```

(Note CONFIG_IP_VS_DEBUG is not a debug level output, so you don't need to add

```
*.=debug                                       /usr/adm/debug
```

to your syslog.conf file)

Finally check whether packets are forwarded successfully through direct routing. (also you can use tcpdump to watch packets between machines.)

> Ratz `ratz@tac.ch` Since some recent lvs-versions extensive debugging can be enabled to get either more information about what's exactly going on or to help you understanding the process of packet handling within the director's kernel. Be sure to have compiled in debug support for LVS (CONFIG_IP_VS_DEBUG=yes in .config)
> You can enable debugging by setting:
>
> ```
> echo $DEBUG_LEVEL > /proc/sys/net/ipv4/vs/debug_level
> ```
>
> where DEBUG_LEVEL is between 0 and 10.
> The do a tail -f /var/log/kernlog and watch the output flying by while connecting to the VIP from a CIP.
> If you want to disable debug messages in kernlog do:

```
echo 0 > /proc/sys/net/ipv4/vs/debug_level
```

If you run tcpdump on the director and see a lot of packets with the same ISN and only SYN and the RST, then either

- you haven't handled the 3 (arp problem) (most likely)
- you're trying to connect directly to the VIP from within the cluster itself

19.18 Security Issues

The HOWTO doesn't discuss securing your LVS (we can't do everything at once). However you need to handle it someway.

Roberto Nibali `ratz@tac.ch` 03 May 2001

It doesn't matter whether you're running an e-gov site or you mom's homepage. You have to secure it anyway, because the webserver is not the only machine on a net. A breach of the webserver will lead to a breach of the other systems too. The load balancer is basically on as secure as Linux itself is. ipchains settings don't affect LVS functionality (unless by mistake you use the same mark for ipchains and ipvsadm). LVS itself has some builtin security mainly to try to secure realserver in case of a DoS attack. There are several parameters you might want to set in the proc-fs.

- /proc/sys/net/ipv4/vs/amemthresh
- /proc/sys/net/ipv4/vs/am_droprate
- /proc/sys/net/ipv4/vs/drop_entry
- /proc/sys/net/ipv4/vs/drop_packet
- /proc/sys/net/ipv4/vs/secure_tcp
- /proc/sys/net/ipv4/vs/debug_level With this you select the debug level (0: no debug output, >0: debug output in kernlog, the higher the number to higher the verbosity)

  The following are timeout settings. For more information see TCP/IP Illustrated Vol. I, R. Stevens.
- /proc/sys/net/ipv4/vs/timeout_close - CLOSE
- /proc/sys/net/ipv4/vs/timeout_closewait - CLOSE_WAIT
- /proc/sys/net/ipv4/vs/timeout_established - ESTABLISHED
- /proc/sys/net/ipv4/vs/timeout_finwait - FIN_WAIT
- /proc/sys/net/ipv4/vs/timeout_icmp - ICMP
- /proc/sys/net/ipv4/vs/timeout_lastack - LAST_ACK
- /proc/sys/net/ipv4/vs/timeout_listen - LISTEN
- /proc/sys/net/ipv4/vs/timeout_synack - SYN_ACK
- /proc/sys/net/ipv4/vs/timeout_synrecv - SYN_RECEIVED
- /proc/sys/net/ipv4/vs/timeout_synsent - SYN_SENT
- /proc/sys/net/ipv4/vs/timeout_timewait - TIME_WAIT
- /proc/sys/net/ipv4/vs/timeout_udp - UDP

You don't want your director replying to pings.

19.19 MTU discovery and ICMP handling

(code for this was added sometime in 2000)

> Eric Mehlhaff wrote: I was just updating ipchains rules and it struck me that I dont know what LVS does with the ICMP needs-fragmentation packets required for path MTU discovery to work. What does LVS do with such packets, when its not immediately obvious which real server they are supposed to go to?

Wensong

Sorry that there is no LVS code to handle ICMP packets and send them to the corresponding real servers. But, I am thinking about adding some code to handle this.

(later, after the code had been added.)

> joern maier 13 Dec 2000 what happens with ICMP messages specified for a Realserver. Or more exactly what happens if for example an ICMP host unreachable messages is send to the LVS because a client got down ? Are the entrys from the connection table removed ?

Julian Anastasov `ja@ssi.bg` Wed, 13 Dec 2000

No

> Are the messages forwarded to the Realservers ?

Julian 13 Dec 2000

Yes, the embedded TCP or UDP datagram is inspected and this information is used to forward the ICMP message to the right real server. All other messages that are not related to existing connections are accepted locally.

Eric Mehlhaff `mehlhaff@cryptic.com` passed on more info

Theoreticaly, path-mtu-discovery happens on every new tcp connection. In most cases the default path MTU is fine. It's weird cases (ethernet LAN conenctions with low MTU WAN connections) that point out broken path-MTU discovery. I.e. for a while I had my home LAN (MTU 1500) hooked up via a modem connection that I had set MTU to 500 for. The minimum MTU in this case was the 500 for my home but there were many broken web sites I could not see because they had blocked out the ICMP-must-fragment packets on their servers. One can also see the effects of broken path mtu discovery on FDDI local networks.

Anyway, here's some good web pages about it:

```
http://www.freelabs.com/~whitis/isp_mistakes.html
http://www.worldgate.com/~marcs/mtu/
```

What happens if a realserver is connected to a client which is no longer reachable? ICMP replies go back to the VIP and will not neccessarily be forwarded to the correct realserver.

> Jivko Velev `jiko@tremor.net` Assume that we have TCP connections...and real server is trying to respond to the client, but it cannot reach it (the client is down, the route doesn't exist anymore, the intermadiate gateway is congested). In these cases your VIP will receive ICMP packets dest unreachable, source quench and friends. If you dont route these packets to the correct realserver you will affect performance of the LVS. For example the realserver will continue to resend packets to the client because they are not confirmed, and gateways will continue to send

you ICMP packets back to VIP for every packets they droped. The TCP stack will drop these kind of connections after his timeouts expired, but if the director forwarded the ICMP packets to the appropriate realserver, this will occur a little bit earlier, and will avoid overloading the redirector with ICMP stuff.

When you receive a ICMP packet it contains the full IP header of the packet that cause this ICMP to be generated + 64bytes of its data, so you can assume that you have the TCP/UDP header too. So it is possible to implement "Persitance rules" for ICMP packages.

Summary: This problem was handled in kernel 2.2.12 and earlier by having the 10.1 (configure script) turn off icmp redirects in the kernel (through the proc interface). For 2.2.13 the ipvs patch handles this. The configure script knows which kernel you are using on the director and does the Right Thing (TM).

Joe: from a posting I picked off Dejanews by Barry Margolin

the criteria for sending a redirect are:

1. The packet is being forwarded out the same physical interface that it was received from,
2. The IP source address in the packet is on the same Logical IP (sub)network as the next-hop IP address,
3. The packet does not contain an IP source route option.

Routers ignore redirects and shouldn't even be receiving them in the first place, because redirects should only be sent if the source address and the preferred router address are in the same subnet. If the traffic is going through an intermediary router, that shouldn't be the case. The only time a router should get redirects is if it's originating the connections (e.g. you do a "telnet" from the router's exec), but not when it's doing normal traffic forwarding.

unknown

Well, remember that ICMP redirects are just bandages to cover routing problems. No one really should be routing that way.

ICMP redirects are easily spoofed, so many systems ignore them. Otherwise they risk having their connectivity being disconnected on whim. Also, many systems no longer send ICMP redirects because some people actually want to pass traffic through an intervening system! I don't know how FreeBSD ships these days, but I suggest that it should ship with ignore ICMP redirects as the default.

19.19.1 LVS code only needs to handle icmp redirects for LVS-NAT

and not for LVS-DR and LVS-Tun

Julian: 12 Jan 2001

Only for LVS-NAT do the packets from the real servers hit the forward chain, i.e. the outgoing packets. LVS-DR and LVS-Tun receive packets only to LOCAL_IN, i.e. the FORWARD chain, where the redirect is sent, is skipped. The incoming packets for LVS/NAT use ip_route_input() for the forwarding, so they can hit the FORWARD chain too and to generate ICMP redirects after the packet is translated. So, the problem always exists for LVS/NAT, for packets in the both directions because after the packets are translated we always use ip_forward to send the packets to the both ends.

I'm not sure but may be the old LVS versions used ip_route_input() to forward the DR traffic to the real servers. But this was not true for the TUN method. This call to ip_route_input() can generate ICMP redirects and may be you are right that for the old LVS versions this is a problem for DR. Looking in the Changelog it seems this change occured in LVS version 0.9.4, near Linux 2.2.13. So, in the HOWTO there is something that is true: there is no ICMP redirect problem for LVS/DR starting from Linux 2.2.13 :) But the problems remains for LVS/NAT even in the latest kernel. But this change in LVS is not created to solve the ICMP redirect problem. Yes, the problem is solved for DR but the goal was to speedup the forwarding for the DR method by skipping the forward chain. When the forward chain is skipped the ICMP redirect is not sent.

ICMP redirects and LVS: (Joe and Wensong)

The test setups shown in this HOWTO for LVS-DR and LVS-Tun have the client, director and realservers on the same network. In production the client will connect via a router from a remote network (and for LVS-Tun the realservers could be remote and all on separate networks).

The client forwards the packet for VIP to the director, the director receives the packet on the eth0 (eth0:1 is an alias of eth0), then forwards the packet to the real server through eth0. The director will think that the packet came and left through the same interface without any change, so an icmp redirect is send to the client to notify it to send the packets for VIP directly to the RIP.

However, when all machines are on the same network, the client is not a router and is directly connected to the director, and ignores the icmp redirect message and the LVS works properly.

If there is a router between the client and the director, and it listens to icmp redirects, the director will send an icmp redirect to the router to make it send the packet for VIP to the real server directly, the router will handle this icmp redirect message and change its routing table, then the LVS/DR won't work.

The symptoms is that once the load balancer sends an ICMP redirect to the router, the router will change its routing table for VIP to the real server, then all the LVS won't work. Since you did your test in the same network, your LVS client is in the same network that the load balancer and the server are, it doesn't need to pass through a router to reach the LVS, you won't have such a symptom. :)

Only when LVS/DR is used and there is only one interface to receive packets for VIP and to connect the real server, there is a need to suppress the ICMP redirects of the interface.

> Joe The ICMP redirects is turned on in the kernel 2.2 by default. The 10.1 (configure.pl) script turns off icmp redirects on the director using sysctl
>
> ```
> echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
> ```

(Wensong) In the reverse direction, replies coming back from the realserver to the client

```
                                |<------------------------|
                                |                  real server
client <--> tunlhost1=======tunlhost2 --> director ------->|
```

After the first response packet arrives from the realserver at the tunlhost2, tunlhost2 will try to send the packet through the tunnel. If the packet is too big, then tunlhost2 will send an ICMP packet to the VIP to fragment the packet. In the previous versions of ipvs, the director won't forward the ICMP packet to (any) real server. With 2.2.13 code has been added to handle the icmp redirects and make the director forward icmp packets to the corresponding servers.

> If a realserver goes down after the connection is established, will the client get a dest_unreachable from the director?

No. Here is a design issue. If the director sends an ICMP_DEST_UNREACH immediately, all tranfered data for the established connection will be lost, the client needs to establish a new connection. Instead, we would rather wait for the timeout of connection, if the real server recovers from the temporary down (such as overloaded state) before the connection expires, then the connection can continue. If the real server doesn't recover before the expire, then an ICMP_DEST_UNREACH is sent to the client.

> If the client goes down after the connection is established, where do the dest_unreachable icmp packets generated by the last router go?

If the client is unreachable, some router will generate an ICMP_DEST_UNREACH packet and sent to the VIP, then the director will forward the ICMP packet to the real server.

> Since icmp packets are udp, are the icmp packets routed through the director independantly of the services that are being LVS'ed. ie if the director is only forwarding port 80/tcp, from CIP to a particular RIP, does the LVS code which handles the icmp forward all icmp packets from the CIP to that RIP. What if the client has a telnet session to one realserver and http to another realserver?

It doesn't matter, because the header of the original packet is encapsulated in the icmp packet. It is easy to identify which connection is the icmp packet for.

> If the client has two connections to the LVS (say telnet and http) each to 2 different realservers and the client goes down, the director gets 2 ICMP_DEST_UNREACH packets. The director knows from the CIP:port which realserver to send the icmp packet to?

Wensong Zhang 21 Jan 2000

The director handles ICMP packets for virtual services long time ago, please check the ChangeLog of the code.

> ChangeLog for 0.9.3-2.2.13 The incoming ICMP packets for virtual services will be forwarded to the right real servers, and outgoing ICMP packets from virtual services will be altered and send out correctly. This is important for error and control notification between clients and servers, such as the MTU discovery.

> Joe If a realserver goes down after the connection is established, will the client get a dest_unreachable from the director?

No. Here is a design issue. If the director sends an ICMP_DEST_UNREACH immediately, all tranfered data for the established connection will be lost, the client needs to establish a new connection. Instead, we would rather wait for the timeout of connection, if the real server recovers from the temporary down (such as overloaded state) before the connection expires, then the connection can continue. If the real server doesn't recover before the expire, then an ICMP_DEST_UNREACH is sent to the client.

> If the client goes down after the connection is established, where do the dest_unreachable icmp packets generated by the last router go?

If the client is unreachable, some router will generate an ICMP_DEST_UNREACH packet and sent to the VIP, then the director will forward the ICMP packet to the real server.

> Since icmp packets are udp, are the icmp packets routed through the director independantly of the services that are being LVS'ed. ie if the director is only forwarding port 80/tcp, from CIP to a particular RIP, does the LVS code which handles the icmp forward all icmp packets from the CIP to that RIP. What if the client has a telnet session to one realserver and http to another realserver?

It doesn't matter, because the header of the original packet is encapsulated in the icmp packet. It is easy to identify which connection is the icmp packet for.

19.19.2 ICMP checksum errors

(This problem pops up in the mailing list occasionally, *e.g.* Ted Pavlic on 2000-08-01.)

Jerry Glomph Black

> The kernel debug log (dmesg) occasionally gets bursts of messages of the following form on the LVS box:

```
IP_MASQ:reverse ICMP: failed checksum from 199.108.9.188!
IP_MASQ:reverse ICMP: failed checksum from 199.108.9.188!
IP_MASQ:reverse ICMP: failed checksum from 199.108.9.188!
IP_MASQ:reverse ICMP: failed checksum from 199.108.9.188!
IP_MASQ:reverse ICMP: failed checksum from 199.108.9.188!
```

> What is this, is it a serious problem, and how to deal with it?

Joe

I looked in dejanews. No-one there knows either and people there are wondering if they are being attacked too. It appears in non-LVS situations, so it probably isn't an LVS problem. The posters don't know the identity of the sending node.

Wensong

I don't think it is a serious problem. If these messages are generated, the ICMP packets must fail in checksum. Maybe the ICMP packets from 199.108.9.188 is malformed for some unknown reason.

Here are some other reports

> Hendrik Thiel `thiel@falkag.de` 18 Jun 2001 I noticed this in dmesg and messages:
>
> kernel: IP_MASQ:reverse ICMP:failed checksum from 213.xxx.xxx.xxx! last message repeated 1522 times. Is this lvs specific (using nat) ? or can this be an attack?
>
> Alois Treindl `alois@astro.ch`
>
> I see those too Jun 17 22:16:19 wwc kernel: IP_MASQ:reverse ICMP: failed checksum from 193.203.8.8!
>
> not as many as you but every few hours a bunch.
>
> Juri Haberland `juri@koschikode.com`
>
> From time to time I see them also on a firewall masquerading the companies net. I always assumed it is a corrupted ICMP packet... Who knows...

19.20 Filesystems for realserver content: the many reader, single writer problem

The client can be assigned to any realserver. One of the assumptions of LVS is that all realservers have the same content. This assumption is easy to fullfill for services like http, where the administrator updates the files on all realservers when needed. For services like mail or databases, the client writes to storage on one realserver. The other realservers do not see the updates unless something intervenes. Various tricks are described elsewhere here for mailservers and databases. These require the realservers to write to common storage (for mail the mailspool is nfs mounted; for databases, the LVS client connects to a database client on each realserver and these database clients write to a single databased on a backend machine, or the databased's on each realserver are capable of replicating).

One solution is to have a file system which can propagate changes to other realservers. We have mentioned gfs and coda in several places in this HOWTO as holding out hope for the future. People now have these working.

Wensong Zhang `wensong@gnuchina.org` 05 May 2001

> It seems to me that *Coda* `<http://www.coda.cs.cmu.edu>` is becoming quite stable. I have run coda-5.3.13 with the root volume replicated on two coda file servers for near two months, I haven't met problem which need manual maintance until now. BTW, I just use it for testing purposes, it is not running in production site.

Mark Hlawatschek `hlawatschek@atix.de` 2001-05-04

> we made some good experiences with the use of *GFS* `<http://www.globalfilesystem.org/>` (now moved to *GFS* `<http://www.sistina.com/gfs/>`. We are using LVS in conjunction with the GFS for about one year in older versions and it worked quite stable. We successfully demonstrated the solution with a newer version of GFS (4.0) at the CEBit 2001. Several domains (i.e. http://www.atix.org) will be served by the new configuration next week.

Mark's slides from his *talk in German at DECUS in Berlin (2001)* `<http://ace73.atix.de/downloads/santime/decus-hlawatschek-release.pdf>` is available.

19.21 netfilter hooks

Tao Zhao `taozhao@cs.nyu.edu` 11 Jul 2001

The source code of LVS adds ip_vs_in() to netfilter hook NF_IP_LOCAL_IN to change the destination of packets. As I understand, this hook is called AFTER routing decisions have reached. So how can it forward the packet to the new assigned destination without routing?

> Henrik Nordstrom `hno@marasystems.com` Instead of rewriting the packet inside the normal packet flow of Linux-2.4, IPVS accepts the packet and constructs a new one, routes it and sends it out.. This approach does not make much sense for LVS-NAT within the netfilter framework, but fits quite well for the other modes.

> Julian LVS does not follow the netfilter recommendations. What happens if we don't change the destination (*e.g.*DR and TUN methods which don't change the IP header). When such packet hits the routing the IP header fields are used for the routing decision. Netfilter can forward only by using NAT methods.
>
> LVS tries not to waste CPU cycles in the routing cache. You can see that there is output routing call involved but there is a optimization you can find even in TCP - the destination cache. The output routing call is avoided in most of the cases. This model is near the one achieved in Netfilter, i.e. to call only once the input routing function (2.2 calls it twice for DNAT). I'm now testing a patch for 2.2 (on top of LVS) that avoids the second input routing call and that can reroute the masqueraded traffic to the right gateway when many gateways are used and mostly when these gateways are on same device. The tests will show how different is the speed between this patched LVS for 2.2 and the 2.4 one (one CPU of course).
>
> We decided to use the LOCAL_IN hook for many reasons. May be you can find more info for the LVS integration into the Netfilter framework by searching in the *LVS mail list*

archive <http://marc.theaimsgroup.com/?l=linux-virtual-server&m=98296653726641&w=2> for "netfilter".

Julian 29 Oct 2001

IPVS uses only the netfilter's hooks. It uses own connection tracking and NAT. You can see how LVS fits into the framework on the *mailing list archive* <http://marc.theaimsgroup.com/?l=linux-virtual-server&m=98296653726641&w=2>.

> Ratz
>
> I see that the defense_level is triggered via a sysctrl and invoked in the sltimer_handler as well as the *_dropentry. If we push those functions on level higher and introduce a metalayer that registers the defense_strategy which would be selectable via sysctrl and would currently contain update_defense_level we had the possibility to register other defense strategies like f.e. limiting threshold. Is this feasible? I mean instead of calling update_defense_level() and ip_vs_random_dropentry() in the sltimer_handler we just call the registered defense_strategy[sysctrl_read] function. In the existing case the defense_strategy[0]=update_defense_level() which also merges the ip_vs_dropentry. Do I make myself sound stupid? ;)

The different strategies work in different places and it is difficult to use one hook. The current implementation allows they to work together. But may be there is another solution considering how LVS is called: to drop packets or to drop entries. There are no many places for such hooks, so may be it is possible something to be done. But first let's see what kind of other defense strategies will come.

> Yes, the project got larger and more reputation than some of us initially thought. The code is very clear and stable, it's time to enhance it. The only very big problem that I see is that it looks like we're going to have to separate code paths one patch for 2.2.x kernels and one for 2.4.x.

Yes, this is the reality. We can try to keep the things not to look different for the user space.

> This would be a pain in the ass if we had two ipvsadm. IMHO the userspace tools should recognize (compile-time) what kernel it is working with and therefore enable the featureset. This will of course bloat it up in future the more feature-differences we will have regarding 2.2.x and 2.4.x series.

Not possible, the sockopt are different in 2.4

> Could you point me to a sketch where I could try to see how the control path for a packet looks like in kernel 2.4? I mean some- thing like I would do for 2.2.x kernels:

```
        ----------------------------------------------------------------
        |            ACCEPT/                              lo interface |
        v           REDIRECT                  _______                  |
--> C --> S --> ______ --> D --> ~~~~~~~~~ -->|forward|----> _______ -->
    h     a    |input |    e    {Routing }   |Chain  |     |output |ACCEPT
    e     n    |Chain |    m    {Decision}   |_______| --->|Chain  |
    c     i    |______|    a     ~~~~~~~~~       |     | ->|_______|
    k     t       |        s       |             |     | |     |
    s     y       |        q       |             v     | |     |
    u     |       v        e       v            DENY/  | |     v
```

```
           m     |     DENY/      r   Local Process   REJECT  | |   DENY/
           |     v    REJECT      a       |                   | |  REJECT
           |   DENY               d       --------------------- |
           v                      e ----------------------------
          DENY
```

Here is some info I maintain (may be not actual, the new ICMP hooks are missing). Look for "LVS" where is LVS placed.

```
Linux IP Virtual Server for Netfilter and Linux 2.4

The Netfilter hooks:

Priorities:
        NF_IP_PRI_FIRST = INT_MIN,
        NF_IP_PRI_CONNTRACK = -200,
        NF_IP_PRI_MANGLE = -150,
        NF_IP_PRI_NAT_DST = -100,
        NF_IP_PRI_FILTER = 0,
        NF_IP_PRI_NAT_SRC = 100,
        NF_IP_PRI_LAST = INT_MAX,

PRE_ROUTING (ip_input.c:ip_rcv):
        CONNTRACK=-200, ip_conntrack_core.c:ip_conntrack_in
        MANGLE=-150, iptable_mangle.c:ipt_hook
        NAT_DST=-100, ip_nat_standalone.c:ip_nat_fn
        FILTER=0, ip_fw_compat.c:fw_in, defrag, firewall, demasq, redirect
        FILTER+1=1, net/sched/sch_ingress.c:ing_hook

LOCAL_IN (ip_input.c:ip_local_deliver):
        FILTER=0, iptable_filter.c:ipt_hook
        LVS=100, ip_vs_in
        LAST-1, ip_fw_compat.c:fw_confirm
        CONNTRACK=LAST-1, ip_conntrack_standalone.c:ip_confirm

FORWARD (ip_forward.c:ip_forward):
        FILTER=0, iptable_filter.c:ipt_hook
        FILTER=0, ip_fw_compat.c:fw_in, firewall, LVS:check_for_ip_vs_out,
                masquerade
        LVS=100, ip_vs_out

LOCAL_OUT (ip_output.c):
        CONNTRACK=-200, ip_conntrack_standalone.c:ip_conntrack_local
        MANGLE=-150, iptable_mangle.c:ipt_local_out_hook
        NAT_DST=-100, ip_nat_standalone.c:ip_nat_local_fn
        FILTER=0, iptable_filter.c:ipt_local_out_hook

POST_ROUTING (ip_output.c:ip_finish_output):
        FILTER=0, ip_fw_compat.c:fw_in, firewall, unredirect,
                mangle ICMP replies
        LVS=NAT_SRC-1, ip_vs_post_routing
        NAT_SRC=100, ip_nat_standalone.c:ip_nat_out
        CONNTRACK=LAST, ip_conntrack_standalone.c:ip_refrag

CONNTRACK:
        PRE_ROUTING, LOCAL_IN, LOCAL_OUT, POST_ROUTING

FILTER:
        LOCAL_IN, FORWARD, LOCAL_OUT

MANGLE:
        PRE_ROUTING, LOCAL_OUT

NAT:
        PRE_ROUTING, LOCAL_OUT, POST_ROUTING

Running variants:

1. Only lvs - the fastest
2. lvs + ipfw NAT
3. lvs + iptables NAT

Where is LVS placed:

LOCAL_IN:100 ip_vs_in

FORWARD:100 ip_vs_out

POST_ROUTING:NF_IP_PRI_NAT_SRC-1 ip_vs_post_routing

The chains:

The out->in LVS packets (for any forwarding method) walk:

pre_routing -> LOCAL_IN -> ip_route_output or dst cache -> POST_ROUTING

        LOCAL_IN
        ip_vs_in        -> ip_route_output/dst cache
                        -> set skb->nfmark with special value
                        -> ip_send -> POST_ROUTING

        POST_ROUTING
        ip_vs_post_routing
                        - check skb->nfmark and exit from the
                        chain

The in->out LVS packets (for LVS/NAT) walk:

pre_routing -> FORWARD -> POST_ROUTING

        FORWARD
        ip_vs_out       -> NAT -> NF_ACCEPT

        POST_ROUTING
        ip_vs_post_routing
                        - check skb->nfmark and exit from the chain
```

I hope there is a nice ascii diagram in the netfilter docs, but I hope the above info is more useful if you already know what each hook means.

> The biggest problem I see here is that maybe the user space daemons don't get enough scheduling time to be accurate enough.

That is definitely true. When the CPU(s) are busy transferring packets the processes can be delayed. So, the director better not spend many cycles in user space. This is the reason I prefer all these health checks to run in the real servers but this is not always good/possible.

> No, considering the fact that not all RS are running Linux. We would need to port the healthchecks to every possible RS architecture.

Yes, this is a drawback.

> Tell me, which scheduler should I take? None of the existing ones gives me good enough results currently with persistency. We have to accept the fact, that 3-Tier application programmers don't know about loadbalancing or clustering, mostly using Java and this is just about the end of trying to load balance the application smoothly.

WRR + load informed cluster software. But I'm not sure in the the case when persistency is on (it can do bad things).

> I currently get some values via an daemon coded in perl on the RS, started via xinetd. The LB connects to the healthcheck port and gets some prepared results. He then puts this stuff into a db and starts calculating the next steps to reconfigure the LVS-cluster to smoothen the imbalance. The longer you let it running the more data you get and the less adjustments you have to make. I reckon some guy showing up on this list once had this idea in direction of fuzzy logic. Hey Julian, maybe we should accept the fact that the wlc scheduler also isn't a very advanced one:
>
> ```
> loh = atomic_read(&least->activeconns)*50+atomic_read(&least->inactconns);
> ```
>
> What would you think would change if we made this 50 dynamic?

Not sure :) I don't have results from experiments with wlc :) You can put it in /proc and to make different experiments, for example :) But warning, ip_vs_wlc can be module, check how lblc* register /proc vars.

Jul 2001

```
        Linux IP Virtual Server for Netfilter and Linux 2.4

Running variants:

1. Only lvs - the fastest
2. lvs + ipfw NAT
```

```
3. lvs + iptables NAT

Where is LVS placed:

LOCAL_IN:100 ip_vs_in

FORWARD:99 ip_vs_forward_icmp
FORWARD:100 ip_vs_out

POST_ROUTING:NF_IP_PRI_NAT_SRC-1 ip_vs_post_routing

The chains:

The out->in LVS packets (for any forwarding method) walk:

pre_routing -> LOCAL_IN -> ip_route_output or dst cache -> POST_ROUTING


        LOCAL_IN
        ip_vs_in        -> ip_route_output/dst cache
                        -> mark skb->nfcache with special bit value
                        -> ip_send -> POST_ROUTING

        POST_ROUTING
        ip_vs_post_routing
                        - check skb->nfcache and exit from the
                        chain if our bit is set


The in->out LVS packets (for LVS/NAT) walk:

pre_routing -> FORWARD -> POST_ROUTING

        FORWARD (check for related ICMP):
        ip_vs_forward_icmp      -> local delivery -> mark
                                skb->nfcache -> POST_ROUTING

        FORWARD
        ip_vs_out               -> NAT -> mark skb->nfcache -> NF_ACCEPT

        POST_ROUTING
        ip_vs_post_routing
                        - check skb->nfcache and exit from the
                        chain if our bit is set
```

Why LVS is placed there:

- LVS creates connections after the packets are marked, i.e. after PRE_ROUTING:MANGLE:-150 or PRE_ROUTING:FILTER:0. LVS can use the skb->nfmark as a virtual service ID.
- LVS must be after PRE_ROUTING:FILTER+1:sch_ingress.c - QoS setups. By this way the incoming traffic can be policed before reaching LVS.

- LVS creates connections after the input routing because the routing can decide to deliver locally packets that are marked or other packets specified with routing rules. Transparent proxying handled from the netfilter NAT code is not always a good solution.
- LVS needs to forward packets not looking in the IP header (direct routing method), so calling ip_route_input with arguments from the IP header only is not useful for LVS
- LVS is after any firewall rules in LOCAL_IN and FORWARD

19.21.1 Requirements for the PRE_ROUTING chain

Sorry, we can't waste time here. The netfilter connection tracking can mangle packets here and we don't know at this time if a packet is for our virtual service (new connection) or for existing connection (needs lookup in the LVS connection table). We are sure that we can't make decisions whether to create new connections at this place but lookup for existing connections is possible under some conditions: the packets must be defragmented, etc.

LVS works with defragmented packets only

There are so many nice modules in this chain that can feed LVS with packets (probably modified)

19.21.2 Requirements for the LOCAL_IN chain

ip_local_deliver() defragments the packets for us

We detect here packets for the virtual services or packets for existing connections. In any case we send them to POST_ROUTING via ip_send and return NF_STOLEN. This means we remove the packet from the LOCAL_IN chain before reaching priority LAST-1. The LocalNode feature just returns NF_ACCEPT without mangling the packet

In this chain if a packet is for LVS connection (even newly created) the LVS calls ip_route_output (or uses a destination cache), marks the packet as a LVS property (sets bit in skb->nfcache) and calls ip_send() to jump to the POST_ROUTING chain. There our ip_vs_post_routing hook must call the okfn for the packets with our special nfcache bit value (Is skb->nfcache used after the routing calls? We rely on the fact that it is not used) and to return NF_STOLEN.

One side effect: LVS can forward packet even when ip_forward=0, only for DR and TUN methods. For these methods even TTL is not decremented nor data checksum is checked.

19.21.3 Requirements for the FORWARD chain

LVS checks first for ICMP packets related to TCP or UDP connections. Such packets are handled as they are received in the LOCAL_IN chain - they are localy delivered. Used for transparent proxy setups.

LVS looks in this chain for in->out packets but only for the LVS/NAT method. In any case new connections are not created here, the lookup is for existing connections only.

In this chain the ip_vs_out function can be called from many places:

FORWARD:0 - the ipfw compat mode calls ip_vs_out between the forward firewall and the masquerading. By this way LVS can grab the outgoing packets for its connection and to avoid they to be used from the netfilter's NAT code.

FORWARD:100 - ip_vs_out is registered after the FILTER=0. We can come here twice if the ipfw compat module is used because ip_vs_out is called once from FORWARD:0 (fw_in) and after that from pri=100 where LVS always registers the ip_vs_out function. We detect this second call by looking in the skb->nfcache bit value. If the bit is set we return NF_ACCEPT. In fact, the second ip_vs_out call is avoided if the first returns NF_STOLEN and after calling the okfn function.

19.21.4 Requirements for the POST_ROUTING chain

LVS marks the packets for debugging and they appear to come from LOCAL_OUT but this chain is not traversed. The LVS requirements from the POST_ROUTING chain include the fragmentation code only. But even the ICMP messages are generated and mangled ready for sending long before the POST_ROUTING chain: ip_send() does not call ip_fragment() for the LVS packets because LVS returns ICMP_FRAG_NEEDED when the mtu is shorter.

LVS makes MTU checks when accepting packets and selecting the output device. So, the ip_refrag POST_ROUTING hook is not used from LVS.

The result is: LVS must hook POST_ROUTING as first (may be only after the ipfw compat filter) and to return NF_STOLEN for its packets (detected by checking the special skb->nfcache bit value).

19.22 What is a session?

Andreas J. Koenig `andreas.koenig@anima.de` 26 Jun 2001

1. What are sessions

When an application written on top of a stateless protocol like HTTP has a need of stateful transactions, it typically writes some data to disk between requests and retrieves these data again on the subsequent request. This mechanism is known as session handling. The session data typically get written to files or databases. Each followup-request sends some sort of token to the server so that the application can retrieve the correct file or correct record in the database.

2. The old-fashined way to identify sessions

At the time when every webserver was a single PC, the session token identified a filename or a record in a database and everything was OK. When an application that relies on this mechanism is ported to a cluster environment, it stops working unless one deteriorates the cluster with a mechanism called persistence. Persistence is a quick and dirty way to get the old-fashioned token to work. It's not a very clever way though.

3. Why persistence is bad

Persistence counteracts two purposes of a cluster: easy maintainance by taking single machines out at any time and optimized balancing between the members of a cluster. Above that, persistence consumes memory on the load balancers.

4. How to do it better

Recall that there is a token being sent back and forth anyway, that identifies a filename or a database record. Extend this token to unambiguously point to the machine where the session data were created and install a session server on each host that delivers session data within the cluster to any of the peers on request. From that moment you can run your application truely distributed, you can take single machines out for maintainance any time: you turn their weight to 0 and wait for maybe an hour or three, depending on how long you want your sessions to last. You get better balancing, and you safe memory on the balancer. Note, that unlike with a dedicated session server, you do not create a single point of failure with this method.

19.23 Developement: Supporting IPSec on LVS

see *Julian's notes on developing code for IPSec over LVS* <http://www.linuxvirtualserver.org/~julian/LVS_IPSEC.txt>.

20 High Availability LVS: Failover protection

20.1 Introduction

In a production system you want to be able to remove, upgrade, add or replace nodes, without interruption of service to the client. Presumably these changes will be planned, but machines may crash too, so a mechanism for automatically handling machine/service crashes is required too.

Redundancy of services on the realservers is one of the useful features of LVS. One machine/service can be removed from the functioning virtual server for upgrade or moving of the machine and can be brought back on line later without interruption of service to the client.

The most common problem found is loss of network access or extreme slowdown. Hardware failure or an OS crash (on unix) is less likely.

Spinning media (disks) fail near the end of the warrantee period (in my experience) - you should replace your disks preemptively. The director(s) don't need hard disks. I've run my director from 30M of files (including perl and full glibc) pulled from my Slackware distribution. Presumably a mini Linux distribution would be even smaller. You should be able to boot off a floppy/cdrom/flash disk and load all files onto a small ramdisk. Logging information (eg for security) can be mailed/scp'ed at intervals to a remote machine via a NIC used for monitoring (not one of the NIC's used to connect to the outside world or to the realservers). Reconfiguring services on the fly with ipvsadm will not interrupt current sessions. You can reasonably expect your director to stay up for a long time without crashing and will not need to be brought down for servicing any more than any other diskless router.

The LVS code itself does not provide high availability. Other software is used in conjunction with LVS to provide high availability (*i.e.* to switch out a failed realserver/service or a failed director). Several families of tools are available to automatically handle failout for LVS. Conceptually they are a separate layer to LVS. Some separately setup LVS and the monitoring layer. Others will setup LVS for you and administratively the two layers are not separable.

There are two types of failures with an LVS.

- 20.2 (director failure)

  This is handled by having a redundant director available. Director failover is handled by the *Ultra Monkey Project* <http://ultramonkey.sourceforge.net> or by the 20.4 (vrrpd) in 20.4 (keepalived).

  The director maintains session information client IP, realserver IP, realserver port), and on failover this information must be available on the new director. On simple failover, where a new director is just swapped in, in place of the old one, the session information is not transferred to the new director and the client will loose their session. Transferring this information is handled by the 20.5 (server state synchronisation demon).

  The *keepalived project* <http://keepalived.sourceforge.net/> by Alexandre Cassen works with both Linux-HA and LVS. 20.4 (keepalived) watches the health of services. It also controls failover of directors using 20.4 (vrrpd).

- 21 (realserver failure), or failure of a service on a realserver

  This is relatively simple to handle (compared to director failover).

An agent running on the director monitors the services on the realservers. If a service goes down, that service is removed from the ipvsadm table. When the service comes back up, the service is added back to the ipvsadm table. There is no separate handling of realserver failure. If the server catches on fire (a concern of Mattieu Marc `marc.mathieu@metcelo.com`), the agent on the director will just remove that realserver's services from the ipvsadm table as they go down.

The 10 (configure script) monitors services with Mon. Setting up mon is covered in 20 (Failover protection) The configure script will set up mon for you. Mon was the first tool used with LVS to handle failover. It does not handle director failover.

In the *Ultra Monkey Project* `<http://ultramonkey.sourceforge.net>`, service failure is monitored by ldirectord.

For service failure on the realserver or director failure (without the 20.5 (server state synchronisation demon)), the client's session with the realserver will be lost. This is no different to what would happen if you were using a single server instead of an LVS. With LVS and failover however, the client will be presented with a new connection when they initiate a reconnect. Since only one of several realservers failed, only some of the clients will experience loss of connection, unlike the single server case where all clients loose their connection. In the case of http, the client will not even realise that the server/service has failed, since they get a new connection when clicking on a link. For session oriented connections (*e.g.*https, telnet) all unsaved data and session information will be lost.

20.2 Director failure

What happens if the director dies? The usual solution is duplicate director(s) with one active and one inactive. If the active director fails, then it is switched out. Although everyone seems to want reliable service, in most cases people are using the redundant boxes in order to maintain service through periods of planned maintenance rather than to handle boxes which just fail at random times.

Automatic detection of failure in unreliable devices by other unreliable devices is not a simple problem. All the code to handle LVS director failure in an LVS comes from the *Linux HA (High Availability) project* `<http://www.linux-ha.org>` The Linux HA solution is to have two directors and to run a heartbeat between them. One director defaults to being the operational director and the other takes over when heartbeat detects that the default director has died.

```
               --------
              |        |
              | client |
              |________|
                  |
                  |
               (router)
                  |
                  |
 -----------      |       -----------
|           |     |  DIP |           |
| director1 |-----|------| director2 |
|___________|     |  VIP |___________|
      |     <- heartbeat->    |
      |---------- | ----------|
                  |
------------------------------------
```

```
         |                   |                  |
         |                   |                  |
      RIP1, VIP          RIP2, VIP         RIP3, VIP
    --------------     --------------     --------------
   |              |   |              |   |              |
   | realserver1  |   | realserver2  |   | realserver3  |
   |______________|   |______________|   |______________|
```

LVS is one of the major uses for the Linux HA code and several of the Linux HA developers monitor the LVS mailing list. Setup problems can be answered on the LVS mailing list. For more detailed issues on the working of Linux HA, you'll be directed to join the Linux HA mailing list.

Fake, heartbeat and mon are available at the *Linux High Availability site* `<http://linux-ha.org/download>`.

There are several overlapping families of code being developed by the Linux HA project and the developers seem to contribute to each other's code. The two main branches of Linux HA used for LVS are 20.3 (UltraMonkey) and 20.4 (vrrpd/keepalived). Both of these have their own documentation and are not covered in this HOWTO.

20.3 UltraMonkey

The UltraMonkey code was the first version of Linux HA packaged for LVS and was written by Horms. The *Ultra Monkey project* `<http://ultramonkey.sourceforge.net>` uses Heartbeat from the Linux-HA project and ldirectord to monitor the realservers. The setup of Ultra Monkey is also covered on the LVS website.

Alternatively the 22 (setup of Linux HA on directors) has been written by Peter Mueller. This being functionally equivelent to the Ultra Monkey code.

20.3.1 Two box HA LVS

> Doug Sisk `sisk@coolpagehosting.com` 19 Apr 2001 Is it possible to create a two server LVS with fault tolerance? It looks straight forward with 4 servers (2 Real and 2 Directors), but can it be done with just two boxes, ie directors, each director being a realserver for the other director and a realserver running localnode for itself?

Horms

Take a look at ultramonkey.org, that should give you all the bits you need to make it happen. You will need to configure heartbeat on each box, and then LVS (ldirectord) on each box to have two real servers: the other box, and localhost.

20.3.2 heartbeat and connection state synch demon

> Michael Cunningham `m.cunningham@xpedite.com` I have heartbeat running between two LVS directors. It is working great. It can fail back and forth without issues. Now I would like to setup connection state synchronization between the two directors. But I am two problems/questions. Can I run the multicast connection sync over my 100 mbit private lan link which is being used by heartbeat?
>
> How can I setup heartbeat to always run..

