LVS-mini-HOWTO

Joseph Mack

jmack (at) wm7d (dot) net

Table of Contents

Abstract

Purpose of this document

To show you how to install a Linux Virtual Server (LVS) and to set up a few demonstration virtual servers. No knowlege of the workings of LVS is needed or explained here. You will need to be familiar with configuring Linux. You are expected to use the instructions here to setup your first LVS, if you can't do it any other way.

Once you can get an LVS to work, you should read the docs and the LVS-HOWTO to understand how an LVS works and to setup the LVS you want. If you have trouble with your own LVS setup, read the instructions in the LVS-HOWTO introduction about getting help, before posting to the mailing list.

1. Introduction

Everything you want to know (code, docs, mailing list, mailing list archives) can be found somewhere on the LVS website

The neccessity of this mini-HOWTO was pointed out by ratz *ratz (at) tac (dot) ch* who made suggestions for content and who proof-read the text. Other suggestions came from John Cronin *jsc3 (at) havoc (dot) gtf (dot) org*. This originally was a set of instructions to get an LVS up quickly without having much of an idea of how an LVS worked. However this lead to duplication of instructions in the HOWTO and the mini-HOWTO. Eventually it became easier to move all the instructions to one place. Now the mini-HOWTO contains more than the minimum you need to know to set up a working demonstration LVS.

This document is written in xml.

1.1. Location of this document

On the

LVS-HOWTO website and also linked from the

LVS documentation page.

1.2. What an LVS does

An LVS is a group of servers with a director that appear to the outside world (a client on the internet) as one server. The LVS can offer more services, or services of higher capacity/throughput, or redundant services (where individual servers can be brought down for maintenance) than is available from a single server. A service is defined here as a connection to a single port, eg telnet, http, https, nntp, ntp, nfs, ntp, ssh, smtp, pop, databases. Multiport services (eg ftp for LVS-NAT) are also handled as a special case. Other multiport protocols (*e.g.* passive ftp, http/https for e-commerce sites) are handled by LVS persistance or more recently by

fwmark in the HOWTO.

In the computer bestiary, and LVS is a layer-4 switch. Standard client-server semantics are preserved. Each client thinks that it has connected directly with

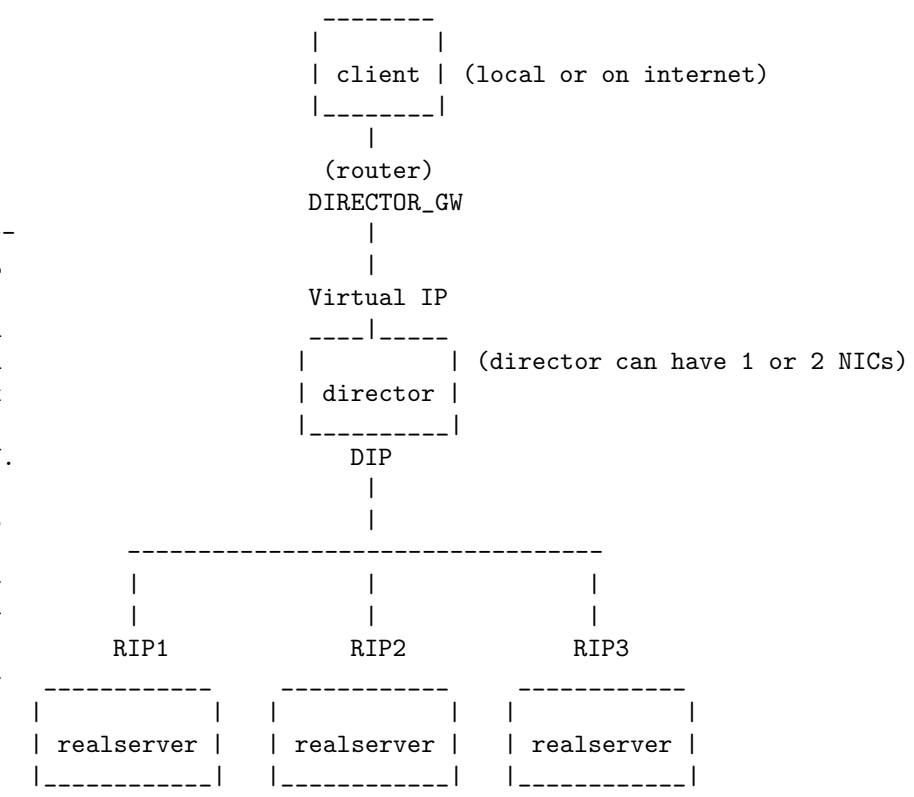
the realserver. Each realserver thinks that it has connected directly to the client. Neither the client nor the realservers have any way of telling that a director has intervened in the connection.

An LVS is not a beowulf - a beowulf is a group of machines each of which is cooperatively calculating a small part of a larger problem. It is not a cluster - a cluster of machines is a set of machines which cooperatively distribute processing. The realservers in an LVS do not cooperate - they have no knowlege of any other realservers in the LVS. All a realserver knows about is that it gets connections from a client.

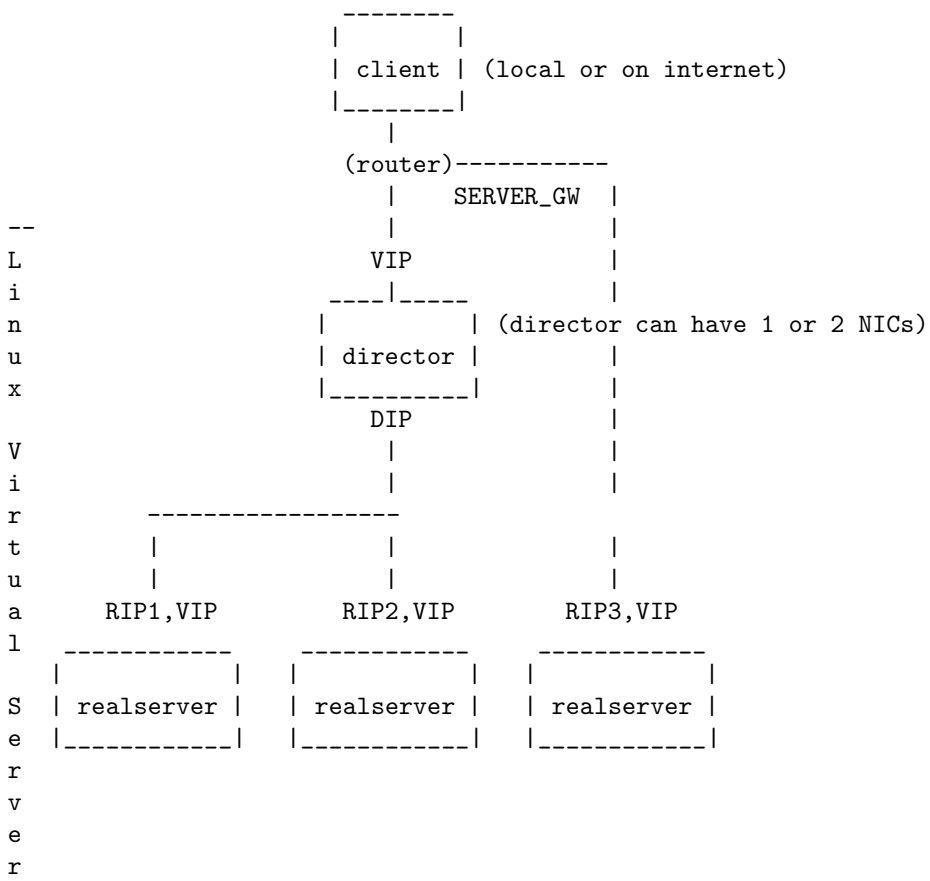
This mini-HOWTO, will demonstrate LVS setup for the services telnet and http.

2. Minimum Hardware/Network requirements

Here's a typical LVS-NAT setup.



Here is a typical LVS-DR or LVS-Tun setup.



For director failover (covered in the

LVS-HOWTO), the VIP and DIP will be moved to a backup director. These two IPs then cannot be the primary IP on the director NIC(s), *i.e.* they must be secondary IPs (or in the old days were called aliases). The primary IP's on the director NIC(s) can be anything compatible with the network.

Any setup procedure (including the configure script) will assume that you have already setup all IPs and network connections, apart from the VIP,DIP: *i.e.* setup RIP1..n, the primary IPs on the director NIC(s) and the DIRECTOR_GW/SERVER_GW and that the link layer joining these IPs is functioning. The configure script will setup the VIP, DIP (0.9.x will set them up on aliases, hopefully later versions will set them up on secondary IPs).

In a test setup, with all machines in the same room, a router (in the ascii diagram) will not be present and the client box will have the IPs for the DIRECTOR_GW or the SERVER_GW.

You will need a minimum of 3 machines (1 client, 1 director, 1 realserver). If you have 4 machines (*i.e.* 2 realservers), then you can see loadbalancing (the

connection from the client to the LVS being sent to one realserver, then the other).

You can setup an LVS with 2 machines (1 client, 1 director) using the

localnode feature of LVS, with the director also functioning as a realserver, but this doesn't demonstrate how to scale up an LVS farm, and you may have difficulty telling whether it is working if you're new to LVS.

The director connects to all boxes. If you have only one keyboard and monitor, you can set everything up from the director.

You need

- **Client**

  Any machine, any OS, with a telnet client and/or an http client. (*e.g.* netscape, lynx, IE). (I used a Mac client for my first LVS).

- **Director:**

  machine running Linux kernel 2.2.x (x¿=14) or 2.4.x patched with ipvs. If you're starting from scratch, use a 2.4.x kernel, as **ipchains** (as is used in 2.2.x kernels) is no longer being developed, having been replaced by **iptables**.

  For a test, this machine doesn't need to be powerful - any 486 machine on 10Mpbs ethernet is fine. In real life a 75MHz Pentium I is capable of delivering 50Mbps of packets using 100Mbps ethernet. In this case the network layer (33MHz bus) is the bottleneck and the CPU is doing little work. Compare this throughput to the capacity of a T-1 connection (1.5Mbps) and you can see that for low capacity networks, the redundancy of the LVS is the most useful feature. It's not till the network layer is 1Gbps that CPU resources become limiting on 300MHz PII's.

- **Realserver(s):**

  These are the machines offering the service of interest (here telnet and/or http). In production environments these can have any operating system, but for convenience I will discuss the case where they are linux machines with kernel ¿=2.2.14. (Support for earlier kernels has been dropped from the HOWTO).