```
ipvsadm --start-daemon=master --mcast-interface=eth1
```

> on the current master.. and

```
ipvsadm --start-daemon=backup --mcast-interface=eth1
```

> on the current slave at all times?
>
> The master can run a script when it starts up/obtains resources but I don't see anyway for the slave to run a script when it starts up or releases resources.

Lars Marowsky-Bree `lmb@suse.de` 02 Feb 2002

The slave runs the resource scripts with the "stop" action when the resources are released, so you could it add in there; anything you want to run before the startup of heartbeat is separate from that and obviously beyond the control of heartbeat.

You are seeing the result of heartbeat's rather limited resource manager, I am afraid.

20.4 Keepalived and Vrrpd

Alexandre Cassen `alexandre.cassen@canal-plus.com`, the co-author of *keepalived* `<http://keepalived.sourceforge.net>` and author of 23.3 (LVSGSP) has produced a vrrpd demon for LVS which enables director failover. The vrrpd is documented on the keepalived site.

The vrrpd fabricates a software ethernet device on the outside of the director (for the VIP) and another for the inside of the director (for the DIP) each with a MAC address from the private range of MAC addresses (*i.e.* will not be found on any manufactured NIC). When a director fails, vrrpd re-creates the ethernet devices, with the original IP and MACs, on the secondary director. The router does not see any change in the link layer and continues to route packets to the same MAC address.

In the Linux-HA situation by contrast, when the IPs (VIP, DIP) are moved to a new hardware NIC, the MAC address changes. Various types of trickery, (*e.g.* using send-arp to flush the router's arp table) is required to tell the router that the IP has moved to a new MAC address. This can possibly interrupt service (some packets will have to be re-sent).

> Padraig Brady `padraig@antefacto.com` 22 Nov 2001 Haven't Cisco got patents on this? What's the legal situation if someone wanted to deploy this?

Michael McConnell `michaelm@eyeball.com> no - ftp:`/ftp.isi.edu/in-notes/rfc2338.txt

Andre

In short yes : http://www.ietf.org/ietf/IPR//VRRP-CISCO, IBM too : http://www.ietf.org/ietf/IPR/NAT-VRRP-IBM

In fact there is 2 patents (*summary* `<http://www.foo.be/vrrp/>`)

- CISCO - http://www.delphion.com/details?pn=US06108300__
- Nortel Network - http://www.delphion.com/details?pn=EP01006702A3

When you read this papers you can't find any OpenSource restriction... All that I can see is the commercial product implementation... I plan to post a message into the IETF mailing list to present the LVS work on VRRP and to enlarge the debate on OpenSource implementation and eventual licence...

9 Jan 2002

answer from Robert Barr, CISCO Systems

> Cisco will not assert any patent claims against anyone for an implementation of IETF standard for VRRP unless a patent claim is asserted against Cisco, in which event Cisco reserves the right to assert patent claims defensively. I cannot answer for IBM, but I suspect their answer will be different.

20.5 Saving connection state on failover: Director demon for server state synchronisation

For seemless director failover, all connection state information from the failed director should be transferred/available to the new director. This is a similar problem to backing up a hot database. This problem had been discussed many times on the mailing list without any code being produced. Grabbing the bull by the horns, Ratz and Julian convened the Bulgarian Summit meeting in March 2001 where a design was set for a server state sync demon.

In ipvs-0.9.2 Wensong released a sync demon.

Wensong Zhang `wensong@gnuchina.org` 20 Jun 2001 The ipvs-0.9.2 tar ball is available on the LVS website. The major change is new connection sychronization feature.

Added the feature of connection synchroniztion from the primary load balancer to the backup load balancers through multicast.

The ipvs syncmaster daemon is started inside the kernel on the primary load balancers, and it multicasts the queue of connection state that need synchronization. The ipvs syncbackup daemon is started inside the kernel too on the backup load balancers, and it accepts multicast messages and create corresponding connections.

Here is simple intructions to use connection synchronization.

On the primary load balancer, run

```
primary_director:# ipvsadm --start-daemon=master --mcast-interface=eth0
```

On the backup load balancers, run

```
backup_director:# ipvsadm --start-daemon=backup --mcast-interface=eth0
```

To stop the daemon, run

```
director:# ipvsadm --stop-daemon
```

Note that the feature of connection synchronization is under experiment now, and there is some performance penalty when connection synchronization, because a highly loaded load balancer may need multicast a lot connection information. If the daemon is not started, the performance will not be affected.

Alexandre Cassen `alexandre.cassen@canal-plus.com` 9 Jul 2001 Using ipvsadm you start the sync daemon on to the master director. So it send adverts to the backups servers using multicast: 224.0.0.81. You need to start ipvsadm sync daemon on the backups servers too...

The master mulitcasts messages to the backup load balancers in the following format.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  Count Conns  |   Reserved    |             Size              |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      |                    IPVS Sync Connection (1)                   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                            .                                  |
      |                            .                                  |
      |                            .                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                   IPVS Sync Connection (n)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

> I have planed to add an ICV like in IPSEC-AH (with anti-replay and all strong dataexchange format) but I steel very busy.

There aren't a lot of people using the server state sync demon yet, so we don't have much experience with it yet.

There is now a *sync demon write up* <http://www.linuxvirtualserver.org/docs/sync.html>.

Here's a comment from the days before the LVS sync demon.

> Lars Marowski-Bree `lmb@suse.de` The 7.3 (-sh and -dh schedulers) in 2.4 should make this possible, as there is no state information to transfer ;-)

21 Realserver failure handled by Mon

21.1 Introduction

Don't even think about doing this till you've got your LVS working properly. If you want the LVS to survive a server or director failure, you can add software to do this after you have the LVS working. For production systems, failover may be useful or required.

An agent external to the ipvs code on the director is used to monitor the services. LVS itself can't monitor the services as LVS is just a packet switcher. If a realserver fails, the director doesn't get the failure, the client does. For the standard LVS-Tun and LVS-DR setups (ie receiving packets by an ethernet device and not by TP), the reply packets from the realserver go to its default gw and don't go through the director, so the LVS can't detect failure even if it wants to. For some of the mailings concerning why the LVS does not monitor itself and why an external agent (eg mon) is used instead, see the postings on 21.12 (external agents).

In a failure protected LVS, if the realserver at the end of a connection fails, the client will loose their connection to the LVS, and the client will have to start with a new connection, as would happen on a regular single server. With a failure protected LVS, the failed realserver will be switched out of the LVS and a working new server will be made available to you transparently (the client will connect to one of the still working servers, or possibly a new server if one is brought on-line).

If the service is http, loosing the connection is not a problem for the client: they'll get a new connection next time they click on a link/reload. For services which maintain a connection, loosing the connection will be a problem.

> ratz `ratz@tac.ch` 16 Nov 2000 This is very nasty for persistent setups in an e-commerce environment. Take for example a simple e-com site providing some subjects to buy. You can browse and view all their goodies. At a certain point you want to buy something. Ok, it is common nowadays that people can buy over the Internet with CC. Obviously this is done f.e. with SSL. SSL needs persistency enabled in the lvs-configuration. Imaging having 1000 users (conn ESTABLISHED) that are entering their VISA information when the database server crashes and the healthcheck takes out the server; or even more simple when the server/service itself crashes. Ok, all already established connections (they have a persistent template in the kernel space) are lost and these 1000 users have to reauthenticate. How does this look from a clients point of view which has no idea about the technology behind a certain site.

Here the functioning and setup of "mon" is described. In the Ultra Monkey version of LVS, ldirectord fills the same role. (I haven't compared ldirectord and mon. I used mon because it was available at the time, while ldirectord was either not available or I didn't know about it.) The 10.1 (configure script) will setup mon to monitor the services on the realservers.

Get "mon" and "fping" from http://www.kernel.org/software/mon/ (I'm using mon-0.38.20)

(from ian.martins@mail.tju.edu - comment out line 222 in fping.c if compiling under glibc)

Get the perl package "Period" from CPAN, ftp://ftp.cpan.org)

To use the fping and telnet monitors, you'll need the tcp_scan binary which can be built from satan. The standard version of satan needs patches to compile on Linux. The patched version is at

ftp://sunsite.unc.edu/pub/Linux/system/network/admin

21.2 ethernet NIC failure, and channel bonding

There was a lengthy thread on using multiple NICs to handle NIC failure. Software/hardware to handle such failures is common for unices which run expensive servers (*e.g.* Solaris) but is less common in Linux.

Beowulfs can use multiple NICs to increase thoughput by bonding them together (channel bonding), but redundancy/HA is not important for beowulfs - if a machine fails, it is fixed on the spot. There is no easy way to un-bond NICs - you have to reboot the computer :-(

Michael McConnell `michaelm@eyeball.com` 06 Aug 2001

> I want to take advantage of dual NICS in the real server to provide redundancy. Unfortunately the default gw issue comes up.

Michael E Brown `michael_e_brown@dell.com` 09 Aug 2001

> Yes, this is a generally available feature with most modern NICS. It is called various things: channel bonding, trunking, link aggregation. There is a native linux driver that implements this feature in a generic way. It is called the bonding driver. It works with any NIC. look in drivers/net/bond*. Each NIC vendor also has a proprietary version that works with only their NIC. I gave urls for intel's product, iANS. Broadcom and 3com also have this feature. I believe there is a standard for this: 802.1q.

John Cronin

> It would be nice if it could work across multiple switches, so if a single switch failed, you would not lose connectivity (I think the adaptive failover can do this, but that does not improve bandwidth). Jake Garver `garver@valkyrie.net` 08 Aug 2001
>
> > No it wouldn't be nice because it would put a tremendous burden on the link connecting the switches. If you are lucky, this link is 1Gb/sec, much slower than back planes which or 10Gb/sec and up. In general, you don't want to "load balance" your switches. Keep as much as you can on the same back plane.
>
> So, are there any Cisco Fast EtherChannel experts out there? Can FEC run across multiple switches, or at least across multiple Catalyst blades? I guess I can go look it up, but if somebody already knows, I don't mind saving myself the trouble.
>
> > Fast EtherChannel cannot run across multiple switches. A colleague spent weeks of our time proving that. In short, each switch will see a distinct link, for a total of two, but your server will think it has one big one. The switches will not talk to each

> other to bond the two links and you don't want them to for the reason I stated above. Over multiple blades, that depends on your switch. Do a "show port capabilities" to find out; it will list the ports that can be grouped into an FEC group.

Michael E Brown `michael_e_brown@dell.com`

If you want HA, have one machine (machine A) with bonded channels connected to switch A, and have another machine (machine B) with bonded channels connected to switch B. If you want to go super-paranoid, and have money to burn on links that won't be used during normal operations: have one machine (machine A) with bonded channels connected to switch A, and have backup bonded channels to switch B. Have software that detects failure of all bonded channels to switch A and fails over your IP to switch B (still on machine A). Have another machine (B), with two sets of bonded channels connected to switch C and switch D. lather, rinse, repeat. On Solaris, IP failover to backup link is called IP Multipathing, IIRC. New feature of Solaris 8. Various HA softwares for Linux, notably Steeleye Lifekeeper and possibly LinuxFailsafe, support this as well.

John Cronin

For the scenario described above (two systems), in many cases machine A is active and machine B is a passive failover, in which case you have already burned some money on an entire system (with bonded channels, no less) that won't be used during normal operations. Considering I can get four (two for each system) SMC EtherPower dual port cards for about $250, including shipping, or four Zynx Netblaster quad cards for about $820, if I shop around carefully, or $1000 for Intel Dual Port Server adapters or $1600 for Adaptec/Cogent ANA-6944 quad cards, if a name brand is important), the cost seems less significant when viewed in this light (not to mention the cost of two Cisco switches that can do FEC too).

Back to channel bonding (John Cronin)

> I presume it's not doable.

I think "not doable" is an incorrect statement - "not done" would be more precise. For the most part, beowulf is about performance, not HA. I know that Intel NICs can use their own channel aggregation or Cisco Fast-EtherChannel to aggregate bandwidth AND provide redundancy. Unfortunately, these features are only available on the closed-source Microsoft and Novell platforms.

http://www.intel.com/network/connectivity/solutions/server_bottlenecks/config_1.htm

> Having 2 NICs on a machine with one being spare, is relatively new. No-one has implemented a protocol for redundancy AFAIK.

I assume that you mean both of these statements to apply to Linux and LVS only. Sun has had trunking for years, but IP multipathing is the way to go now as it is easier to set up. You do get some bandwidth improvements for OUTBOUND connections only, on a per connection basis, but the main feature is redundancy.

Look in http://docs.sun.com/ for IP, multipathing, trunking.

Sun also has had Network Adapter Fail-Over groups (NAFO groups) in Sun Cluster 2.X for years, and in Sun Cluster 3.0. Veritas Cluster Server has an IPmultiNIC resource that provides similar functionality. Both of these allow for a failed NIC to be more or less seamlessly replaced by another NIC. I would be surprised if IBM HACMP has not had a similar feature for quite some time. In most cases these solutions do not provide improved bandwidth.

> The next question then is how often does a box fail in such a way that only 1 NIC fails and everything else keeps working? I would expect this to be an unusual failure mode and not worth protecting against. You might be better off channel bonding your 2 NICs and using the higher throughput (unless you're compute bound).

I would agree, with one exception. If you have the resources to implement redundant network paths farther out into your infrastructure, then having redundant NICs is much more likely to lead to improved availability. For example if you have two NICs, which are plugged into to two different switches, which are in turn plugged into two different routers, then you start to get some real benefit. It is more complicated to setup (HA isn't easy most of the time), but with the dropping prices of switches and routers, and the increased need for HA in many environments, this is not as uncommon as it might sound, at least not in the ISP and hosting arena.

I am not trying to slam LVS and Linux HA products - to the contrary; I am trying to inspire some talented soul to write a multipathing NIC device driver we can all benefit from. ;) I make my living doing work on Sun boxes, but I use Linux on my Dell Inspiron 8000 laptop (my primary workstation, actually - it's a very capable system). I would recommend Linux solutions in many situations, but in most cases my employers won't bite, as they prefer vendor supported solutions in virtually every instance, while complaining about the official vendor support.

for channel bonding both NICS on the host have the same IP and MAC address. You need to split the cabling for the two lots of NICs, so you don't have address collisions - you'll need two switches.

> John Cronin You either need multiple switches, or switches that understand and are willing participants in the channel aggregation method being used. Cisco makes switches that do Fast EtherChannel, and Intel makes adapters that understand this protocol (but again, not currently using Linux). Intel adapters also have their own channel aggregation scheme, and I think the Intel switches could also facilitate this scheme, but Intel is getting out of the switch business. Unfortunately, none of the advanced Intel NIC features are available using Linux (it would be nice to have the hardware IPsec support on their newest adapters, for example).

Michael E Brown `michael_e_brown@dell.com`

> Depends on which kind of bonding you do. Fast Etherchannel depends on all of the nics being connected to the same switch. You have to do configure the switch for trunking. Most of the standardized trunking methods I have seen require you to configure the switch and have all your nics connected to the same switch.

You either need multiple switches, or switches that understand and are willing participants in the channel aggregation method being used. Cisco makes switches that do Fast EtherChannel, and Intel makes adapters that understand this protocol (but again, not currently using Linux). Michael E Brown `michael_e_brown@dell.com`

> Not true. You can download the iANS software from Intel. Not open source, but that is different from "not available". look in http://isearch.intel.com for ians+linux.
>
> Also, if you want channel bonding without intel proprietary drivers, see

```
/usr/src/linux/drivers/net/bonding.c:
/*
 * originally based on the dummy device.
 *
 * Copyright 1999, Thomas Davis, tadavis@lbl.gov.
 * Licensed under the GPL. Based on dummy.c, and eql.c devices.
 *
 * bond.c: a bonding/etherchannel/sun trunking net driver
 *
 * This is useful to talk to a Cisco 5500, running Etherchannel, aka:
 *      Linux Channel Bonding
 *      Sun Trunking (Solaris)
 *
 * How it works:
 *    ifconfig bond0 ipaddress netmask up
 *      will setup a network device, with an ip address.  No mac address
 *      will be assigned at this time.  The hw mac address will come from
 *      the first slave bonded to the channel.  All slaves will then use
 *      this hw mac address.
 *
 *    ifconfig bond0 down
 *         will release all slaves, marking them as down.
 *
 *    ifenslave bond0 eth0
 *      will attache eth0 to bond0 as a slave.  eth0 hw mac address will
either
 *      a: be used as initial mac address
 *      b: if a hw mac address already is there, eth0's hw mac address
 *         will then  be set from bond0.
 *
 * v0.1 - first working version.
 * v0.2 - changed stats to be calculated by summing slaves stats.
 *
 */
```

Michael McConnell

This definately does it! It create this excellent kernel module, it contains ALL. I just managed to get this running on a Tyan 2515 Motherboard that has two Onboard Intel Nics.

I've just tested failover mode, works *PERFECT* not even a single packet dropped! I'm gonna try out adaptive load balancing next, and i'll let you know how I make out.

ftp://download.intel.com/df-support/2895/eng/ians-1.3.34.tar.gz

Michael E Brown `michael_e_brown@dell.com`

Broadcom also has proprietary channel bonding drivers for linux. The problem is getting access to this driver. I could not find any driver downloads from their website. It is possible that only OEMs have this driver. Dell factory installs this driver for RedHat 7.0 (and will be on 7.1, 7.2). You might want to e-mail Broadcom and ask. Also

> Broadcom also has an SSL offload card which is coming out and it has open source drivers for linux. http://www.broadcom.com/products/5820.html
>
> You need the openssl library and kernel.
>
> The next release of Red Hat linux will have this support integrated in. The Broadcom folks are working closely with the OpenSSL team to get their userspace integrated directly into 0.9.7. Red Hat has backported this functionality into their 0.9.6 release.
>
> If you look at Red Hat's latest public beta, all the support is there and is working.

> Since there aren't docs yet, the "bcm5820" rpm is the one you want to install to enable everything. Install this RPM, and it contains an init script that enables and disables the OpenSSL "engine" support as appropriate. Engine is the new OpenSSL feature that enables hardware offload.

21.2.1 more on channel bonding

> Paul wrote Interface bond0 comes up fine with eth1 and eth2 no problem. Bond1 fails miserably every time. I'm going to take that issue up on the bonding mailing list.

Roberto Nibali `ratz@drugphish.ch` 07 Mar 2002

Which patch did you try? Is it the following:

http://prdownloads.sourceforge.net/bonding/bonding-2.4.18-20020226

Did you pass max_bonds=2 when you loaded the bonding.o module? Without that you have no chance. Read the source (if you haven't already) to see what other fancy parameters you might want to pass.

> This is driven in part by our desire to see how far we can push lvs. I know it does 100mb/s in and out. If it can keep 2 channels full, I'll add a thirds, fourth,fifth, etc as necessary.

Read http://www.sfu.ca/acs/cluster/nic-test.html to get the impression of what happens if you try to bond too many NICs.

21.3 Service/realserver failout

To activate realserver failover, you can install mon on the director. Several people have indicated that they have written/are using other schemes. RedHat's piranha has monitoring code, and handles director failover and is documented there.

ldirectord handles director failover and is part of the Linux High Availability project. The author of ldirectord is Jacob Rief `jacob.rief@tis.at` with most of the later add-ons and code cleanup by Horms. ldirectord needs Net::SSLeay only if you are monitoring https (Emmanuel Pare `emman@voxtel.com`, Ian S. McLeod `ian@varesearch.com`)

To get ldirectord -

Jacob Rief `jacob.rief@tis.at`

```
the newest version available from
cvs.linux-ha.org:/home/cvs/
user guest,
passwd guest
module-name is: ha-linux
file: ha-linux/heartbeat/resource.d/ldirectord
documentation: ha-linux/doc/ldirectord
```

ldirectord is also available from *http://reserve.tiscover.com/linux-ha/* `<http://reserve.tiscover.com/linux-ha/>`.

Andreas Koenig `andreas.koenig@anima.de` 7 Jun 2001

cvs access is described in http://lists.community.tummy.com/pipermail/linux-ha-dev/1999-October/000212.html

Here's a possible alternative to mon -

> Doug Bagley doug@deja.com 17 Feb 2000 Looking at mon and ldirectord, I wonder what kind of work is planned for future service level monitoring?
>
> mon is okay for general purposes, but it forks/execs each monitor process, if you have 100 real services and want to check every 10 seconds, you would fork 10 monitor processes per second. This is not entirely untenable, but why not make an effort to make the monitoring process as lightweight as possible (since it is running on the director, which is such an important host)?
>
> ldirectord, uses the perl LWP library, which is better than forking, but it is still slow. It also issues requests serially (because LWP doesn't really make parallel requests easy).
>
> I wrote a very simple http monitor last night in perl that uses non-blocking I/O, and processes all requests in parallel using select(). It also doesn't require any CPAN libraries, so installation should be trivial. Once it is prototyped in perl, conversion to C should be straightforward. In fact, it is pretty similar to the Apache benchmark program (ab).
>
> In order for the monitor (like ldirectord) to do management of the ipvs kernel information, it would be easier if the /proc interface to ipvs gave a more machine readable format.

From: Michael Sparks zathras@epsilon3.mcc.ac.uk

Agreed :-)

```
It strikes me that rather than having:
type serviceIP:port mechanism
  -> realip:port tunnel weight active inactive
  -> realip:port tunnel weight active inactive
  -> realip:port tunnel weight active inactive
  -> realip:port tunnel weight active inactive
```

If the table was more like:

```
type serviceIP:port mechanism realip:port tunnel weight active inactive
```

Then this would make shell/awk/perl/etc scripts that do things with this table easier to cobble together.

> That seems like a far reaching precedent to me. On the other hand, if the ipvsadm command wished to have a option to represent that information in XML, I can see how that could be useful.

This reminds me I should really finish tweaking the prog I wrote to allow pretty printing of the ipvsadm table, and put it somewhere else for others to play with if they like - it allows you to specify a template file for formatting the output of of ipvsadm, making displaying the stuff as XML, HTML, plain text, etc simpler/quicker. (It's got a few hardcoded settings at the mo which I want to ditch first :-)

21.4 Mon for server/service failout

Here's the prototype LVS

```
                                 ________
                                |        |
                                | client |
                                |________|
                                    |
                         |
                      (router)
                         |
                         |
                         |       ----------
                         |  DIP |          |
                         |------| director |
                         |  VIP |__________|
                         |
                         |
                         |
       ------------------------------------
       |                 |                |
       |                 |                |
   RIP1, VIP         RIP2, VIP        RIP3, VIP
 --------------    --------------    --------------
|              |  |              |  |              |
| realserver1  |  | realserver2  |  | realserver3  |
|______________|  |______________|  |______________|
```

Mon has two parts:

- monitors: these are programs (usually perl scripts) which are run periodically to detect the state of a service. E.g. the telnet monitor attempts to login to the machine of interest and checks whether a program looking like telnetd responds (in this case looking for the string "login:"). The program returns success/fail. Monitors have been written for many services and new monitors are easy to write.
- Mon demon: reads a config file, specifying the hosts/services to monitor and how often to poll them (with the monitors). The conf file lists the actions for failure/success of each host/service. When a failure (or recovery) of a service is detected by a monitor, an "alert" (another perl script) is run. There are alerts which send email, page you or write to a log. LVS supplies a "virtualserver.alert" which executes ipvsadm commands to remove or add servers/services, in response to host/services changing state (up/down).

21.5 BIG CAVEAT

Trap for the unwary

Mon runs on the director, but...

Remember that you cannot connect to any of the LVS controlled services from within the LVS (including from the director) (see 4.2 (gotchas)). You can only connect to the LVS'ed services from the outside (eg from the client). If you are on the director, the packets will not return to you and the connection will hang. If you are connecting from the outside (ie from a client) you cannot tell which server you have connected to. This means that mon (or any agent), running on the director (which is where is needs to be to execute ipvsadm commands), cannot tell whether an LVS controlled service is up or down.

With LVS-NAT an agent on the director can access services on the RIP of the realservers (on the director you can connect to the http on the RIP of each realserver). Normal (i.e. non LVS'ed) IP communication is unaffected on the private director/realserver network of LVS-NAT. If ports are not re-mapped then a monitor running on the director can watch the httpd on server-1 (at 10.1.1.2:80). If the ports are re-mapped (eg the httpd server is listening on 8080), then you will have to either modify the http.monitor (making an http_8080.monitor) or activate a duplicate http service on port 80 of the server.

For LVS-DR and LVS-Tun the service on the realserver is listening to the VIP and you cannot connect to this from the director. The solution to monitoring services under control of the LVS for LVS-DR and LVS-Tun is to monitor proxy services whose accessability should closely track that of the LVS service. Thus to monitor an LVS http service on a particular server, the same webpage should also be made available on another IP (or to 0.0.0.0), not controlled by LVS on the same machine.

Example:

```
VS-Tun, LVS-DR
lvs IP (VIP): eth0 192.168.1.110
director:     eth0 192.168.1.1/24 (normal login IP)
              eth1 192.168.1.110/32 (VIP)
realserver:  eth0 192.168.1.2/24 (normal login IP)
              tunl0 (or lo:0) 192.168.1.110/32 (VIP)
```

On the realserver, the LVS service will be on the tunl (or lo:0) interface of 192.168.1.110:80 and not on 192.168.1.2:80. The IP 192.168.1.110 on the realserver 192.168.1.2 is a non-arp'ing device and cannot be accessed by mon. Mon running on the director at 192.168.1.1 can only detect services on 192.168.1.2 (this is the reason that the director cannot be a client as well). The best that can be done is to start a duplicate service on 192.168.1.2:80 and hope that its functionality goes up and down with the service on 192.168.1.110:80 (a reasonable hope).

```
VS-NAT
lvs IP (VIP): eth0 192.168.1.110
director:     eth0 192.168.1.1/24 (outside IP)
              eth0:1 192.168.1.110/32 (VIP)
              eth1 10.1.1.1/24 (DIP, default gw for realservers)
realserver:  eth0 10.1.1.2/24
```

Some services listen to 0.0.0.0:port, ie will listen on all IPs on the and you will not have to start a duplicate service.

21.6 About Mon

Mon doesn't know anything about the LVS, it just detects the up/down state of services on remote machines and will execute the commands you tell it when the service changes state. We give Mon a script which runs ipvsadm commands to remove services from the ipvsadm table when a service goes down and another set of ipvsadm commands when the service comes back up.

Good things about Mon:

- It's independant of LVS, ie you can setup and debug the LVS without Mon running.
- You can also test mon independantly of LVS.
- The monitors and the demon are independant.
- Most code is in perl (one of he "run anywhere" languages) and code fixes are easy.

Bad things about Mon:

- I upgraded to 0.38.20 but it does't run properly. I downgraded back to v0.37l. (Mar 2001 - Well I was running 0.37l. I upgraded to perl5.6 and some of the monitors/alerts don't work anymore. Mon-0.38.21 seems to work, with minor changes in the output and mon.cf file.)

- the author doesn't reply to e-mail.

mon-0.37l, keeps executing alerts every polling period following an up-down transition. Since you want your service polled reasonable often (eg 15secs), this means you'll be getting a pager notice/email every 15secs once a service goes down. Tony Bassette `kult@april.org` let me know that mon-0.38 has a numalert command limiting the number of notices you'll get.

Mon has no way of merging/layering/prioritising failure info. If a node fails you get an avalanch of notices that all the services on that node died too.

> Ted Pavlic `tpavlic@netwalk.com` 2 Dec 2000 If you are careful, you can avoid this. For example, it is easy to write a simple script (a super monitor) which runs every monitor you want. This way only one message is sent when one or more services goes down on a system. Unless you specify the actual failure in the message, you will bring the entire system goes down when one service fails.
>
> There are other alternatives to make sure you only receive one notice when a service goes down, but are you sure you would want that? What if genuinely two services on a system go down but all the other services are up? I, personally, would want to receive notifications about both services.
>
> Ideally, what you want is a dependency setup. Most other system monitors (like the popular "WhatsUp" by IpSwitch (or whatever they call themselves now)) that will only send you one notification if, for example, the ICMP monitor reports a failure. This functionality can also be easily built into the monitors.
>
> So the way I see it, mon is fine, but sometimes the monitors one uses with mon might need a little work. Mon is nice because it is so versatile and pluggable. It's modular and doesn't lock one into using some proprietary scripting language to build monitors. Of course, this also makes it very slow.
>
> All of these things can be improved, of course, in a number of ways which can be addressed and fixed one by one. However, most people have gone with ldirectord, so mon, in this application, seems to really have been forgotten. :(
>
> I use mon and have been using it for a long time now and have no real problems with it. My biggest problem was that both of my redundant linuxdirectors notified me when things went down. I just wrote in a simple addition to the mailto script which figured out if the machine on which it was running was a master and if and ONLY if it was, it would send a message. That solved that problem.
>
> Now you should note that I have a made quite a few changes to mon to make it more LinuxDirector-friendly. Rather than configuring mon through its configuration scripts, I configure mon and ipvsadm all through some very simple configuration files I use that control both equally. This required quite a bit of hacking inside mon to get it to dynamically create configuration data, but all of this modification isn't needed for the average LVS-admin.
>
> Abstract: mon ain't really that bad. If ldirectord was everything I'd want it to be, it'd be mon.... consequence: I use mon. :)

21.7 Mon Install

Mon is installed on the director.

Most of mon is a set of perl scripts. There are few files to be compiled - it is mostly ready to go (rpc.monitor needs to be compiled, but you don't need it for LVS).

You do the install by hand.

```
$ cd /usr/lib
$ tar -zxvof /your_dirpath/mon-x.xx.tar.gz
```

this will create the directory /usr/lib/mon-x.xx/ with mon and its files already installed.

LVS comes with virtualserver.alert (goes in alert.d) and ssh.monitor (goes in mon.d).

Make the directory "mon-x.xx" accessable as "mon" by linking it to "mon" or by renaming it

```
$ln -s mon-x.xx mon
or
$mv mon-x.xx mon
```

Copy the man files (mon.1 etc) into /usr/local/man/man1

Check that you have the perl packages required for mon to run

$perl -w mon

do the same for all the perl alerts and monitors that you'll be using (telnet.monitor, dns.monitor, http_t.monitor, ssh.monitor).

DNS in /etc/services is known as "domain" and "nameserver" but not "dns". To allow the use of the string "dns" in the lvs_xxx.conf files and to enable 10.1 (configure_lvs.pl) to autoinclude the dns.monitor, add the string "dns" to the port 53 services in /etc/services with an entry like

```
    domain              53/tcp          nameserver dns  # name-domain server
    domain              53/udp          nameserver dns
```

Mon expects executables to be in /bin, /usr/bin or /usr/sbin. The location of perl in the alerts is #!/usr/bin/perl (and not /usr/local/bin/perl) - make sure this is compatible with your setup. (Make sure you don't have one version of perl in /usr/bin and another in /usr/local/bin).

The 10.1 (configure script) will generate the required mon.cf file for you (and if you like copy it to the cannonical location of /etc/mon).

Add an auth.cf file to /etc/mon

I use

```
#auth.cf ----------------------------------
# authentication file
#
# entries look like this:
# command: {user|all}[,user...]
#
# THE DEFAULT IS TO DENY ACCESS TO ALL IF THIS FILE
# DOES NOT EXIST, OR IF A COMMAND IS NOT DEFINED HERE
#

list:           all
reset:          root
loadstate:      root
savestate:      root
term:           root
stop:           root
```

```
start:          root
set:            root
get:            root
dump:           root
disable:        root
enable:         root

#end auth.cf ----------------------------
```

21.8 Mon Configure

This involves editing /etc/mon/mon.cf, which contains information about

- nodes monitored
- how to detect if a node:service is up (does the node ping, does it serve http...?)
- what to do when the node goes down and what to do later when it comes back up.

The mon.cf generated by 10.1 (configure_lvs.pl)

- assigns each node to its own group (nodes are brought down one at a time rather than in groups - I did this because it was easier to code rather than for any good technical reason).
- detects whether a node is serving some service (eg telnet/http) selecting, if possible, a monitor for that service, otherwise defaulting to fping.monitor which detects whether the node is pingable.
- on failure of a realserver, mon sends mail to root (using mail.alert) and removes the realserver from the ipvsadm table (using virtualserver.alert).
- on recovery sends mail to root (using mail.alert) and adds the realserver back to the pool of working realservers in the ipvsadm table (using virtualserver.alert).

21.9 Testing mon without LVS

The instructions here show how to get mon working in two steps. First show that mon works independantly of LVS, then second bring in LVS.

The example here assumes a working LVS-DR with one realserver and the following IPs. LVS-DR is chosen for the example here as you can set up LVS-DR with all machines on the same network. This allows you to test the state of all machines from the client (ie using one kbd/monitor). (Presumably you could do it from the director too, but I didn't try it.)

```
lvs IP (VIP): eth0 192.168.1.110
director:     eth0 192.168.1.1/24 (admin IP)
              eth0:1 192.168.1.110/32 (VIP)
realserver:  eth0 192.168.1.8/24
```

On the director, test ping.monitor (in /usr/lib/mon/mon.d) with

```
$ ./fping.monitor 192.168.1.8
```

You should get the command prompt back quickly with no other output. As a control test for a machine that you know is not on the net

```
$ ./fping.monitor 192.168.1.250
192.168.1.250
```

ping.monitor will wait for a timeout (about 5secs) and then return the IP of the unpingable machine on exit.

Check test.alert (in /usr/lib/mon/alert.d) - it writes a file in /tmp

$ ./test.alert foo

you will get the date and "foo" in /tmp/test.alert.log

As part of generating the rc.lvs_dr script, you will also have produced the file mon_lvsdr.cf. To test mon, place this in /etc/mon/mon.cf

```
#-----------------------------------------------------
#mon.cf
#
#mon config info, you probably don't need to change this very much
#

alertdir   = /usr/lib/mon/alert.d
mondir     = /usr/lib/mon/mon.d
#maxprocs    = 20
histlength = 100
#delay before starting
#randstart = 60s

#------
hostgroup LVS1 192.168.1.8

watch LVS1
#the string/text following service (to OEL) is put in header of mail messages
#service "http on LVS1 192.168.1.8"
service fping
        interval 15s
        #monitor http.monitor
        #monitor telnet.monitor
        monitor fping.monitor
        allow_empty_group
        period wd {Sun-Sat}
        #alertevery 1h
                #alert mail.alert root
                #upalert mail.alert root
                alert test.alert
                upalert test.alert
                #-V is virtual service, -R is remote service, -P is protocol, -A is add/delete (t|u)
                #alert virtualserver.alert -A -d -P -t -V 192.168.1.9:21 -R 192.168.1.8
                #upalert virtualserver.alert -A -a -P -t -V 192.168.1.9:21 -R 192.168.1.8 -T -m -w 1

#the line above must be blank
```

```
#mon.cf---------------------------
```

Now we will test mon on the realserver 192.168.1.8 independantly of LVS. Edit /etc/mon/mon.cf and make sure that all the monitors/alerts except for fping.monitor and test.alert are commented out (there is an alert/upalert pair for each alert, leave both uncommented for test.alert).

Start mon with rc.mon (or S99mon)

Here is my rc.mon (copied from the mon package)

```
# rc.mon -------------------------------
# You probably want to set the path to include
# nothing but local filesystems.
#

echo -n "rc.mon "

PATH=/bin:/usr/bin:/sbin:/usr/sbin
export PATH

M=/usr/lib/mon
PID=/var/run/mon.pid

if [ -x $M/mon ]
        then
        $M/mon -d -c /etc/mon/mon.cf -a $M/alert.d -s $M/mon.d -f 2>/dev/null
        #$M/mon -c /etc/mon/mon.cf -a $M/alert.d -s $M/mon.d -f
fi
#-end-rc.mon----------------------------
```

After starting mon, check that mon is in the ps table (ps -auxw | grep perl). When mon comes up it will read mon.cf and then check 192.168.1.8 with the fping.monitor. On finding that 192.168.1.8 is pingable, mon will run test.alert and will enter a string like

Sun Jun 13 15:08:30 GMT 1999 -s fping -g LVS3 -h 192.168.1.8 -t 929286507 -u -l 0

into /tmp/test.alert.log. This is the date, the service (fping), the hostgroup (LVS), the host monitored (192.168.1.8), unix time in secs, up (-u) and some other stuff I didn't need to figure out to get everything to work.

Check for the "-u" in this line, indicating that 192.168.1.8 is up.

If you don't see this file within 15-30secs of starting mon, then look in /var/adm/messages and syslog for hints as to what failed (both contain extensive logging of what's happening with mon). (Note: syslog appears to be buffered, it may take a few more secs for output to appear here).

If neccessary kill and restart mon

```
$ kill -HUP `cat /var/run/mon.pid`
```

Then pull the network cable on machine 192.168.1.8. In 15secs or so you should hear the whirring of disks and the following entry will appear in /tmp/test.alert.log

Sun Jun 13 15:11:47 GMT 1999 -s fping -g LVS3 -h 192.168.1.8 -t 929286703 -l 0

Note there is no "-u" near the end of the entry indicating that the node is down.

Watch for a few more entries to appear in the logfile, then connect the network cable again. A line with -u should appear in the log and no further entries should appear in the log.

If you've got this far, mon is working.

Kill mon and make sure root can send himself mail on the director. Make sure sendmail can be found in /usr/lib/sendmail (put in a link if neccessary).

Next activate mail.alert and telnet.monitor in /etc/mon/mon.cf and comment out test.alert. (Do not restart mon yet)

Test mail.alert by doing

```
$ ./mail.alert root
hello
^D
```

root is the address for the mail, hello is some arbitrary STDIN and controlD exits the mail.alert. Root should get some mail with the string "ALERT" in the subject (indicating that a machine is down).

Repeat, this time you are sending mail saying the machine is up (the "-u")

```
$ ./mail.alert -u root
hello
^D
```

Check that root gets mail with the string "UPALERT" in the subject (indicating that a machine has come up).

Check the telnet.monitor on a machine on the net. You will need tcp_scan in a place that perl sees it. I moved it to /usr/bin. Putting it in /usr/local/bin (in my path) did not work.

```
$ ./telnet.monitor 192.168.1.8
```

the program should exit with no output. Test again on a machine not on the net

```
$ ./telnet.monitor 192.168.1.250
192.168.1.250
```

the program should exit outputting the IP of the machine not on the net.

Start up mon again (eg with rc.mon or S99mon), watch for one round of mail sending notification that telnet is up (an "UPALERT) (note: for mon-0.38.21 there is no initial UPALERT). There should be no further mail while the machine remains telnet-able. Then pull the network cable and watch for the first ALERT mail. Mail should continue arriving every mon time interval (set to 15secs in mon_lvs_test.cf). Then plug the network cable back in and watch for one UPALERT mail.

If you don't get mail, check that you re-edited mon.cf properly and that you did kill and restart mon (or you will still be getting test.alerts in /tmp). Sometimes it takes a few seconds for mail to arrive. If this happens you'll get an avalanche when it does start.

If you've got here you are really in good shape.

Kill mon (kill ‘cat /var/run/mon.pid‘)

21.10 Can virtualserver.alert send commands to LVS?

(virtualserver.alert is a modified version of Wensong's orginal file, for use with 2.2 kernels. I haven't tested it back with a 2.0 kernel. If it doesn't work and the original file does, let me know)

run virtualserver.alert (in /usr/lib/mon/alert.d) from the command line and check that it detects your kernel correctly.

$ ./virtualserver.alert

you will get complaints about bad ports (which you can ignore, since you didn't give the correct arguments). If you have kernel 2.0.x or 2.2.x you will get no other output. If you get unknown kernel errors, send me the output of 'uname -r'. Debugging print statements can be uncommented if you need to look for clues here.

Make sure you have a working LVS-DR LVS serving telnet on a realserver. If you don't have the telnet service on realserver 192.168.1.8 then run

$ipvsadm -a -t 192.168.1.110:23 -r 192.168.1.8

then run ipvsadm in one window.

$ipvsadm

and leave the output on the screen. In another window run

$ ./virtualserver.alert -V 192.168.1.110:23 -R 192.168.1.8

this will send the down command to ipvsadm. The entry for telnet on realserver 192.168.1.8 will be removed (run ipvsadm again).

Then run

$ ./virtualserver.alert -u -V 192.168.1.110:23 -R 192.168.1.8

and the telnet service to 192.168.1.8 will be restored in the ipvsadm table.

21.11 Running mon with LVS

Connect all network connections for the LVS and install a LVS-DR LVS with INITIAL_STATE="off" to a single telnet realserver. Start with a file like lvs_dr.conf.single_telnet_off adapting the IP's for your situation and produce the mon_xxx.cf and rc.lvs_xxx file. Run rc.lvs_xxx on the director and then the realserver.

The output of ipvsadm (on the director) should be

```
grumpy:/etc/mon# ipvsadm
IP Virtual Server (Version 0.9)
Protocol Local Address:Port Scheduler
      -> Remote Address:Port   Forward Weight ActiveConn FinConn
TCP 192.168.1.110:23 rr
```

showing that the scheduling (rr) is enabled, but with no entries in the ipvsadm routing table. You should NOT be able to telnet to the VIP (192.168.1.110) from a client.

Start mon (it's on the director). Since the realserver is already online, mon will detect a functional telnet on it and trigger an upalert for mail.alert and for virtualserver.alert. At the same time as the upalert mail arrives run ipvsadm again. You should get

```
grumpy:/etc/mon# ipvsadm
```

```
IP Virtual Server (Version 0.9)
Protocol Local Address:Port Scheduler
      -> Remote Address:Port   Forward Weight ActiveConn FinConn
TCP 192.168.1.110:23 rr
      -> 192.168.1.8:23        Route   1      0          0
```

which shows that mon has run ipvsadm and added direct routing of telnet to realserver 192.168.1.8. You should now be able to telnet to 192.168.1.110 from a client and get the login prompt for machine 192.168.1.8.

Logout of this telnet session, and pull the network cable to the realserver. You will get a mail alert and the entry for 192.168.1.8 will be removed from the ipvsadm output.

Plug the network cable back in and watch for the upalert mail and the restoration of LVS to the realserver (run ipvsadm again).

If you want to, confirm that you can do this for http instead of telnet.

You're done. Congratulations. You can use the mon_xxx.cf files generated by 10.1 (configure.pl) from here.

21.12 Why is the LVS monitored for failures/load by an external agent rather than by the kernel?

> Patrick Kormann `pkormann@datacomm.ch` Wouldn't it be nice to have a switch that would tell ipvsadm 'If one of the realservers is unreachable/connection refused, take it out of the list of real servers for x seconds' or even 'check the availability of that server every x seconds, if it's not available, take it out of the list, if it's available again, put it in the list'.

Lars

That does not belong in the kernel. This is definetely the job of a userlevel monitoring tool.

I admit it would be nice if the LVS patch could check if connections directed to the realserver were refused and would log that to userlevel though, so we could have even more input available for the monitoring process.

> and quirks to make lvs a real high-availability system. The problem is that all those external checks are never as effective as a decition be the 'virtual server' could be.

That's wrong.

A userlevel tool can check reply times, request specific URLs from the servers to check if they reply with the expected data, gather load data from the real servers etc. This functionality is way beyond kernel level code.

> Michael Sparks `zathras@epsilon3.mcc.ac.uk` Personally I think monitoring of systems is probably one of the things the lvs system shouldn't really get into in it's current form. My rationale for this is that LVS is a fancy packet forwarder, and in that job it excels.
>
> For the LVS code to do more than this, it would require for TCP services the ability to attempt to connect to the *service* the kernel is load balancing - which would be a horrible thing for a kernel module to do. For UDP services it would need to do more than pinging... However, in neither case would you have a convincing method for determining if the *services* on those machines was still running effectively, unless you put a large amount of protocol knowledge into the kernel. As a result, you would still need to have external monitoring systems to find out whether the services really are working or not.

> For example, in the pathological case (of many that we've seen :-) of a SCSI subsystem failure resulting in indestructable inodes on a cache box, a cache box can reach total saturation in terms of CPU usage, but still respond correctly to pings and TCP connections. However nothing else (or nothing much) happens due to the effective system lockup. The only way round such a problem is to have a monitoring system that knows about this sort of failure, and can then take the service out.
>
> There's no way this sort of failure could be anticipated by anyone, so putting this sort of monitoring into the kernel would create a false illusion of security - you'd still need an auxillary monitoring system. Eg - it's not just enough for the kernel to mark the machine out of service - you need some useful way of telling people what's gone wrong (eg setting off people's pager's etc), and again, that's not a kernel thing.

Lars

Michael, I agree with you.

However, it would be good if LVS would log the failures it detects. ie, I _think_ it can notice if a client receives a port unreachable in response to a forwarded request if running masquerading, however it cannot know if it is running DR or tunneling because in that case it doesn't see the reply to the client.

> _think_ it can notice if a client receives a port unreachable in response to a

Wensong

Currently, the LVS can handle ICMP packets for virtual services, and forward them to the right place. It is easy to set the weight of the destination zero or temperarily remove the dest entry directly, if an PORT_UNREACH icmp from the server to the client passes through the LVS box.

If we want the kernel to notify monitoring software that a real server is down in order to let monitoring software keep consistent state of virtual service table, we need design efficient way to notify monitoring, more code is required. Anyway, there is a need to develop efficient communication between the LVS kernel and monitoring software, for example, monitoring software get the connection number efficiently, it is time-consuming to parse the big IPVS table to get the connection numbers; how to efficiently support ipvsadm -L <protocol, ip, port>? it is good for applications like Ted's 1024 virtual services. I checked the procfs code, it still requires one write_proc and one read_proc to get per virtual service print, it is a little expensive. Any idea?

> Currently, the LVS can handle ICMP packets for virtual services, and forward them to the right place. It is easy to set the weight of the destination zero or temperarily remove the dest entry directly, if an PORT_UNREACH icmp from the server to the client passes through the LVS box.

Julian Anastasov `uli@linux.tu-varna.acad.bg`

PORT_UNREACH can be returned when the packet is rejected from the real server's firewall. In fact, only UDP returns PORT_UNREACH when the service is not running. TCP returns RST packet. We must carefully handle this (I don't know how) and not to stop the real server for all clients if we see that one client is rejected. And this works only if the LVS box is default gw for the real servers, i.e. for any mode: MASQ(it's always def gw), DROUTE and TUNNEL (PROT_UNREACH can be one of the reasons not to select other router for the outgoing traffic for these two modes). But LVS cn't detect the outgoing traffic for DROUTE/TUNNEL mode. For TUNNEL it can be impossible if the real servers are not on the LAN.

So, the monitoring software can solve more problems. The TCP stack can return PORT_UNREACH but if the problem with the service in the real server is more complex (real server died, daemon blocked) we can't expect PORT_UNREACH. It is send only when the host is working but the daemon is stooped. Please restart this daemon. So, don't rely on the real server, in most of the cases he can't tell "Please remove me from the VS configuration, I'm down" :) This is job for the monitoring software to exclude the destinations and even to delete the service (if we switch to local delivery only, i.e. when we switch from LVS to WEB only mode for example). So, I vote for the monitoring software to handle this :)

Wensong

Yeah, I prefer that monitoring software handles this too, because it is a unified approach for LVS-NAT, LVS-Tun and LVS-DR, and monitoring software can detect more failures and handle more things according to the failures.

What we discuss last time is that the LVS kernel sets the destination entry unavailable in virtual server table if the LVS detect some icmp packets (only for LVS-NAT) or RST packet etc. This approach might detect this kinds of problems just a few seconds earlier than the monitoring software, however we need more code in kernel to notify the monitoring software that kernel changes the virtual server table, in order to let the monitoring software keep the consistent view of the virtual server table as soon as possible. Here is a tradeoff. Personally, I prefer to keeping the system simple (and effective), only one (monitoring software) makes decision and keeps the consistent view of VS table.

22 Setting up Linux-HA for directors using rpms

This was posted to the mailing list by Peter Mueller Peter Mueller `pmueller@sidestep.com` on 17Sep2001. (The original was in html with DOS carriage control. I've converted it by hand. There may be some parsing errors. Joe)

original mon files from Juri, data posted from personal experience or mailing list (linux-ha or LVS) or respective websites.

urls

- *mon* `<http://www.kernel.org/pub/software/admin/mon/html>`
- *ultramonkey* `<http://ultramonkey.sourceforge.net/>`
- *getting started with Linux-HA* `<http://linux-ha.org/download/GettingStarted.html>`

note : these scripts assume mon 0.99.2. For simplicity in install I downloaded mon-0.38rpm (couldn't find new 0.99.2 rpm) and upgraded to 0.99.2 via source. I then changed appropriate lines in /etc/rc.d/init.d/mon.

22.1 linux-ha howto

This document is a mini how-to get heartbeat working between two individually working LVS boxes. It is certainly not intended to be all-encompasing document detailing everything imagineable. What it is intended to deliver is an 'essential steps' to getting LVS-HA functional. And you definitely should have two individually functioning boxes before even attempting this. (Yes, go back and test your setup with each box to insure it works!).

Another important note to add is that I have only tested this setup with Ultramonkey RPMs. I don't know if your setup will work. I wouldn't trust this document unless you do the same. (I would be interested in knowing if the HA features are the same for all 'heartbeat' setups..)

PS. - apologies if this document is RedHat biased, I'm running from VALinux boxes that are RedHat configured.

22.1.1 Fix the (possible) ethernet alias issue.

By now you've setup a dummy alias device on each LVS box (most likely eth0:0). This alias device is unecessary and potentially problematic in the HA-setup. The reason for this is that the heartbeat software (/etc/ha.d/resource.d/) actually creates a new eth0:0 device on the active box. If you have an eth0:0 (or whatever) alias configured for your VIP on the standby director box, you might get a " VSbox2 kernel: Uh Oh, MAC address 00:02:B3:03:9A:13 claims to have our IP address (vip.ip.goes.here) (duplicate IP conflict likely)" error! Not good...

If I were you I'd move your alias script out of your /etc/sysconfig/network-scripts/ directory and restart networking to clear out that alias. Alternatively, if you are using shell scripts then you should modify those to not control alias ips.

22.1.2 Configure /etc/ha.d/. files.

- authkeys authkeys MUST be permission-set to 600 or 400 from what I have read. Be sure this is the case. authkeys should contain something like

  ```
  auth 2
  #1 crc
  2 sha1 passwordhere
  #3 md5 Hello!
  ```

  Since you want to make sure this file is the same on both machines, get it setup on one box and scp or ftp the file over to the other.

- haresources haresources is convoluted to understand until you have a working setup. The example config show things like :

  ```
  #just.linux-ha.org      135.9.216.110 httpa
  ```

  when something like : primary.director.box.goes.here shared.resources.address.here http

  ```
  #vs1.foo.com vip.foo.com http # <-- put actual IP down instead of vip.foo.com
  vs1.foo.com IPaddr::10.10.10.10 ldirectord::ldirectord.cf # <-- if you use ldirector like this
  # multiple VIP example follows
  # vs1.so.com IPaddr::10.10.10.10 IPaddr::10.10.10.254 ldirectord::ldirectord.cf
  ```

  It's important to note that the box listed in the first box is considered the 'primary' director box and usually takes control in the event of uncertainty. (Definitely look at nice_failback in ha.cf if you're interested in this thread).

- ha.cf high-availability configuration file. yep, looks like the meat of the subject! I'll just post my config, which assumes you use ttyS0 and eth0 for your links to the other director.

  ```
  #       File to wirte debug messages to debugfile /var/log/ha-debug
  #       File to write other messages to logfile /var/log/ha-log
  #       Facility to use for syslog()/logger logfacility     local0
  #       keepalive: how many seconds between heartbeats
  keepalive 1
  #       deadtime: seconds-to-declare-host-dead
  deadtime 20
  #       hopfudge maximum hop count minus number of nodes in config
  ```

```
    #hopfudge 1
    #       serial  serialportname ...
    serial  /dev/ttyS0
    #       Only for serial ports.  It applies to both PPP/UDP and "raw" ports
    #       This means run PPP over ports ttyS1 and ttyS2
    #       Their respective IP addresses are as listed.
    #       Note that I enforce that these are local addresses.
    #       Other addresses are almost certainly a mistake.
    #ppp-udp        /dev/ttyS1 10.0.0.1 /dev/ttyS2 10.0.0.2
    #       Baud rate for both serial and ppp-udp ports...
    baud    19200
    #       What UDP port to use for udp or ppp-udp communication?
    udpport 694
    #       What interfaces to heartbeat over?
    udp     eth0
    #       Watchdog is the watchdog timer.
    #       If our own heart doesn't beat for
    #       a minute, then our machine will reboot.
    #watchdog /dev/watchdog
    #       Nice_failback sets the behavior when performing a failback:
    #
    #       - if it's on, when the primary node starts or comes back from any
    #         failure and the cluster is already active, i.e. the secondary
    #         server performed a failover, the primary stays quiet, acting as a
    #         secondary.  This way some operations like syncing disks can be
    #         easily done.
    #       - if it's off (default), the primary node will always be the primary,
    #         whenever it's powered on.
    nice_failback off               # <-- might want to turn this on after you get things working
    #       Tell what machines are in the cluster
    #       node    nodename ...    -- must match uname -n
    node    vs1.foo.com     # <-- must match uname -n !
    node    vs2.foo.com     # <-- must match uname -n !
```

22.2 Stop ldirectord from starting, ensure heartbeat starts on reboot

```
/etc/rc.d/init.d/ldirectord stop.
/usr/sbin/chkconfig --level 2345 ldirectord off
/usr/sbin/chkconfig --level 345 heartbeat on # <-- run on whatever init levels you want
```

22.3 starting heartbeat and verifying functionality

At this point you should have linux-director NOT running on both boxes. If you type ipvsadm -L on either box you should get:

```
[root@vs1 ha.d]# ipvsadm -L
IP Virtual Server version 0.9.11 (size=3D4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
```

Now start up heartbeat. tail /var/log/messages, and /var/log/ha-log for important log information. My /var/log/messages looks like :

```
Apr 24 13:12:38 vs1 heartbeat[2070]: Configuration validated. Starting heartbeat.
Apr 24 13:12:39 vs1 heartbeat[2075]: Starting serial heartbeat on tty /dev/ttyS0
Apr 24 13:12:39 vs1 heartbeat[2075]: UDP heartbeat started on port 694 interface eth0
Apr 24 13:12:39 vs1 heartbeat[2077]: node vs1.internal.smartbasket.com -- link eth0: status up
Apr 24 13:12:39 vs1 heartbeat[2077]: node stage-monitor -- link /dev/ttyS0: status up
Apr 24 13:12:39 vs1 heartbeat[2077]: node stage-monitor -- link eth0: status up
```

And a quick check of ifconfig on the primary director shows the alias interface (eth0:0) appears. Note that eth0:0 is *NOT* present when heartbeat isn't running.

```
[root@vs1 ha.d]# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:02:B3:06:B6:45 =20
          inet addr:10.0.1.5  Bcast:10.0.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:106550 errors:0 dropped:0 overruns:0 frame:0
          TX packets:75338 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100=20
          Interrupt:10 Base address:0xd000=20

eth0:0    Link encap:Ethernet  HWaddr 00:02:B3:06:B6:45 =20
          inet addr:10.0.1.10  Bcast:10.0.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:10 Base address:0xd000=20
```

A ps aux on the active director shows :

```
root      1648  0.0  0.1  1444  868 ttyS0    SL   13:17   0:00 /usr/lib/heartbeat/heartbeat
root      1650  0.0  0.1  1332  748 ttyS0    SL   13:17   0:00 /usr/lib/heartbeat/heartbeat
root      1651  0.0  0.1  1332  736 ttyS0    SL   13:17   0:00 /usr/lib/heartbeat/heartbeat
root      1652  0.0  0.1  1328  736 ttyS0    S    13:17   0:00 /usr/lib/heartbeat/heartbeat
root      1653  0.0  0.1  1332  732 ttyS0    SL   13:17   0:00 /usr/lib/heartbeat/heartbeat
root      1654  0.0  0.1  1328  728 ttyS0    S    13:17   0:00 /usr/lib/heartbeat/heartbeat
root      1775  0.0  0.8  5352 4388 ttyS0    S    13:17   0:00 perl /etc/ha.d/resource.d/ldirectord
root      1869  0.0  0.1  2344  724 pts/0    R    13:20   0:00 ps aux
```

22.4 Test your fail-over features, understand HA.

At this point you should test around your failover functionality and learn how your setup works. You also need to customize your ha.cf file to the specifications for your site.

As noted in the 'getting started' document mentioned in the url section above, be certain to NOT yank all heartbeat medium cables at once! This will cause a 'split brain' scenario and you won't be happy! Test failover possibilities one at a time, or catastrophically!

22.5 Configuration of mon - recommended

- add lines to /etc/services

```
mon             2583/tcp                        # MON
mon             2583/udp                        # MON traps
```

- install Perl modules

  get modules,files from 10.0.0.34 ftp, directory /mon, install.

  Convert-BER, Period, Time-HiRes, Mon, fping

- convert headers into perl headers

  ```
  cd /usr/include
  h2ph *.h sys/*.h asm/*.h
  ```

- install mon-rpm

  to get /etc/rc.d/init.d/mon & other nice features automate-installed. afterwards update to the latest source available to get bug fixes. I recommend untarring in /usr/src/mon-x and symlinking that directory to /usr/src/mon for simplicity and ease of upgrade.

- install mon.cf file into /etc/mon/

  modify if appropriate (ie, change the gateway that it monitors). the mon.cf file contains lots of configuration options which you should be familiar with, such as log locations. Example included below.

- copy any specific monitors from staging or production to your new box.

  In this example we are using a few extraneous monitors : fping.monitor, pid.monitor, heartbeat.alert, and restartheartbeat.alert. all '.monitor' files go in the 'mon.d' folder, and all '.alert' files go in the alert.d folder.

- change the /etc/rc.d/init.d/mon file to point to appropriate paths

  change /usr/lib/mon to /usr/src/mon and the cf line to /etc/mon/mon.cf. (either that or copy from a working server).

- make sure a copy of fping is in the restricted path solicited by /etc/rc.d/init.d/mon.

  one way of fixing this is via a simple 'cp /usr/local/sbin/fping /usr/sbin' (or /usr/local/bin or anywhere in your path).

- create /etc/mon/monusers.cf. Instructions in man file (man mon).

  ```
  #!/bin/bash
  # example heartbeat.alert from Juri
  # Script to start/stop heartbeat daemon
  # Put a line like
  # alert heartbeat.alert
  # or
  # upalert heartbeat.alert
  # in your mon config file

  HEARTBEAT="/etc/rc.d/init.d/heartbeat"
  if [ $9 = "-u" ]; then
          $HEARTBEAT start
  else
          $HEARTBEAT stop
  fi
  ```

```
#!/bin/sh
# example pid.mon from Juri
# Script for mon to check wether a process is running or not.
# Invoke with
# monitor pid.monitor process

/sbin/pidof -s $1 > /dev/null 2>&1

if [ $? -eq "0" ]; then
        echo "$1 running"
        exit 0
else
        echo "$1 not running"
        exit 1
fi

# Sample mon.cf configuration file for mon, originally from Juri
#
# You have to restart mon after editing this file in order for your
# changes to take effect.

authtype        =       userfile
userfile        =       /etc/mon/monusers.cf
cfbasedir       =       /etc/mon
alertdir        =       /usr/src/mon/alert.d
mondir          =       /usr/src/mon/mon.d
logdir          =       /var/log/mon
dtlogfile       =       /var/log/mon/downtime
dtlogging       =       yes
historicfile    =       /var/log/mon/history
maxprocs        =       20
histlength      =       100
monerrfile      =       /var/log/mon/errfile

# Hostgroup entries
#hostgroup node1 stage-monitor
#
#hostgroup node2 vs1
#
hostgroup gateway 10.0.1.2

#hostgroup heartbeat localhost


###########
# Gateway #
###########

watch gateway
        service fping
                interval 10s
                monitor fping.monitor
                #comp_alerts    <-- default starting in mon 0.99.1
                period NORMAL: wd {Sun-Sat}
                        numalerts 1
                        alertafter 3
                        alert heartbeat.alert
                        upalert restartheartbeat.alert # read mon file

#############
# Heartbeat #
#############

#watch heartbeat
#       service heartbeat
#               interval 15s
#               monitor pid.monitor /usr/lib/heartbeat/heartbeat
#               depend gateway:fping
#               dep_behavior m
#               period NORMAL: wd {Sun-Sat}
#                       #alert restartheartbeat.alert
#                       upalert heartbeat.alert

##############
# First node #
##############
#
#watch node1
#       service http
#               interval 10s
#               monitor http.monitor
#               period NORMAL: wd {Sun-Sat}
#                       alert restart.alert httpd ;;
#               period ADVANCED: wd {Sun-Sat}
#                       alert mail.alert root
#                       alert reboot.alert
#                       alertafter 3
#                       alertevery 1m
# Example for testing disk operations
#       service disk
#               interval 10s
#               monitor nfs.monitor /vol/shared/0/ ;;
#               period wd {Sun-Sat}
#                        alert mail.alert -s 'REBOOT: Disk not responding!' root
#                       alert hardreboot.alert
#                       alertafter 2
#                       alertevery 1m
# Example for testing rpc based services such as nfs, nis etc.
#       service rpc
#               interval 10s
#               monitor rpc.monitor -r mountd -r nfs
#               period wd {Sun-Sat}
#                        alert mail.alert root
#                       alertafter 2
#                        alertevery 1m
###############
# Second node #
###############
#
#watch node2
# Example for testing disk operations
#       service disk
#               interval 10s
#               monitor nfs.monitor /tmp ;;
#               period wd {Sun-Sat}
#                       alert mail.alert root
#                       alert hardreboot.alert
#                       alertafter 2
#                       alertevery 1m
# Example for testing samba
#       service samba
#               interval 15s
#               monitor tcp.monitor -p 139 localhost
#               period wd {Sun-Sat}
#                        alert restart.alert smb
#               period ADVANCED: wd {Sun-Sat}
#                        alert mail.alert root
#                       alert reboot.alert
#                       alertafter 3
#                       alertevery 1m
```

23 Monitoring director throughput

Much of this is relatively recent (end 2001).

23.1 ipvsadmm

The number of active/inactive connections are available from the output of ipvsadm.

> Julian 22 May 2001 Conns is a counter and is incremented when a new connection is created. It is *not* incremented when a client re-uses a port to make a new connection (Joe, - this is common with Linux).

```
director:/etc/lvs# ipvsadm
IP Virtual Server version 0.2.12 (size=16384)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  lvs2.mack.net:0 rr persistent 360
  -> bashfull.mack.net:0            Route   1      0          0
  -> sneezy.mack.net:0              Route   1      0          0
```

```
TCP  lvs2.mack.net:telnet rr
  -> bashfull.mack.net:telnet       Route   1      0          0
  -> sneezy.mack.net:telnet         Route   1      0          0
```

23.2 /proc system (originally /proc/net/ip_vs_stats)

Cyril Bouthors

Where can I get the info originally in /proc/net/ip_vs_stats and removed since 0.9.4?

> Wensong Zhang `wensong@gnuchina.org` 20 Nov 2001 /proc/net/ip_vs_stats is now for global stats.
>
> You can get per-service statistics by ipvsadm -Ln –stats -t|u|f service-address
>
> If you want to program to get statistics info, you can let your program to use libipvs. It is easier to get into in this way.

Here's the write that went with the original code.

Packet throughput (in 64 bit integers) is in /proc/net/ip_vs_stats or /proc/net/ip_masq/vs_stats. The counters are not resetable, you have to keep the previous reading and substract. Output is in hexadecimal.

```
kernel 2.4:#cat /proc/net/ip_vs_stats
kernel 2.2:#cat /proc/net/ip_masq/vs_stats
```

Here's the statistics

```
director:# more /proc/net/ip_vs_stats
TotalConns   InPkts  OutPkts          InBytes         OutBytes
      98F9 13945999 13609E49        613AD3B2F       4F90FE6F9E

Virtual Service
Pro VirtService      Conns   InPkts  OutPkts          InBytes         OutBytes
TCP C0A8026E:0000        4       12        0        00000043B        000000000
TCP C0A8026E:0017        7      3A9        0        00000C3A5        000000000
Real Service
Pro VirtService   RealService      Conns   InPkts  OutPkts          InBytes         OutBytes
TCP C0A8026E:0000 C0A8010C:0000        4       14        0        0000004B4        000000000
TCP C0A8026E:0000 C0A8010B:0000        1        3        0        0000000B4        000000000
TCP C0A8026E:0017 C0A8010C:0017        4       A2        0        00000232A        000000000
TCP C0A8026E:0017 C0A8010B:0017        4      32A        0        00000A827        000000000
```

Joe

Can I zero out these counters if I want to get rates, or should I store the last count?

> Ratz, May 2001 There was a recent (2 months ago) talk about zeroing in-kernel counters and I'm not so sure if all the kernel hacker gurus agreed but:
>
> *You must not zero a counter in the kernel!*
>
> I didn't really understand the arguments against or pro zeroing counters so I'm not a big help here, but if others agree we certainly can add this feature. It would be ipvsadm -Z as an analogy to ip{chains|tables}. BTW, we are proud of haveing 64 bit counters in the kernel :)
>
> Storing ... there are different approaches to this (complexity order):

- Use a script that extracts the info and writes it flat to a file
- Use MRTG or rrdtool since I reckon you wanted to use the stats to generate some graphics anyway. These tools handle the problem for you.

  MRTG requires SNMP, but you can have a slightly modified snmpd.conf and execute a script that parses /proc/net/ip_masq/vs_stats and writes it into a file. The advantage of this over the first one is, that you can write the current number into one file and mrtg will know how to draw the graph.

  I give you an example:

  We have a customer named plx. Now he has only one service and 2 realserver. We extended the snmpd.conf with following lines:

  ```
  exec lbsessions /bin/sh /opt/tac/snmp/lbsessions
  exec lbsessions.plx.total /bin/sh /opt/tac/snmp/lbsessions.plx.total
  exec lbsessions.plx.web-web1 /bin/sh /opt/tac/snmp/lbsessions.plx.web-web1
  exec lbsessions.plx.web-web2 /bin/sh /opt/tac/snmp/lbsessions.plx.web-web2
  ```

  The scripts are awk scripts that get the information accordingly to the service or the realserver. You can then do a table walk of the OID 1.3.6.1.4.1.2021.8 to see what your values are:

  snmpwalk $IP $COMMUNITY .1.3.6.1.4.1.2021.8

  Example output if everything is ok:

  ```
  enterprises.ucdavis.extTable.extEntry.extNames.1 = lbsessions
  enterprises.ucdavis.extTable.extEntry.extNames.2 = lbsessions.plx.total
  enterprises.ucdavis.extTable.extEntry.extNames.3 = lbsessions.plx.web-web1
  enterprises.ucdavis.extTable.extEntry.extNames.4 = lbsessions.plx.web-web2
  enterprises.ucdavis.extTable.extEntry.extCommand.1 = /bin/sh
  /opt/tac/snmp/lbsessions
  enterprises.ucdavis.extTable.extEntry.extCommand.2 = /bin/sh
  /opt/tac/snmp/lbsessions.plx.total
  enterprises.ucdavis.extTable.extEntry.extCommand.3 = /bin/sh
  /opt/tac/snmp/lbsessions.plx.web-web1
  enterprises.ucdavis.extTable.extEntry.extCommand.4 = /bin/sh
  /opt/tac/snmp/lbsessions.plx.web-web2
  enterprises.ucdavis.extTable.extEntry.extResult.1 = 0
  enterprises.ucdavis.extTable.extEntry.extResult.2 = 0
  enterprises.ucdavis.extTable.extEntry.extResult.3 = 0
  enterprises.ucdavis.extTable.extEntry.extResult.4 = 0
  enterprises.ucdavis.extTable.extEntry.extOutput.1 = 292
  enterprises.ucdavis.extTable.extEntry.extOutput.2 = -1
  enterprises.ucdavis.extTable.extEntry.extOutput.3 = -1
  enterprises.ucdavis.extTable.extEntry.extOutput.4 = -1
  ```

  Here you see that the total amount of sessions of the load balancer serving about 8 customers is 292 currently and that customer plx has no connections so far.

- Write a 23.4 (MIB) for LVS stats.

23.3 MRTG and LVSGSP

Alexandre Cassen alexandre.cassen@canal-plus.com, the author of *keepalived* <http://www.linuxvirtualserver.org/~acassen> has produced a package, *LVSGSP* <http://www.

linuxvirtualserver.org/~acassen> that runs with *MRTG* <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/> to output LVS status information. Currently *active and inactive connections are plotted* <http://www.linuxvirtualserver.org/~acassen/lvsgsp-sample/> (html/png).

The LVSGSP package includes directions for installing and a sample mrtg.cfg file for monitoring one service. The mrtg.cfg file can be expanded to multiple services

```
WorkDir: /usr/local/mrtg
IconDir: /usr/local/mrtg/images/

# VS1 10.10.10.2:1358
Target[VS1]: `/usr/local/bin/LVSGSP 10.10.10.2 1358`
Directory[VS1]: LVS
MaxBytes[VS1]: 150
.
.

# VS2 10.10.10.2:8080
Target[VS2]: `/usr/local/bin/LVSGSP 10.10.10.2 8080`
Directory[VS2]: LVS
MaxBytes[VS2]: 150
.
.
```

A note from Alexandre

> Concerning the use of MRTG directly onto the director, we must take care of the computing CPU time monopolised by the MRTG graph generation. On a very overloaded director, the MRTG processing can degrade LVS performance.

23.4 MIB/SNMP

A MIB has been written for LVS by Romeo Benzoni rb@ssn.tp (Nov 2001). It's available as *code and documentation* <http://anakin.swiss-support.net/~romeo/lvs-snmp/>.

the latest (Mar 2002) is at http://anakin.swiss-support.net/ romeo/lvs-snmp/ucd-snmp-lvs-module-0.0.2.tar.bz2

24 Newer networking tools: Priority Routing

The standard network tools (eg ifconfig and route/netstat) aren't capable of setting up some of the features used in newer LVSs *e.g.* 13.10 (routing based on src_addr). For this we use iproute2, which allows routing based on almost any of the parameters of a packet (src, dest, proto, tos...). iproute2 is available at *iproute2-current.tar.gz* <ftp://ftp.inr.ac.ru/ip-routing/>. iproute2 implements similar functionality to cisco's IOS.

If there is only one possible route for packets, then ifconfig and route are just fine. If multiple routes exist then iproute2 is needed.

Presumably routing in Linux and the setup of LVS will move more toward using iproute2. The 10.1 (configure script) will use the iproute2 package to do some configuration if you have it installed.

iproute2 doesn't use ip_aliases (*e.g.* eth0:110) and just attaches all addresses to the machine. ip_tables is based on the same underlying code and also doesn't recognise ip_aliases. If you want to see the network as ip_tables sees it, you need the iproute2 tools. Presumably ip_aliases are in 2.4 for compatibility with 2.2 scripts, but they don't work well with iproute2.

iproute2 is not compatible with ifconfig/route/netstat. The entries added by the iproute2 tools are not seen by ifconfig/route etc and the output of ifconfig/route etc will be incorrect. You can't tell from looking at the output of ifconfig/route whether iproute2 commands have been run - you just have to know. The iproute2 tools correctly interpret the results of ifconfig/route commands and will give the correct state of the network.

Unfortunately the user interface to iproute2 is not easy.

- The documentation is not easy to read (although it was all Julian needed). Ratz suggested "Policy Routing Using Linux" by Matthew G. Marsh, Pub Sams 2001, ISBN 0-672-32052-5, to get you started (it helped me).

  Padraig Brady `padraig@antefactor.com` suggests *Linux Advanced Routing & Traffic Control HOWTO* `<http://lartc.org/HOWTO/cvs/2.4routing/output/2.4routing.html>`.

- The output from the commands is difficult to parse (see the comments in the 10.1 (configure script) for more details) - *i.e.* it's not machine readable. Ratz is developing a wrapper for iproute2 that will give machine readable output.

24.1 Priority Routing and ifconfig

Example:

> In a normally functioning LVS-DR, with routing setup by "route" the realservers will be sending packets with the following routing
>
> - src_addr=VIP dest_addr=0/0. dest=0/0 - route via default gw
> - src_addr=RIP dest_addr=RIP network. dest=RIP network - route to RIP network
>
> In LVS-DR a packet leaving the realserver can exit via the default gw or the director. In the standard setup, packets with dst_addr=RIPnetwork are put onto the local network and all other packets are sent to the default gw.
>
> If instead the routing is setup by "iproute2", packets with src_addr=VIP are sent to the default gw, while packets with src_addr=RIP are put onto the local network. The realservers will be sending packets with the following routing
>
> - src_addr=VIP dest_addr=0/0. src=VIP - route via default gw
> - src_addr=RIP dest_addr=RIP network. src=RIP - route to RIP network
>
> The result for a normal working LVS, will be the same (*i.e.* the LVS will still work). However with the standard setup, packets with scr_addr=RIP cannot get to the outside world (the director does not have a default route to 0/0, or else will not forward packets from RIP network). If a process needs this (*e.g.* the operator needs to telnet out or the realserver needs DNS), then those packets from the RIP can be NAT'ed out via the director. For security all packets from the VIP have to go out the default gw (including those to say the DIP, which will be dropped by rules on the default gw). This prevents spoofing.
>
> - src_addr=VIP dest_addr=RIP network. src=VIP - route via default gw, will be dropped
> - src_addr=RIP dest_addr=0/0. src=RIP - route to RIP network. If the director has the correct NAT rules, then these packets can pass to the outside world.

24.2 Various debugging techniques for routes

(with Julian)

(I needed this information to setup a one-net LVS-NAT LVS. However since it is about routing and not LVS specifically, maybe I should move it elsewhere.)

The routes added with the route command go into the kernel FIB (Forwarding information base) route table. The contents are displayed with the route (or netstat -a) command.

Following an icmp redirect, the route updates go into the kernel's route cache (route -C).

You can flush the route cache with

```
        echo 1 > /proc/sys/net/ipv4/route/flush
or
        ip route flush cache
```

Here's the route cache on the realserver before any packets are sent.

```
realserver:/etc/rc.d# route -C
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
realserver      director        director              0      1        0 eth0
director        realserver      realserver      il    0      0        9 lo
```

With icmp redirects enabled on the director, repeatedly running traceroute to the client shows the routes changing from 2 hops to 1 hop. This indicates that the realserver has received an icmp redirect packet telling it of a better route to the client.

```
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.932 ms  0.562 ms  0.503 ms
 2  client (192.168.1.254)  1.174 ms  0.597 ms  0.571 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.72 ms  0.581 ms  0.532 ms
 2  client (192.168.1.254)  0.845 ms  0.559 ms  0.5 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  client (192.168.1.254)  0.69 ms *  0.579 ms
```

Although the route command shows no change in the FIB, the route cache has changed. (The new route of interest is bracketted by >< signs.)

```
 realserver:/etc/rc.d# route -C
 Kernel IP routing cache
 Source          Destination     Gateway         Flags Metric Ref    Use Iface
 client          realserver      realserver      l     0      0        8 lo
 realserver      realserver      realserver      l     0      0     1038 lo
 realserver      director        director              0      1      138 eth0
>realserver      client          client                0      0        6 eth0<
 director        realserver      realserver      l     0      0        9 lo
 director        realserver      realserver      l     0      0      168 lo
```

Packets to the client now go directly to the client instead of via the director (which you don't want).

It takes about 10mins for the client's route cache to expire (experimental result). The timeouts may be in /proc/sys/net/ipv4/route/gc_*, but their location and values are well encrypted in the sources :) (some more info from Alexey at *LVS archives* <http://marc.theaimsgroup.com/?l=linux-kernel&m=91754108723334&w=2>)

Here's the route cache after 10mins.

```
realserver:/etc/rc.d# route -C
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
realserver      realserver      realserver      l     0      0     1049 lo
realserver      director        director              0      1      139 eth0
director        realserver      realserver      l     0      0        0 lo
director        realserver      realserver      l     0      0      236 lo
```

There are no routes to the client anymore. Checking with traceroute, shows that 2 hops are initially required to get to the client (i.e. the routing cache has reverted to using the director as the route to the client). After 2 iterations, icmp redirects route the packets directly to the client again.

```
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.908 ms  0.572 ms  0.537 ms
 2  client (192.168.1.254)  1.179 ms  0.6 ms  0.577 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.695 ms  0.552 ms  0.492 ms
 2  client (192.168.1.254)  0.804 ms  0.55 ms  0.502 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  client (192.168.1.254)  0.686 ms  0.533 ms *
```

If you now turn off icmp redirects on the director.

```
director:/etc/lvs# echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
director:/etc/lvs# echo 0 > /proc/sys/net/ipv4/conf/default/send_redirects
director:/etc/lvs# echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

Checking routes on the realserver -

```
realserver:/etc/lvs# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         director        0.0.0.0         UG        0 0          0 eth0
```

nothing has changed here.

Flush the kernel routing table and show the kernel routing table -

```
realserver:/etc/lvs# ip route flush cache
realserver:/etc/lvs# route -C
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
realserver      director        director              0      1        0 eth0
director        realserver      realserver      l     0      0        1 lo
```

There are no routes to the client.

Now when you send packet to the client, the route stays via the director needing 2 hops to get to the client. There are no one hop packets to the client.

```
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.951 ms  0.56 ms  0.491 ms
 2  client (192.168.1.254)  0.76 ms  0.599 ms  0.574 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.696 ms  0.562 ms  0.583 ms
 2  client (192.168.1.254)  0.62 ms  0.603 ms  0.576 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.692 ms *  0.599 ms
 2  client (192.168.1.254)  0.667 ms  0.603 ms  0.579 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.689 ms  0.558 ms  0.487 ms
 2  client (192.168.1.254)  0.61 ms  0.63 ms  0.567 ms
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.705 ms  0.563 ms  0.526 ms
 2  client (192.168.1.254)  0.611 ms  0.595 ms *
realserver:/etc/rc.d# traceroute client
traceroute to client (192.168.1.254), 30 hops max, 40 byte packets
 1  director (192.168.1.9)  0.706 ms  0.558 ms  0.535 ms
 2  client (192.168.1.254)  0.614 ms  0.593 ms  0.573 ms
```

The kernel route cache

```
 realserver:/etc/rc.d# route -C
 Kernel IP routing cache
 Source          Destination     Gateway         Flags Metric Ref    Use Iface
 client          realserver      realserver      l     0      0       17 lo
 realserver      realserver      realserver      l     0      0        2 lo
 realserver      director        director              0      1        0 eth0
>realserver      client          director              0      0       35 eth0<
 director        realserver      realserver      l     0      0       16 lo
 director        realserver      realserver      l     0      0       63 lo
```

shows the the only route to the client (labelled with ><) is via the director.