  The director can forward packets by 3 methods. The OS for the realservers can be

  - LVS-NAT - any machine, any OS, running some service of interest (eg httpd, ftpd, telnetd, smtp, nntp, dns, daytime). The realserver just has to have a tcpip stack - even a network printer will do.
  - LVS-DR - OS known to work are listed in the
    HOWTO: almost all unices and Microsoft OS.
  - LVS-Tun - realserver needs to run an OS that can tunnel (only Linux so far).

2.1. Link layer

Must be ethernet, can be coax or twisted pair with a hub or switch. Only 10Mbps ethernet is needed for a demonstration. (ATM doesn't work, at least yet, see the HOWTO).

2.1.1. Choose number of networks There are two logical networks to consider.

- **VIP**: The network that the director's VIP is on. The client connects to the the VIP (and must be able to route to it). For a LVS webserver, the VIP will be a public, routable IP. If the LVS is local, then the VIP will be in one of your own networks (*e.g.* a private network). In the examples here, the VIP is on a private network, for convenience of testing. In production, the VIP will usually be a public IP.

- **RIP**: The network that the realservers are on. In general these will be private IPs, since the client doesn't connect to the realservers directly and doesn't need to know that the realservers even exist (the client only thinks there is one machine at the IP of the VIP).

If the client(s) are on the same network as the realservers, then the two logical networks can be collapsed into one network.

Note The VIP is usually on an ethernet device (*e.g.* eth0). However *dec (at) rm-f (dot) net* 01 Jun 2002 put it on lo and the LVS worked just fine.

2.1.2. Choose number of NICs on director The number of networks (1 or 2) is a separate matter from the number of NICs. A NIC can have multiple IPs and be on multiple networks. You can have 1 NIC on the director with a 2 network LVS.

For an initial LVS-DR setup, you only need 1 NIC on the director. For an initial LVS-NAT setup, you need 2 NICs on the director (to use one NIC with LVS-NAT, see

One Network LVS-NAT). Having 2 NICs on the director separates outside and inside packets, simplifying filter rules and security. 100Mbps NICs are cheap and for production, you can have as many NICs as you can fit into your motherboard (there are some nice quad NICs, I use the D Link quad tulip on my director). The packet throughput will be limited by the PCI bus/tcpip stack (400Mbps for mobos in 2000) - *i.e.* 4 NICs. The director could have a NIC for each realserver even if each NIC is not being used at full speed. Increasing the number of NICs will increase throughput as long as something else doesn't become rate limiting, eg the CPU speed or the PCI bus.

The configure script will handle 1 or 2 NICs on the director. In the 1 NIC case, the NIC connects to both the outside world and to the realserver network. In the 2 NIC case, these two networks are physically separated, one connecting to the outside world, and the other connecting to the realservers.

2.1.3. `nameif` Change names of NICS eth0¡-¿eth1 If you need to exchange the names of two NICs (*e.g.* two NICs on a mobo)..

Peter Mueller *pmueller (at) sidestep (dot) com* 03 Dec 2004

I just saw on another mailing list about a tool called nameif. nameif: renames network interfaces based on mac addresses. When no arguments are given /etc/mactab is read. Each line of it contains an interface name and a Ethernet MAC address. Comments are allowed starting with #. Otherwise the interfaces specified on the command line are pro- cessed. nameif looks for the interface with the given MAC address and renames it to the name given.

Jacob Coby *jcoby (at) listingbook (dot) com* 03 Dec 2004

I just looked at the source for `ifup`, and it calls `nameif`

```
if [ -n "\${HWADDR}" ]; then
     FOUNDMACADDR=‘LC_ALL= LANG= ip -o link show \${REALDEVICE} | \
         sed ’s/.*link\/ether \([[:alnum:]:]*\).*/\1/’‘
     if [ "\${FOUNDMACADDR}" != "\${HWADDR}" ]; then
         /sbin/nameif "\${REALDEVICE}" "\${HWADDR}" || {
            echo \$"Device \${DEVICE} has different MAC address than
expected, ignoring."
            exit 1
         }
     fi
fi
```

Looks like the HWADDR directive will work for me if I reboot the machine. Thanks Peter, and secroft@micron.com!

2.2. Gotchas: you need an outside client (the director and realservers can't access the virtual service)

To set up and test/run LVS, you need a minimum of 3 machines: client, director, realserver(s).

From the outside, the LVS functions as one machine. The client cannot be one of the machines in the LVS (the director, or realserver). You need an outside client. If you try to access an LVS controlled service (eg http, smtp, 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.

For limited conditions, people have found ways around this restriction.

- Apr 2003: Jacob Rief has figured out

  how to have a realserver be a client with LVS-NAT.

- May 2004: Joshua Goodall has figured out

  how to have a director be a client with LVS-DR.

- Ludo has

  proposed a scheme to allow an LVS-DR realserver to be an LVS-DR client.

2.3. Initial service for testing should be telnet (or netcat)

All testing of your LVS should be done with simple services (*telnet, http*). 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
- 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 forwarded by LVS, then you should fix this first.

Another useful client is
netcat or its replacement, phatcat.

2.4. Test without filter (iptables) rules

You don't need them to set up a standard LVS. You can add them after you get the LVS working. They usually stop the LVS working and result long exchanges and a great waste of time on the mailing list.