For send_redirects, what's the difference between all, default and eth0?

> Julian see the *LVS archives* <http://marc.theaimsgroup.com/?l=linux-virtual-server&m=97932487110806&w=2>
>
> When the kernel needs to check for one feature (send_redirects for example) it uses such calls: if (IN_DEV_TX_REDIRECTS(in_dev)) ... These macros are defined in /usr/src/linux/include/linux/inetdevice.h The macro returns a value using expression from
>
> `all/<var> and <dev>/<var>`
>
> So, these macros check for example for:

```
all/send_redirects || eth0/send_redirects
```

or

```
all/hidden && eth0/hidden
```

when you create eth0 for first time using ifconfig eth0 ... up default/send_redirects is copied to eth0/send_redirects from the kernel, internally. I.e. default/ contains the initial values the device inherits when it is created. This is the safest way a device to appear with correct

```
conf/<dev>/
```

values.

When we put value in

```
all/<var>
```

you can assume that we set the

```
<var>
```

When we put value in

```
all/<var>
```

you can assume that we set the

```
<var>
```

for all devices in this way:

```
                all/<var>       the macro returns:
for &&          0               0
for &&          1               the value from <dev>/<var>
for ||          0               the value from <dev>/<var>
for ||          1               1
```

This scheme allows the different devices to have different values for their vars. For example, if we set 0 to all/send_redirects, the 3th line applies to the values, i.e. the result from the macro is the real value in <dev>/send_redirects. If we set 1 to all/send_redirects according to the 4th line, the macro always returns 1 regardless of the <dev>/send_redirects

how to debug/understand TCP/IP packets?

Julian The RFC documents are your friends:
http://www.ietf.cnri.reston.va.us/rfc.html
The numbers you need:

```
793     TRANSMISSION CONTROL PROTOCOL
1122    Requirements for Internet Hosts -- Communication Layers
1812    Requirements for IP Version 4 Routers
826     An Ethernet Address Resolution Protocol
```

for tcpdump, see man tcpdump.

for Microsoft NT _server_

`Steve.Gonczi@networkengines.com` there is a uSoft supplied packet capture utility as well.

also -W. Richard Stevens: TCP-IP Illustrated, Vol 1, a good intro into packet layouts and protocol basics. (anything by Stevens is good - Joe).

Ivan Figueredo `idf@weewannabe.com` for windump - http://netgroup-serv.polito.it/windump/

24.3 checking source routed packets

Packets leaving a LVS-DR realserver can have src_addr=VIP or src_addr=RIP. If the default gw is different for each packet, it would be nice to have a command line testing tool like ping or traceroute to test the route. The normal tools will create packets with src_addr=RIP and you won't be able to test the packets with src_addr=VIP.

Roberto Nibali `ratz@tac.ch` 22 May 2001

maybe hping can help you *development* `<http://www.kyuzz.org/antirez/hping3.html>`, *old hping2, but working* `<http://www.kyuzz.org/antirez/hping.html>`.

> Joe Ah, the file hping2.8 is the man page ie {hping2}.8 - I thought it was v2.8 of hping.

How about:

```
ip route get $IP?
```

> didn't know about "get". yes that works. It's like a -C with iptables. I'd still like to send a packet and see where it goes rather than getting an answer about where it is expected to go.

Julian

Not possible with src interface "lo" but possible with source address configured in "lo". Oh yes, "source interface" for some tools means "get one address from this iface and use it". In most of the cases these tools don't do the Right Thing.

from iproute2

```
$ ping -I src dst
```

or

```
arping -I if -s src dst
```

24.4 handling arp problem with iproute2

see *Julian's notes and patches to handle the arp problem with iproute2* `<http://www.linuxvirtualserver.org/~julian/##noarp>` (this is somewhat developemental).

24.5 ip commands you mightn't know about

24.5.1 ip route get

from Julian

This will look at the routing tables and tell you the route to xxx.xxx.xxx.xxx

```
ip route get xxx.xxx.xxx.xxx
```

24.5.2 ip route append

If you already have a route from A to B, and want to add another, you can't, you have to *append* the extra route.

dynnema dynnema@yahoo.com Mar 22 2002
Lets say I got one RS and two NAT DIRs.

```
RS:
RIP1:   192.168.1.2/24 dev eth0
RIP2    192.168.2.2/24 dev eth0:10

DIR1:
VIP:    x.x.x.69        eth0:110
DIP     192.168.1.1

DIR2:
VIP:    x.x.x.70        eth0:110
DIP     192.168.2.1
```

I add the first route

```
ip route add src 192.168.1.2 via 192.168.1.1
```

but then I can't add the second route:

```
ip route add src 192.168.2.2 via 192.168.2.1:
 "RTNETLINK answers: File exists"
```

Careful reading of IProute mailing list was very useful. It should be

```
ip route append src 192.168.2.2 via 192.168.2.1
```

25 Misc/FAQ/Wisdom from the mailing list

These topics were too short or not central enough to LVS operation to have their own section.

25.1 Multiple VIPs

Multiple VIPs (and their associated services) can co-exist independantly on an LVS. On the director, add the extra IPs to a device facing the internet. On the realservers, for LVS-DR||VS-Tun, add the VIPs to a device and setup services listening to the ports. On the realservers, for LVS-NAT, add the extra services to the RIP.

Tao Zhao 6 Nov 2001 what if I need multiple VIPs on the realserver?

Julian Anastasov ja@ssi.bg 06 Nov 2001

```
for i in 180 182 182
do
        ip addr add X.Y.Z.$i dev dummy0
done
```

There is also an example for 22.1.2 (setting up multiple VIPs on HA).

25.2 Who is connecting to my LVS?

On the realservers you can look with 'netstatn -an'. With LVS, the director also has information.

> malalon@poczta.onet.pl 18 Oct 2001 How do I know who is connecting to my LVS?

Julian

- Linux 2.2: netstat -Mn (or /proc/net/ip_masquerade)
- Linux 2.4: ipvsadm -Lcn (or /proc/net/ip_vs_conn)

25.3 Limiting number of clients connecting to LVS

Milind Patil mpatil@iqs.co.in> 24 Sep 2001

I want to limit number of users accessing the LVS services at any given time. How can I do it.

> Julian
>
> - for non-NAT cluster (maybe stupid but interesting)
>   May be an array from policers, for example, 1024 policers or an user-defined value, power of 2. Each client hits one of the policers based on their IP/Port. This is mostly a job for QoS ingress, even the distributed attack but may be something can be done for LVS? May be we better to develop a QoS Ingress module? The key could be derived from CIP and CPORT, may be something similar to SFQ but without queueing. It can be implemented may be as a patch to the normal policer but with one argument: the real number of policers. Then this extended policer can look into the TCP/UDP packets to redirect each packet to one of the real policers.
> - for NAT only Run SFQ qdisc on your external interface(s). It seems this is not a solution for DR method. Of course, one can run SFQ on its uplink router.
> - Linux 2.4 only iptables has support to limit the traffic but I'm not sure whether it is useful for your requirements. I assume you want to set limit to each one of these 1024 aggregated flows.

Wenzhuo Zhang

Is anybody actually using the ingress policer for anti-DoS? I tried it several days ago using the script in the iproute2 package: iproute2/examples/SYN-DoS.rate.limit. I've tested it against different 2.2 kernels (2.2.19-7.0.8(redhat kernel), 2.2.19, 2.2.20preX, with all QoS related functions either compiled into the kernel or as modules) and different versions of iproute2. In all cases, tc fails to install the ingress qdisc policer:

```
    root@panda:~# tc qdisc add dev eth0 handle ffff: ingress
    RTNETLINK answers: No such file or directory
    root@panda:~# /tmp/tc qdisc add dev eth0 handle ffff: ingress
    RTNETLINK answers: No such file or directory
```

Julian For 2.2, you need the ds-8 package, at *Package for Differentiated Services on Linux* <http://diffserv.sourceforge.net/>. Compile tc by setting TC_CONFIG_DIFFSERV=y in Config. The right command is:

```
tc qdisc add dev eth0 ingress
```

Ratz
The 2.2.x version is not supported anymore. The advanced routing documentation says to only use 2.4.

For 2.4 ingress is in the kernel but it is still unusable for more than one device (look in linux-netdev for reference).

25.4 Setting up an LVS with inetd

from Ratz ratz@tac.ch

We're going to set up a LVS cluster from scratch. you need

- 4 machines (2 realserver, 1 load balancer, 1 client) wired like described in various sketches throughout this howto.
- fun and some spare time (actually quite some if it doesn't work out the first time like described)

The goal is to set up an own loadbalanced tcp application. The application will consist of a own written shell script being invoked by inetd. As you might have guessed, security is very low priority, you should get the idea behind this. Of course I should take xinetd and of course I should use a tcpwrapper and maybe even SecurID authentication but here the goal is to understand the fundamental design principals of a LVS cluster and its deploy. All instructions will be done as root.

Setting up the realserver

```
Edit /etc/inetd.conf and add following line:
lvs-test        stream  tcp     nowait  root    /usr/bin/lvs-info       lvs-info

Edit /etc/services and add following line:
lvs-test        31337/tcp              # supersecure lvs-test port
```

Now you need to get inetd running. This is different for every Unix. So please have a look at it yourself. You verify if it's running with 'ps ax|grep [i]netd' And to verify if it really runs this port you do a 'netstat -an|grep LISTEN' and if there is a line:

```
tcp        0      0 0.0.0.0:31337           0.0.0.0:*               LISTEN
```

you're one step closer to the truth. Now we have to supply the script that will be called if you connect to realserver# port 31337. So simply do this on your command line (copy 'n' paste):

```
cat > /usr/bin/lvs-info << EOF && chmod 755 /usr/bin/lvs-info
#!/bin/sh

echo "This is a test of machine `ifconfig -a | grep HWaddr | awk '{print $1}'`"
echo
EOF
```

Now you can test if it really works with telnet or netcat:

```
telnet localhost 31337
netcat localhost 31337
```

This should spill out something like:

```
hog:/ # netcat localhost 31337
This is a test of machine 192.168.1.11

hog:/ #
```

If it worked, do the same procedure to set up the second realserver. Now we're ready to set up the load balancer. These are the required commands to set it up for our example:

```
ipvsadm -A -t 192.168.1.100:31337 -s wrr
ipvsadm -a -t 192.168.1.100:31337 -r 192.168.1.11 -g -w 1
ipvsadm -a -t 192.168.1.100:31337 -r 192.168.1.12 -g -w 1
```

Check it with ipvsadm -L -n:

```
hog:~ # ipvsadm -L -n
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.100:31337 wrr
  -> 192.168.1.12:31337          Route   1      0          0
  -> 192.168.1.11:31337          Route   1      0          0
hog:~ #
```

Now if you connect from outside with the client node to the VIP=192.168.1.100 you should get to one of the two realserver (presumably to .12) Reconnect to the VIP again an you should get to the other realserver. If so, be happy, if not go back, check netstat -an, ifconfig -a, arp-problem, routing tables and so on ...

25.5 How to bring down a realserver for maintenance (eg swap disks)

I want to use virtual server functionality to allow switching over from one pool of server processes to another without an interruption in service to clients.

Michael Sparks `sparks@mcc.ac.uk` current realservers : A,B,C servers to swap into the system instead D,E,F

- Add servers D,E,F into the system all with fairly high weights (perhaps ramping the weights up slowly so as not to hit them too hard:-)
- Change the weights of servers A,B,C to 0.
- All new traffic should now go to D,E,F
- When the number of connections through A,B,C reaches 0, remove them from the service. This can take time I know but...

from Joe

A planned feature for ipvsadm will be to give a realserver a weight of 0 (now implemented). This realserver will not be sent any new connections and will continue serving its current connections till they close. You may have to wait a while if a user is downloading a 40M file from the realserver.

25.6 Howto turn your single node ftp/http server into an LVS without taking it off-line

eg if you want to test LVS on your BIG Sunserver and how to restore an LVS to a single node server again.

```
current ftp server:        standalone  A

planned LVS (using LVS-DR): realserver A
                            director   Z
```

Setup the LVS in the normal way with the director's VIP being a new IP for the network. The IP of the standalone server will now also be the IP for the realserver. You can access the realserver via the VIP while the outside users continue to connect to the original IP of A. When you are happy that the VIP gives the right service, change the DNS IP of your ftp site to the VIP. Over the next 24hrs as the new DNS information is propagated to the outside world, users will change over to the VIP to access the server.

To expand the number of servers (to A, B,...), add another server with duplicated files, add an extra entry into the director's tables with ipvsadm.

To restore - in your DNS, change the IP for the service to the realserver IP. When no-one is accessing the VIP anymore, unplug the director.

25.7 shutdown of LVS

You can't shutdown an LVS. However you can stop it forwarding by clearing the ipvsadm table (ipvsadm -C), then allow all connections to expire (check the active connections with ipvsadm) and then remove the ipvs modules (rmmod). Since ip_vs.o requires ip_vs_rr.o etc, you'll have to remove ip_vs_rr.o first.

> Do you know how to shutdown LVS? I tried rmmod but it keeps saying that the device is busy.

Kjetil Torgrim Homme `kjetilho@linpro.no` 18 Aug 2001

Run ipvsadm -C. You also need to remove the module(s) for the balancing algorithm(s) before rmmod ip_vs. Run lsmod to see which modules these are.

Roy Walker `Roy.Walker@GEZWM.com` 18 Mar 2002 could not cleanly shutdown his director (LVS 1.0, 2.4.18) which hung at "Send TERM signal". The suggested cure, was to bring down the LVS first (we haven't heard back if it works).

25.8 Other projects like LVS - Beowulf

The difference between a beowulf and an LVS:

The Beowulf project has to do with processor clustering over a network – parallel computing... Basically putting 64 nodes up and running that all are a part of a collective of resources. Like SMP – but between a whole bunch of machines with a fast ethernet as a backplane.

LVS, however, is about load-balancing on a network. Someone puts up a load balancer in front of a cluster of servers. Each one of those servers is independent and knows nothing about the rest of the servers in the farm. All requests for services go to the load balancer first. That load balancer then distributes requests to each server. Those servers respond as if the request came straight to them in the first place. So – with the more servers one adds – the less load goes to each server.

A person might go to a web site that is load balanced, and their requests would be balanced between four different machines. (Or perhaps all of their requests would go to one machine, and the next person's request would go to another machine)

However, a person who used a Beowulf system would actually be using one processing collaborative that was made up of multiple computers...

I know that's not the best explanation of each, and I apologize for that, but I hope it at least starts to make things a little clearer. Both projects could be expanded on to a great extent, but that might just confuse things farther.

(Joe) -

both use several (or a lot of) nodes.

A beowulf is a collection of nodes working on a single computation. The computation is broken into small pieces and passed to a node, which replies with the result. Eventually the whole computation is done. THe beowulf usually has a single user and the computations can run for weeks.

An LVS is a group of machines offering a service to a client. A dispatcher connects the client to a particular server for the request. When the request is completed, the dispatcher removes the connection between the client and server. The next request from the same client may go to a different server but the client cannot tell which server it has connected to. The connection between client and server may only be seconds long

from a posting to the beowulf mailing list by Alan Heirich -

Thomas Sterling and Donald Becker made "Beowulf" a registered service mark with specific requirements for use:

```
-- Beowulf is a cluster
-- the cluster runs Linux
-- the O/S and driver software are open source
-- the CPU is multiple sourced (currently, Intel and Alpha)
```

I assume they did this to prevent profit-hungry vendors from abusing this term; can't you just imagine Micro$oft pushing a "Beowulf" NT-cluster?

(Joe - I looked up the Registered Service Marks on the internet and Beowulf is not one of them.)

(Wensong) Beowulf is for parallel computing, Linux Virtual Server is for scalable network services.

They are quite different now. However, I think they may be unified under "single system image" some day. In the "single system image", every node can see a single system image (the same memory space, the same process space, the same external storage), and the processes/threads can be transparently migrated to other nodes in order to achieve load balance in the cluster. All the processes are checkpointed, they can be restarted in the node or the others if they fails, full fault tolerant can be made here. It will be easy for programmers to code because of single space, they don't need to statically partition jobs to different sites and let them communicate through PVM or MPI. They just need identify the parallelism of his scientific application, and fork the processes or generate threads, because processes/threads will be automatically load balanced on different nodes. For network services, the service daemons just need to fork the processes or generates threads, it is quite simple. I think it needs lots of investigation in how to implement these mechanisms and make the overhead as low as possible.

What Linux Virtual Server has done is very simple, Single IP Address, in which parallel services on different nodes is appeared as a virtual service on a single IP address. The different nodes have their own space, it is far from "single system image". It means that we have a long way to run. :)

25.9 Projects like LVS - Eddie

Eddie http://www.eddieware.org

(Jacek Kujawa `blady@cnt.pl`) Eddie is a load balancing software, using NAT (only NAT), for webservers, written in language erlang. Eddie include intelligent HTTP gateway and Enhanced DNS.

(Joe) Erlang is a language for writing distrubuted applications.

25.10 Any recommendations for a NIC?

Martin Seigert at Simon Fraser U posted *Benchmarks for various NICS* `<http://www.sfu.ca/acs/cluster/nic-test.html>` to the beowulf mailing list. The conclusion was that for fast CPUs (*i.e.*600MHz, which can saturate 100Mbps ethernet) the 3c95x and tulip cards were equivelent. For slower CPUs (166MHz) which cannot saturate 100Mbs ethernet, the on-board processing on the 3Com hards allowed marginally better throughput.

If you are going into production, you should test that your NIC works well with your hardware. Give it a good exercising with a netpipe test (see the *performance page* `<http://www.linuxvirtualserver.org/Joseph.Mack/performance/single_realserver_performance.html>`).

I use Netgear FA310TX (tulip), and eepro100 for single port NICs. The related FA311 card seems to be Linux incompatible (postings to the beowulf mailing list), currently (Jul 2001) requiring a driver from Netgear (this was the original situation with the FA310 too). I also use a quad DLink DFE-570TX (tulip) on the director. I'm happy with all of them.

The eepro100 has problems as Intel seems to change the hardware without notice and the linux driver writers have trouble handling all the versions of hardware. One kernel (2.2.18?) didn't work with the eepro100. and new kernels seem to have problems occassionally. I bought all of my eepro100's at once and presumably they are identical. There have been a relatively large number of posting of people with 25.11 (eepro100 problems) on the LVS mailing list. You should expect continuing problems with this card, which will be solved by kernel patches.

25.11 NIC problems - eepro100

25.11.1 counter overflows

(This is from 1999 I think)

linux with an eepro100 can't pass more than 2ˆ31-1 packets. This may not still be a problem.

> Jerry Glomph Black `black@real.com` Subject: 2-billion-packet bug? I've seen several 2.2.12/2.2.13 machines lose their network connections after a long period of fine operation. Tonight our main LVS box fell off the net. I visited the box, it had not crashed at all. However, it was not communicating via its (Intel eepro100) ethernet port.
>
> The evil evidence:

```
eth0      Link encap:Ethernet  HWaddr 00:90:27:50:A8:DE
          inet addr:172.16.0.20  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
RX packets:15 errors:288850 dropped:0 overruns:0 frame:0
TX packets:2147483647 errors:1 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
Interrupt:10 Base address:0xd000
```

Check out the TX packets number! That's 2^31-1. Prior to the rollover, In-and-out packets were roughly equal. I think this has happened to non-LVS systems as well, on 2.2 kernels. ifconfigging eth0 down-and-up did nothing. A reboot (ugh) was necessary.

It's still happening 2yrs later. This time the counter stops, but the network is still functional.

Hendrik Thiel `thiel@falkag.de` 20 Nov 2001 using lvs with eepro100 cards (kernel 2.2.17) and encountered a TX packets value stopping at 2147483647 (2^32-1) thats what ifconfig tells...the system still runs fine ...

it seems to be a ifconfig Bug. Check out the TX packets number! That's 2^31-1.

```
eth0 Link encap:Ethernet HWaddr 00:90:27:50:A8:DE
inet addr:172.16.0.20 Bcast:172.16.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:15 errors:288850 dropped:0 overruns:0 frame:0
TX packets:2147483647 errors:1 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
Interrupt:10 Base address:0xd000
```

Simon A. Boggis Hmmm, I have a couple of eepro100-based linux routers - the one thats been up the longest is working fine (167 days, kernel 2.2.9) but the counters are jammed - for example, 'ifconfig eth0' gives:

```
eth0 Link encap:Ethernet HWaddr 00:90:27:2A:55:48
inet addr:138.37.88.251 Bcast:138.37.88.255 Mask:255.255.255.0
IPX/Ethernet 802.2 addr:8A255800:0090272A5548
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2147483647 errors:41 dropped:0 overruns:1 frame:0
TX packets:2147483647 errors:13 dropped:0 overruns:715 carrier:0
Collisions:0
Interrupt:15 Base address:0xa000
```

BUT /proc/net/dev reports something more believable:

```
hammer:/# cat /proc/net/dev
Inter-| Receive | Transmit
face |bytes packets errs drop fifo frame compressed multicast|bytes packets errs drop fifo
eth0:2754574912 2177325200 41 0 1 0 0 0 2384782514 3474415357 13 0 715 0 0 0
```

Thats RX packets: 2177325200 and TX packets: 3474415357 compared to : 2147483647 from ifconfig eth0

25.11.2 new drivers

Andrey Nekrasov After I changed to kernel 2.4.x with "arp hidden patch"

```
eepro100: wait_for_cmd_done timeout!
```

I haven't had any problems before with NIC Intel EEPRO/100.

Julian 4 Feb 2002

This problem happens not only with LVS. Search the web or linux-kernel:

http://marc.theaimsgroup.com/?t=100444264400003&r=1&w=2

25.12 NIC problems - tulip

(Joe, Nov 2001 I don't know if this is still a problem, we haven't heard any more about it and haven't had any other tulip problems, unlike the 25.11 (eepro100).)

> John Connett `jrc@art-render.com` 05 May 1999 Any suggestions as to how to narrow it down? I have an Intel EtherExpress PRO 100+ and a 3COM 3c905B which I could try instead of the KNE 100TX to see if that makes a difference.
>
> A tiny light at the end of the tunnel! Just tried an Intel EtherExpress PRO 100+ and it works! Unfortunately, the hardware is fixed for the application I am working on and has to use a single Kingston KNE 100TX NIC ...
>
> Some more information. The LocalNode problem has been observed with both the old style (21140-AF) and the new style (21143-PD) of Kingston KNE 100TX NIC. This suggests that there is a good chance that it will be seen with other "tulip" based NICs. It has been observed with both the "v0.90 10/20/98" and the "v0.91 4/14/99" versions of tulip.c.
>
> I have upgraded to vs-0.9 and the behaviour remains the same: the EtherExpress PRO 100+ works; the Kingston KNE 100TX doesn't work.
>
> It is somewhat surprising that the choice of NIC should have this impact on the LocalNode behaviour but work successfully on connections to slave servers.
>
> Any suggestions as to how I can identify the feature (or bug) in the tulip driver would be gratefully received. If it is a bug I will raise it on the tulip mailing list.

25.13 Recommendations for a redundant file system, RAID

> Shain Miley 4 Jun 2001 any recommendations for Level 5 SCSI RAID?

Matthew S. Crocker `matthew@crocker.com` 04 Jun 2001

I have had very good luck with Mylex. We use the DAC960 which is a bit old now but if the newer stuff works as well as what I have I would highly recommend it. You might also want to think about putting your data on a NAS and seperate your CPU from your harddrives

Don Hinshaw `dwh@openrecording.com` 04 Jun 2001

Mylex work well. I use *ICP-Vortex* `<http://www.icp-vortex.com/index_e.html>` which are supported by the Linux kernel. I've also had good luck with

25.14 Thundering herd problem, when down machine(s) come on line

(now handled by code added to the scheduler)

From: Christopher Seawood `cls@aureate.com`

LVS seems to work great until a server goes down (this is where mon comes in). Here's a couple of things to keep in mind. If you're using the Weighted Round-Robin scheduler, then LVS will still attempt to hit the server once it goes down. If you're using the Least Connections scheduler, then all new connections will be directed to the down server because it has 0 connections. You'd think using mon would fix these problem but not in all cases.

Adding mon to the LC setup didn't help matters much. I took one of three servers out of the loop and waited for mon to drop the entry. That worked great. When I started the server back up, mon added the entry. During that time, the 2 running servers had gathered about 1000 connections apiece. When the third server came back up, it immediately received all of the new connections. It kept receiving all of the connections until it had an equal number of connections with the other servers (which by this time...a minute or so later...had fallen to 700). By this time, the 3rd server had been restarted after due to triggering a high load sensor also monitoring the machine (a necessary evil or so I'm told). At this point, I dropped back to using WRR as I could envision the cycle repeating itself indefinitely.

25.15 on the need for extended testing

(this must have been solved, no-one is complaining about memory leaks now :-)

> Jerry Glomph Black `black@real.com` We have successfully used 2.0.36-vs (direct routing method), but it does fail at extremely high loads. Seems like a cumulative effect, after about a billion or so packets forwarded. Some kind of kernel memory leak, I'd guess.

25.16 loopback on Solaris

The thing I have found out is that on Solaris 2.6, and probably other versions of Solaris, you have to to some magic to get the loopback alias setup. You must run the following commands one at a time:

```
ifconfig lo0:1 <VIP>
ifconfig lo0:1 <VIP> <VIP>
ifconfig lo0:1 netmask 255.255.255.255
ifconfig lo0:1 up
```

Which works well and is actually a pointopoint link like ppp which must be the way Solaris defines aliases to the lo interface. It will not let you do this all at once, just each step at a time or you have to start over from scratch on the interface.

```
Chris Kennedy
I-Land Internet Services
<tt/ckennedy@iland.net/
```

25.17 Having one director handling multiple LVS sites

> Keith Rowland wrote: Can I use Virtual Server to host multiple domains on the cluster? Can VS be setup to respond to multiple 10-20 different IP addresses and use the clusters to reposnd to any one of them with the proper web directory.

James CE Johnson `jjohnson@mobsec.com`

If I understand the question correctly, then the answer is yes :-) I have one system that has two IP addresses and responds to two names:

```
  foo.mydomain.com  A.B.C.foo  eth1
  bar.mydomain.com  A.B.C.bar  eth1:0
```

On that system (kernel 2.0.36 BTW) I have LVS setup as:

```
ippfvsadm -A -t A.B.C.foo:80 -R 192.168.42.50:80
ippfvsadm -A -t A.B.C.bar:80 -R 192.168.42.100:80
```

To make matters even more confusing, 192.168.42.(50|100) are actually one system where eth0 is 192.168.42.100 and eth0:0 is 192.168.42.50. We'll call that 'node'.

Apache on 'node' is setup to serve foo.mydomain.com on ...100 and bar.mydomain.com on ...50.

It took me a while to sort it out but it all works quite nicely. I can easily move bar.mydomain.com to another node within the cluster by simply changing the ippfvsadm setup on the externally addressable node.

25.18 Running multiple directors (each with their own IP)

On a normal LVS (one director, multiple realservers being failed-over with mon), the single director is a SPOF (single point of failure). Director failure can be handled (in principle) with heartbeat, but no-one is doing this yet. In the meantime, you can have two directors each with their own VIP known to the users and set them up to talk to the same set of realservers. (You can have two VIP's on one director box too). (The 10.1 (configure.pl) script doesn't handle this yet.)

> Michael Sparks `michael.sparks@mcc.ac.uk` Also has anyone tried this using 2 or more masters - each master with it's own IP? (*) From what I can see theoretically all you should have to do is have one master on IP X, tunneled to clients who recieve stuff via tunl0, and another master on IP Y, tunneled to clients on tunl1 - except when I just tried doing that I can't get the kernel to accept the concept of a tunl1... Is this a limitation of the IPIP module ???

Stephen D. WIlliams `sdw@lig.net`

Do aliasing. I don't see a need for tunl1. In fact, I just throw a dummy address on tunl0 and do everything with tunl0:0, etc.

We plan to run at least two LinuxDirector/DR systems with failover for moving the two (or more) public IP's between the systems. We also use aliased, movable IP's for the real server addresses so that they can failover also.

25.19 Running clients (eg telnet) on realservers

There are two types of clients on realservers from the point of view of LVS.

- Clients which have src_addr=RIP (eg telnet run from the command line). These are simpler to handle.
- Clients which need to have src_addr=VIP (but call from RIP). These are usually call-backs from the LVS'ed service to a demon on the LVS client. Handling these is somewhat problematic. The instances that we know about of this.
  - 17 (authd)
  - 11.26 (rshd)
  - 11.5 (passive ftp)

Both types of clients require the same understanding of LVS, but because the first case is simple, it is discussed here. The second case has all sorts of ramifications for LVS and for that reason is discussed in the 17 (section on authd).

You might have valid reasons for running clients on realservers, e.g. so that the sysadmin could telnet to a remote site. The way to allow clients on the realservers to connect to outside servers is to configure these requests so that they are independant of the LVS setup (you do have to use the network and default gw set by the LVS). The solution is to NAT the client requests.

25.19.1 client requests from realservers in a LVS-NAT LVS

This is simple

- the director is already the default gw for the realserver (a requirement for NAT).
- each realserver is replying to LVS packets with its RIP, which is unique (there is no VIP on the realservers with LVS-NAT). NAT'ed client requests will return to the correct realserver.

Here's the command to run on a 2.2.x director to allow realserver1 to telnet to the outside world.

```
director:# ipchains -A forward -p tcp -j MASQ -s realserver1 telnet -d 0.0.0.0/0
```

You may have to turn off icmp redirects, if you have a 12.12 (one network LVS-NAT).

```
director: #echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
director: #echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

After running this command you can telnet from the realservers. You can do this even if telnet is an LVS'ed service, since the telnet client and demon operate independantly of each other. You can use NAT the rshd and identd clients in the same way (replace telnet with rsh/identd and clients on the realserver can connect to their demons on outside machines).

25.19.2 client requests from realservers in LVS-DR or LVS-Tun LVS's

In general this has not been solved. Calls initiated by the identd client on a realserver will come from the VIP, not the RIP. Some hare-brained schemes have been tried but did not work (NAT'ing out the request from the VIP, so that it emerges from the realserver with src_addr=RIP and then NAT'ing the packet again on the director, so it emerges with src_addr=VIP).

There are specific solutions

In LVS-DR/VS-Tun, if the client and RIP are on the same network. Usually the RIP's on LVS-DR realservers are private addresses. However if the LVS clients and the LVS are all local and on the same network, this will work.

Clients not associated with the LVS'ed services (ie telnet even if telnetd is LVSed, but not authd or rshd) can still be NAT'ed out, since the connect request will come from the RIP and not the VIP. Since the default gw for the realserver in LVS-DR is not the director, you can handle this 2 ways

- do the NAT'ing on the default gw box (you may not have access to this machine)
- make the director the default gw for packets from the RIP (see setting up 13.10 (NAT for clients on LVS-DR)).

25.20 ICMP

> Laurent Lefoll Laurent.Lefoll@mobileway.com 14 Feb 2001 what is the usefulness of the ICMP packets that are sent when new packets arrives for a TCP connection that timed out for in LVS box ? I understand obviously for UDP but I don't see their role for a TCP connection...

Julian

I assume your question is about the reply after ip_vs_lookup_real_service.

It is used to remove the open request in SYN_RECV state in the real server. LVS replies for more states and may be some OSes report them as soft errors (Linux), others can report them as hard errors, who knows.

> it's about ICMP packets from a LVS-NAT director to the client. For example, a client accesses a TCP virtual service and then stops sending data for a long time, enough for the LVS entry to expire. When the client try to send new data over this same TCP connection the LVS box sends ICMP (port unreachable) packets to the client. For a TCP connection how do these ICMP packets "influence" the client ? It will stop sending packets to this expired (for the LVS box...) TCP connection only after its own timeouts, doesn't it ?

By default TCP replies RST to the client when there is no existing socket. LVS does not keep info for already expired connections and so we can only reply with an ICMP rather than sending a TCP RST. (If we implement TCP RST replies, we could reply TCP RST instead of ICMP).

What does the client do with this ICMP packet? By default, the application does not listen for ICMP errors and they are reported as soft errors after a TCP timeout and according to the TCP state. Linux at least allows the application to listen for such ICMP replies. The application can register for these ICMP errors and detect them immediately as they are received by the socket. It is not clear whether it is a good idea to accept such information from untrusted sources. ICMP errors are reported immediately for some TCP (SYN) states.

25.21 tcpdump

> Joseph Mack, 16 Mar 2001 I'm looking at packets after they've been accepted by TP and I'm using (among other things) tcpdump.
>
> Where in the netfilter chain does tcpdump look at incoming and outgoing packets? When they are put on/received from the wire? After the INPUT, before the OUTPUT chain...?

Julian

Before/after any netfilter chains. Such programs hook at packet level before/after the IP stack just before/after the packet is received/must be sent from/by the device. They work for other protocols. tcpdump is a packet receiver just like the IP stack is in the network stack.

25.22 Bringing down aliased devices

(without bringing them all down)

Problem: if down/delete an aliased device (eg eth0:1) you also bring down the other eth0 devices. This means that you can't bring down an alias remotely as you loose your connection (eth0) to that machine. You then have to go the console of the remote machine to fix it by rmmod'ing the device driver for the device and bring it up again.

The 10.1 (configure script) handles this for you and will exit (with instructions on what to do next) if it finds that an aliased device needs to be removed by rmmod'ing the module for the NIC.

(I'm not sure that all of the following is accurate, please test yourself first).

(Stephen D. WIlliams sdw@lig.net) whenever you want to down/delete an alias, first set its netmask to 255.255.255.255. This avoids also automatically downing aliases that are on the same netmask and are considered 'secondaries' by the kernel.

(Joe) To bring up an aliased device

$ifconfig eth0:1 192.168.1.10 netmask 255.255.255.0

to bring eth0:1 down without taking out eth0, you do it in 2 steps, first change the netmask

$ifconfig eth0:1 192.168.1.10 netmask 255.255.255.255

then down it

$ifconfig eth0:1 192.168.1.10 netmask 255.255.255.255 down

then eth0 device should be unaffected, but the eth0:1 device will be gone.

This works on one of my machines but not on another (both with 2.2.13 kernels). I will have to look into this. Here's the output from the machine for which this procedure doesn't work.

Examples: Starting setup. The realserver's regular IP/24 on eth0, the VIP/32 on eth0:1 and another IP/24 for illustration on eth0:2. Machine is SMP 2.2.13 net-tools 1.49

```
chuck:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:90:27:71:46:B1
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:6071219 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6317319 errors:0 dropped:0 overruns:4 carrier:0
          collisions:757453 txqueuelen:100
          Interrupt:18 Base address:0x6000

eth0:1    Link encap:Ethernet  HWaddr 00:90:27:71:46:B1
          inet addr:192.168.1.110  Bcast:192.168.1.110  Mask:255.255.255.255
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          Interrupt:18 Base address:0x6000

eth0:2    Link encap:Ethernet  HWaddr 00:90:27:71:46:B1
          inet addr:192.168.1.240  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          Interrupt:18 Base address:0x6000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:299 errors:0 dropped:0 overruns:0 frame:0
          TX packets:299 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

chuck:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.110   0.0.0.0         255.255.255.255 UH        0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

```
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0

Deleting eth0:1 with netmask /32

chuck:~# ifconfig eth0:1 192.168.1.110 netmask 255.255.255.255 down
chuck:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:90:27:71:46:B1
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:6071230 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6317335 errors:0 dropped:0 overruns:4 carrier:0
          collisions:757453 txqueuelen:100
          Interrupt:18 Base address:0x6000

eth0:2    Link encap:Ethernet  HWaddr 00:90:27:71:46:B1
          inet addr:192.168.1.240  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          Interrupt:18 Base address:0x6000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:299 errors:0 dropped:0 overruns:0 frame:0
          TX packets:299 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0


If you do the same thing with eth0:2 with the /24 netmask

chuck:~# ifconfig eth0:2 192.168.1.240 netmask 255.255.255.0 down
chuck:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:90:27:71:46:B1
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:6071237 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6317343 errors:0 dropped:0 overruns:4 carrier:0
          collisions:757453 txqueuelen:100
          Interrupt:18 Base address:0x6000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:299 errors:0 dropped:0 overruns:0 frame:0
          TX packets:299 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

tunl0     Link encap:IPIP Tunnel  HWaddr
          unspec addr:[NONE SET]  Mask:[NONE SET]
          NOARP  MTU:1480  Metric:1
```

```
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
```

25.23 Malicious attacks (SYN floods)

LVS has been tested with a 100Mbit/sec syn-flooding attack by Alan Cox and Wensong.

Each connection requires 128 bytes. A machine with 128M of free memory could hold 1M concurrent connections. An average connection lasts 300secs. Connections which just receive the syn packet are expired in 30secs (starting ipvs 0.8). An attacker would have to initiate 3k connections/sec (600Mbps) to maintain the memory at the 128M mark and would require several T3 lines to keep up the attack.

25.24 Does SMP help on the director?

only one CPU can be in the kernel with 2.2. Since LVS is all kernel code, there is no benefit to LVS by using SMP with 2.2.x. Kernel 2.[3-4] can use multiple CPUs. While standard (300MHz pentium) directors can easily handle 100Mbps networks, they cannot handle an LVS at Gbps speeds. Either SMP directors with 2.4.x kernels or multiple directors (each with a separate VIP all pointing to the same realservers) are needed.

Since LVS-NAT requires computation on the director (to rewrite the packets) not needed for LVS-DR and LVS-Tun, SMP would help throughput.

> Joe If you're using LVS-NAT then you'll need a machine that can handle the full bandwidth of the expected connections. If this is T1, you won't need much of a machine. If it's 100Mbps you'll need more (I can saturate 100Mbps with a 75MHz machine). If you're running LVS-DR or LVS-Tun you'll need less horse power. Since most LVS is I/O I would suspect that SMP won't get you much. However if the director is doing other things too, then SMP might be useful

Julian Anastasov `uli@linux.tu-varna.acad.bg`

Yep, LVS in 2.2 can't use both CPUs. This is not a LVS limitation. It is already solved in the latest 2.3 kernels: softnet. If you are using the director as real server too, SMP is recommended.

> Pat O'Rourke `orourke@mclinux.com` 03 Jan 2000 In our *performance tests* `<http://www.linuxvirtualserver.org/performance/lvs.ps.gz>` we've been seeing an SMP director perform significantly worse than a uni-processor one (using the same hardware - only difference was booting an SMP kernel or uni-processor).
>
> We've been using a 2.2.17 kernel with the 1.0.2 LVS patch and bumped the send / recv socket buffer memory to 1mb for both the uni-processor and SMP scenarios. The director is an Intel based system with 550 mhz Pentium 3's.
>
> In some tests I've done with FTP, I have seen *significant* improvements using dual and quad processors using 2.4. Under 2.2, there are improvements, but not astonishing ones.
>
> Things like 90% saturation of a Gig link using quad processors, 70% using dual processors and 55% using a single processor under 2.4.0test. Really amazing improvements.
>
> > Michael E Brown `michael_e_brown@dell.com` 26 Dec 2000 What are the percentage differences on each processor configuration between 2.2 and 2.4? How does a 2.2 system compare to a 2.4 system on the same hardware?
>
> I haven't had much of a chance to do a full comparison of 2.2 vs 2.4, but most of the evidence on tests that I have run points to a > 100% improvement for *network intensive* tasks.

> In our experiments we've been seeing an SMP director perform significantly worse than a uni-processor one (using the same hardware - only difference was booting an SMP kernel or uni-processor).

25.25 Multiple IPs on the Director

Michael Sparks

It's useful for the director to have 3 IP addresses. One which is the real machines base IP address, one which is the virtual service IP address, and then another virtual IP address for servicing the director. The reason for this is associated with director failover.

Suppose:

- X realservers pinging director on real IP A (assume a heartbeat style monitor) serving pages off virtual IP V. (IP A would be in place of hostip above)
- Director on IP A fails, backup director (*) on IP B comes online taking over the virtual IP V. By not taking over IP A, IP B can watch for IP A to come back online via the network, rather than via a serial link (etc).
- Problem is the realservers are still sending to IP A for the heartbeat code to be valid on IP B, the realservers need to send their pings to IP B instead. IMO the easiest solution is to allocate a we need a "heartbeat"/monitor virtual IP. (this is the vhostip)

25.26 Performance Hints from the Squid people

There is information on the Squid site about tuning a squid box for performance. I've lost the original URL, but here's one about *file descriptors* `<http://www.squid-cache.org/Doc/FAQ/FAQ-11.html##ss11.4>` and another by *Joe Cooper* `<http://www.devshed.com/Server_Side/Administration/SQUID/>` (occasional contributor to the LVS mailing list) that also addresses the FD_SETSIZE problem (*i.e.* not enough filedescriptors). The squid performance information should apply to an LVS director. For a 100Mbps network, current PC hardware on a director can saturate a network without these optimizations. However current single processor hardware cannot saturate 1Gpbs network, and optimizations are helpful. The squid information is as good a place to start as any.

Here's some more info

> Michael E Brown `michael_e_brown@dell.com` 29 Dec 2000
>
> How much memory do you have? How fast of network links? There are some kernel parameters you can tune in 2.2 that help out, and there are even more in 2.4. From the top of my head,
>
> 1) /proc/sys/net/core/*mem* <– tune to your memory spec. The defaults are not optimized for network throughput on large memory machines.
>
> 2.) 2.4 only /proc/sys/net/ipv4/*mem*
>
> 3.) For fast links, with multiple adapters (Two gig links, dual CPU) 2.4 has NIC–>CPU IRQ binding. That can really help also on heavily loaded links.
>
> 4.) For 2.2 I think I would go into your BIOS or RCU (if you have one) and hardcode all NIC adapters (Assuming identical/multiple NICS) to the same IRQ. You get some gain due to cache affinity, and one interrupt may service IRQs from multiple adapters in one go, on heavily loaded links.
>
> 5.) Think "Interrupt coalescing". Figure out how your adapter driver turns this on and do it. If you are using Intel Gig links, I can send you some info on how to tune it. Acenic Gig adapters are pretty well documented.

> For a really good tuning guide, go to spec.org, and look up the latest TUX benchmark results posted by Dell. Each benchmark posting has a full list of kernel parameters that were tuned. This will give you a good starting point from which to examine your configuration.
>
> The other obvious tuning recommendation: Pick a stable 2.4 kernel and use that. Any (untuned) 2.4 kernel will blow away 2.2 in a multiprocessor configuration. If I remember correctly 2.4.0test 10-11 are pretty stable.

Some information is on

http://www.LinuxVirtualServer.org/lmb/LVS-Announce.html

25.27 Testimonials

This isn't particularly inclusive. We don't pester people for testimonials as we don't want to scare people from posting to the mailing list and we don't want inflated praise. People seem to understand this and don't pester us with their performance data either. The quotes below aren't scientific data, but it is nice to hear. The people who don't like LVS presumably go somewhere else, and we don't hear any complaints from them.

> "Daniel Erdös" 2 Feb 2000 How many connections did you really handled? What are your impressions and experiences in "real life"? What are the problems?

Michael Sparks `zathras@epsilon3.mcc.ac.uk`

Problems - LVS provides a load balancing mechanism, nothing more, nothing less, and does it *extremely* well. If your back end real servers are flakey in anyway, then unless you have monitoring systems in place to take those machines out of service as soon as there are problems with those servers, then users will experience glitches in service.

NB, this is essentially a real server stability issue, not an LVS issue - you'd need good monitoring in place anyway if you weren't using LVS!

Another plus in LVS's favour in something like this over the commercial boxes, is the fact that the load balancer is a Unix type box - meaning your monitoring can be as complex or simple as you like. For example load balancing based on wlc could be supplemented by server info sent to the director.

> Drew Streib `ds@varesearch.com` 23 Mar 2000 I can vouch for all sorts of good performance from lvs. I've had single processor boxes handle thousands of simultaneous connections without problems, and yes, the 50,000 connections per second number from the VA cluster is true.
>
> lvs powers SourceForge.net, Linux.com, Themes.org, and VALinux.com. SourceForge uses a single lvs server to support 22 machines, multiple types of load balancing, and an average 25Mbit/sec traffic. With 60Mbit/sec of traffic flowing through the director (and more than 1000 concurrent connections), the box was having no problems whatsoever, and in fact was using very little cpu.
>
> Using DR mode, I've sent request traffic to an director box resulting in near gigabit traffic from the real servers. (Request traffic was on the order of 40Mbit.)
>
> I can say without a doubt that lvs toasts F5/BigIP solutions, at least in our real world implementations. I wouldn't trade a good lvs box for a Cisco Local Director either.

```
> The 50,000 figure is unsubstantiated and was _not_ claimed by anyone at VA
> Linux Systems. A cluster with 16 apache servers and 2 LVS servers in a was
> configured for Linux World New York but due to interconnect problems the
> performance was never measured - we weren't happy with the throughput of the
> NICs so there didn't seem to be a lot of point. This problem has been
> resolved and there should be an opportunity to test this again soon.
```

> In recent tests, I've taken multinode clusters to tens of thousands of connections per second. Sorry for any confusion here. The exact 50,000 number from LWCE NY is unsubstantiated.

Jerry Glomph Black `black@real.com` 23 Mar 2000

We ran a very simple LVS-DR arrangement with one PII-400 (2.2.14 kernel)directing about 20,000 HTTP requests/second to a bank of about 20 Web servers answering with tiny identical dummy responses for a few minutes. Worked just fine.

Now, at more terrestrial, but quite high real-world loads, the systems run just fine, for months on end. (using the weighted-least-connection algorithm, usually).

We tried virtually all of the commercial load balancers, LVS beats them all for reliability, cost, manageability, you-name-it.

25.28 Transport Layer Security(TLS)

Noma wrote Nov 2000

Are you going to implement TLS(Transport Layer Security) Ver1.0 on LVS?

> Wensong I haven't read the TLS protocol, so don't know if the TLS transmits IP address and/or port number in payload. In most cases, it should not, because SSL doesn't.
>
> If it doesn't, you can use either of three VS/NAT, VS/TUN and VS/DR methods. If it does, VS/TUN and VS/DR can still work.

Ted Pavlic `tpavlic@netwalk.com`, Nov 2000

> I don't see any reason why LVS would have any bearing on TLS. As far as LVS was concerned, TLS connections would just be like any other connections.
>
> Perhaps you are referring to HTTPS over TLS? Such a protocol has not been completed yet in general, and when it does it still will not need any extra work to be done in the LVS code.
>
> The whole point of TLS is that one connects to the same port as usual and then "upgrades" to a higher level of security on that port. All the secure logic happens at a level so high that LVS wouldn't even notice a change. Things would still work as usual.

Julian Anastasov `ja@ssi.bg`

> This is an end-to-end protocol layered on another transport protocol. I'm not a TLS expert but as I understand TLS 1.0 is handled just like the SSL 3.0 and 2.0 are handled, i.e. they require only a support for persistent connections.

25.29 rcp and friends on LVS (better to use ssh)

David Lambe `david.lambe@netunlimited.com` Mon, 13 Nov 2000

I've recently completed "construction" of a LVS cluster consisting of 1 LVS and 3 real servers. Everything seems to work OK with the setup except for rcp. All it ever gives is "Permission Denied" when running rcp blahfile node2:/tmp/blahfile from a console on node1. Both rsh and rlogin function, BUT require the password to be entered twice.

> Joe sounds like you are running RedHat. You have to fix the pam files. The beowulf people have been through all of this. You can either recompile the r* executables without pam (my

solution), or you can fiddle with the pam files. For suggestions, go to the beowulf mailing list search engine at *scyld beowulf* <http://www.scyld.com/search.html> and look for "rsh", "root", "rlogin". (hmm it seems to have gone. Looks like you have to download the whole archive at *whole archive* <http://www.beowulf.org/pipermail/beowulf/> and grep through it.)

If you go to the beowulf site, you'll find people are moving to replace rsh etc with ssh etc on sites which could be attacked from outside (and turning off telnet, r* etc)

My machines aren't connected to the outside world so I have root with no passwd. To compile ssh do

./configure –with-none

and use the config file I've attached (the docs on passwordless root logins were not helpfull)

```
# This is ssh server systemwide configuration file.

Port 22
#Protocol 2,1
ListenAddress 0.0.0.0
#ListenAddress ::
HostKey /usr/local/etc/ssh_host_key
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
#PermitRootLogin without-password
#
# Don't read ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes
StrictModes yes
X11Forwarding no
X11DisplayOffset 10
PrintMotd yes
KeepAlive yes

# Logging
SyslogFacility AUTH
LogLevel INFO
#obsoletes QuietMode and FascistLogging

RhostsAuthentication no
#
# For this to work you will also need host keys in /usr/local/etc/ssh_known_hosts
#RhostsRSAAuthentication no
RhostsRSAAuthentication yes
#
RSAAuthentication yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
PermitEmptyPasswords yes
```

```
# Uncomment to disable S/key passwords
#SkeyAuthentication no
#KbdInteractiveAuthentication yes

# To change Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#AFSTokenPassing no
#KerberosTicketCleanup no

# Kerberos TGT Passing does only work with the AFS kaserver
#KerberosTgtPassing yes

CheckMail no
#UseLogin no

# Uncomment if you want to enable sftp
#Subsystem      sftp    /usr/local/libexec/sftp-server
#MaxStartups 10:30:60
```

25.30 Forwarding an httpd request based on file name not load (mod_proxy)

On Mon, 25 Dec 2000, Sean wrote:

I need to forward request using the Direct Routing method to a server. However I determine which server to send the request to depending on the file it has requested in the HTTP GET not based on it's load. For this I am

> Michael E Brown michael_e_brown@dell.com Mon, 25 Dec 2000 Use LVS to balance the load among several servers set up to reverse-proxy your realservers, set up the proxy servers to load-balance to realservers based upon content.
>
> atif.ghaffar@4unet.net On the LVS servers you can run apache with mod_proxy comiled in, then redirect traffic with it.
> Example
>
> ```
> ProxyPass /files/downloads/ http://internaldownloadserver/ftp/
> ProxyPass /images/ http://internalimagesserver/images/
> ```
>
> See *more on Proxy pass* <http://www.linuxfocus.org/English/March2000/article147.html> and *transparent proxy module for apache* <http://www.stevek.com/projects/mod_tproxy/>. You can use mod_rewrite if your realservers are reachable from the net.

25.31 URL parsing

Is there any way to do URL parsing for http requests (ie send cgi-bin requests to one server group, static to another group?)

> John Cronin jsc3@havoc.gtf.org 13 Dec 2000 Probably the best way to do this is to do it in the html code itself; make all the cgis hrefs to cgi.your-domain.com. Similarly, you can make images hrefs to image.your-domain.com. You then set these up as additional virtual servers, in

addition to your www virtual server. That is going to be a lot easier than parsing URLs; this is how they have done it at some of the places I have done consulting for; some of those places were using Extreme Networks load balancers, or Resonate, or something like that, using dozens of Sun and Linux servers, in multiple hosting facilities.

25.32 can I run my ipchains firewall and LVS on the same box?

"K.W." kathiw@erols.com

can I run my ipchains firewall and LVS (piranha in this case) on the same box? It would seem that I cannot, since ipchains can't understand virtual interfaces such as eth0:1, etc.

Brian Edmonds bedmonds@antarcti.ca 21 Feb 2001 I've not tried to use ipchains with alias interfaces, but I do use aliased IP addresses in my incoming rulesets, and it works exactly as I would expect it to.

Julian I'm not sure whether piranha already supports kernel 2.4, I have to check it. ipchains does not understand interfaces aliase even in Linux 2.2. Any setup that uses such aliases can be implemented without using them. I don't know for routing restrictions that require using aliases.

I have a full ipchains firewall script, which works (includes port forwarding), and a stripped-down ipchains script just for LVS, and they each work fine separately. When I merge them, I can't reach even just the firewall box. As I mentioned, I suspect this is because of the virtual interfaces required by LVS.

LVS does not require any (virtual) interfaces. LVS never checks the devices nor any aliases. I'm not sure what is the port forwarding support in ipchains too. Is that the support provided from ipmasqadm: the portfw and mfw modules? If yes, they are not implemented (yet). And this support is not related to ipchains at all. Some good features are still not ported from Linux 2.2 to 2.4 including all these autofw useful things. But you can use LVS in the places where use ipmasqadm portfw/mfw but not for the autofw tricks. LVS can perfectly do the portfw job and even to extend it after the NAT support: there are DR and TUN methods too.

Lorn Kay lorn_kay@hotmail.com I ran into a problem like this when adding firewall rules to my LVS ipchains script. The problem I had was due to the order of the rules.

Remember that once a packet matches a rule in a chain it is kicked out of the chain–it doesn't matter if it is an ACCEPT or REJECT rule(packets may never get to your FWMARK rules, for example, if they do not come before your ACCEPT and REJECT tests).

I am using virtual interfaces as well (eg, eth1:1) but, as Julian points out, I had no reason to apply ipchains rules to a specific virtual interface (even with an ipchains script that is several hundred lines long!)

unknown FWMARKing does not have to be a part of an ACCEPT rule.

If you have a default DENY policy and then say:

```
/sbin/ipchains -A input -d $VIP -j ACCEPT
/sbin/ipchains -A input -d $VIP 80 -p tcp -m 3
/sbin/ipchains -A input -d $VIP 443 -p tcp -m 3
```

To maintain persistence between port 80 and 443 for https, for example, the packets will match on the ACCEPT rule, get kicked out of the input chain tests, and never get marked.

25.33 Setting up a hot spare server

Mark Miller markm@cravetechnology.com 09 May 2001 We want a configuration where two Solaris based web servers will be setup in a primary and secondary configuration. Rather than load balancing between the two we really want the secondary to act as a hot spare for the primary.

Here is a quick diagram to help illustrate this question:

```
     Internet             LD1,LD2 - Linux 2.4 kernel
         |                RS1,RS2 - Solaris
      Router
         |
  -------+-------
  |             |
-----         -----
|LD1|         |LD2|
-----         -----
  |             |
  -------+-------
         |
       Switch
         |
  ---------------
  |             |
-----         -----
|RS1|         |RS1|
-----         -----
```

Paul Baker pbaker@where2getit.com 09 May 2001

Just use heartbeat on the two firewall machines and heartbeat on the two solaris machines.

Horms horms@vergenet.net 09 May 2001 You can either add and remove servers from the virtual service (using ipvsadm) or toggle the weights of the servers from zero to non-zero values.

Alexandre Cassen alexandre.cassen@canal-plus.com 10 May 2001 For your 2 LDs you need to run a Hot standby protocol. Hearthbeat can be used, you can also use vrrp or hsrp. I am actually working on the IPSEC AH implementation for vrrp. That kind of protocol can be usefull because your LD backup server can be used even if it is in backup state (you simply create 2 LDs VIP and set default gateway of your serveur pool half on LD1 and half on LD2).

For your webserver hot-spare needs, you can use the next *keepalived* <http://keepalived.sourceforge.net> in which there will be "sorry server" facility. This mean exactly what you need => You have a RS server pool, if all the server of this RS server pool are down then the sorry server is placed into the ipvsadm table automaticaly. If you use keepalived keep in mind that you will use NAT topology.

Joe 11 May 2001 Unless there's something else going on that I don't know about, I expect this isn't a great idea. The hot spare is going to degrade (depreciate, disk wear out - although not quite as fast, software need upgrading) just as fast idle as doing work.

You may as well have both working all the time and for the few hours of down time a year that you'll need for planned maintenance, you can make do with one machine. If you only need the capacity of 1 machine, then you can use two smaller machines instead.

25.34 An LVS of LVSs

Since an LVS obeys unix client/server semantics, an LVS can replace a realserver (at least in principle, no-one has done this yet). Each LVS layer could have its own forwarding method, independantly of the other LVSs. The LVS of LVSs would look like this, with realserver_3 being in fact the director of another LVS and having no services running on it.

```
                        --------
                       |        |
                       | client |
                       |________|
                           |
                           |
                        (router)
                           |
                           |
                           |       ------------
                           |  DIP |            |
                           |------| director_1 |
                           |  VIP |____________|
                           |
                           |
                           |
         -------------------------------------
         |                  |                |
         |                  |                |
     RIP1, VIP          RIP2, VIP        RIP3, VIP
   --------------     --------------    -------------
  |              |  |              |  |             |
  | realserver1  |  | realserver2  |  | realserver3 |
  |              |  |              |  | =director_2 |
  |______________|  |______________|  |_____________|

                                             |
                                             |
         -------------------------------------
         |                  |                |
         |                  |                |
     RIP4, VIP          RIP5, VIP        RIP6, VIP
   --------------     --------------    --------------
  |              |  |              |  |              |
  | realserver4  |  | realserver5  |  | realserver6  |
  |              |  |              |  |              |
  |______________|  |______________|  |______________|
```

If all realservers were offering http and only realservers1..4 were offering ftp, then you would (presumably) setup the directors with the following weights for each service:

- director_1: realserver1 http,ftp=1; realserver2 http,ftp=1;realserver3 http=3,ftp=1
- director_2: realserver4 http,ftp=1; realserver5 http=1 (no ftp);realserver3 http=1 (no ftp)

You might want to do this if realservers4..6 were on a different network (*i.e.* geographically remote). In this case director_1 would be forwarding by LVS-Tun, while director_2 could use any forwarding method.

25.34.1 An LVS of LVSs: using Windows/Solaris machines with LVS-Tun

This is the sort of ideas we were having in the early days. It turns out that not many people are using LVS-Tun, most people are using Linux realservers, and not many people are using geographically distributed LVSs.

Joe, Jun 99

For the forseeable future many of the servers who could benefit from the LVS will be microsoft or solaris. The problem is that they don't have tunneling. A solution would be to have a linux box in front of each real server on the link from the director to the real server. The linux box appears to be the server to the director (it has the real IP eg 192.168.1.2) but does not have the VIP (eg 192.168.1.110). The linux box decapsulates the packet from the director and now has a packet from the client to the VIP. Can the linux box route this packet to the real server (presumably to an lo device on the real server)?

The linux box could be a diskless 486 machine booting off a floppy with a patched kernel, like the machines in the Linux router project.

> Wensong 29 Jun 1999 We can use nested (hyprid) LinuxDirector approach. For example,
>
> ```
>  LVS-Tun   ---->   LVS-NAT ---->  RealServer1
>       |                 |       ...
>       |                 ----->  RealServer2
>       |
>       |           ....
>       |
>       |
>       -------->   LVS-NAT  ....
> ```
>
> Real Servers can run any OS. A LVS-NAT load balancer usually can schedule over 10 general servers. And, these LVS-NATs can be geographically distributed.
>
> By the way, LinuxDirector in kernel 2.2 can use LVS-NAT, VS-TUN and LVS-DR together for servers in a single configuration.

25.35 Connecting through multiple parallel links to the clients

(This is not an LVS problem, just a normal routing problem.)

> Logu `lvslog@yahoo.com` 5 Oct I have two isdn internet connection from two different isps. I am going to put an lvs_nat between the users and these two links so as to loadbalace the bandwidth.

Julian

You can use the Linux's multipath feature:

```
# ip ru
0:      from all lookup local
50:     from all lookup main
...
```

```
100:    from 192.168.0.0/24 lookup 100
200:    from all lookup 200
32766:  from all lookup main
32767:  from all lookup 253

# ip r l t 100
default  src DUMMY_IP
        nexthop via ISP1  dev DEV1 weight 1
        nexthop via ISP2  dev DEV2 weight 1

# ip r l t 200
default via ISP1 dev DEV1  src MY_IP1
default via ISP2 dev DEV2  src MY_IP2
```

You can add my *dead gateway detection extension* `<http://www.linuxvirtualserver.org/~julian/>` (for now only against 2.2)

This way you will be able fully to utilize the both lines for masquerading. Without this patch you will not be able to select different public IPs to each ISP. They are named "Alternative routes". Of course, in any case the management is not an easy task. It needs understanding.

25.36 performance testing tools

> Dennis Kruyt, 9 Jan 2002 I am looking for software for testing my lvs webservers with persistant connection. With normal http benchmarking tools all request come from one IP, but I want to simulate a few hundred connections from different IPss with persistant connections.

Joe Cooper `joe@swelltech.com` 09 Jan 2002

Web Polygraph `<http://www.web-polygraph.org>` is a benchmarking framework originally designed for web proxies. It will generate thousands of IPs on the client box if you request them. It does not currently have a method to test existing URLs, as far as I know (it provides its own realserver(s) and content, so that data is two-sided).

It currently works very well for stress-testing an LVS balancer, but for the realservers themselves it probably needs a pretty good amount of coding. The folks who developed it will add features for pretty good rates, particularly if the new features fit in with their future plans.

It does have some unfortunate licensing restrictions, but is free to get, use and modify for your own internal purposes.

25.37 Conntrack, effect on throughput

Conntrack is part of netfilter. It keeps track of connections and knows the relationship of a packet to existing connections. This enables filtering to reject or allow packets *e.g.* a packet appearing to be part of a passive ftp connection is only valid after a call to setup a passive ftp connection.

> Rodger Erickson `rerickson@aventail.com` 17 Dec 2001
> Does anyone have any comments they can make on the effect of conntrack on LVS performance? The LVS device I'm using also has to do some DNAT and SNAT, which require conntrack to be enabled.

Julian

We need to port the 2.2 masquerade to 2.4 :) LVS reused some code from 2.2 but much of it is removed and I'm not sure it can be added back easily. It would be better to redesign some parts of Netfilter for 2.5 or 2.7 :) You can use the ipchains compat module. But may be it does not work for FTP and is broken at some places.

The best approach might be to test the slowdown when using both LVS and conntracking and if it's not fast enough, buy faster hardware. It will take less time :) You can test the slowdown with some app or even with testlvs.

> patrick edwards My lvs works with no problem. However with in a matter of an hour or two my bandwidth drops to virtually nothing and the CPU load goes ballistic. I have a 100Mbit internal network, but at times i'm lucky to see 50Kps.

Christian Bronk `chris@isg.de` 15 Jan 2002

For our 2.4 kernel test servers, it turned out that ipchains under kernel 2.4 does full connection-tracking and makes the system slow. Try to use iptables or the arp-patch instead.

> Fabrice `fabrice@urbanet.ch` 17 Dec 2001 When I ran testlvs, with conntrack enabled on the client machine (the one that runs testlvs), I had a mean of about 2000 SYN/s. When I removed thoses modules (there are many conntracks) I reached 54'000 SYN/s!

Julian Anastasov `ja@ssi.bg` 22 Dec 2001

I performed these tests with 40K SYN/s incoming (director near its limits), VS-NAT, 2.4.16, noapic, SYN flood from testlvs with -srcnum 32000 to 1 realserver.

- only IPVS 0.9.8able to send the 40K/sec (same as incoming), 3000 context switches/sec
- after modprobe ip_conntrack hashsize=131072

  10-15% performance drop, 500 context switches/sec
- after modprobe iptable_nat (no NAT rules)5% performance drop, same number of context switches/sec
- Additional test: -j DNAT instead of IPVS

  ```
  # modprobe ip_conntrack hashsize=131072
  # modprobe iptable_nat
  # cat /proc/sys/net/ipv4/ip_conntrack_max
  1048576
  ```

  Tragedy: 1000P/s out, director is overloaded.

I looked into the ip_conntrack hashing, it is not perfect for incoming traffic between two IPs, but note that testlvs uses different IPs, so after little tuning, it seems that the DNAT's problem is not the bad hash function. Maybe I'm missing some NF tuning parameters.

25.38 LVS on a Linux/IBM mainframe

Kyle Sparger `ksparger@dialtoneinternet.net` 18 Sep 2001

I'm familiar with the s/390; the zSeries 900 will be similar, but on a 'next-gen' scale – It's 64 bit and I expect 2-3 times the maximum capacity.

- The s/390 is ONLY, at most, a 12-way machine in a single frame, 24-way in a two-frame configuration. The CPU's are not super-powered; they're normal CPU's, so imagine a normal 12-24 way, and you have a good idea. It does have special crypto-processors built in, if you can find a way to use them.
- The s/390, however, has an obnoxiously fast bus – 24GByte/s. Yes, I did mean gigabytes. Also, I/O takes up almost no CPU time, as the machines have sub-processors to take care of it.
- The s/390 is a 31bit machine – yes, 31. One bit defines whether the code is 16 or 31 bit code. The z/900 is a 64bit machine. Note that the s/390, afaik, suffers when attempting to access memory over a certain amount, like any 31/32 bit machine would – 2 gigs can be addressed in a single clock cycle; greater than that takes longer to process, since it requires more than 32 bits to address.
- From top to bottom, the entire machine is redundant. There is no single point of failure anywhere in the machine. According to IBM's docs, the MTBF is 30 years. It calls IBM when it's broken, and they come out and fix it. The refrigerator ad was no joke ;) Of course, this doesn't protect you from power outages, but interestingly enough, if I recall correctly, all RAM is either SRAM, or battery backed – the machine will come back up and continue right where it left off when it lost power. No restarting instances or apps required. No data lost.

There are five premises for the cost-savings:

- You don't have to design a redundant system – it's already built in.
- One machine is easier to manage than n number servers.
- One machine uses less facilities than n number servers.
- A single machine, split many ways, can result in higher utilization.
- Linux, Linux, Linux. All the free software you can shake a stick at.

On the flip-side, there are some constraints:

- If you have 500 servers, all at 80% CPU usage, there's no way you're going to cram them all onto the mainframe. Part of the premise is that most servers sit at only a fraction of their maximum capacity.
- The software must be architecture compatible.
- Mainframe administrators and programmers are rare and expensive.

The ideal situation for an s/390 or z/Series is an application which is not very CPU intensive, but is highly I/O intensive, that must _NEVER_ go down. Could that be why many companies do databases on them? Think airline ticketing systems, financial systems, inventory, etc :) Realize, however, that your cost of entry is probably going to be well over a million dollars, unless you want a crippled entry-level box. You probably don't want to buy this server to run your web site. You probably want to buy it to run your database. That being said, if you happen to order more than you really need – a reasonably common phenomenon in IT shops – you can now run Linux instances with that extra capacity. :)

25.39 How do I check to see if my kernel has the ip-vs patch installed?

Short answer: If you need the HOWTO, you shouldn't be using other people's kernels - go compile up an ipvs patched kernel yourself.

Long answer: If you have the kernel binary, then you have a bit of a job ahead of you. If you've compiled it yourself, then you should give it a name like

```
bzImage-0.9.3-2.4.9-module-forward-shared
```

to remind you of what it is (here ip_vs 0.9.3 compiled as modules, kernel 2.4.9, forward-shared patch).

Otherwise

- If ipvs is compiled into the kernel

  ```
  $ grep ip_vs_init System.map
  ```

- if it's a recent kernel and ip_vs is compiled as a module, running ipvsadm will load the module for you. From there check the loaded modules with lsmod.
- for older kernels you need to load the module before running ipvsadm. If the kernel doesn't have the ip_vs patch, the module probably won't load.

25.40 Is it possible to forward a range of ports/range of IPs?

> Miri Groentman, 11 Jul 2001
> Is it possible to configure a range of ports rather than a single port

Joe

if you mean ports for services, yes, see fwmark in the HOWTO. You can also forward a range of IPs.

25.41 Running a test LVS (director, backup director and realservers) on one box

> Can I load both the ipvs code and the failover code in a single stand alone machine?

Joe 09 Jul 2001

VMWare?

Henrik Nordstrom `hno@marasystems.com`

user-mode-linux `<http://user-mode-linux.sourceforge.net/>` works beautifully for simulating a network of Linux boxes on a single CPU. Use it extensively when hacking on netfilter/iptables, or when testing our patches on new kernels and/or ipvs versions. Also has the added benefit that you can run the kernel under full control of gdb, which greatly simplifies tracking kernel bugs down if you get down to kernel hacking.

Joe

I attended a talk by the UML author at OLS 2001. It's pretty smart software. You can have virtual CPUs, NICs... - you can have a virtual 64-way SMP machine running on your 75MHz pentium I. The performance is terrible, but you can test your application on it.

25.42 mqseries

The LVS worked for a client connected directly to the director, but not from a client on the internet.

> Carlos J. Ramos `cjramos@genasys.es` 12 Mar 2002 Now, it seems to be solved by using static routes to hosts instead of using static routes to networks.

There is also another important note. Directors uses MQSeries from IBM, the starting sequence in haresources was mqseries masq.lvs (script for NAT), it looks that the 1 minute needed by mqseries to get up was confusing(!?) masq.lvs or ldirectord. We have just change the order to get up mqseries and masq.lvs, rising up first masq.lvs and finally mqseries.

With these two changes it works perfectly.

25.43 LVS log files

Chris Ruegger Does LVS maintain a log file or can I configure it to use one so I can see a history of the requests that came in and how it forwarded them?

Joe 1 Apr 2002

It doesn't but it could. LVS does make statistics available.

Another question is whether logging is a good idea. The director is a router with slightly different rules than a regular router. It is designed to handle 1000's requests/sec and operate with no spinning media (eg on a flash card). There's no way you can log all connections to a disk and maintain throughput. You couldn't even review the contents of the logs. People do write filter rules, looking for likely problems and logging suspicious packets. Even reviewing those files overwhelmes most people.

Ratz 2 Apr 2002

LVS works on L4. Maybe the following command will make you happy:

```
echo 666 > /proc/sys/net/ipv4/vs/debug_level
```

25.44 LVS and linux vlan

Matt Stockdale Does the current LVS code work in conjuction with the linux vlan code? We'd like to have a central load balancing device, that connects into our core switch w/ a dot1q trunk, and can have virtual interfaces on any of our many netblocks/vlans.

Benoit Gaussen `bgaussen@fr.colt.net` 20 Mar 2002

I tested it and it works. The only problem I encountered is a MTU problem with eepro100 driver and 8021q code. However there is a small patch on 8021q website. My config was linux 2.4.18/lvs 1.0.0 configured with LVS-NAT.

26 L7 Switching

Switching packets based on the content of the payload (L7 switching) is something of a holy grail in this area. However, inspecting packet contents is slow (or CPU intensive) compared to L4 switching and should not be regarded as a panacea. Commercial L7 switches are expensive. It's better to handle the problem at the L4 layer, something that can often by done by 11.25 (correct design of applications).

Netfilter `<http://netfilter.samba.org/>` has code for inspecting the contents of packets through the *u32* option. Presumably this could be coupled with fwmark to setup an LVS.

Preliminary code, *KTCPVS* `<http://www.linuxvirtualserver.org/software/ktcpvs/ktcpvs.html>`, for this has been written by Wensong. Some documentation is in the source code.

26.1 from the mailing list about L7 switching

Michael Sparks `Michael.Sparks@wwwcache.ja.net`> 12 Jul 1999

Some of the emerging redirection switches on the market support something known as Level 7 redirection which essentially allows the redirector to look at the start of the TCP data stream, by spoofing the initial connection and making load balancing based on what it sees there. (Apologies if I'm doing the equivalent of "teaching your grandma to suck eggs", but at least this way there's less mis-understanding of what I'm getting at/after)

For example if we have X proxy-cache servers, we could spoof an HTTP connection, grab the requested URL, and hash it to one of those X servers. If it was possible to look inside individual UDP packets as well, then we would be able to route ICP (inter cache prototol) packets in the same way. The result is a cluster that looks like a single server to clients.

> Wensong Do you mean that these X proxy-cache servers are not equivalent, and they are statically partitioned to fetch different objects? for example, the proxy server 1 is to cache European URLs, and the proxy server 2 is to cache Asian URLs, then there is a need to parse the packets to grab the requested URL. Right?
>
> If you want to do this, I think Apache's mod_rewrite and mod_proxy can be used to group these X proxy-cache servers as a single proxy server. Since the overhead of dispatching requests in application level is high, its scalability may not be very good, the load balancer might be a bottleneck when there are 4 proxy servers or more.
>
> The other way is to copy data packet to userspace to grap the request if the request is in a single UDP packet, the userspace program select a server based on the request and pass it back to the kernel.

For generic HTTP servers this could also allow the server to farm cgi-requests to individual machines, and the normal requests to the rest. (eg allowing you to buy a dual/quad processor to handle soley cgi-requests, but cheap/fast servers for the main donkey work.)

> Wensong It is statically partitioned. Not very flexible and scalable.

We've spoken to a number of commercial vendors in the past who are developing these things but they've always failed to come up with the goods, mainly citing incompatibility between our needs and their designs :-/

Any ideas how complicated this would be to add to your system?

> Wensong If these proxy-cache servers are identical(they are the same of all kind of URL requests), I have a good solution to use LVS to build a high-performance proxy server.

```
request                            |-<--fetch objects directly
|                    |-----Squid 1---->reply users directly
|->LinuxDirector  ---|
  (VS-TUN/VS-DR)     |-----Squid 2
                     |      ...
                     |-----Squid i
```

> Since LVS-Tun/VS-DR is on client-to-server half connection, squid servers can fetch object directly from the Internet and return objects directly to users. The overheading of forwarding request is low, scalabilty is very good.
>
> The ICP is used to query among these Squid servers. In order to avoid the mulitcasting storm, we can add one more NIC in each squid server for ICP query, we can call it multicast channel.

again Michael,

The reason I asked if the code could be modified to do this:

```
> look at the start of the TCP data stream, by spoofing the initial
> connection and making load balancing based on what it sees there.
```

Is to enable us to do this:

1. Spoof the connection until we're able to grab the URL requested in the HTTP header.
2. Based on that info make a decision as to which server should deal with the request.
3. Fake the start of a TCP request to that server, making it look like it's come from the original client, based on the info we got from the original client in 1) And when we reach the end of the data we recieved from 1) stop spoofing and hand over the connection.

   Boxes like the Arrowpoint CS100/CS800 do this sort of thing, but their configurable options for 2) isn't complex enough.
4. Forward UDP packets after looking inside the payload, seeing if it's an ICP packet and if so forwarding based on the ICP payload.

The reason for 1,2 & 3 is to have deterministic location of cached data, to eliminate redundancy in the cache system, and to reduce intra-cache cluster communication. The reason for 4 is because the clients of our caches are caches themselves - they're the UK National Academic root level caches servicing about a 3/4 billion requests per month during peak periods.

Also 2) can be used in future to implement a cache-digest server to serve a single cache digest for the entire cluster to eliminate delays for clients caused by ICP. (During peak periods this is large.)

The boxes from Arrowpint can do 1-3, but not 4 for example, and being proprietory hardware...

Essentially the ICP+cache digest thing for the cluster is the biggest nut - squid 2.X in a CARP mode can do something similar to 1,2 & 3, at the expense of having to handle a large number of TCP streams, but wouldn't provide a useful ICP service (it would always return ICP MISS), and can't provide the cache digest service. (Or would at least return empty digests)

```
> I have a good solution to use LVS to build a high-performance proxy
> server.
```

Fine (to an extent) for the case where requests are coming from browsers, but bad where the clients are caches:

- Client sends ICP packet to Linux Director, forwarded to cache X resulting in a ICP HIT reply, so client sends HTTP request to Linux Director, which forwards request to cache Y, resulting in a MISS, Y grabs from X, no major loss.
- However same scenario with X & Y flipped, results in ICP MISS, where there would've been a cluster hit. Pretty naff really!

It's even worse with cache digests...

We've had to say this sort of thing to people like Alteon, etc too... (Lots more details in an overview at http://epsilon3.mcc.ac.uk/ zathras/WIP/Cache_Cooperation/).

later...

A slightly better discussion of how these techniques can be used is at: http://www.ircache.net/Cache/Workshop99/Papers/johnson-0.ps.gz)

> Wensong I have read the paper "Increasing the performance of transparent caching with content-aware cache bypass". If no inter-cache cooperation is concerned, it can be easily done on Linux, you don't need to buy expensive arrowpoint. As for availability, Linux boxes are reliable now. :)
>
> I can modify the transparent proxy on Linux to do such a content-aware bypass and content-aware switching. The content-aware bypass will enable fetch non-cacheable objects directly, and the content-aware switching can let your cache cluster not overlapped.

27 Geographically distributed load balancing

VS-Tun offers the opportunity for your realservers to be anywhere (including on different continents).

Even better would be the ability to determine the closest server (by some internet metric) to any particular client. This has been done at least for http by Horms (of Ultra Monkey fame) with the 1.4.5 (Super Sparrow Project). Super Sparrow works differently than and is incompatible with LVS, and because of it works, it is correspondingly less likely that anyone will go to the pain of developing an LVS compatible version of geographical load balancing.

Super Sparrow uses *zebra* `<http://www.zebra.org>` to determine the number of AS hops between client and server using BGP4 routing information. Documentation on BGP is hard to find (the early zebra docs had none). Horms suggests a *BGP Routing Part 1* `<http://www.netaxs.com/~freedman/bgp/bgp.html>` by Avi Freeman of Akami. It's somewhat Cisco centric and there is no part 2 yet but is applicable to zebra.

However many people thought about geographically distributed LVSs. For historical completeness here are some of their musings.

> Michael Sparks `zathras@epsilon3.mcc.ac.uk` I'm curious about the physical architecture of a cluster of servers where "the realservers have their own route to the client." (Like in LVS-DR and LVS-Tun) How have people achived this in real life? Does each real server actually have it's own dedicated router and Internet connection? Do you set up groups of real servers where each group shares one line?

It could do or it can share things. We've got 3 LVS based clusters, based around LVS-Tun. The reason for this is because one of the clusters is at a different location (about 200 miles from where I'm sitting) , and this allows us to configure all the realservers in the same way thus:

```
tunl0:1 - IP of LVS balanced cluster1
tunl0:2 - IP of LVS balanced cluster2
tunl0:3 - IP of LVS balanced cluster3 (remote)
```

The only machines that ends up getting configured differently then are just the directors.

So whilst machines are nominally in one of the three clusters, if (say) the remote cluster is overloaded, it can take advantage of the extra machines in the other two clusters, which then reply directly back to the client - and vice versa.

In that situation a client in (say) Edinburgh, could request an object via the director at Manchester, and if the machines are overloaded there, have the request forwarded to London, which then requests the object via a network completely separate from the director's and returns the object to the client.

That UK Nat cache likely to be introducing another node at another location in the country at some point in the near future which will be very useful. (The key advantage is that at each location we gain X more Mbit/s of bandwidth to utilise making service better for users.)

Joe

how do I get BGP info from a BGP router to the director?