Somsak Sriprayoonsakul *somsaks (at) gmail (dot) com* 06 Feb 2006

The problem has been solved. It's something related to iptables. Stopping iptables on director and the connection rate goes from 200 to Nx2000, where N is the number of real server. After that, I tried to figure out which iptables rules conflict with ipvs and found that it's default argument generated from system-config-securitylevel that cause this. Replace "-m state –state NEW -m tcp -p tcp –dport 80" with just "-m tcp -p tcp –dport 80" make everything works perfectly.

Ratz

It's not a conflict, it's the connection tracking core which is extremely slow. There's ongoing effort from the netfilter people to improve this state. If you want high performance load balancing, do _not_ use netfilter; especially the connection tracking. It just does not scale. Simply loading ip_conntrack into the kernel makes your packet rate drop by 60 kpps on a 1Gbit/s connection.

unknown

If it can help, here is a small document named "netfilter conntrack performance tweaking" : http://www.wallfire.org/misc/netfilter_conntrack_perf.txt

Ratz

No, I'm afraid it does not help ;). A conntrack entry is at least 192 bytes. If I have 800kpps traffic flow and 5000 rules, even Herve's tuning tips are not helping anymore. I've seen some improvements regarding RCU conversion lately and I definitely need to redo my tests with a recent 2.6.x kernel, however I have little hope regarding throughput with ip_conntrack.

Graeme Fowler *graeme (at) graemef (dot) net* 06 Feb 2006

That's because in DR mode the connection state as seen by the director never reaches ESTABLISHED - in the majority of cases, the packets returning from realserver to client do not go via the director. Your new rule ignores the connection state and simply processes packets destined for port 80 regardless. This is what you want, as you've seen.

2.5. ip aliases

Since the 2.0.x kernels, you have been able to put multiple IPs onto an ethernet device (*e.g.* lo, eth0) with ip aliases (*e.g.* lo:0, eth0:1). Starting with the 2.4.x kernels, ip aliases are deprecated, although still work. It seems with the 2.6.x kernels, that ip aliases no longer work (there's some conflict here - the configure script works with 2.6.x it seems) and you have to use the `iproute2` tools which acheive the same functionality as ip aliases by using primary and secondary addresses. This documentation is written for ip aliases. Be aware that eventually, you'll have to substitute the iproute2 commands for the ip alias commands written here.

3. Software

3.1. Compiler

Wensong is using egcs-2.91.66 on RedHat 6.2 and gcc-2.96 on RedHat 7.3.

May 2003 - I have had a lot of trouble with newer kernels and gcc-3.x. Kernels build but don't work properly (on one of my machines, the scsi, and networking don't work when I turn on SMP). The number of parameters in ip_select_ident has changed leading to compile failures (the new parameter has the value '0', I just deleted it - it compiled but I don't know if the code will bight me now). I also find that some packages won't compile with gcc-3.3.

Feb 2004 - whatever the problems were, they seem to have gone now. Apr 2004 - seems that gcc-2.95.3 is still the recommended compiler for kernels. Use others at your own risk.

3.2. Use the standard kernel

You need the standard kernel for both the director and the realservers. The standard kernel is obtainable from the kernel ftp site.

If you're using a market enhanced version of the kernel (*i.e.* from a Linux distribution), it will likely **not** patch correctly with LVS. The Redhat kernel has LVS patches pre-applied, with a version which will be old by the time you get to talk to us. SuSE (May 2003) is doing the same thing. If you can get LVS

to work with the standard kernel, but not with your distribution, 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're really interested in getting the kernel in your distro working with LVS, you can do that after you've done it with the standard kernel. If you come up on the mailing list with a problem and you're using a market enhanced version of the kernel, make sure you tell us that you're using a non-standard kernel.

3.3. Use standard utilities (not ifup)

Use standard utilities, *e.g.* the `iproute2` tools (or the older **ifconfig/route** commands, which really only work well for a leaf node with one IP). Note We've been setting up LVSs since the first days with **ifconfig** and ip_aliases, but ip_aliases are not compatible with most current networking tools, so be careful if you're using **ifconfig**.

People find that their LVS doesn't work if they set it up with RedHat's **ifup** (see the LVS archives). The problem is that **ifup** runs **arping** which it has no business doing, and which hoses your LVS setup. Other market enhanced networking tools add strange routes. If you want to outsmart yourself by using commands that are designed to enhance the uniqueness of a distro in the market place, rather than to do the job that needs to be done, then you fully deserve all the trouble you get into, but we on the mailing list have to figure out how you broke your setup and we assume that you're using the normal utilities. We aren't happy to find out after many exchanges that you're using deliberately broken tools written by people who don't understand networking. You may as well be running Windows.

hapless user

Before ran the script, I can connect to all my realservers, but after running it I lost connecion and the only difference in the route table is this:

```
\$VIP dev lo  scope link  src \$VIP
```

Francois JEANMOUGIN *Francois (dot) JEANMOUGIN (at) 123multimedia (dot) com* 16 Jan 2004 and Apr 07, 2004

You should not add any lo route. Never use ifup to mount noarp interfaces. In the ifup script, you will find an arping command. If this is the first realserver you are configuring, this is not really a problem, but if you try to add a realserver to an existing pool of servers, this will mess all the LVS routing structure by sending an arp announce.