> Lars Marowsky-Bree `lmb@teuto.net` 23 Jul 1999 If you telnet to the BGP4 port of the the router running BGP4, and do a "sh ip route www.yahoo.com" for example, you will get something like this

```
Routing entry for 204.71.192.0/20, supernet
  Known via "bgp 8925", distance 20, metric 0
  Tag 1270, type external
  Last update from 139.4.18.2 1w5d ago
  Routing Descriptor Blocks:
  * 139.4.18.2, from 139.4.18.2, 1w5d ago
      Route metric is 0, traffic share count is 1
      AS Hops 4
```

> This address is 4 AS hops away from me. You can also find out this information using SNMP if I recall correctly.
>
> The most cool idea would be to actually run a routing daemon on the cluster manager (like gated or Zebra (see www.zebra.org)), then we wouldn't even need to telnet to the router but could run fully self contained using an IBGP feed. Zebra is quite modular even and could possibly be made to integrated more tightly with the dispatcher...

It must have been your other mail where you said that this was simple but not everyone knew about it. I just found out why. My cisco tcipip routing book has 2 pages on BGP. They told me to find a cisco engineer to "discuss my needs" with if I wanted to know more about BGP

> There is actually some sort of nice introduction hidden on www.cisco.com, search for BGP4. "Internet Routing Architecture" from Cisco Press covers everything you might want to know about BGP4. _All_ routers, participating in global Internet routing, hold a full view of the routing table, reflecting their view of the network. They know how many AS (autonomous systems) are in between them and any reachable destination on the network. If they have multiple choices (ie multile connections, so called multi homed providers), they select the one with the shortest AS path and install it into their routing table.
>
> Now, one sets up a dispatcher which has BGP4 peerings with all participatin g clusters. Since the dispatcher only installs the best routes to all targets in it's routing table, it is a simple lookup to see which cluster is closest to the client.
>
> If a cluster fails, the peering with the dispatcher is shutdown and the backup routes (the views learned from other clusters) take over.
>
> This is actually very nice and well tested technology, since it is what makes the internet work.
>
> It requires cooperation on part of the ISP hosting the cluster, that he provides a read-only "IBGP" feed to the cluster manager inside his AS.
>
> BGP4 AS hops may not be directly related to latency. However, it tells you how many backbones are in between you and how many are not, which does have a tight relationship to the latency. And you can use the BGP4 route-maps etc to influnce the load balancing on an easy way - if you got one cluster from which a certain part of the Internet is reached via a slow satellite link, you can automatically lower the preference for all routes comeing in via that satellite link and not use that.

Ted Pavlic `tpavlic@netwalk.com` 9 Sep 1999

For now AS hops probably are useful - we have two mirrors on different continents.

> You do NOT and cannot run OSPF here. OSPF is an "IGP" (interior routing protocol) which can't be usefully applied here.

I suppose I figured large networks might all share OSPF information, but I guess that they wouldn't share too much OSPF information between different geographical locations. (And I'm guessing that the latency between the load balancer and the user will PROABABLY almost exactly the same as the latency between the end-servers and the user...so...) I never claimed that I knew much of anything about BGP or OSPF, but thought that if BGP wasn't very helpful... OSPF might be. :) (It was a shot in the dark - a request for comments, if anything)

> Of course, you not only want to figure BGP4, but also load and availability. We should investigate what other geogrpahical load balancers do. A lot of them setup large proxy'ing networks.

AFAICT, a lot of geographical load balancing systems seem to use their own means of finding which server is best per end-user. I think, for decent load balancing on that sort of scale, most balancers have to invent their own wheel.

This by no means is an easy task – doing decent geographical load balancing. Companies put in a good deal of R&D to do just this and customers pay a good deal of money for the results.

> > Worse comes to worst, have a perl script look-up the name of the machine that made the request... grab the first-level domain... figure out which continent it should go on.
>
> DNS has no guaranteed reply times at all.

it wasn't a serious suggestion for production. Just simply a way to divy out which mirror got which request.

28 Useful things that have other place (yet)

28.1 Files which are kernel version dependant eg System.map and ipvsadm

I'm always booting into versions of the kernel which don't match the version of the kernel in /usr/src/linux. This gives me the warning (eg when running 'ps')

```
Warning: /usr/src/linux/System.map has an incorrect kernel version.
```

As well, my version of ipvsadm doesn't work anymore (and will probably tell me that I don't have ipvsadm installed when I do).

My /usr/src directory has entries like

```
linux-1.0.3-2.2.18-TP
```

for the 2.2.18 kernel with the 1.0.3 ipvs patch applied and built with transparent proxy. I link this to linux-2.2.18

My /sbin director has ipvsadm-2.2.18 and ipvsadm-2.4.1 (ipvsadm versions match ipvs versions and not kernel versions, but it's close enough to kernel versions that labelling them by kernel versions works).

At bootup I run this script to get the right versions of these files.

```
#always have the right System.map, ipvsadm...
#no matter which kernel version you're running
#Joseph Mack (C) 2001 released under GPL, jmack@wm7d.net,

#To use:
#for files/directories which are kernel version specific.
#
#eg you'll get error messages about System.map being the wrong version unless
#you have /usr/src/linux linked to the same version of linux as you booted from.
#eg in /sbin: ipvsadm-2.4.1, ipvsadm-2.2.18 which is called as /sbin/ipvsadm
#
#in main you send a list of directories/files like /usr/src/linux/, /sbin/ipvsadm
#for each item in this list, this program will
#a. delete /usr/src/linux if linux is a link, otherwise do nothing if linux is a file
#b. look for linux-x.x.x where x.x.x is the kernel version
#c. ln -s linux-x.x.x linux, else issue notice if there is no linux-x.x.x
#
#---------------------

ip_vs(){
#IPVS_VERSION is assigned the version number of IPVS currently loaded

if [ "$LINUX_KERNEL_SERIES" = "2.2" ]
then
        IPVS_VERSION=`grep "IP Virtual Server version" /proc/net/ip_vs | awk '{print \$5}'`
        echo $IPVS_VERSION

elif [ "$LINUX_KERNEL_SERIES" = "2.4" ]
then
        IPVS_VERSION=`grep "IP Virtual Server version" /proc/net/ip_vs | awk '{print \$5}'`
        echo $IPVS_VERSION

else
        echo "Error: kernel $UNAME_R not from 2.2 or 2.4 series"
        exit -1
fi
}

make_link(){
BASENAME=`/usr/bin/basename $1`
#echo "BASENAME $BASENAME"

cd `/usr/bin/dirname $1`

#if $BASENAME a link or doesn't exist
if [ -L $BASENAME -o ! \( -e $BASENAME \) ]
then
        #only do something if we have a target
        if [ -d $BASENAME-${UNAME_R}-${IPVS_VERSION} -o -f $BASENAME-${UNAME_R}-${IPVS_VERSION} ]
        then
                if [ -L $BASENAME ] #if the filename is a link, delete it
                then
                        /bin/rm $BASENAME
                fi
                #there is no $BASENAME now.
                /bin/ln -s $BASENAME-${UNAME_R}-${IPVS_VERSION} $BASENAME
        elif [ -d $BASENAME-${IPVS_VERSION}-${UNAME_R} -o -f $BASENAME-${IPVS_VERSION}-${UNAME_R} ]
        then
                if [ -L $BASENAME ] #if the filename is a link, delete it
                then
                        /bin/rm $BASENAME
                fi
                #there is no $BASENAME now.
                /bin/ln -s $BASENAME-${IPVS_VERSION}-${UNAME_R} $BASENAME
        elif [ -d $BASENAME-${IPVS_VERSION} -o -f $BASENAME-${IPVS_VERSION} ]
        then
                if [ -L $BASENAME ] #if the filename is a link, delete it
                then
                        /bin/rm $BASENAME
                fi
                #there is no $BASENAME now.
                /bin/ln -s $BASENAME-${IPVS_VERSION} $BASENAME
        elif [ -d $BASENAME-${UNAME_R} -o -f $BASENAME-${UNAME_R} ]
        then
                if [ -L $BASENAME ] #if the filename is a link, delete it
                then
                        /bin/rm $BASENAME
                fi
                #there is no $BASENAME now.
                /bin/ln -s $BASENAME-${UNAME_R} $BASENAME
        else
                echo "no $BASENAME-${UNAME_R} or $BASENAME-${IPVS_VERSION} to link to"
        fi
else
        echo "cannot delete $BASENAME, doesn't exist or is a regular file"
fi
cd -
}
#-----------------
#main:
echo "rc.system_map "
UNAME_R=`/bin/uname -r`
LINUX_KERNEL_SERIES=${UNAME_R%.*}

ip_vs
if [ $? != "0" ]
```

```
then
        echo "Error: unable to determine IPVS version"
        exit -1
fi

make_link /usr/src/linux
make_link /sbin/ipvsadm
make_link /sbin/ipvsadm-save
make_link /sbin/ipvsadm-restore
make_link /usr/src/ipvs
#make_link /bin/foo #a test, foo-x.x.x doesn't exist

#-----------------
```

Note:klogd is supposed to read files like /boot/System.map-<kernel_version> allowing you to have several kernels in /. However this doesn't solve the problem for general executables.

28.2 Ramdisk

I needed a ramdisk for testing once and couldn't find the instructions for setting it up. Here they are

Jerry Glomph Black `black@real.com`

You specify the ramdisk size when you load the rd module, as an option.

```
/sbin/insmod rd rd_size=32768
/sbin/mke2fs -m0 /dev/ram0
mount -t ext2 /dev/ram0 /mnt
```

28.3 cscope

cscope is a code symbol navigating tool.

from Patrick O'Rourke, `orourke@missioncriticallinux.com`

A cscope for Linux is at http://sourceforge.net/projects/cscope/

29 Patches

These are patches not in the standard ipvs distribution, but which I expect to be useful to some/many people.

Note the original sgml requires & to be represented as &amp;. If you extract the patch from the sgml file you will get &amp; rather than & in your patch. To get the correct patch, save the html representation as txt.

29.1 machine readable error codes from ipvsadm

Computers can talk to each other and read from and write to other programs. You shouldn't have to get a person to sit at the console to parse the output of a program. Here's a patch to make the output of ipvsadm machine readable

Padraig Brady padraig@antefacto.com 07 Nov 2001

This 1 line patch is useful for me and I don't think it will break anything. It's against ipvsadm-0.8.2 and returns a specific error code.

```
--Boundary_(ID_nuebet+LsBGYFsmRPljqqA)
Content-type: text/plain; name="ipvsadm-0.8.2-returncode.diff"
Content-disposition: inline; filename="ipvsadm-0.8.2-returncode.diff"
Content-transfer-encoding: 7bit

--- //ipvs-0.8.2/ipvs/ipvsadm/ipvsadm.c Fri Jun 22 16:03:08 2001
+++ ipvsadm.c   Wed Nov  7 16:29:11 2001
@@ -938,6 +938,7 @@
         result = setsockopt(sockfd, IPPROTO_IP, op,
                             (char *)&urule, sizeof(urule));
         if (result) {
+                result = errno; /* return to caller */
                 perror("setsockopt failed");

                 /*

--Boundary_(ID_nuebet+LsBGYFsmRPljqqA)--
```

29.2 machine compatible ipsvadm entries

With commands like ifconfig, you can repeat the same valid command without errors. With ipvsadm, if a VIP:port entry already exists, then you will get an error on re-entering it. If no entry exists, then you put it into the ipvsadm table by adding (-a) it; if the entry exists, then you edit (-e) it. You will get an error if you use the wrong command.

This is a problem for automated control of an LVS:

- a demon will need to hold a duplicate of the state held in ipvsadm (either by journalling the ipvsadm commands or parsing the output of ipvsadm) or
- you'll have to try both commands and see which one runs and then have the script figure out if both the error and the non-error was valid.

What is needed is a version of ipvs that accepts valid entries without error. Here's a patch by Horms against ipvs-0.9.0. It modifies several ipvs files, including ipvsadm.

```
diff -ruN ipvs-0.9.0/ipvs/ip_vs.h ipvs-0.9.0.new/ipvs/ip_vs.h
--- ipvs-0.9.0/ipvs/ip_vs.h     Wed May  9 08:01:14 2001
+++ ipvs-0.9.0.new/ipvs/ip_vs.h Fri May 11 14:54:35 2001
@@ -69,13 +69,13 @@
 #define IP_VS_SO_SET_NONE      IP_VS_BASE_CTL          /* just peek */
 #define IP_VS_SO_SET_INSERT    (IP_VS_BASE_CTL+1)
 #define IP_VS_SO_SET_ADD       (IP_VS_BASE_CTL+2)
-#define IP_VS_SO_SET_EDIT      (IP_VS_BASE_CTL+3)
+#define IP_VS_SO_SET_EDIT      (IP_VS_BASE_CTL+3)      /* Depreciated */
 #define IP_VS_SO_SET_DEL       (IP_VS_BASE_CTL+4)
 #define IP_VS_SO_SET_FLUSH     (IP_VS_BASE_CTL+5)
 #define IP_VS_SO_SET_LIST      (IP_VS_BASE_CTL+6)
 #define IP_VS_SO_SET_ADDDEST   (IP_VS_BASE_CTL+7)
 #define IP_VS_SO_SET_DELDEST   (IP_VS_BASE_CTL+8)
```

```
-#define IP_VS_SO_SET_EDITDEST  (IP_VS_BASE_CTL+9)
+#define IP_VS_SO_SET_EDITDEST  (IP_VS_BASE_CTL+9)      /* Depreciated */
 #define IP_VS_SO_SET_TIMEOUTS  (IP_VS_BASE_CTL+10)
 #define IP_VS_SO_SET_MAX       IP_VS_SO_SET_TIMEOUTS

diff -ruN ipvs-0.9.0/ipvs/ip_vs_ctl.c ipvs-0.9.0.new/ipvs/ip_vs_ctl.c
--- ipvs-0.9.0/ipvs/ip_vs_ctl.c Wed May  9 08:01:15 2001
+++ ipvs-0.9.0.new/ipvs/ip_vs_ctl.c     Fri May 11 14:46:26 2001
@@ -739,32 +739,21 @@


 /*
- *  Add a destination into an existing service
+ * Actually add a new destination to a service
+ * Note: You should call ip_vs_add_dest() which will add or delete
+ * a destination as appropriate.
  */
-static int ip_vs_add_dest(struct ip_vs_service *svc,
+
+static int __ip_vs_add_dest(struct ip_vs_service *svc,
                          struct ip_vs_rule_user *ur)
 {
-       struct ip_vs_dest *dest;
        __u32 daddr = ur->daddr;
        __u16 dport = ur->dport;
        int ret;
+       struct ip_vs_dest *dest;

        EnterFunction(2);

-       if (ur->weight < 0) {
-               IP_VS_ERR("ip_vs_add_dest(): server weight less than zero\n");
-               return -ERANGE;
-
-       /*
-        * Check if the dest already exists in the list
-        */
-       dest = ip_vs_lookup_dest(svc, daddr, dport);
-       if (dest != NULL) {
-               IP_VS_DBG(1, "ip_vs_add_dest(): dest already exists\n");
-               return -EEXIST;
-       }
-
        /*
         * Check if the dest already exists in the trash and
         * is from the same service
@@ -838,10 +827,12 @@


 /*
- *  Edit a destination in the given service
+ * Edit a destination in the given service
+ * Note: You should call ip_vs_add_dest() which will add or delete
+ * a destination as appropriate.
  */
```

```
-static int ip_vs_edit_dest(struct ip_vs_service *svc,
-                          struct ip_vs_rule_user *ur)
+static int __ip_vs_edit_dest(struct ip_vs_service *svc,
+                          struct ip_vs_rule_user *ur)
 {
        struct ip_vs_dest *dest;
        __u32 daddr = ur->daddr;
@@ -850,11 +841,6 @@

        EnterFunction(2);

-       if (ur->weight < 0) {
-               IP_VS_ERR("ip_vs_add_dest(): server weight less than zero\n");
-               return -ERANGE;
-       }
-
        /*
         *  Lookup the destination list
         */
@@ -873,6 +859,42 @@


 /*
+ *  Add a destination into an existing service
+ *  If the destination alrady exists it will be edited
+ *  using __ip_vs_edit_dest()
+ *  Else add the destination using __ip_vs_add_dest()
+ */
+static int ip_vs_add_dest(struct ip_vs_service *svc,
+                         struct ip_vs_rule_user *ur)
+{
+       __u32 daddr = ur->daddr;
+       __u16 dport = ur->dport;
+       struct ip_vs_dest *dest;
+
+       EnterFunction(2);
+
+       if (ur->weight < 0) {
+               IP_VS_ERR("ip_vs_add_dest(): server weight less than zero\n");
+               return -ERANGE;
+       }
+
+       /*
+        * Check if the dest already exists in the list
+        * If it does, edit the existing entry
+        * Else add a new one
+        */
+
+       dest = ip_vs_lookup_dest(svc, daddr, dport);
+       if (dest != NULL)
+               return __ip_vs_edit_dest(svc, ur);
+       else
+               return __ip_vs_add_dest(svc, ur);
+
+       LeaveFunction(2);
+}
+
+
+/*
  *  Delete a destination (must be already unlinked from the service)
  */

@@ -1788,14 +1810,11 @@

        switch (cmd) {
        case IP_VS_SO_SET_ADD:
-               if (svc != NULL)
-                       ret = -EEXIST;
-               else
-                       ret = ip_vs_add_service(urule, &svc);
-               break;
        case IP_VS_SO_SET_EDIT:
-               if (svc == NULL || svc->protocol != urule->protocol)
-                       ret = -ESRCH;
+               if (svc != NULL && svc->protocol != urule->protocol)
+                       svc = NULL;
+               if (svc == NULL)
+                       ret = ip_vs_add_service(urule, &svc);
                else
                        ret = ip_vs_edit_service(svc, urule);
                break;
@@ -1809,16 +1828,11 @@
                }
                break;
        case IP_VS_SO_SET_ADDDEST:
-               if (svc == NULL || svc->protocol != urule->protocol)
-                       ret = -ESRCH;
-               else
-                       ret = ip_vs_add_dest(svc, urule);
-               break;
        case IP_VS_SO_SET_EDITDEST:
                if (svc == NULL || svc->protocol != urule->protocol)
                        ret = -ESRCH;
                else
-                       ret = ip_vs_edit_dest(svc, urule);
+                       ret = ip_vs_add_dest(svc, urule);
                break;
        case IP_VS_SO_SET_DELDEST:
                if (svc == NULL || svc->protocol != urule->protocol)
diff -ruN ipvs-0.9.0/ipvs/ipvsadm/VERSION ipvs-0.9.0.new/ipvs/ipvsadm/VERSION
--- ipvs-0.9.0/ipvs/ipvsadm/VERSION     Thu Mar 22 04:57:46 2001
+++ ipvs-0.9.0.new/ipvs/ipvsadm/VERSION Sat May 12 11:38:19 2001
@@ -1 +1 @@
-1.17
+1.18
diff -ruN ipvs-0.9.0/ipvs/ipvsadm/debian/changelog ipvs-0.9.0.new/ipvs/ipvsadm/debian/changelog
--- ipvs-0.9.0/ipvs/ipvsadm/debian/changelog    Thu Jan 11 06:14:59 2001
+++ ipvs-0.9.0.new/ipvs/ipvsadm/debian/changelog        Sat May 12 11:39:08 2001
@@ -1,8 +1,8 @@
-ipvsadm (1.13-1) nstable; urgency=low
```

```
+ipvsadm (1.18-1) nstable; urgency=low

   * A release

- -- Horms <horms@vergenet.net>  Thu, 14 Dec 2000 17:00:00 -0800
+ -- Simon Horman <horms@vergenet.net>  Sat, 12 May 2001 11:39:05 -0700

 Local variables:
 mode: debian-changelog
diff -ruN ipvs-0.9.0/ipvs/ipvsadm/ipvsadm.8 ipvs-0.9.0.new/ipvs/ipvsadm/ipvsadm.8
--- ipvs-0.9.0/ipvs/ipvsadm/ipvsadm.8   Thu Mar 22 04:57:46 2001
+++ ipvs-0.9.0.new/ipvs/ipvsadm/ipvsadm.8       Sat May 12 11:41:31 2001
@@ -14,6 +14,8 @@
 .\"       Horms            :  Tidy up some of the description and the
 .\"                           grammar in the -f and sysctl sections
 .\"       Wensong Zhang    :  -s option description taken from ipchains(8)
+.\"       Horms            :  Depreciated Edit options, they are now
+.\"                           handled by add.
 .\"
 .\"     This program is free software; you can redistribute it and/or modify
 .\"     it under the terms of the GNU General Public License as published by
@@ -30,12 +32,12 @@
 .\"     Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 .\"
 .\"
-.TH IPVSADM 8 "11th January 2001" "LVS Administration" "Linux Administrator's Guide"
+.TH IPVSADM 8 "11th May 2001" "LVS Administration" "Linux Administrator's Guide"
 .UC 4
 .SH NAME
 ipvsadm \- Linux Virtual Server administration
 .SH SYNOPSIS
-.B ipvsadm -A|E -t|u|f \fIservice-address\fP [-s \fIscheduler\fP]
+.B ipvsadm -A -t|u|f \fIservice-address\fP [-s \fIscheduler\fP]
 .ti 15
 .B [-p [\fItimeout\fP]] [-M \fInetmask\fP]
 .br
@@ -47,7 +49,7 @@
 .br
 .B ipvsadm -S [-n]
 .br
-.B ipvsadm -a|e -t|u|f \fIservice-address\fP
+.B ipvsadm -a -t|u|f \fIservice-address\fP
 .ti 15
 .B -r|R \fIserver-address\fP [-g|i|m] [-w \fIweight\fP]
 .br
@@ -99,10 +101,8 @@
 .B -A, --add-service
 Add a virtual service. A service address is uniquely defined by a
 triplet: IP address, port number, and protocol. Alternatively, a
-virtual service may be defined by a firewall-mark.
-.TP
-.B -E, --edit-service
-Edit a virtual service.
+virtual service may be defined by a firewall-mark. If the service
+already exists then it will be modified according to the values given.
```

```
 .TP
 .B -D, --delete-service
 Delete a virtual service, along with any associated real servers.
@@ -127,10 +127,8 @@
 This option only works if \fIipvsadm\fP is compiled against \fBpopt\fR(3).
 .TP
 .B -a, --add-server
-Add a real server to a virtual service.
-.TP
-.B -e, --edit-server
-Edit a real server in a virtual service.
+Add a real server to a virtual service. If the real server already exists
+then it will be modified according to the values given.
 .TP
 .B -d, --delete-server
 Remove a real server from a virtual service.
diff -ruN ipvs-0.9.0/ipvs/ipvsadm/ipvsadm.c ipvs-0.9.0.new/ipvs/ipvsadm/ipvsadm.c
--- ipvs-0.9.0/ipvs/ipvsadm/ipvsadm.c   Wed Apr 11 19:59:31 2001
+++ ipvs-0.9.0.new/ipvs/ipvsadm/ipvsadm.c       Sat May 12 11:41:56 2001
@@ -50,6 +50,9 @@
  *                             in an informative error message rather
  *                             than the usage information
  *        Horms               :   added -v option
+ *        Horms               :   Merged funtionality of add and edit options
+ *                                Depreciated edit option
+ *
  *
  *
  *      ippfvsadm - Port Fowarding & Virtual Server ADMinistration program
@@ -119,7 +122,7 @@
 #endif

 #define IPVSADM_VERSION_NO             "v" VERSION
-#define IPVSADM_VERSION_DATE           "2001/03/18"
+#define IPVSADM_VERSION_DATE           "2001/05/12"
 #define IPVSADM_VERSION        IPVSADM_VERSION_NO " " IPVSADM_VERSION_DATE

 #define MINIMUM_IPVS_VERSION_MAJOR      0
@@ -218,7 +221,7 @@
        struct poptOption add_service_option =
        {"add-service", 'A', POPT_ARG_NONE, NULL, 'A'};
        struct poptOption edit_service_option =
-       {"edit-service", 'E', POPT_ARG_NONE, NULL, 'E'};
+       {"edit-service", 'E', POPT_ARG_NONE, NULL, 'E'};     /* Depreciated */
        struct poptOption delete_service_option =
        {"delete-service", 'D', POPT_ARG_NONE, NULL, 'D'};
        struct poptOption clear_option =
@@ -229,8 +232,8 @@
        {"list", 'l', POPT_ARG_NONE, NULL, 'l'};
        struct poptOption add_server_option =
        {"add-server", 'a', POPT_ARG_NONE, NULL, 'a'};
-       struct poptOption edit_server_option =
-       {"edit-server", 'e', POPT_ARG_NONE, NULL, 'e'};
+       struct poptOption edit_server_option =
+       {"edit-server", 'e', POPT_ARG_NONE, NULL, 'e'};      /* Depreciated */
```

```
         struct poptOption delete_server_option =
         {"delete-server", 'd', POPT_ARG_NONE, NULL, 'd'};
         struct poptOption set_option =
@@ -359,11 +362,8 @@

         switch (cmd) {
         case 'A':
-                *op = IP_VS_SO_SET_ADD;
-               options_sub = options_service;
-                break;
         case 'E':
-                *op = IP_VS_SO_SET_EDIT;
+                *op = IP_VS_SO_SET_ADD;
                options_sub = options_service;
                 break;
         case 'D':
@@ -371,11 +371,8 @@
                options_sub = options_delete_service;
                 break;
         case 'a':
-                *op = IP_VS_SO_SET_ADDDEST;
-               options_sub = options_server;
-                break;
         case 'e':
-                *op = IP_VS_SO_SET_EDITDEST;
+                *op = IP_VS_SO_SET_ADDDEST;
                options_sub = options_server;
                 break;
         case 'd':
@@ -621,12 +618,12 @@
        struct option long_options[] =
        {
                {"add-service", 0, 0, 'A'},
-               {"edit-service", 0, 0, 'E'},
+               {"edit-service", 0, 0, 'E'},                /* Depreciated */
                {"delete-service", 0, 0, 'D'},
                {"clear", 0, 0, 'C'},
                {"list", 0, 0, 'L'},
                {"add-server", 0, 0, 'a'},
-               {"edit-server", 0, 0, 'e'},
+               {"edit-server", 0, 0, 'e'},                 /* Depreciated */
                {"delete-server", 0, 0, 'd'},
                {"help", 0, 0, 'h'},
                {"version", 0, 0, 'v'},
@@ -664,11 +661,8 @@

         switch (cmd) {
         case 'A':
-                *op = IP_VS_SO_SET_ADD;
-                optstr = "t:u:f:s:M:p::";
-                break;
         case 'E':
-                *op = IP_VS_SO_SET_EDIT;
+                *op = IP_VS_SO_SET_ADD;
                 optstr = "t:u:f:s:M:p::";
                break;
        case 'D':
@@ -676,11 +670,8 @@
                optstr = "t:u:f:";
                break;
        case 'a':
-               *op = IP_VS_SO_SET_ADDDEST;
-               optstr = "t:u:f:w:r:R:gmi";
-               break;
        case 'e':
-               *op = IP_VS_SO_SET_EDITDEST;
+               *op = IP_VS_SO_SET_ADDDEST;
                optstr = "t:u:f:w:r:R:gmi";
                break;
        case 'd':
@@ -886,7 +877,7 @@
               return 0;
       }

-       if (op == IP_VS_SO_SET_ADD || op == IP_VS_SO_SET_EDIT) {
+       if (op == IP_VS_SO_SET_ADD) {
                /*
                 * Make sure that port zero service is persistent
                 */
@@ -907,13 +898,12 @@
         * i.e. make sure that a -r accompanies a -[t|u|f]
         */
        if ((op == IP_VS_SO_SET_ADDDEST
-            || op == IP_VS_SO_SET_EDITDEST
             || op == IP_VS_SO_SET_DELDEST)
            && !urule.daddr) {
                fail(2, "No destination specified");
        }

-       if (op == IP_VS_SO_SET_ADDDEST || op == IP_VS_SO_SET_EDITDEST) {
+       if (op == IP_VS_SO_SET_ADDDEST) {
                /*
                 * Set the default weight 1 if not specified
                 */
@@ -947,16 +937,7 @@
                 */
                switch (op) {
                case IP_VS_SO_SET_ADD:
-                       if (errno == EEXIST)
-                               printf("Service already exists\n");
-                       else if (errno == ENOENT)
-                               printf("Scheduler not found: ip_vs_%s.o\n",
-                                      urule.sched_name);
-                       break;
-               case IP_VS_SO_SET_EDIT:
-                       if (errno==ESRCH)
-                               printf("No such service\n");
-                       else if (errno == ENOENT)
+                       if (errno == ENOENT)
                                printf("Scheduler not found: ip_vs_%s.o\n",
                                              urule.sched_name);
                         break;
@@ -967,10 +948,6 @@
                 case IP_VS_SO_SET_ADDDEST:
                         if (errno == ESRCH)
                                 printf("Service not defined\n");
-                        else if (errno == EEXIST)
-                                printf("Destination already exists\n");
-                        break;
-                case IP_VS_SO_SET_EDITDEST:
                 case IP_VS_SO_SET_DELDEST:
                         if (errno==ESRCH)
                                 printf("Service not defined\n");
@@ -1124,14 +1101,14 @@
        version(stream);
         fprintf(stream,
                 "Usage:\n"
-                "  %s -A|E -t|u|f service-address [-s scheduler] [-p [timeout]] [-M netmask]\n"
+                "  %s -A -t|u|f service-address [-s scheduler] [-p [timeout]] [-M netmask]\n"
                 "  %s -D -t|u|f service-address\n"
                 "  %s -C\n"
 #ifdef HAVE_POPT
                 "  %s -R\n"
                 "  %s -S [-n]\n"
 #endif
-                "  %s -a|e -t|u|f service-address -r|R server-address [-g|i|m] [-w weight]\n"
+                "  %s -a -t|u|f service-address -r|R server-address [-g|i|m] [-w weight]\n"
                 "  %s -d -t|u|f service-address -r|R server-address\n"
                 "  %s -L|l [-c] [-n]\n"
                 "  %s -s tcp tcpfin udp\n"
@@ -1145,16 +1122,14 @@
         fprintf(stream,
                 "Commands:\n"
                 "Either long or short options are allowed.\n"
-                "  --add-service     -A       add virtual service with options\n"
-                "  --edit-service    -E       edit virtual service with options\n"
+                "  --add-service     -A       add/edit virtual service with options\n"
                 "  --delete-service  -D       delete virtual service\n"
                 "  --clear           -C       clear the whole table\n"
 #ifdef HAVE_POPT
                 "  --restore         -R       restore rules from stdin\n"
                 "  --save            -S       save rules to stdout\n"
 #endif
-                "  --add-server      -a       add real server with options\n"
-                "  --edit-server     -e       edit real server with options\n"
+                "  --add-server      -a       add/edit real server with options\n"
                 "  --delete-server   -d       delete real server\n"
                 "  --list            -L|-l    list the table\n"
                 "  --set|-s tcp tcpfin udp    set connection timeout values\n"
```

29.3 Threshhold patch

ratz `ratz@tac.ch` 2001-01-29 16:16:27

This patch on top of ipvs-1.0.3-2.2.18 adds support for threshhold settings per realserver for all schedulers that have the -w option.

29.3.1 Description/Purpose

I was always thinking of how a kernel based implementation of connection limitation per real server would work and how it could be implemented so while waiting in the hospital for the x-ray I had enough time to write up some little dirty hack to show a proof of concept. It works like follows. I added three new entries to the ip_vs_dest() struct, u_thresh and l_thresh in ip_vs.* and I modified the ipvsadm to add the two new options x and y. A typical setup would be:

```
ipvsadm -A -t 192.168.100.100:80 -s wlc
ipvsadm -a -t 192.168.100.100:80 -r 192.168.100.3:80 -w 3 -x 1145 -y 923
ipvsadm -a -t 192.168.100.100:80 -r 192.168.100.3:80 -w 2 -x 982 -y 677
ipvsadm -a -t 192.168.100.100:80 -r 127.0.0.1:80 -w 1 -x 100 -y 50
```

So, this means, as soon as (dest->inactconns + dest->activeconns) exceed the x value the weight of this server is set to zero. As soon as the connections drop below the lower threshhold (y) the weight is set back to the initial value. What is it good for? Yeah well, I don't know exactly, imagine yourself, but first of all this is proposal and I wanted to ask for a discussion about a possible inclusion of such a feature or even a derived one into the main code (of course after fixing the race conditions and bugs and cleaning up the code) and second, I found out with tons of talks with customers that such a feature is needed, because also commercial lb have this and managers always like to have a nice comparision of all features to decide which product they take. Doing all this in user- space is unfortunately just not atomic enough.

Anyway, if anybody else thinks that such a feature might be vital for inclusion we can talk about it. If you look at the code, it wouldn't break anything and just add two lousy CPU cycles for checking if u_thresh is < 0. This feature can easily be disabled by just setting u_thresh to zero or not even initialize it.

Well, I'm open for discussion and flames. I have it running in production :) but with a special SLA. I implemented the last server of resort which works like this: If all RS of a service are down (healthcheck took it out or treshhold check set weight to zero), my userspace tool automagically invokes the last server of resort, a tiny httpd with a static page saying that the service is currently unavailable. This is also useful if you want to do maintainance of the realservers.

I already implemented a dozen of such setups and they work all pretty well.

```
--- ipvsadm.c-old       Mon Jan 29 08:39:45 2001
+++ ipvsadm.c   Mon Jan 29 08:56:24 2001
@@ -265,6 +265,10 @@
         {"gatewaying",'g', POPT_ARG_NONE, NULL, 'g'};
         struct poptOption weight_option =
         {"weight", 'w', POPT_ARG_STRING, &optarg, 'w'};
+        struct poptOption u_thresh_option =
+        {"u_thresh", 'x', POPT_ARG_STRING, &optarg, 'x'};
+        struct poptOption l_thresh_option =
+        {"l_thresh", 'y', POPT_ARG_STRING, &optarg, 'y'};
         struct poptOption numeric_option =
         {"numeric", 'n', POPT_ARG_NONE, NULL, 'n'};
         struct poptOption NULL_option =
@@ -326,6 +330,8 @@
                 udp_service_option,
                 fwmark_service_option,
                 weight_option,
+                u_thresh_option,
+               l_thresh_option,
                real_server_option,
                real_server2_option,
                gatewaying_option,
@@ -517,6 +523,17 @@
                              string_to_number(optarg,0,65535)) == -1)
                                 fail(2, "illegal weight specified");
                         break;
+               case 'x':
+                       if ((mc->u.vs_user.u_thresh=
+                            string_to_number(optarg, 0, 65535)) == -1)
+                               fail(2, "illegal u_thresh specified");
+                       break;
+               case 'y':
+                       if ((mc->u.vs_user.l_thresh=
+                            string_to_number(optarg, 0, 65535)) == -1)
+                               fail(2, "illegal l_thresh specified");
+                       break;
+
                 case 'n':
                         *format |= FMT_NUMERIC;
                         break;
@@ -611,6 +628,8 @@
                {"ipip", 0, 0, 'i'},
                {"gatewaying", 0, 0, 'g'},
                {"weight", 1, 0, 'w'},
+               {"u_thresh", 1, 0, 'x'},
+               {"l_thresh", 1, 0, 'y'},
                {"numeric", 0, 0, 'n'},
                {"help", 0, 0, 'h'},
                {0, 0, 0, 0}
@@ -624,7 +643,7 @@
        /* Re-process the arguments each time options is called*/
        optind = 1;

-       if ((cmd = getopt_long(argc, argv, "AEDCSRaedlLhv",
+       if ((cmd = getopt_long(argc, argv, "AEDCSRaedlLhvxy",
                               long_options, NULL)) == EOF)
                usage_exit(argv[0], -1);

@@ -643,11 +662,11 @@
                 break;
         case 'a':
                 mc->m_cmd = IP_MASQ_CMD_ADD_DEST;
-                optstr = "t:u:f:w:r:R:gmi";
+                optstr = "t:u:f:w:r:R:gmi:x:y";
                 break;
         case 'e':
                 mc->m_cmd = IP_MASQ_CMD_SET_DEST;
-                optstr = "t:u:f:w:r:R:gmi";
+                optstr = "t:u:f:w:r:R:gmi:x:y";
                 break;
         case 'd':
                 mc->m_cmd = IP_MASQ_CMD_DEL_DEST;
@@ -787,6 +806,20 @@
```

```
                            string_to_number(optarg,0,65535)) == -1)
                               fail(2, "illegal weight specified");
                        break;
+               case 'x':
+                       if (mc->u.vs_user.u_thresh != -1)
+                               fail(2, "multiple server u_thresh specified");
+                       if ((mc->u.vs_user.u_thresh=
+                            string_to_number(optarg, 0, 65535)) == -1)
+                               fail(2, "illegal u_thresh specified");
+                       break;
+               case 'y':
+                       if (mc->u.vs_user.l_thresh != -1)
+                               fail(2, "multiple server l_thresh specified");
+                       if ((mc->u.vs_user.l_thresh=
+                            string_to_number(optarg, 0, 65535)) == -1)
+                               fail(2, "illegal l_thresh specified");
+                       break;
                case 'n':
                        *format |= FMT_NUMERIC;
                        break;
@@ -814,6 +847,9 @@
         ctl.m_target = IP_MASQ_TARGET_VS;
         /* weight=0 is allowed, which means that server is quiesced */
         ctl.u.vs_user.weight = -1;
+       /* set u_thresh and l_thresh to zero -> disabled */
+       ctl.u.vs_user.u_thresh = 0;
+       ctl.u.vs_user.l_thresh = 0;
         /* Set direct routing as default forwarding method */
         ctl.u.vs_user.masq_flags = IP_MASQ_F_VS_DROUTE;
         /* Set the default persistent granularity to /32 masking */
@@ -1078,7 +1114,7 @@
                 "  %s -R\n"
                 "  %s -S [-n]\n"
 #endif
-                "  %s -[a|e] -[t|u|f] service-address -[r|R] server-address \
[-g|-i|-m] [-w weight]\n" +                "  %s -[a|e] -[t|u|f] service-address \
                -[r|R] server-address [-g|-i|-m] [-w weight] [-x u_thresh] [-y \
                l_thresh]\n"
                 "  %s -d -[t|u|f] service-address -[r|R] server-address\n"
                 "  %s -[L|l] [-n]\n"
                 "  %s -h\n\n",
@@ -1128,6 +1164,8 @@
         fprintf(stream,
                 "  --ipip        -i                    ipip encapsulation \
                (tunneling)\n"
                 "  --masquerading -m                   masquerading (NAT)\n"
+                "  --u_thresh    -x <u_thresh>         max. connections\n"
+                "  --l_thresh    -y <l_thresh>         weight fallback \
                connections\n"
                 "  --weight      -w <weight>           capacity of real server\n"
                 "  --numeric     -n                    numeric output of addresses \
and ports\n"  );
@@ -1230,7 +1268,7 @@
         }
         if (fgets(buffer, sizeof(buffer), handle) && !(format & FMT_RULE))
```

```
                printf("  -> RemoteAddress:Port           "
-                       "Forward Weight ActiveConn InActConn\n");
+                       "Forward Weight ActiveConn InActConn u_thresh l_thresh\n");

        /*
         * Print the VS information according to the format
@@ -1280,6 +1318,8 @@
        int weight;
        int activeconns;
        int inactconns;
+       unsigned int u_thresh;
+       unsigned int l_thresh;

        int n;
        unsigned long temp;
@@ -1289,11 +1329,11 @@

        if (buf[0] == ' ') {
                /* destination entry */
-               if ((n = sscanf(buf, " %s %lX:%hX %s %d %d %d",
+               if ((n = sscanf(buf, " %s %lX:%hX %s %d %d %d %d %d",
                                arrow, &temp, &dport, fwd, &weight,
-                               &activeconns, &inactconns)) == -1)
+                               &activeconns, &inactconns, &u_thresh, &l_thresh)) == \
-1)  exit(1);
-               if (n != 7)
+               if (n != 9)
                        fail(2, "unexpected input data");

                daddr.s_addr = (__u32) htonl(temp);
@@ -1315,8 +1355,9 @@
                                        dname, get_fwd_switch(fwd), weight);
                        }
                } else {
-                       printf("  -> %-27s %-7s %-6d %-10d %-10d\n",
-                              dname , fwd, weight, activeconns, inactconns);
+                       printf("  -> %-27s %-7s %-6d %-10d %-10d %-10d %-10d\n",
+                              dname , fwd, weight, activeconns, inactconns,
+                              u_thresh, l_thresh);
                }
                free(dname);
        } else if (buf[0] == 'F') {

["per_rs_thresh-kernel_2.2.18.diff" (text/plain)]

Only in linux-2.2.18.vanilla/include/linux: coda_opstats.h
Only in linux-2.2.18.vanilla/include/linux: dasd.h
diff -ur linux-2.2.18.vanilla/include/linux/ip_masq.h \
linux-2.2.18/include/linux/ip_masq.h --- \
linux-2.2.18.vanilla/include/linux/ip_masq.h    Fri Jan 26 22:28:59 2001 +++ \
linux-2.2.18/include/linux/ip_masq.h    Thu Jan 25 08:54:06 2001 @@ -121,6 +121,9 @@
        u_int16_t       dport;
        unsigned        masq_flags;     /* destination flags */
        int             weight;         /* destination weight */
+       int             old_weight;     /* old destination weight */
+       u_int16_t       u_thresh;       /* upper threshold */
+       u_int16_t       l_thresh;       /* lower threshold */
 };


diff -ur linux-2.2.18.vanilla/include/net/ip_vs.h linux-2.2.18/include/net/ip_vs.h
--- linux-2.2.18.vanilla/include/net/ip_vs.h    Fri Jan 26 22:28:59 2001
+++ linux-2.2.18/include/net/ip_vs.h    Sun Jan 28 10:10:28 2001
@@ -110,7 +110,10 @@
         atomic_t               activeconns;    /* active connections */
         atomic_t               inactconns;     /* inactive connections */
         atomic_t               refcnt;         /* reference counter */
+       __u16                   u_thresh;       /* upper threshold */
+       __u16                   l_thresh;       /* lower threshold */
         int                    weight;         /* server weight */
+        int                    old_weight;     /* old server weight */
        struct list_head        d_list;   /* table with all dests */

         /* for virtual service */
@@ -215,6 +218,8 @@
 extern int ip_vs_wrr_init(void);
 extern int ip_vs_lc_init(void);
 extern int ip_vs_wlc_init(void);
+extern int ip_vs_lblc_init(void);
+extern int ip_vs_lblcr_init(void);


 /*
diff -ur linux-2.2.18.vanilla/net/ipv4/ip_vs.c linux-2.2.18/net/ipv4/ip_vs.c
--- linux-2.2.18.vanilla/net/ipv4/ip_vs.c       Fri Jan 26 22:28:59 2001
+++ linux-2.2.18/net/ipv4/ip_vs.c       Sat Jan 27 14:07:42 2001
@@ -69,6 +69,7 @@
  *     Wensong Zhang           :    changed to two service hash tables
  *     Julian Anastasov        :    corrected trash_dest lookup for both
  *                                  normal service and fwmark service
+ *     Roberto Nibali         :    added per realserver threshhold (hospital version)
  *
  */

@@ -1274,6 +1275,8 @@
          *    Set the weight and the flags
          */
         dest->weight = mm->weight;
+       dest->u_thresh = mm->u_thresh;
+       dest->l_thresh = mm->l_thresh;
         dest->masq_flags = mm->masq_flags;

         dest->masq_flags |= IP_MASQ_F_VS;
@@ -1817,9 +1820,21 @@
         ms->dest = dest;

         /*
-         *    Increase the refcnt counter of the dest.
+         *    Increase the refcnt counter of the dest and set the weight
+        *    accordingly. I don't why dest->refcnt is conns+1?
```

```
          */
         atomic_inc(&dest->refcnt);
+       if ( dest->u_thresh != 0) {
+               if (( (atomic_read(&dest->inactconns) + atomic_read(&dest->activeconns)+1) >= \
dest->u_thresh) && (dest->weight > 0)){ +                       IP_VS_DBG(7, "Bind-masq [changing weight]
conns:%d " +                            "weight=%d oldweight=%d\n",
+                               atomic_read(&dest->inactconns) +
+                               atomic_read(&dest->activeconns),
+                               dest->weight, dest->old_weight);
+                       dest->old_weight=dest->weight;
+                       dest->weight=0;
+               }
+       }

         IP_VS_DBG(9, "Bind-masq fwd:%c s:%s c:%u.%u.%u.%u:%d v:%u.%u.%u.%u:%d "
                 "d:%u.%u.%u.%u:%d flg:%X cnt:%d destcnt:%d\n",
@@ -1862,6 +1877,21 @@
                         }
                 }

+               /*
+                  if all connections are smaller then lower threshhold and the
+                  old weight isn't zero.
+               */
+               if (dest->u_thresh != 0) {
+                       if (((atomic_read(&dest->inactconns) + atomic_read(&dest->activeconns)) <= \
dest->l_thresh) && (dest->old_weight > 0)){ +                           IP_VS_DBG(7, "Unbind-masq conns:%d
weight=%d " +                                   "oldweight=%d\n",
+                                   atomic_read(&dest->inactconns) +
+                                   atomic_read(&dest->activeconns),
+                               dest->weight, dest->old_weight);
+                               dest->weight=dest->old_weight;
+                               dest->old_weight=0;
+                       }
+               }
                 /*
                  *  Decrease the refcnt of the dest, and free the dest
                  *  if nobody refers to it (refcnt=0).
@@ -2415,7 +2445,7 @@
         size = sprintf(buf+len,
                        "IP Virtual Server version %d.%d.%d (size=%d)\n"
                        "Prot LocalAddress:Port Scheduler Flags\n"
-                       "  -> RemoteAddress:Port Forward Weight ActiveConn \
InActConn\n", +                        "  -> RemoteAddress:Port Forward Weight \
ActiveConn InActConn u_thresh l_thresh\n",  NVERSION(IP_VS_VERSION_CODE), \
IP_VS_TAB_SIZE);  pos += size;
         len += size;
@@ -2456,13 +2486,15 @@
                                 dest = list_entry(q,struct ip_vs_dest,n_list);
                                 size = sprintf(buf+len,
                                                "  -> %08X:%04X      %-7s "
-                                               "%-6d %-10d %-10d\n",
+                                               "%-6d %-10d %-10d %-10d %-10d\n",
                                                ntohl(dest->addr),
                                                ntohs(dest->port),
                                                 ip_vs_fwd_name(dest->masq_flags),
                                                 dest->weight,
                                                 atomic_read(&dest->activeconns),
-                                                atomic_read(&dest->inactconns));
+                                                atomic_read(&dest->inactconns),
+                                               dest->u_thresh,
+                                               dest->l_thresh);
                                  len += size;
                                  pos += size;

@@ -2505,13 +2537,15 @@
                                  dest = list_entry(q,struct ip_vs_dest,n_list);
                                  size = sprintf(buf+len,
                                                 "  -> %08X:%04X      %-7s "
-                                                "%-6d %-10d %-10d\n",
+                                                "%-6d %-10d %-10d %-10d %-10d\n",
                                                 ntohl(dest->addr),
                                                 ntohs(dest->port),
                                                 ip_vs_fwd_name(dest->masq_flags),
                                                 dest->weight,
                                                 atomic_read(&dest->activeconns),
-                                                atomic_read(&dest->inactconns));
+                                                atomic_read(&dest->inactconns),
+                                               dest->u_thresh,
+                                               dest->l_thresh);
                                  len += size;
                                  pos += size;

@@ -2565,6 +2599,7 @@
                          atomic_read(&ip_vs_concurrentconns),
                          atomic_read(&ip_vs_connshandled),
                          atomic_read(&ip_vs_packetshandled));
+       /* Here we should add a per svc and per rs statistics */
         pos += size;
         len += size;
```

How we will defend against DDoS (distributed DoS)?

> I'm using a packetfilter and in special zones a firewall after the packetfilter ;) No seriously, I personally don't think the LVS should take too much part on securing the realservers It's just another part of the firewall setup.

The problem is that LVS has another view for the real server load. The director sees one number of connections the real server sees another one. And under attack we observe big gap between the active/inactive counters and the used threshold values. In this case we just exclude all real servers. This is the reason I prefer the more informed approach of using agents.

Using the number of active or inactive connections to assign a new weight is _very_ dangerous.

> I know, but OTOH, if you set a threshhold and my code takes the server out, because of a well formated DDoS attack, I think it is even better than if you would allow the DDoS and maybe kill the realservers http-listener.
>
> > No, we have two choices: - use SYN cookies and much memory for open requests, accept more valid requests

> > - don't use SYN cookies, drop the requests exceeding the backlog length, drop many valid requests but the real servers are not overloaded
> >
> > In both cases the listeners don't see requests until the handshake is completed (Linux).
>
> BTW, what if you enable the defense strategies of the loadbalancer? I've done some tests and I was able to flood the realservers by sending forged SYNs and timeshifted SYN-ACKs with the expected seq-nr. It was impossible to work on the realservers unless of course I enabled the TCP_SYNCOOKIES.

Yes, nobody claims the defense strategies guard the real servers. This is not their goal. They keep the director with more free memory and nothing more :) Only drop_packet can control the request rate but only for the new requests.

> I then enabled my patch and after the connections exceeded the threshhold, the kernel took the server out temporarily by setting the weight to 0. In that way the server was usable and I could work on the server.

Yes but the clients can't work, you exclude all servers in this case because the LVS spreads the requests to all servers and the rain becomes deluge :)

In theory, the number of connections is related to the load but this is true when the world is ideal. The inactive counter can be set with very high values when we are under attack. Even the WLC method loads proportionatly the real servers but they are never excluded from operation.

> True, but as I already said. I think LVS shouldn't replace a fw. I normally have a router configured properly, then a packetfilter, then a firewall or even another but stateful packetfilter. See, the patch itself is not even mandatory. I normal setup, my code is not even touched (except the "if":).
>
> > I have some thoughts about limiting the traffic per connection but this idea must be analyzed.
>
> Hmm, I just want to limit the amount of concurrent connections per realserver and in the future maybe per service. This saved me quite some lines of code in my userspace healthchecking daemon.

Yes, you vote for moving some features from user to the kernel space. We must find the right balance: what can be done in LVS and what must be implemented in the user space tools.

The other alternatives are to use the Netfilter's "limit" target or QoS to limit the traffic to the real servers.

> But then you have to add quite some code. The limit target has no idea about LVS tables. How should this work, f.e. if you would like to rate limit the amount of connections to a realserver?

May be we can limit the SYN rate. Of course, that not covers all cases, so my thought was to limit the packet rate for all states or per connection, not sure, this is an open topic. It is easy to open a connection through the director (especially in LVS-DR) and then to flood with packets this connection. This is one of the cases where LVS can really guard the real servers from packet floods. If we combine this with the other kind of attacks, the distributed ones, we have better control. Of course, some QoS implementations can cover such problems, not sure. And this can be a simple implementation, of course, nobody wants to invent the wheel :)

Let's analyze the problem. If we move new connections from "overloaded" real server and redirect them to the other real servers we will overload them too.

> No, unless you use a old machine. This is maybe a requirement of an e-commerce application. They have some servers and if the servers are overloaded (taken out by my user-space healthchecking daemon because the response time it to high or the application daemon is not listening anymore on the port) they will be taken out. Now I have found out that by setting threshholds I could reduce the down- time of flooded server significantly. In case all servers were taken out or their weights were set to 0 the userspace application sets up a temporarily (either local route or another server) new realserver that has nothing else to do then pushing a static webpage saying that the service is currently unavailable due to high server load or DDoS attack or whatever. Put this page behind a TUX 2.0 and try to overflow it. If you can, apply the zero-copy patches of DaveM. No way you will find such a fast (88MBit/s requests!!) Link to saturate the server.

Yes, I know that this is a working solution. But see, you exclude all real servers :) You are giving up. My idea is we to find a state when we can drop some of the requests and to keep the real servers busy but responsive. This can be a difficult task but not when we have the help from our agents. We expect that many valid requests can be dropped but if we keep the real server in good health we can handle some valid requests because nobody knows when the flood will stop. The link is busy but it contains valid requests. And the service does not see the invalid ones.

IMO, the problem is that there are more connection requests than the cluster can handle. The solutions to try to move the traffic between the real servers can only cause more problems. If at the same time we set the weights to 0 this leads to more delay in the processing. May be more useful is to start to reduce the weights first but this again returns us to the theory for the smart cluster software.

So, we can't exit from this situation without dropping requests. There is more traffic that can't be served from the cluster.

The requests are not meaningful, we care how much load they introduce and we report this load to the director. It can look, for example, as one value (weight) for the real host that can be set for all real services running on this host. We don't need to generate 10 weights for the 10 real services running in our real host. And we change the weight on each 2 seconds for example. We need two syscalls (lseek and read) to get most of the values from /proc fs. But may be from 2-3 files. This is in Linux, of course. Not sure how this behaves under attack. We will see it :)

> Obviously yes, but if you also include the practical problem of SLA with customers and guaranteed downtime per month I still have to say that for my deploition (is this the correct noun?) I go better with my patch in case of a DDoS and enabled LVS defense strategies then without.

If there is no cluster software to keep the real servers equally loaded, some of them can go offline too early.

> The scheduler should keep them equally loaded IMO even in case of let's say 70% forged packets. Again, if you don't like to set a threshold, leave it. The patch is open enough. If you like to set it, set it, maybe set it very high. It's up to you.

The only problem we have with this scheme is the ipvsadm binary. It must be changed (the user structure in the kernel :)) The last change is dated from 0.9.10 and this is a big period :) But you know what means a change in the user structures :)

The cluster software can take the role to monitor the load instead of relying on the connection counters. I agree, changing the weights and deciding how much traffic to drop can be explained with a complex formula. But I can see it only as a complete solution: to balance the load and to drop the exceeding requests, serve as many requests as possible. Even the drop_packet strategy can help here, we can explicitly enable it specifying the proper drop rate. We don't need to use it only to defend the LVS box but to drop the exceeding traffic. But someone have to control the drop rate :) If there is no exceeding traffic what problems we can expect? Only from the bad load balancing :)

The easiest way to control the LVS is from user space and to leave in LVS only the basic needed support. This allows us to have more ways to control LVS.

29.4 Martian modification patchs

29.4.1 Martian modification patch for 2.2.x

```
--- linux/include/net/ip_fib.h.orig     Wed Feb 23 16:54:27 2000
+++ linux/include/net/ip_fib.h  Wed Mar 15 13:46:22 2000
@@ -200,7 +200,7 @@
 extern int inet_rtm_getroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg);
 extern int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb);
 extern int fib_validate_source(u32 src, u32 dst, u8 tos, int oif,
-                              struct device *dev, u32 *spec_dst, u32 *itag);
+                       struct device *dev, u32 *spec_dst, u32 *itag, int our);
 extern void fib_select_multipath(const struct rt_key *key, struct fib_result *res);

 /* Exported by fib_semantics.c */
--- linux/net/ipv4/fib_frontend.c.orig  Wed Feb 23 16:54:27 2000
+++ linux/net/ipv4/fib_frontend.c       Wed Mar 15 14:44:45 2000
@@ -189,7 +189,7 @@
  */

 int fib_validate_source(u32 src, u32 dst, u8 tos, int oif,
-                       struct device *dev, u32 *spec_dst, u32 *itag)
+                       struct device *dev, u32 *spec_dst, u32 *itag, int our)
 {
        struct in_device *in_dev = dev->ip_ptr;
        struct rt_key key;
@@ -206,7 +206,8 @@
                return -EINVAL;
        if (fib_lookup(&key, &res))
                goto last_resort;
-       if (res.type != RTN_UNICAST)
+       if ((res.type != RTN_UNICAST) &&
+               ((res.type != RTN_LOCAL) || our))
                return -EINVAL;
        *spec_dst = FIB_RES_PREFSRC(res);
        if (itag)
@@ -216,13 +217,20 @@
 #else
        if (FIB_RES_DEV(res) == dev)
 #endif
+       {
+               if (res.type == RTN_LOCAL) {
+                       *itag = 0;
+                       return -EINVAL;
+               }
                return FIB_RES_NH(res).nh_scope >= RT_SCOPE_HOST;
+       }
```

```
        if (in_dev->ifa_list == NULL)
                goto last_resort;
        if (IN_DEV_RPFILTER(in_dev))
                return -EINVAL;
        key.oif = dev->ifindex;
+       if (res.type == RTN_LOCAL) key.iif = loopback_dev.ifindex;
        if (fib_lookup(&key, &res) == 0 && res.type == RTN_UNICAST) {
                *spec_dst = FIB_RES_PREFSRC(res);
                return FIB_RES_NH(res).nh_scope >= RT_SCOPE_HOST;
--- linux/net/ipv4/route.c.orig Wed Feb 23 17:00:07 2000
+++ linux/net/ipv4/route.c      Wed Mar 15 13:07:28 2000
@@ -1037,7 +1037,7 @@
                if (!LOCAL_MCAST(daddr))
                        return -EINVAL;
                spec_dst = inet_select_addr(dev, 0, RT_SCOPE_LINK);
-       } else if (fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag) < 0)
+       } else if (fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag, our) < 0)
                return -EINVAL;

        rth = dst_alloc(sizeof(struct rtable), &ipv4_dst_ops);
@@ -1181,7 +1181,7 @@
        if (res.type == RTN_LOCAL) {
                int result;
                result = fib_validate_source(saddr, daddr, tos, loopback_dev.ifindex,
-                                            dev, &spec_dst, &itag);
+                                            dev, &spec_dst, &itag, 1);
                if (result < 0)
                        goto martian_source;
                if (result)
@@ -1206,7 +1206,7 @@
                return -EINVAL;
        }

-       err = fib_validate_source(saddr, daddr, tos, FIB_RES_OIF(res), dev, &spec_dst, &itag);
+       err = fib_validate_source(saddr, daddr, tos, FIB_RES_OIF(res), dev, &spec_dst, &itag, 0);
        if (err < 0)
                goto martian_source;

@@ -1279,7 +1279,7 @@
        if (ZERONET(saddr)) {
                spec_dst = inet_select_addr(dev, 0, RT_SCOPE_LINK);
        } else {
-               err = fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag);
+               err = fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag, 1);
                if (err < 0)
                        goto martian_source;
                if (err)
```

29.4.2 Martian modification patch for 2.4.0

```
--- linux-2.4.0/include/net/ip_fib.h~   Sat Jan 20 13:16:50 2001
+++ linux/include/net/ip_fib.h  Sun Mar 11 11:07:22 2001
@@ -203,7 +203,7 @@
 extern int inet_rtm_getroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg);
 extern int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb);
 extern int fib_validate_source(u32 src, u32 dst, u8 tos, int oif,
-                              struct net_device *dev, u32 *spec_dst, u32 *itag);
+                              struct net_device *dev, u32 *spec_dst, u32 *itag, int our);
 extern void fib_select_multipath(const struct rt_key *key, struct fib_result *res);

 /* Exported by fib_semantics.c */
--- linux-2.4.0/net/ipv4/fib_frontend.c~        Fri Jun  2 07:25:21 2000
+++ linux/net/ipv4/fib_frontend.c       Sun Mar 11 11:07:22 2001
@@ -204,7 +204,8 @@
  */

 int fib_validate_source(u32 src, u32 dst, u8 tos, int oif,
-                       struct net_device *dev, u32 *spec_dst, u32 *itag)
+                       struct net_device *dev, u32 *spec_dst, u32 *itag,
+                       int our)
 {
        struct in_device *in_dev;
        struct rt_key key;
@@ -233,7 +234,8 @@

        if (fib_lookup(&key, &res))
                goto last_resort;
-       if (res.type != RTN_UNICAST)
+       if ((res.type != RTN_UNICAST) &&
+               ((res.type != RTN_LOCAL) || our))
                goto e_inval_res;
        *spec_dst = FIB_RES_PREFSRC(res);
        if (itag)
@@ -244,6 +246,10 @@
        if (FIB_RES_DEV(res) == dev)
 #endif
        {
+               if (res.type == RTN_LOCAL) {
+                       *itag = 0;
+                       goto e_inval_res;
+               }
                ret = FIB_RES_NH(res).nh_scope >= RT_SCOPE_HOST;
                fib_res_put(&res);
                return ret;
@@ -254,6 +260,7 @@
        if (rpf)
                goto e_inval;
        key.oif = dev->ifindex;
+       if (res.type == RTN_LOCAL) key.iif = loopback_dev.ifindex;

        ret = 0;
        if (fib_lookup(&key, &res) == 0) {
--- linux-2.4.0/net/ipv4/route.c~       Sun Nov  5 23:10:48 2000
+++ linux/net/ipv4/route.c      Sun Mar 11 11:07:22 2001
@@ -1177,7 +1177,7 @@
                if (!LOCAL_MCAST(daddr))
                        goto e_inval;
                spec_dst = inet_select_addr(dev, 0, RT_SCOPE_LINK);
-       } else if (fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag) < 0)
```

```
+       } else if (fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag, our) < 0)
                goto e_inval;

        rth = dst_alloc(&ipv4_dst_ops);
@@ -1339,7 +1339,7 @@
        if (res.type == RTN_LOCAL) {
                int result;
                result = fib_validate_source(saddr, daddr, tos, loopback_dev.ifindex,
-                                            dev, &spec_dst, &itag);
+                                            dev, &spec_dst, &itag, 1);
                if (result < 0)
                        goto martian_source;
                if (result)
@@ -1364,7 +1364,7 @@
                goto e_inval;
        }

-       err = fib_validate_source(saddr, daddr, tos, FIB_RES_OIF(res), dev, &spec_dst, &itag);
+       err = fib_validate_source(saddr, daddr, tos, FIB_RES_OIF(res), dev, &spec_dst, &itag, 0);
        if (err < 0)
                goto martian_source;

@@ -1447,7 +1447,7 @@
        if (ZERONET(saddr)) {
                spec_dst = inet_select_addr(dev, 0, RT_SCOPE_LINK);
        } else {
-               err = fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag);
+               err = fib_validate_source(saddr, 0, tos, 0, dev, &spec_dst, &itag, 1);
                if (err < 0)
                        goto martian_source;
                if (err)
```

29.5 fwmark name-number translation table

ipvsadm allows entry of fwmark as numbers. In some cases, it would be more convenient to enter/display the fwmark as a name; *e.g.* an e-commerce site, serving multiple customers (*i.e.* VIPs) and which is linking http and https by a fwmark. The output of ipvsadm then would list the fwmark as "bills_business", "fred_inc" rather than "14","15"...

Horms has written a patch which allows the use of fwmarks as text as well as the default method of numbers, using a table in /etc that looks like the /etc/hosts table.