Christopher DeMarco *cdemarco (at) md (dot) com (dot) my* 07 Jul 2004

RedHat uses **ifup** at startup.

```
[08:55:05][root@sunset root]# cat /etc/redhat-release
Red Hat Enterprise Linux WS release 3 (Taroon Update 2)
[08:55:14][root@sunset root]# grep  -n ifup /etc/init.d/network
73:     action \$"Bringing up loopback interface: " ./ifup ifcfg-lo
137:            action \$"Bringing up interface \$i: " ./ifup \$i boot
148:            action \$"Bringing up interface \$i: " ./ifup \$i boot
```

Why not just comment out the bits of **ifup** that invoke arp(8) and arping(8)? It's not ideal insofar as any system upgrades will need to re-clean the **ifup** code,

but to my mind it's the least invasive procedure. Re-writing clean init scripts will break the entire RH point-n-drool configurization functionality that users may want to use. My understanding is that the manipulation of arp caches is the single impediment to using RH products with LVS, or are there others?

Horms

Yes, that is certainly what I would do. Perhaps RH could be encouraged (through a bug report on bugzilla) to make this a configuration option. RH is the only distribution that I am aware of that does this. But it seems reasonable to think that others might do the same.

3.4. getting files in packages

If you want files in a package format for some distribution, in general you'll have to go to that distributor. Sometimes people on the mailing list package up the files for general use. LVS is a spare time activity for some of us. We're happy to generate files that everyone can use, but there's not much point in generating packages for the myriad of distributions, when we can't test them and we don't even know if anyone will use them.

My personal view is that unless you can compile the code, you may as well be running Windows. I know that some of us are subject to management who wants code packaged from an approved distributor, and are prepared to pay money for it. We aren't set up to do that.

3.5. Getting Files: Director

Software is available at

http://www.linuxvirtualserver.org/software/ipvs.html.

The director code is well tested on Intel CPUs. As well several people are running on Alpha hardware. It is expected to work on any hardware which runs Linux.

3.5.1. Kernel Patch You need a kernel patched with the ip_vs code

- 2.6.x kernels:

  the kernel (from ftp.kernel.org) is already patched with the ip_vs code.

- 2.4.x kernels:

  for 2.4.23 and later, the kernel (from ftp.kernel.org) is already patched with the ip_vs code. Note Do not run **make patchkernel** on the 2.4.23 kernel (or later). These kernels are already patched with LVS. Despite its name **make patchkernel** is replacing files, rather than patching them and the command will succeed but produce a kernel which will not compile. Note 2.4.23 has a root exploit problem which has been fixed for 2.4.24. 2.4.25 had a fix for a root exploit too. It's probably best to skip 2.4.23/24 for production systems.

for 2.4.22 and earlier, you need to download the ip_vs patch for your kernel version and patch the kernel.

- 2.2.x kernels:

  all versions require you to download the ip_vs patch for your kernel version and patch the kernel.

In all cases, when you compile your kernel, you have to turn on the ip_vs option in **make configure** (Networking options -¿ IP: Virtual Server Configuration). If you don't know what you're doing, compile all options as modules and don't change the connection table size (see further down for more details on compile options).

Horms 14 Nov 2003

Generally each release is made in two forms (take your pick for kernel build style).

- A tar ball which you can use to build LVS outside or inside the kernel tree.
- And a patch which can only be used to build LVS inside the kernel tree.

For 2.4.22, patch `linux-2.4.21-ipvs-1.0.11.patch.gz` can be applied. Unless you've got good reason to use an older kernel, then use 2.4.23 or later with LVS code already in it.

3.5.2. ipvsadm: the user level interface to ip_vs You control and configure the kernel code with **ipvsadm**

- If you have a kernel prepatched with ip_vs code: download `ipvsadm` from http://www.linuxvirtualserver.org/software/ipvs.html.

  Note Make sure you get the `ipvsadm` for your kernel series. The various kernel verions (2.2.x, 2.4.x, 2.6.x) require different versions of `ipvsadm`. However there is only one numbering series for versions of `ipvsadm`. You can't tell the target kernel from the ipvsadm verion number (*e.g.* `ipvsadm-1.24.tar.gz` is for 2.6 kernels, while `ipvsadm-1.21.tar.gz` is for the 2.4 kernels). If you just get the latest `ipvsadm` (Jan 2004 is v1.24) it won't work on 2.4 kernels.

- If you are using an ip_vs patch to patch a kernel: `ipvsadm` will be included with the patch. Note Compile `ipvsadm` after you've compiled and booted to your new ip_vs kernel, so that `ipvsadm` is compiled against the correct version of the kernel header files. This will require you to remember that after you boot to your new kernel, you still have one more step, of compiling and installing `ipvsadm`.

3.5.3. 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, you have to bring the director down.

I use the **rc.system_map** script (comes with the 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).

Horms 01 Feb 2005

LVS is part of the kernel. You can compile it as modules or into the kernel itself. In the case of the former, you can change these modules at any time, though LVS gets reset and any establised connections will be broken - though this can be mitigated by synchonising to a backup linux director. However, if you need to upgrade some other part of the kernel, which is more common than just upgrading LVS, then you need to reboot. This includes a situation where updating LVS relies on a change somewhere else in the kernel.

3.5.4. LVS in CVS The IPVS CVS respository is available at

```
cvs -d :pserver:anonymous@cvs.linuxvirtualserver.org:/home/lvscvsroot login
(type anything for the password of user anonymous)
cvs -d :pserver:anonymous@cvs.linuxvirtualserver.org:/home/lvscvsroot co ipvs
```

3.6. Realserver OS by forwarding method

Realservers can have almost any OS. Here they are listed by forwarding method

- LVS-NAT - any OS. Since only one IP is needed, anything with a tcpip stack is OK. You can use a networked printer as a realserver.
- LVS-DR - most OS's (all expected to succumb eventually).
- LVS-Tun - only linux. You need the tunl0 device to decapsulate the ipip packets (turn on in the kernel configuration under "Network options"). Windows had tunneling for a short while in their later versions, but then dropped it.

The problem for realservers is handling the arp problem for LVS-DR and LVS-Tun. The problem arises because all machines (director, realservers) have the VIP and only the director can reply to arp requests for the MAC address of the VIP, if the LVS is to work. Ratz has assembled instructions for setting up realservers using non Linux OS. All the standard unices (except Linux in 2.0 kernels and later, and HPUX) honour the `-noarp` flag to **ifconfig**, so handling the arp problem is trivial. For linux you need to apply software to the realserver kernel, or use routing tricks (Lars method).

3.7. Realservers: Do you need to handle the arp problem?

- LVS-NAT: no arp problem (the realservers don't have the VIP)

- For realservers with LVS-DR (and possibly LVS-Tun), you'll need to handle the ARP problem.

The theory for the arp problem can be skipped on a first read. Eventually you'll need to understand this to set up your own configuration of 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, 2.6.x kernel on a realserver)
       AND
       (
               the VIP on the realserver is on an ethernet device eg lo:0, tunl0:0
               i.e. packets to the VIP are 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|segment as the director)
       )
    )
THEN
    {
    YOU MUST HANDLE THE ARP PROBLEM
    }
FI
```

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

3.8. Realservers: arp problem by Linux kernel version

- **2.0.x**: honours `-noarp` flag to **ifconfig**. No kernel patches needed. Use the NOARP option in ifconfig to setup VIP on lo:0 (as for other unices)

- **2.2.x**: does not honour `-noarp` flag.

  Julian's `hidden` patch is in the standard kernels for 2.2.x (where x¿=12). You do not need any kernel patches.

  You hide the device from arp calls by bit twiddling the `/proc` filesystem. This gives the same effect as configuring the device with the `-noarp` option.

- **2.4.x**: does not honour the `-noarp` flag.

  There are **no** patches in the kernel to handle the arp problem.

  You can patch the kernel with Julian's hidden patch or load Maurizio's `noarp` module. For 2.4.26 and later, Julian has another patch (see 2.6.x).

- **2.6.x**: does not honour the `-noarp` flag.

  There are **no** patches in the kernel to handle the arp problem.

  For 2.6.x Julian has new code

  Julian Anastasov *ja (at) ssi (dot) bg* 25 Feb 2004

2.4.26pre and 2.6.4pre will come with 2 new device flags for tuning the ARP stack: `arp_announce` and `arp_ignore`. All IPVS like setups can use arp_announce=2 and arp_ignore=1/2/3 to solve the "ARP problem" with DR/TUN setups. These flags are going to replace the "hidden" functionality which does not work well for directors when they are changing their role between master/slave for a particular VIP. The risk is that other hosts can probe for VIP using unicast packets which the hidden flag always replies. I'll continue to support the hidden flag for 2.4 and 2.6 to help existing setups but switching to the new device flags (or other solutions) is recommended.

3.9. Realservers: Methods of handling the arp problem

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

The methods in current use for handling the arp problem

- Julian's hidden patch available at

  Julian's patches and software page. This stops replies to arp requests by device (*e.g.* lo, where all IPs on the device are hidden). This is the oldest of the methods in use and so it most familiar to people who've been using LVS for long time.

  As of late 2001, both the hidden patch and the ipvs patch can be applied simultaneously to the kernel, allowing you to use the same kernel for the director and realservers. The part of the kernel being patched doesn't change much with kernel versions, so the hidden patch against 2.4.5 is still OK against 2.4.19pre4.

  You patch by doing