> Horms `horms@vergenet.net` Nov 14 2001 while we were at OLS in June, Joe suggested that we have a file to associate names with firewall marks. I have attached a patch that enables ipvsadm to read names for firewall marks from /etc/fwmarks. This file is intended to be analogous to /etc/hosts.
>
> The patch to the man page explains the format more fully, but briefly the format is "fwmark name..." newline delimited
>
> ```
> e.g.
>
> 1 a_name
> 2 another_name yet_another_name
> ```

```
     Which leads to

     ipvsadm -A -f a_name

--Boundary_(ID_7gtcNW35aSCKH29xS00dLw)
Content-type: text/plain; charset=us-ascii
Content-disposition: attachment; filename="ipvs-0.9.5.fwmarks-file.patch"

diff -ruN ipvs-0.9.5/ipvs/ipvsadm/Makefile ipvs-0.9.5.new/ipvs/ipvsadm/Makefile
--- ipvs-0.9.5/ipvs/ipvsadm/Makefile    Sat Oct 20 04:14:00 2001
+++ ipvs-0.9.5.new/ipvs/ipvsadm/Makefile        Sat Oct 27 20:31:58 2001
@@ -63,7 +63,7 @@
 POPT_DEFINE = -DHAVE_POPT
 endif

-OBJS = ipvsadm.o config_stream.o dynamic_array.o
+OBJS = ipvsadm.o config_stream.o dynamic_array.o fwmark_lookup.o
 LIBS = $(POPT_LIB)
 DEFINES = -DVERSION=\"$(VERSION)\" -DSCHEDULERS=\"$(SCHEDULERS)\" \
          $(POPT_DEFINE) $(IP_VS_H_DEFINE)
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/VERSION ipvs-0.9.5.new/ipvs/ipvsadm/VERSION
--- ipvs-0.9.5/ipvs/ipvsadm/VERSION     Wed Sep 19 03:42:54 2001
+++ ipvs-0.9.5.new/ipvs/ipvsadm/VERSION Sat Oct 27 20:31:56 2001
@@ -1 +1 @@
-1.20
+1.21
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/config_stream.c ipvs-0.9.5.new/ipvs/ipvsadm/config_stream.c
--- ipvs-0.9.5/ipvs/ipvsadm/config_stream.c     Fri Mar 23 00:57:46 2001
+++ ipvs-0.9.5.new/ipvs/ipvsadm/config_stream.c Mon Oct 29 10:49:29 2001
@@ -1,12 +1,29 @@
 /*
- *      Code to convert a stream input into a dynamic array
- *      that can be parsed as argc and argv.
+ *      dynamic_array.c -  Code to convert a stream input into a
+ *                         that can be parsed as argc and argv.
  *
  *      Authors: Horms <horms@vergenet.net>
  *
- *      Released under the terms of the GNU GPL
+ *      Version: $Id: config_stream.c,v Exp $
  *
- *      ChangeLog
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ *     Changes:
  *      Horms         :   scanf Glibc under Red Hat 7 does not appear
  *                        to return EOF when input ends. Fall through
  *                        code has been added to handle this case correctly
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/config_stream.h ipvs-0.9.5.new/ipvs/ipvsadm/config_stream.h
--- ipvs-0.9.5/ipvs/ipvsadm/config_stream.h     Wed May 31 14:36:21 2000
+++ ipvs-0.9.5.new/ipvs/ipvsadm/config_stream.h Mon Oct 29 10:50:10 2001
@@ -1,14 +1,33 @@
 /*
- *      Code to convert a stream input into a dynamic array
- *      that can be parsed as argc and argv.
+ *      dynamic_array.h -  Code to convert a stream input into a
+ *                         that can be parsed as argc and argv.
  *
  *      Authors: Horms <horms@vergenet.net>
  *
- *      Released under the terms of the GNU GPL
+ *      Version: $Id: config_stream.h,v Exp $
+ *
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ *     Changes:
  */

-#ifndef CONFIG_STREAM_FLIM
-#define CONFIG_STREAM_FLIM
+#ifndef _CONFIG_STREAM_H
```

```
+#define _CONFIG_STREAM_H

 #include "dynamic_array.h"

@@ -16,4 +35,4 @@

 dynamic_array_t *config_stream_read (FILE *stream, const char *first_element);

-#endif
+#endif /* _CONFIG_STREAM_H */
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/del ipvs-0.9.5.new/ipvs/ipvsadm/del
--- ipvs-0.9.5/ipvs/ipvsadm/del Thu Jan  1 12:00:00 1970
+++ ipvs-0.9.5.new/ipvs/ipvsadm/del     Mon Oct 29 10:42:13 2001
@@ -0,0 +1,30 @@
+/*
+ *      dynamic_array.c -  Code to convert a stream input into a
+ *                         that can be parsed as argc and argv.
+ *
+ *      Author: Horms <horms@vergenet.net>
+ *
+ *      Version: $Id: config_stream.c,v Exp $
+ *
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ *     Changes:
+ *      Horms         :   scanf Glibc under Red Hat 7 does not appear
+ *                        to return EOF when input ends. Fall through
+ *                        code has been added to handle this case correctly
+ */
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/dynamic_array.c ipvs-0.9.5.new/ipvs/ipvsadm/dynamic_array.c
--- ipvs-0.9.5/ipvs/ipvsadm/dynamic_array.c     Fri Mar 23 00:57:46 2001
+++ ipvs-0.9.5.new/ipvs/ipvsadm/dynamic_array.c Mon Oct 29 10:49:33 2001
@@ -1,4 +1,6 @@
 /*
+ *      dynamic_array.c
+ *
  *      Dynamic array, to store all your flims in Includes macros required
  *      to create an array of strings but as the primitive type for the
  *      array is void * providing your own duplicate_primitive and
@@ -7,8 +9,29 @@
  *
  *      Authors: Horms <horms@vergenet.net>
  *
- *      Released under the terms of the GNU GPL
+ *      Version: $Id: dynamic_array.c,v Exp $
  *
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ *     Changes:
+ *      Horms         :   scanf Glibc under Red Hat 7 does not appear
+ *                        to return EOF when input ends. Fall through
+ *                        code has been added to handle this case correctly
  */

 #include "dynamic_array.h"
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/dynamic_array.h ipvs-0.9.5.new/ipvs/ipvsadm/dynamic_array.h
--- ipvs-0.9.5/ipvs/ipvsadm/dynamic_array.h     Fri Mar 23 00:57:46 2001
+++ ipvs-0.9.5.new/ipvs/ipvsadm/dynamic_array.h Mon Oct 29 10:50:11 2001
@@ -1,4 +1,7 @@
+
 /*
+ *      dynamic_array.h
+ *
  *      Dynamic array, to store all your flims in Includes macros required
  *      to create an array of strings but as the primitive type for the
  *      array is void * providing your own duplicate_primitive and
@@ -7,12 +10,33 @@
  *
  *      Authors: Horms <horms@vergenet.net>
  *
```

```
- *      Released under the terms of the GNU GPL
+ *      Version: $Id: dynamic_array.h,v Exp $
+ *
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
  *
+ *     Changes:
+ *      Horms         :   scanf Glibc under Red Hat 7 does not appear
+ *                        to return EOF when input ends. Fall through
+ *                        code has been added to handle this case correctly
  */

-#ifndef DYNAMIC_ARRAY_FLIM
-#define DYNAMIC_ARRAY_FLIM
+#ifndef _DYNAMIC_ARRAY_H
+#define _DYNAMIC_ARRAY_H

 #include <stdio.h>
 #include <stdlib.h>
@@ -174,4 +198,4 @@

 dynamic_array_t *dynamic_array_split_str(char *string, const char delimiter);

-#endif
+#endif /* _DYNAMIC_ARRAY_H */
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/fwmark_lookup.c ipvs-0.9.5.new/ipvs/ipvsadm/fwmark_lookup.c
--- ipvs-0.9.5/ipvs/ipvsadm/fwmark_lookup.c     Thu Jan  1 12:00:00 1970
+++ ipvs-0.9.5.new/ipvs/ipvsadm/fwmark_lookup.c Mon Oct 29 10:49:36 2001
@@ -0,0 +1,786 @@
+/*
+ *      fwmark_lookup.c
+ *
+ *      Look up firwall marks in /etc/fwmarks. This is intended to be
+ *      analogous to /etc/hosts
+ *
+ *      Authors: Horms <horms@vergenet.net>
+ *
+ *      Version: $Id: fwmark_lookup.c,v Exp $
+ *
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ *     Changes:
+ */
+
+#include <stdio.h>
+#include <stdlib.h>
+#include <sys/types.h>
+#include <sys/stat.h>
+#include <fcntl.h>
+#include <unistd.h>
+#include <ctype.h>
+#include <string.h>
+#include <errno.h>
+#include <netdb.h>
+
+#include "fwmark_lookup.h"
+
+#define __FWM_LOOKUP_MAX_LINE_LENGTH 4096
+#define __FWM_BLOCKING ((size_t) 7)
+
+static struct hostent __fwm_hostent;
+static char **__fwm_h_aliases = NULL;
+static size_t __fwm_no_h_aliases = 0;
+static char **__fwm_h_addr_list = NULL;
+static size_t __fwm_no_h_addr_list = 0;
+static int __fwm_fd = -1;
+
+static struct hostent *__fwm_simple_hostent(const fwm_t fwm, const char *name);
+static int __fwm_beginfwment(void);
+
+static unsigned char __fwm_getc_buffer[__FWM_LOOKUP_MAX_LINE_LENGTH];
```

```
+static ssize_t __fwm_getc_nread = 0;
+static ssize_t __fwm_getc_offset = 0;
+
+static int __fwm_getc(void);
+
+static int __fwm_add_fwmark(fwm_t key);
+static int __fwm_add_fwmark_str(char *key);
+static int __fwm_add_name(const char *name);
+
+static void __fwm_reset_global(void);
+static struct hostent *__fwm_reset_hostent(struct hostent *hostent);
+static void __fwm_array_destroy(void **a, const size_t count,
+                                const size_t fixed_size);
+static void **__fwm_array_add_element(void **a, size_t * count,
+                                      void *new_elem, size_t fixed_size);
+
+
+/**********************************************************************
+ * __fwm_reset_global
+ * Reset the global hostent structure and associated arrays.
+ * pre: none
+ * post: __fwm_hostent is reset
+ *       __fwm_h_aliases and __fwm_h_addr_list are freed and set to NULL
+ *       __fwm_no_h_aliases and __fwm_no_h_addr_list are set to 0
+ * return: none
+ **********************************************************************/
+
+static void __fwm_reset_global(void)
+{
+       __fwm_reset_hostent(&__fwm_hostent);
+
+       __fwm_array_destroy((void **) __fwm_h_aliases, __fwm_no_h_aliases,
+                           0);
+       __fwm_h_aliases = NULL;
+       __fwm_no_h_aliases = 0;
+
+       __fwm_array_destroy((void **) __fwm_h_addr_list,
+                           __fwm_no_h_addr_list, sizeof(fwm_t));
+       __fwm_h_addr_list = NULL;
+       __fwm_no_h_addr_list = 0;
+}
+
+
+/**********************************************************************
+ * __fwm_reset_hostent
+ * Zero a hostent structure
+ * pre: hostent: hostent to zero
+ * post: values of hostent are zeroed
+ * return: hostent
+ * Note: Any memory associated with elements of hostent are
+ *       not freed.
+ ********************************************************************/
+
+static struct hostent *__fwm_reset_hostent(struct hostent *hostent)
+{
+       if (hostent == NULL) {
+               return (NULL);
+       }
+
+       hostent->h_name = NULL;
+       hostent->h_aliases = NULL;
+       hostent->h_addrtype = AF_INET;
+       hostent->h_length = sizeof(fwm_t);
+       hostent->h_addr_list = NULL;
+
+       return (hostent);
+}
+
+
+/********************************************************************
+ * __fwm_array_destroy
+ * Destroy an array of strings
+ * pre: a: array of strings to free
+ *      count: number of elements allocated in the array
+ *      fixed_size: Size of elements if they are a fixed size
+ *                  0 if they are variable size
+ * post: If fixed_size is zero free each element in the array
+ *       Free the array.
+ *       Note: If elements are fixed size they are allocated as part
+ *             of the array itself and hence are freed anyway,
+ * return: none
+ ********************************************************************/
+
+static void __fwm_array_destroy(void **a, const size_t count,
+                               const size_t fixed_size)
+{
+       int i;
+
+       if (a == NULL) {
+               return;
+       }
+
+       if (!fixed_size) {
+               for (i = 0; i < count; i++) {
+                       if (a[i] != NULL) {
+                               free(a[i]);
+                       }
+               }
+       }
+
+       free(a);
+}
+
+
+/*********************************************************************
+ * __fwm_array_add_element
+ * Add an element to an array of strings.
+ * If the array is empty create it, if there is not enough space
+ * allocate more space.
+ * pre: a: array to add element to
+ *      count: number of elements allocated in the array
+ *      new_elem: element to add to the array
+ *      fixed_size: Size of elements if they are a fixed size
+ *                  0 if they are variable size
+ * post: If a is NULL
+ *           allocate a block of elements in the array and
+ *           make elem the fist element
+ *       Else
+ *           Find the last unused element
+ *           If this is the last allocated element in the array
+ *               Allocate another block of elemnts
+ *           Make the last unused element in the array new_elem
+ *       Note: If elements are fixed size they are allocated as part
+ *             of the array itself and copied into the array
+ *             using memcpy. Otherewise the only the pointer
+ *             to the element is stored and well be freed by
+ *             __fwm_array_destroy.
+ * return: none
+ *********************************************************************/
+
+static void **__fwm_array_add_element(void **a, size_t * count,
+                                      void *new_elem, size_t fixed_size)
+{
+       void **old_a;
+       size_t elem;
+       size_t elem_alloc;
+
+       elem_alloc = (fixed_size ? fixed_size : sizeof(void *));
+
+       if (a == NULL || *count == 0) {
+               *count = __FWM_BLOCKING;
+               a = (void **) malloc(__FWM_BLOCKING * elem_alloc);
+               if (a == NULL) {
+                       *count = 0;
+                       return (NULL);
+               }
+               memset(a, '\0', __FWM_BLOCKING * elem_alloc);
+               elem = 0;
+       } else {
+               old_a = a;
```

```
+
+               for (elem = 0; elem < *count; elem++) {
+                       if (a[elem] == NULL) {
+                               break;
+                       }
+               }
+
+               if (elem + 2 > *count) {
+                       *count += __FWM_BLOCKING;
+                       a = (void **) realloc((void *) a,
+                                             *count * elem_alloc);
+                       if (a == NULL) {
+                               __fwm_array_destroy(old_a,
+                                                   *count -
+                                                   __FWM_BLOCKING,
+                                                   fixed_size);
+                               *count = 0;
+                               return (NULL);
+                       }
+                       memset(a + (*count) - __FWM_BLOCKING, '\0',
+                              __FWM_BLOCKING * elem_alloc);
+               }
+       }
+
+       if (fixed_size) {
+               memcpy(&(a[elem]), new_elem, fixed_size);
+       } else {
+               a[elem] = new_elem;
+       }
+
+       return (a);
+}
+
+
+/**********************************************************************
+ * __fwm_str_to_fwm
+ * Convert a string to a fwmark
+ * pre: str: ASCII representation of a fwmark
+ *      fwm: pointer to fwm_t to store result in
+ * post: str is converted to a fwm
+ * return: 0 on success
+ *         -1 on erroe
+ **********************************************************************/
+
+static int __fwm_str_to_fwm(const char *str, fwm_t *fwm)
+{
+       long l;
+       char *end;
+
+       if(str == NULL || *str == '\0' || fwm == NULL) {
```

```
+               return(-1);
+       }
+
+       l = strtol(str, &end, 10);
+       if(*end != '\0' || errno == ERANGE || l < 0 || l > UINT_MAX) {
+               return(-1);
+       }
+
+       *fwm = (fwm_t) l;
+       return(0);
+}
+
+
+/*********************************************************************
+ * __fwm_getc
+ * read a single character from the fwmarks file
+ * Note: This is buffered internally and hence should
+ * be reasonably efficient.
+ * pre: fwmarks file is open.
+ * post: If __fwm_getc_buffer is empty or has been exhausted
+ *       then it is filled by reding
+ *       __FWM_LOOKUP_MAX_LINE_LENGTH bytes from __fwm_fd.
+ *       One character is returned from __fwm_getc_buffer and
+ *       the offset is advanced.
+ * return: Once character from the fwmarks file.
+ *         EOF on error or end of file.
+ *********************************************************************/
+
+static int __fwm_getc(void)
+{
+       ssize_t nread;
+
+       if (__fwm_getc_offset < __fwm_getc_nread) {
+               return ((int) (__fwm_getc_buffer[__fwm_getc_offset++]));
+       }
+
+       while (1) {
+               nread = read(__fwm_fd, __fwm_getc_buffer,
+                               __FWM_LOOKUP_MAX_LINE_LENGTH);
+               if (nread < 0) {
+                       if (errno == EINTR) {
+                               continue;
+                       }
+                       return (EOF);
+               }
+               if (nread == 0) {
+                       return (EOF);
+               }
+
+               __fwm_getc_offset = 0;
+               __fwm_getc_nread = nread;
+               break;
+       }
+
+       return ((int) (__fwm_getc_buffer[__fwm_getc_offset++]));
+}
+
+
+/**********************************************************************
+ * __fwm_add_fwmark
+ * Add a fwmark to __fwm_hostent
+ * pre: fwm to add to the __fwm_hostent
+ * post: fwm is added to h_addr_list
+ * return: 0 on success
+ *        -1 on error
+ **********************************************************************/
+
+static int __fwm_add_fwmark(fwm_t fwm)
+{
+       void **a;
+
+       fwm = htonl(fwm);
+
+       a = __fwm_array_add_element((void **) __fwm_h_addr_list,
+                                   &__fwm_no_h_addr_list, (void *) &fwm,
+                                   sizeof(fwm_t));
+       if (a == NULL) {
+               __fwm_reset_global();
+               endfwment();
+               return (-1);
+       }
+
+       __fwm_h_addr_list = (char **) a;
+       __fwm_hostent.h_addr_list = __fwm_h_addr_list;
+
+       return (0);
+}
+
+
+/**********************************************************************
+ * __fwm_add_fwmark_str
+ * Add a fwmark to __fwm_hostent
+ * pre: fwm_str: fwm to add to the __fwm_hostent, as a string.
+ *      The string must be the ASCII representation a posigive integer.
+ * post: fwm is added to h_addr_list
+ * return: 0 on success
+ *        -1 on error
+ **********************************************************************/
+
+static int __fwm_add_fwmark_str(char *fwm_str)
```

```
+{
+       fwm_t fwm;
+
+       if(__fwm_str_to_fwm(fwm_str, &fwm) < 0) {
+               return(-1);
+       }
+
+       return (__fwm_add_fwmark(fwm));
+}
+
+
+/*********************************************************************
+ * __fwm_add_name
+ * Add a name to __fwm_hostent
+ * pre: name: name to add to __fwm_hostent
+ * post: name is copied using strcpy.
+ *       __fwm_hostent is added to h_addr_list. If h_name is
+ *       null, it is set to name.
+ * return: 0 on success
+ *         -1 on error
+ *********************************************************************/
+
+static int __fwm_add_name(const char *name)
+{
+       void **a;
+       char *name_cpy;
+
+       name_cpy = strdup(name);
+       if (name_cpy == NULL) {
+               return (-1);
+       }
+
+       a = __fwm_array_add_element((void **) __fwm_h_aliases,
+                                   &__fwm_no_h_aliases, (void *) name_cpy, 0);
+       if (a == NULL) {
+               __fwm_reset_global();
+               endfwment();
+               return (-1);
+       }
+
+       __fwm_h_aliases = (char **) a;
+       __fwm_hostent.h_aliases = __fwm_h_aliases;
+
+       if (__fwm_hostent.h_name == NULL) {
+               __fwm_hostent.h_name = __fwm_hostent.h_aliases[0];
+       }
+
+       return (0);
+}
+
+
+/*****************************************************************
+ * __fwm_simple_hostent
+ * hostent with fwm as the only entry in h_addr_list and
+ * name as the h_name and only entry in h_aliases
+ * pre: fwm: fwm to be the "address" in the hostent
+ *      name: "name" for the hostent
+ * post: The global __fwm_hostent is set with fwm and name
+ * return: pointer to __fwm_hostent
+ *         NULL on error
+ *****************************************************************/
+
+static struct hostent *__fwm_simple_hostent(const fwm_t fwm, const char *name)
+{
+       __fwm_reset_global();
+       if (__fwm_add_fwmark(fwm) < 0) {
+               endfwment();
+               return (NULL);
+       }
+       if (__fwm_add_name(name) < 0) {
+               endfwment();
+               return (NULL);
+       }
+       return(&__fwm_hostent);
+}
+
+/*****************************************************************
+ * __fwm_beginfwment
+ * Open the fwmarks file and reset the read buffers.
+ * pre: FWMARKS_FILE is defined
+ * post: globals __fwm_getc_* globals are zeroed
+ *       __fwm_fd is an open, read-only file descriptor to FWMARKS_FILE.
+ * return: none.
+ *****************************************************************/
+
+static int __fwm_beginfwment(void)
+{
+       __fwm_getc_nread = 0;
+       __fwm_getc_offset = 0;
+       memset(__fwm_getc_buffer, '\0', __FWM_LOOKUP_MAX_LINE_LENGTH);
+
+       if (__fwm_fd > 0) {
+               endfwment();
+       }
+
+       if ((__fwm_fd = open(FWMARKS_FILE, O_RDONLY)) < 0) {
+               endfwment();
+       }
+
+       return (__fwm_fd);
+}
+
+
+/**********************************************************************
+ * setfwment
+ * Rewind the fwmarks file
+ * pre: FWMARKS_FILE is defined
+ * post: If fwmarks file is already open it is rewound to the begining
+ *       else it is opened.
+ * return: none
+ **********************************************************************/
+
+void setfwment(void)
+{
+       if (__fwm_fd > -1) {
+               if (lseek(__fwm_fd, SEEK_SET, 0) < 0) {
+                       endfwment();
+               }
+       } else {
+               __fwm_beginfwment();
+       }
+}
+
+
+/**********************************************************************
+ * endfwment
+ * Close the fwmarks file
+ * pre: none
+ * post: fwmarks file is closed if it was open.
+ * return: none
+ **********************************************************************/
+
+void endfwment(void)
+{
+       if (__fwm_fd > -1) {
+               close(__fwm_fd);
+       }
+       __fwm_fd = -1;
+}
+
+
+/**********************************************************************
+ * getfwment
+ * Read the next valid line from the fwmarks file
+ * pre: none
+ * return: hostent for line for fwmarks file if a valid line is found
+ *         NULL if the end of the file is reached without finding a new
+ *         line, or on error.
+ *
+ * Note: Values are stored in internally allocated memory which is not
+ *       persistant across calls. If you wish to keep values you should
+ *       copy them.  Don not free the memory returned by this function.
+ *********************************************************************/
+
+
+/* Mark the begining of a key */
+#define BEGIN_KEY \
+       if(!in_escape && !in_comment && !in_quote){ \
+               in_key=1; \
+               valid_key = 0; \
+       }
+
+/* Mark the end of a key.
+ * If a valid key has been read then store it */
+#define END_KEY \
+       if(!in_escape && in_key && !in_quote){ \
+               if(in_key && token_pos){ \
+                       *(token_buffer+token_pos)='\0'; \
+                       if(__fwm_add_fwmark_str(token_buffer) < 0) { \
+                               __fwm_reset_global(); \
+                               endfwment(); \
+                               return(NULL); \
+                       } \
+                       else { \
+                               valid_key = 1; \
+                       } \
+               } \
+               token_pos=0; \
+               in_key=0; \
+       }
+
+/* Mark the begining of a value */
+#define BEGIN_VALUE \
+       if(!in_key && !in_comment && !in_quote){ \
+               in_value=1; \
+       } \
+
+/* Mark the end of a key.
+ * If a valid value and a valid key have been read then store the value */
+
+#define END_VALUE \
+       if(!in_escape && in_value && !in_quote){ \
+               if(in_value && valid_key){ \
+                       *(token_buffer+token_pos)='\0'; \
+                       if(__fwm_add_name(token_buffer) < 0 ) { \
+                               __fwm_reset_global(); \
+                               endfwment(); \
+                               return(NULL); \
+                       } \
+               } \
+               token_pos=0; \
+               in_value=0; \
+       }
+
+/* Mark the end of a comment if it is not escaped*/
+#define END_COMMENT \
+       if(!in_escape){ \
+               in_comment=0; \
+       }
+
+/* Mark the begining of a comment if it is not escaped or in a commnet */
+#define BEGIN_COMMENT \
+       if(!in_escape && !in_quote){ \
+               in_comment=1; \
+       }
+
+/* Mark the begining of an escape */
+#define BEGIN_ESCAPE \
+       in_escape=1;
+
+/* Mark the end of an escape */
+#define END_ESCAPE \
+       in_escape=0;
+
+#define __FWM_SINGLE_QUOTE 1
+#define __FWM_DOUBLE_QUOTE 2
+
+struct hostent *getfwment(void)
+{
+       ssize_t token_pos;
+       char token_buffer[__FWM_LOOKUP_MAX_LINE_LENGTH];
+       char c;
+       int max_token_pos = __FWM_LOOKUP_MAX_LINE_LENGTH - 3;
+       int i;
+
+       int in_escape = 0;
+       int in_comment = 0;
+       int skip_char = 0;
+       int in_value = 0;
+       int in_quote = 0;
+       int in_key = 0;
+       int valid_key = 0;
+
+       if (__fwm_fd < 0 && __fwm_beginfwment() < 0) {
+               return (NULL);
+       }
+
+       token_pos = 0;
+
+       __fwm_reset_global();
```

```
+
+       BEGIN_KEY;
+
+       while ((i = __fwm_getc()) != EOF) {
+               c = (char) i;
+
+               switch (c) {
+               case ' ':
+               case '\t':
+                       END_KEY;
+                       END_VALUE;
+                       if (in_escape) {
+                               BEGIN_VALUE;
+                       }
+                       END_ESCAPE;
+                       break;
+               case '\n':
+               case '\r':
+                       END_KEY;
+                       END_COMMENT;
+                       END_VALUE;
+                       if (!in_escape && !in_quote && valid_key) {
+                               if (__fwm_hostent.h_aliases == NULL) {
+                                       return (NULL);
+                               }
+                               return (&__fwm_hostent);
+                       }
+                       BEGIN_KEY;
+                       END_ESCAPE;
+                       break;
+               case '\\':
+                       if (in_escape || in_quote) {
+                               END_ESCAPE;
+                       } else {
+                               BEGIN_ESCAPE;
+                       }
+                       BEGIN_VALUE;
+                       break;
+               case '#':
+                       BEGIN_COMMENT;
+                       END_KEY;
+                       END_VALUE;
+                       BEGIN_VALUE;
+                       END_ESCAPE;
+                       break;
+               case '"':
+                       BEGIN_VALUE;
+                       if (!in_escape && !in_comment
+                           && !(in_quote & __FWM_SINGLE_QUOTE)) {
+                               if (in_quote & __FWM_DOUBLE_QUOTE) {
```

```
+                                       in_quote ^=
+                                           in_quote & __FWM_DOUBLE_QUOTE;
+                               } else {
+                                       in_quote |= __FWM_DOUBLE_QUOTE;
+                               }
+                               skip_char = 1;
+                       }
+                       END_ESCAPE;
+                       break;
+               case '\'':
+                       BEGIN_VALUE;
+                       if (!in_escape && !in_comment) {
+                               if (in_quote & __FWM_SINGLE_QUOTE) {
+                                       in_quote ^= __FWM_SINGLE_QUOTE;
+                               } else {
+                                       in_quote |= __FWM_SINGLE_QUOTE;
+                               }
+                               skip_char = 1;
+                       }
+                       END_ESCAPE;
+                       break;
+               default:
+                       BEGIN_VALUE;
+                       END_ESCAPE;
+                       break;
+               }
+
+               if (in_key | in_value &&
+                   c != '\n' &&
+                   c != '\r' &&
+                   !in_escape && !skip_char
+                   && token_pos < max_token_pos) {
+                       *(token_buffer + token_pos) = c;
+                       token_pos++;
+               }
+               skip_char = 0;
+
+       }
+
+       __fwm_reset_global();
+       return (NULL);
+}
+
+
+/**********************************************************************
+ * getfwmbyfwm
+ * Find a the names for a firewall mark
+ * pre: fwm: firewall mark to find names of
+ * post: fwmarks file is serached for the first line corresponding
+ *       to fwm
+ * return: hostent connresponding to fwm as per the fwmarks file.
+ *
+ * Note: Values are stored in internally allocated memory which is not
+ *       persistant across calls. If you wish to keep values you should
+ *       copy them.  Don not free the memory returned by this function.
+ **********************************************************************/
+
+struct hostent *getfwmbyfwm(fwm_t fwm)
+{
+       struct hostent *hostent;
+       char **elem;
+       char buf[11];
+
+       setfwment();
+       while ((hostent = getfwment()) != NULL) {
+               for (elem = hostent->h_addr_list; *elem != NULL; elem++) {
+                       if (ntohl((fwm_t) * elem) == fwm) {
+                               endfwment();
+                               return (hostent);
+                       }
+               }
+       }
+
+       endfwment();
+
+       /* No entry was found in the fwmarks file, so return
+        * the fwm as its own name */
+       if (snprintf(buf, 11, "%u", fwm) < 0) {
+               endfwment();
+               return (NULL);
+       }
+       buf[10] = '\0';
+       return(__fwm_simple_hostent(fwm, buf));
+}
+
+
+/**********************************************************************
+ * getfwmbyname
+ * Find a the fwmark and aliases for a name
+ * pre: fwm: name to find the firewall mark and aliases of
+ * post: If name is the ASCII representation of a fwm
+ *           A hostent with that as the name and fwm is returned.
+ *       Else
+ *          fwmarks file is serached for the first line corresponding
+ *          to name
+ * return: hostent connresponding to name as per the fwmarks file.
+ *
+ * Note: Values are stored in internally allocated memory which is not
+ *       persistant across calls. If you wish to keep values you should
+ *       copy them.  Don not free the memory returned by this function.
+ ********************************************************************/
+
+struct hostent *getfwmbyname(const char *name)
+{
+       struct hostent *hostent;
+       char **elem;
+       fwm_t fwm;
+
+       /* If name is just the ASCII representation of an fwm then
+        * use that as the fwm */
+       if(__fwm_str_to_fwm(name, &fwm) > -1) {
+               return(__fwm_simple_hostent(fwm, name));
+       }
+
+       setfwment();
+       while ((hostent = getfwment()) != NULL) {
+               for (elem = hostent->h_aliases; *elem != NULL; elem++) {
+                       if (strcmp(name, *elem) == 0) {
+                               endfwment();
+                               fflush(NULL);
+                               return (hostent);
+                       }
+               }
+       }
+
+       endfwment();
+       return (NULL);
+
+}
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/fwmark_lookup.h ipvs-0.9.5.new/ipvs/ipvsadm/fwmark_lookup.h
--- ipvs-0.9.5/ipvs/ipvsadm/fwmark_lookup.h     Thu Jan  1 12:00:00 1970
+++ ipvs-0.9.5.new/ipvs/ipvsadm/fwmark_lookup.h Mon Oct 29 10:50:23 2001
@@ -0,0 +1,144 @@
+/*
+ *      fwmark_lookup.h
+ *
+ *      Look up firwall marks in /etc/fwmarks. This is intended to be
+ *      analogous to /etc/hosts
+ *
+ *      Authors: Horms <horms@vergenet.net>
+ *
+ *      Version: $Id: fwmark_lookup.h,v Exp $
+ *
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ *     Changes:
+ */
+
+/*
+ * /etc/fwmarks
+ *
+ * The format for this file is
+ *
+ * fwnark name ...
+ *
+ * Blank lines are ignored, as is anything including and after a # (hash)
+ * on a line. If a \ precedes a new line then the lines will be
+ * concatenated.  if a \ precedes any other character, including a #
+ * (hash) it will be treated as a literal. Anything inside single quotes
+ * (') will be treated as a litreal. Anything other than a (') inside
+ * double quotes (") will be treated as a litreal. Whitespace in names must
+ * be escaped or quoted.
+ *
+ * e.g.
+ *
+ * 1 jimmy james
+ * 2 john
+ *
+ * This associates the names jimmy ad james with fwmark 1
+ * and the name john with fwmark 2
+ *
+ */
+
+#ifndef _FWMARK_LOOKUP_H
+#define _FWMARK_LOOKUP_H
+
+
+/* This should be set by a configure check */
+typedef unsigned int uint32;
+
+typedef uint32 fwm_t;
+
+#ifndef FWMARKS_FILE
+#define FWMARKS_FILE "/etc/fwmarks"
+#endif
```

```
+
+
+/**********************************************************************
+ * setfwment
+ * Rewind the fwmarks file
+ * pre: FWMARKS_FILE is defined
+ * post: If fwmarks file is already open it is rewound to the begining
+ *       else it is opened.
+ * return: none
+ **********************************************************************/
+
+void setfwment(void);
+
+
+/**********************************************************************
+ * endfwment
+ * Close the fwmarks file
+ * pre: none
+ * post: fwmarks file is closed if it was open.
+ * return: none
+ **********************************************************************/
+
+void endfwment(void);
+
+
+/**********************************************************************
+ * getfwment
+ * Read the next valid line from the fwmarks file
+ * pre: none
+ * return: hostent for line for fwmarks file if a valid line is found
+ *         NULL if the end of the file is reached without finding a new
+ *         line, or on error.
+ *
+ * Note: Values are stored in internally allocated memory which is not
+ *       persistant across calls. If you wish to keep values you should
+ *       copy them.  Don not free the memory returned by this function.
+ **********************************************************************/
+
+
+struct hostent *getfwment(void);
+
+
+/**********************************************************************
+ * getfwmbyfwm
+ * Find a the names for a firewall mark
+ * pre: fwm: firewall mark to find names of
+ * post: fwmarks file is serached for the first line corresponding
+ *       to fwm
+ * return: hostent connresponding to fwm as per the fwmarks file.
+ *
+ * Note: Values are stored in internally allocated memory which is not
+ *       persistant across calls. If you wish to keep values you should
+ *       copy them.  Don not free the memory returned by this function.
+ **********************************************************************/
+
+struct hostent *getfwmbyfwm(fwm_t fwm);
+
+
+/**********************************************************************
+ * getfwmbyname
+ * Find a the fwmark and aliases for a name
+ * pre: fwm: name to find the firewall mark and aliases of
+ * post: If name is the ASCII representation of a fwm
+ *           A hostent with that as the name and fwm is returned.
+ *       Else
+ *          fwmarks file is serached for the first line corresponding
+ *          to name
+ * return: hostent connresponding to name as per the fwmarks file.
+ *
+ * Note: Values are stored in internally allocated memory which is not
+ *       persistant across calls. If you wish to keep values you should
+ *       copy them.  Don not free the memory returned by this function.
+ **********************************************************************/
+
+struct hostent *getfwmbyname(const char *name);
+
+#endif /* _FWMARK_LOOKUP_H */
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/header ipvs-0.9.5.new/ipvs/ipvsadm/header
--- ipvs-0.9.5/ipvs/ipvsadm/header      Thu Jan  1 12:00:00 1970
+++ ipvs-0.9.5.new/ipvs/ipvsadm/header  Mon Oct 29 10:42:16 2001
@@ -0,0 +1,30 @@
+/*
+ *      dynamic_array.c -  Code to convert a stream input into a
+ *                         that can be parsed as argc and argv.
+ *
+ *      Author: Horms <horms@vergenet.net>
+ *
+ *      Version: $Id: config_stream.c,v Exp $
+ *
+ *      Copyright (c) 2001 Horms
+ *      All rights reserved.
+ *
+ *      This program is free software; you can redistribute it and/or modify
+ *      it under the terms of the GNU General Public License as published by
+ *      the Free Software Foundation; either version 2 of the License, or
+ *      (at your option) any later version.
+ *
+ *      This program is distributed in the hope that it will be useful,
+ *      but WITHOUT ANY WARRANTY; without even the implied warranty of
+ *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ *      GNU General Public License for more details.
+ *
+ *      You should have received a copy of the GNU General Public License
+ *      along with this program; if not, write to the Free Software
+ *      Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ *     Changes:
+ *      Horms         :   scanf Glibc under Red Hat 7 does not appear
+ *                        to return EOF when input ends. Fall through
+ *                        code has been added to handle this case correctly
+ */
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/ipvsadm.8 ipvs-0.9.5.new/ipvs/ipvsadm/ipvsadm.8
--- ipvs-0.9.5/ipvs/ipvsadm/ipvsadm.8   Wed Sep 19 03:42:54 2001
+++ ipvs-0.9.5.new/ipvs/ipvsadm/ipvsadm.8       Mon Oct 29 11:05:22 2001
@@ -4,16 +4,18 @@
 .\"    $Id: ipvsadm.8,v 1.8 2001/09/18 15:42:54 wensong Exp $
 .\"
 .\"     Authors: Mike Wangsmo <wanger@redhat.com>
-.\"              Wensong Zhang <wensong@linux.com>
+.\"              Wensong Zhang <wensong@linuxvirtualserver.org>
+.\"              Horms <horms@vergenet.net>
 .\"
 .\"     Changes:
 .\"       Horms            :  Updated to reflect recent change of ipvsadm
 .\"                        :  Style guidance taken from ipchains(8)
 .\"                           where appropriate.
-.\"       Wensong Zhang    :  Added a short note about the defense strategies
+.\"       Wensong Zhang    :  Added a short note about the defence strategies
 .\"       Horms            :  Tidy up some of the description and the
 .\"                           grammar in the -f and sysctl sections
 .\"       Wensong Zhang    :  -s option description taken from ipchains(8)
+.\"       Horms            :  documented /etc/fwmarks support
 .\"
 .\"     This program is free software; you can redistribute it and/or modify
 .\"     it under the terms of the GNU General Public License as published by
@@ -30,7 +32,7 @@
 .\"     Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 .\"
 .\"
-.TH IPVSADM 8 "18th September 2001" "LVS Administration" "Linux Administrator's Guide"
+.TH IPVSADM 8 "29th October 2001" "LVS Administration" "Linux Administrator's Guide"
 .UC 4
 .SH NAME
 ipvsadm \- Linux Virtual Server administration
@@ -69,7 +71,7 @@
 nodes. The active node of the cluster redirects service requests to a
 collection of server hosts that will actually perform the
 services. Supported features include two protocols (TCP and UDP),
-three packet-forwarding methods (NAT, tunneling, and direct routing),
+three packet-forwarding methods (NAT, tunnelling, and direct routing),
 and six load balancing algorithms (round robin, weighted round robin,
 least-connection, weighted least-connection, locality-based
 least-connection, and locality-based least-connection with
@@ -149,9 +151,9 @@
 the current timeout value of the  corresponding  entry  is preserved.
 .TP
 .B --start-daemon \fIstate\fP
-Start the connection synchronization daemon. The \fIstate\fP is to
+Start the connection synchronisation daemon. The \fIstate\fP is to
 indicate that the daemon is started as master or backup. The
-connection synchronization daemon is implemented inside the Linux
+connection synchronisation daemon is implemented inside the Linux
 kernel. The master daemon running on the primary load balancer
 multicasts changes of connections periodically, and the backup daemon
 running the backup load balancers receives multicast message and
@@ -161,7 +163,7 @@
 can continue to access the service.
 .TP
 .B --stop-daemon
-Stop the connection synchronization daemon.
+Stop the connection synchronisation daemon.
 .TP
 \fB-h, --help\fR
 Display a description of the command syntax.
@@ -187,8 +189,8 @@
 virtual service instead of an address, port and protocol (UDP or
 TCP). The marking of packets with a firewall-mark is configured using
 the -m|--mark option to \fBiptables\fR(8). It can be used to build a
-virtual service assoicated with the same real servers, covering
-multiple IP addresss, port and protocol tripplets.
+virtual service associated with the same real servers, covering
+multiple IP addresses, port and protocol triplets.
 .sp
 Using firewall-mark virtual services provides a convenient method of
 grouping together different IP addresses, ports and protocols into a
@@ -270,10 +272,10 @@
 service name of port.  In the case of the masquerading method, the
 host address is usually an RFC 1918 private IP address, and the port
 can be different from that of the associated service. With the
-tunneling and direct routing methods, \fIport\fP must be equal to that
+tunnelling and direct routing methods, \fIport\fP must be equal to that
 of the service address. For normal services, the port specified  in
 the service address will be used if \fIport\fP is not specified. For
-fwmark services, \fIport\fP may be ommitted, in which case  the
+fwmark services, \fIport\fP may be omitted, in which case  the
 destination port on the real server will be the destination port of
 the request sent to the virtual service.
 .TP
@@ -281,7 +283,7 @@
 .sp
 \fB-g, --gatewaying\fR  Use gatewaying (direct routing). This is the default.
 .sp
-\fB-i, --ipip\fR  Use ipip encapsulation (tunneling).
+\fB-i, --ipip\fR  Use ipip encapsulation (tunnelling).
 .sp
 \fB-m, --masquerading\fR  Use masquerading (network access translation, or NAT).
 .sp
@@ -405,13 +407,14 @@
 modprobe ip_vs_ftp
 .fi
 .SH NOTES
-The Linux Virtual Server implements three defense strategies against
+.SS DENIAL OF SERVICE DEFENCE STRATEGIES
+The Linux Virtual Server implements three defence strategies against
 some types of denial of service (DoS) attacks. The Linux Director
 creates an entry for each connection in order to keep its state, and
 each entry occupies 128 bytes effective memory. LVS's vulnerability to
 a DoS attack lies in the potential to increase the number entries as
 much as possible until the linux director runs out of memory. The
-three defense strategies against the attack are: Randomly drop some
+three defence strategies against the attack are: Randomly drop some
 entries in the table. Drop 1/rate packets before forwarding them. And
 use secure tcp state transition table and short timeouts. The
 strategies are controlled by sysctl variables and corresponding
@@ -422,7 +425,7 @@
 /proc/sys/net/ipv4/vs/secure_tcp
 .PP
 Valid values for each variable are 0 through to 3. The default value
-is 0, which disables the respective defense strategy. 1 and 2 are
+is 0, which disables the respective defence strategy. 1 and 2 are
 automatic modes - when there is no enough available memory, the
 respective strategy will be enabled and the variable is automatically
 set to 2, otherwise the strategy is disabled and the variable is set
@@ -433,6 +436,36 @@
 .sp
 /proc/sys/net/ipv4/vs/amemthresh
 /proc/sys/net/ipv4/vs/timeout_*
+.SS NAMING FIREWALL MARKS
+ipvsadm understands names for firwall marks. These are assigned
+by adding entries to /etc/fwmarks which is intended to
+be analogous to /etc/hosts.
+.PP
+The format of the file is "fwmark name ...".
+Blank lines are ignored, as is anything including and after a # (hash) on a
+line. If a \\ precedes a new line then the lines will be concatenated.  if a
+\\ precedes any other character, including a # (hash) it will be treated as
+a literal. Anything inside single quotes (') will be treated as a literal.
+Anything other than a (') inside double quotes (") will be treated as a
+literal. Whitespace in names must be escaped or quoted.
+.sp
```

```
+.nf
+e.g.
+# /etc/fwmarks
+1 a-name
+2 another-name yet-another-name
+.fi
+.PP
+This associates a-name with fwmark 1 and associates another-name
+and yet-another-name with fwmark 2.
+In this way names may be used instead of numeric values when
+defining fwmark virtual services
+.sp
+.nf
+e.g.
+ipvsadm -A -f a-name
+ipvsadm -a -f a-name -r 127.0.0.1
+.fi
 .SH FILES
 .I /proc/net/ip_masq/vs
 .br
@@ -469,6 +502,8 @@
 .I /proc/sys/net/ipv4/vs/timeout_timewait
 .br
 .I /proc/sys/net/ipv4/vs/timeout_udp
+.br
+.I /etc/fwmarks
 .SH SEE ALSO
 \fBiptables\fP(8), \fBinsmod\fP(8), \fBmodprobe\fP(8)
 .SH AUTHORS
@@ -477,5 +512,5 @@
          Peter Kese <peter.kese@ijs.si>
 man page - Mike Wangsmo <wanger@redhat.com>
           Wensong Zhang <wensong@linuxvirtualserver.org>
-          Horms <horms@valinux.com>
+          Horms <horms@vergenet.net>
 .fi
diff -ruN ipvs-0.9.5/ipvs/ipvsadm/ipvsadm.c ipvs-0.9.5.new/ipvs/ipvsadm/ipvsadm.c
--- ipvs-0.9.5/ipvs/ipvsadm/ipvsadm.c   Sat Oct 20 04:05:17 2001
+++ ipvs-0.9.5.new/ipvs/ipvsadm/ipvsadm.c       Mon Oct 29 10:49:39 2001
@@ -52,6 +52,7 @@
  *        Horms               :   added -v option
  *        Wensong Zhang       :   rewrite most code of parsing options and
  *                                processing options.
+ *        Horms               :   added /etc/fwmarks support
  *
  *
  *      ippfvsadm - Port Fowarding & Virtual Server ADMinistration program
@@ -122,6 +123,7 @@
 #endif
 #include "config_stream.h"
+#include "fwmark_lookup.h"
 #include "libipvs/libipvs.h"

 #define IPVSADM_VERSION_NO              "v" VERSION
@@ -241,13 +243,15 @@
 int str_is_digit(const char *str);
 int string_to_number(const char *s, int min, int max);
 int host_to_addr(const char *name, struct in_addr *addr);
+int name_to_fwm(const char *name, fwm_t *fwm);
 char * addr_to_host(struct in_addr *addr);
 char * addr_to_anyname(struct in_addr *addr);
 int service_to_port(const char *name, unsigned short proto);
-char * port_to_service(int port, unsigned short proto);
-char * port_to_anyname(int port, unsigned short proto);
-char * addrport_to_anyname(struct in_addr *addr, int port,
+char * port_to_service(unsigned int port, unsigned short proto);
+char * port_to_anyname(unsigned int port, unsigned short proto);
+char * addrport_to_anyname(struct in_addr *addr, unsigned int port,
                           unsigned short proto, unsigned int format);
+char * fwmark_to_anyname(fwm_t fwm, unsigned int format);
 int parse_service(char *buf, u_int16_t proto,
                  u_int32_t *addr, u_int16_t *port);
 int parse_netmask(char *buf, u_int32_t *addr);
@@ -932,20 +936,26 @@


 /*
- * Parse IP fwmark from the argument.
+ * Parse fwmark from the argument.
  */
 unsigned int parse_fwmark(char *buf)
 {
        unsigned long l;
+       unsigned int i;
        char *end;

        errno = 0;
        l = strtol(buf, &end, 10);
-       if (*end != '\0' || end == buf ||
-           errno == ERANGE || l <= 0 || l > UINT_MAX)
-               fail(2, "invalid fwmark value `%s' specified", buf);
+       if (*end == '\0' && end != buf &&
+                       errno != ERANGE && l > 0 && l < UINT_MAX) {
+               return(l);
+       }
+       else if(name_to_fwm(buf, &i) > -1) {
+               return(ntohl(i));
+       }
```

```
-       return 1;
+       fail(2, "invalid fwmark value ‘%s’ specified", buf);
+       return 0;
 }


@@ -1401,19 +1411,22 @@
 {
        struct ip_vs_get_dests *d;
        char svc_name[64];
+       char *vname;
        int i;

        if (!(d = ipvs_get_dests(svc)))
                exit(1);

        if (svc->fwmark) {
+               if(!(vname = fwmark_to_anyname(svc->fwmark, format)))
+                       fail(2, "fwmark_to_anyname");
                if (format & FMT_RULE)
-                       sprintf(svc_name, "-f %d", svc->fwmark);
+                       sprintf(svc_name, "-f %s", vname);
                else
-                       sprintf(svc_name, "FWM  %d", svc->fwmark);
+                       sprintf(svc_name, "FWM  %s", vname);
+               free(vname);
        } else {
                struct in_addr vaddr;
-               char *vname;
                vaddr.s_addr = svc->addr;

                if (!(vname = addrport_to_anyname(&vaddr, ntohs(svc->port),
@@ -1597,6 +1610,20 @@
 }


+int name_to_fwm(const char *name, fwm_t *fwm)
+{
+       struct hostent *hostent;
+
+       if ((hostent = getfwmbyname(name)) == NULL) {
+               return -1;
+       }
+
+       /* warning: we just handle h_addr_list[0] here */
+       *fwm = (fwm_t) hostent->h_addr_list[0];
+       return 0;
+}
+
+
 char * addr_to_host(struct in_addr *addr)
 {
        struct hostent *host;
@@ -1635,7 +1662,7 @@
 }


-char * port_to_service(int port, unsigned short proto)
+char * port_to_service(unsigned int port, unsigned short proto)
 {
        struct servent *service;

@@ -1650,7 +1677,7 @@
 }


-char * port_to_anyname(int port, unsigned short proto)
+char * port_to_anyname(unsigned int port, unsigned short proto)
 {
        char *name;
        static char buf[10];
@@ -1658,13 +1685,13 @@
        if ((name = port_to_service(port, proto)) != NULL)
                return name;
        else {
-               sprintf(buf, "%d", port);
+               sprintf(buf, "%u", port);
                return buf;
        }
 }


-char * addrport_to_anyname(struct in_addr *addr, int port,
+char * addrport_to_anyname(struct in_addr *addr, unsigned int port,
                           unsigned short proto, unsigned int format)
 {
        char *buf;
@@ -1678,6 +1705,28 @@
        } else {
                snprintf(buf, 60, "%s:%s", addr_to_anyname(addr),
                         port_to_anyname(port, proto));
+       }
+
+       return buf;
+}
+
+
+char * fwmark_to_anyname(fwm_t fwm, unsigned int format)
+{
+       char *buf;
+       struct hostent *h;
+
+       if (!(buf=malloc(60)))
+               return NULL;
+
+       if (format & FMT_NUMERIC) {
+               snprintf(buf, 60, "%u", fwm);
+       } else {
+               if((h = getfwmbyfwm(fwm)) == NULL) {
+                       free(buf);
+                       return NULL;
+               }
+               snprintf(buf, 60, "%s", h->h_name);
        }

        return buf;

--Boundary_(ID_7gtcNW35aSCKH29xS00dLw)--
```

30 FAQ

30.1 Help! My LVS doesn't work

30.1.1 Here's the new (Mar 2002) answer.

Setting up an LVS from scratch for the first time requires some thinking, is somewhat tedious and non-obvious and there are many ways of getting it wrong. Confounding the problem, there are several ways in which the LVS appears to work, but you are just connecting directly to the realservers rather than forwarding packets through the director. Understanding how an LVS works will require some investment in time.

Our answer till now has been to try to figure out what you've done wrong. This requires us to translate your posting into our language and then try to figure out what you've probably done wrong. We haven't seen anything new here for a long time, but it does take a lot of our time. (We still see plenty of things on the mailing list about how an LVS does or doesn't work that no-one has yet thought about.) Occassionally after 10 exchanges with 3 people trying to figure out what's wrong, the original poster will suddenly remember that they have filter rules that they "forgot to tell us about" and which they'd checked out thoroughly and were sure were working.

In the last 3 yrs, substantial effort has been put into the HOWTO, the mini-HOWTO and the 10.1 (configure script) to enable people with minimal or no understanding of LVS to setup a working LVS. There is also enough information here for you to setup a working LVS by hand from the command line if you understand how an LVS works.

We need to get on with our real jobs too, so I'd like to try something new: -

If you can't setup an LVS, with 2 realservers serving telnet, then rather than getting us to figure out what you've done wrong, how about you use the tools we've provided, to setup an LVS. Once you have a working LVS, then you can try to set it up your own way. If you can't get the scripts to work, then we'll try to fix those.

30.1.2 Here's the old answer

To you, all problems look the same ("it doesn't work"). We need more information than this. For suggestions look at the 1.10 (information needed to solve problems).

30.2 My LVS doesn't work: ipvsadm shows entries in InActConn, but none in ActiveConn

The usual mistake is to have the default gw for the realservers set incorrectly.

- VS-NAT: the default gw *must* be the director. There *cannot* be any other path from the realservers to the client, except through the director.
- VS-DR, LVS-Tun: the default gw *cannot* be the director - use some local router.

Setting up an LVS by hand is tedious. You can use the 10.1 (configure script) which will trap most errors in setup.

30.3 My LVS still doesn't work: what do I do now?

You have to debug your setup. If you have first setup a simple LVS using the mini-HOWTO and now you have your particular setup which doesn't work then:

First take your realserver offline (pull the network cable) and start the service listening on RIP:port (VS-NAT) or VIP:port (VS-DR, LVS-Tun). Note: no matter what service you'll run in production, as a test telnet is a much easier service to debug. I know you want your try out your superduper shockwave webserver with java applets, but you can get that to run later. Try with telnet first OK? It will save you a round of questions on the LVS mailing list.

Check that the service (initially telnet) is running (netstat -an). Connect to the service from a simple client running on the realserver (eg telnet VIP 80, lynx VIP). Then use your usual client (a web browser say). Make sure your service can do all the things that you expect the client on the internet to be able to do.

Then connect the realserver to the LVS network and set its default gw (to the DIP for LVS-NAT, to the router for LVS-DR, LVS-Tun), add the VIP:services to the director with ipvsadm (or the 10.1 (configure script)) using rr scheduling. Check that ipvsadm shows your realserver:service.

Make sure you have no firewalls between the client and the LVS, otherwise all bets are off.

Then connect to the VIP:service from a simple client (*e.g.* telnet VIP port), check that you can get to all the realservers one by one (watch on the director with ipvsadm). Then use your usual client for the service (eg a webbrowser).

30.4 initial connection is delayed, but once connected everything is fine

Usually you have problems with 17 (authd/identd). Simplest thing is to stop your service from calling the identd server on the client (*i.e.*disconnect your service from identd).

30.5 How fast/big should my director be?

There isn't a simple answer.

The speed of the director is determined by the packet throughput from/to the clients and not by the number of realservers. From the mailing list, 3-500MHz UMP directors running 2.2.x kernels with the ipvs patch can handle 100Mbps throughput. We don't know what is needed for 1Gpbs throughput, but postings on the mailing list show that top end UMP machines (eg 800MHz) can't handle it.

For the complicated answer, see the 5.2 (section on estimating director performance).

30.6 What is the minimum hardware requirements for a director

Enough for the machine to boot,*i.e* 486CPU, 8M memory, no disk.

30.7 Does the director handle ICMP?

yes. For LVS'ed services, the director handles 19.19 (ICMP redirects) and 19.19 (MTU discovery) delivering them to the correct realserver. ICMP packets for non-LVS'ed services are delivered locally.

30.8 I get "connection refused" from the client

This means that no service is listening for your client's requests and that some machine at the end is replying that it is not listening for that service. Possibly

- ipvsadm (on the director) has not added the service to the forwarding table (seen in the output of ipvsadm)
- if the service is in the ipvsadm table, then the director is forwarding packets to a realserver which doesn't have the service running.

The LVS is acting like a single machine which doesn't have the service running.

30.9 Does SMP help?

LVS is kernel code. In particular the network code is kernel code. Kernel code is only SMP in 2.4.x kernels. To take advantage of SMP for LVS then you must be running a 2.4.x kernel.

Michael Brown `michael_e_brown@dell.com` wrote on 26 Dec 2000

> I've seen significant improvements using dual and quad processors with 2.4. Under 2.2 there are improvements but not astonishing ones. Things like 90% saturation of a Gig link using quad processors. 70% using dual processors and 55% using a single processor under 2.4.0test.
>
> I haven't had much of a chance to do a full comparison of 2.2 vs 2.4, but most evidence points to >100% improvement for network intensive tasks.

30.10 When will LVS be ported to Solaris, xxxBSD...?

LVS is all kernel code. It's hard enough to write for one kernel. No-one is contemplating porting LVS to any other OS's.

30.11 Is there a HOWTO in Japanese, French, Italian, Chinese...?

One person offered to translate the HOWTO into Italian a long time ago and I haven't heard from him since. Another person recently (May 2001) offered to translate it into Japanese. From the number of Japanese HOWTO's I find in google searches, I'd say there is a good chance of having a Japanese HOWTO sometime.