  ```
  realserver:/usr/src/linux# patch -p1 <../arch/hidden-2.4.4-1.diff
  ```

- Maurizio Sartori's noarp module . `noarp` module. You don't have to patch the kernel, you just load the module. Maurizio's module acts on the IP, not the device, so you can make the VIP `noarp` while letting other IPs on the same device reply to arp requests. This is currently (Feb 2004) available for 2.4.x kernels, with a version for 2.6.x expected shortly.

- Julian's arp filtering. Julian has added arp extensions to the iptables filtering rules. This would appear to be the simplest way of handling the problem, except that people are wary of writing iptables rules. No-one seems to be using this method.

- Routing tricks *e.g.*Lar's method.

- For earlier kernels (2.0.x and for at least early 2.2.x kernels), 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 kernel with the hidden patch. (Note, 2002: - this doesn't work for 2.4.x kernels).

There is some dispute (from Ratz and Julian) whether this method should even work (they say it won't). However the results of the tests are in This property is probably moot now, since no-one is using these kernels anymore.

3.10. Why netmask=/32 for the VIP in LVS-DR?

Horms 29 Oct 2003

If you are using LVS-DR then the packets that arrive on the real servers have the destination IP address set to the VIP. So the real servers need some way of accepting this traffic as local. One way is to add an interface on the loopback device and hide it so it won't answer ARP requests.

The netmask has to be 255.255.255.255 because the loopback interface will answer packets for _all_ hosts on any configured interface. So 192.168.1.110 with netmask of 255.255.255.0 will cause the machine to accept packets for _all_ addresses in the range 192.168.1.0 - 192.168.1.255, which is probably not what you want.

Josef Pospisil Jul 15, 2005

Why does the VIP on the realservers need to be on a lo interface and not like on director for example on eth0:05?

Horms

You can do that if you would rather, however be careful to make sure that the real server does not make arp adverisments for the VIP. If you use lo, then this can easily be done using arp_ignore and arp_announce. If you want to use eth0, then arptables is probably the way to go.

3.11. Choose LVS Forwarding Type: LVS-NAT, LVS-DR and LVS-Tun

The realservers must be configured appropriately for the LVS forwarding method. To do this you must

- handle the arp problem.

  For more info see

  arp problem.

- set the default gw.
  - LVS-NAT: the DIP
  - LVS-DR, LVS-Tun: a router (anything but the director).

    If you apply

Julian's martian modification , you can route packets back through an LVS-DR director. You don't want to do this for your first LVS.

The examples below will show how to setup LVS-NAT and LVS-DR on a 1 NIC director.

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 2.2.x kernels, LVS-NAT is much more CPU intensive than LVS-DR (LVS-NAT requires packet rewritting). For 2.4.x kernels LVS-NAT has better performance and approaches that of LVS-DR. However for a simple test, LVS-NAT only requires patching 1 machine (the director) and an unmodified machine of any OS for a realservers.

For production LVS-DR is the first choice. 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 (remote or	on internet local)	local -
realserver number	low	high	high
client connnects to	VIP	VIP	VIP
realserver default gw	director	own router	own router

Until you know which forwarding method to use, choose in this order

- VS-DR default, has high throughput, can be setup on most OS's. The realservers must be on the same network as the director (the realservers and director can arp each other).
- VS-NAT, lower throughput, higher latency. Realserver only needs tcpip stack (ie any OS, even a network printer). Works for multiple instances of services (*i.e.* 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).

3.12. multiple forwarding methods

Although it is not often done, you can set up a director to forward services using different forwarding methods at the same time. Thus you could have telnet forwarded by LVS-NAT to one set of realservers, and http forwarded by LVS-DR to another (possibly overlapping) set of realservers.

3.13. Configure scripts, tools

There are some tools to help you configure an LVS.

- Joe wrote configure script for all LVS types, which does a lot of sanity checking on setup but doesn't handle director failover (single director only). The configure script will setup **mon** to handle failure of services on realservers.
- tools which include director failover, *e.g.* Ultra Monkey. by Horms, which handles director failover, but has to be setup by hand. In Apr 2005, Horms released UltraMonkey v3 (http://www.ultramonkey.org/download/3)
  - Connection Syncronisation allows connections to continue even if the active linux-director fails and the stand-by is brought online
  - Native health checking of MySQL servers using ldirectord
  - Improved handling of ARP packets
  - Custom kernel is no longer required (the LVS patches are already in the standard distribution kernel and Horms doesn't have to distribute his own kernel anymore)

  According to some survey that Horms read, UltrMonkey is the most often used method of setting up an LVS.
- vrrpd/keepalived by Alexandre Cassen, which sets everything up for you and is at the keepalived site. It handles director and realserver failure.
- Sylvain Maugiron *sm (at) edatis (dot) net* 05 Feb 2003, has released a cgi interface.
- Malcolm Turnbull *malcolm (dot) turnbull (at) crocus (dot) co (dot) uk* 28 Jan 2003, has written a php based web interface to lvs/ldirectord
- Per Andreas Buer *perbu (at) linpro (dot) no* 27 Jan 2003 has written a CLI-controlled LVS demon lvs-kiss which uses **ipvsadm** for load balancing and fail-over.
- Horms wrote lvs-gui. This is from the early days of LVS, only sets up a LVS-DR LVS, is no longer being maintained and should not be used for current work.

- Sylvain Maugiron *sm (at) edatis (dot) net* 27 Jan 2003

  I've placed my cgi interface to **ipvsadm** at Virtux I will try to go on developing this interface.

Unfortunately several of these scripts produce files with the name `rc.lvs` (just to make things interesting when reading the mailing list).

A survey of production LVS's (by Lorn Kay *lorn_kay (at) hotmail (dot) com*, results on the

LVS website) 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.

3.13.1. Configure script 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.

The configure script is designed to quickly (so I could do testing) setup an LVS with `mon` directed service failover. It's perl and writes out a shell script, which you run on all machines (it figures out whether it's running on a director or a realserver). This means that you don't need perl running on the realservers. The current version (0.9.x) does a wide range of checks, catching all the errors and deliberately pathological cases I could find in the last 2 yrs. Elementary checks are done on routing and connectivity and the script has always produced a working LVS when no errors are reported.

The configure script is run from the director console (or on the director over a connection from the RIP network or through an admin NIC that not used in the LVS). tcpip connections through the NIC facing the internet will be broken.

The configure script is on the

LVS software page (down at the bottom). This only sets up a single director and cannot handle director failover. For production systems you will probably want to use tools that handle

director failover. Note

the configure script tests and uses the links between all machines - the network should be working already. You should be able to ping between all machines on the same network. You will need NICs configured with all the RIPs, and the SIP and PIP (for versions 0.10.x and older) or DIP (0.9.x and earlier).

I never released v0.10.x.

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. 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 to produce the `rc.lvs` file. The `rc.lvs` file is then run on the director and then the realservers. This allows the `rc.lvs` script to check the connections to the now configured director.

I have the configure files/directory nfs exported to the realservers. After running the `rc.lvs` script on the director, I can then run the same `rc.lvs` script from the realservers. (After setup, the network connections neccessary to export this directory can be removed.) Alternately the configure script will copy and execute the `rc.lvs` file on the realservers using **ssh**, using the "-i" option (to check that this will work run **'ssh realserver_name_or_IP hostname'** - you should get back the hostname of the realserver). The `rc.lvs` file knows whether it's running on a director or realserver by looking for IPs on the machine it is running on and comparing them to the IPs in the conf file.

The `rc.lvs` script is verbose and reports on its progress. Watch for any errors. The script is idempotent *i.e.*- you can run the script as many times as you like on the same machine. You can copy the `rc.lvs` file from the director to the realservers for running. I have the conf files on the director in `/etc/lvs` directory (you can have them anywhere) which is nfs exported to the same directory on the realservers. In this way the `rc.lvs` script on the director can be run by the realservers. In a production system, you can umount the directories on the realservers after setup. Alternately, if you run configure with the -i (install flag), the configure script will use ssh to copy the output files to the realserver and execute the script on the realserver.

The configure script has been tested with directors on 2.2 and 2.4 kernels. It will issue warnings when it encounters a kernel later than it knows about. These warnings can usually be ignored as ipvs is not expected to change its configuration method a whole lot.

The conf file and the configure script can use either IPs or hostnames (from `/etc/hosts`) and can use either port numbers (eg 23) or service names (eg telnet) (from `/etc/services`). 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. (Note: versions of the configure script from 0.10.x onward will move to the iproute2 tools and will not be using aliases. 'ifconfig' and 'netstat -rn' will not work for networking set up by the iproute2 tools - you'll have to use the iproute2 tools only).

Documentation on the configure script is in perldoc format.

```
director:/etc/lvs:\$ perldoc configure.pl
```

3.13.2. configure script may need help for the 2.6.x kernels tito *extlists (dot) e-inventa (dot) net* 26 May 2004

I solved the problem of the configure script not being 2.6 ready by changing just the rc.lvs_dr script generated by configure-lvs, hiding the version of my kernel and also modifying the check_function_in_kernel() function. The problem with this function was that it compares the result of searching with grep -c into System.map for ip_vs_init module, with 1, and in my kernel there are two references to ip_vs_init (ip_vs_init and ip_vs_init_hash_table)

Neamtu Dan 12 Mar 2006

I finally did it, I've run the script manually on my 2.6.x machine and I found the problem. It wasn't the ethernet aliases as you said, they work fine, it was about the iptables which weren't flushed.

3.13.3. Installing the configure script The script is perl and requires perl modules not needed in the demonstrations here. However the script will not run without them, so check that you have the required modules by running

```
\$perl -w configure.pl
```

If you're missing a perl module (*e.g.* DNS.pm) you'll get an error message like

```
director:/etc/lvs# ./configure lvs_dr.conf.IP.two_NIC_two_network -i
Can't locate Net/DNS.pm in @INC (@INC contains: \
/usr/local/lib/perl5/5.6.0/i586-linux /usr/local/lib/perl5/5.6.0 \
/usr/local/lib/perl5/site_perl/5.6.0/i586-linux \
/usr/local/lib/perl5/site_perl/5.6.0 /usr/local/lib/perl5/site_perl\
 .) at ./configure line 1365.
BEGIN failed--compilation aborted at ./configure line 1365.
```

Fetch the extra modules from the perl website or download them directly with the perl cpan module, which you can do with

```
director:# cpan
or
director:# perl -MCPAN -e "shell"
```

You will get the cpan prompt. Help, which you get by typing "help" at the cpan prompt, is cryptic. To get the hang of cpan, try the 'r' command, which will list the most recent updates for modules you currently have installed. You should be able to install `Net::DNS` with

```
\begin{scriptsize}
cpan> install Net::DNS
```

If your upgrade has any dependencies, you will be prompted to download them too.

The `rc.lvs` script uses `fping` to test connections. `fping` is available from

ftp://ftp.kernel.org/pub/software/admin/mon

The newer versions of ping have the functionality of fping and the current configure script tests for this functionality. For older versions of configure, if you don't want to download fping, you can use this script instead of fping (put it in /usr/local/bin and make it executable).

```
#!/bin/sh
#fping replacement

ping -c 1 \$1
#-----fping-------------
```

3.13.4. running the configure script 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. There are example conf files for the various forwarding types, number of networks and number of NICs. The format of the conf file is the same for all LVS setups - only the IPs/networks change when you change from a 1-NIC director to a 2-NIC director. Several example conf files are provided but the only thing that changes between the diagrams is the network diagrams (and targets for default gw), not the format of the conf file directives. 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.

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).

3.13.5. 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.

3.13.6. 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.

3.14. Install - General

3.14.1. 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).
Note

the DIP and the VIP will be moved to another director on director failover. These IPs will be setup as secondary IPs (aliases in the language of 2.0 and 2.2 kernels) so that you can later use director failover. The configure script sets up these IPs for you (*i.e.* you shouldn't have the VIP and the DIP as primary IPs already installed on your director).

Julian Anastasov *ja (at) ssi (dot) bg* 06 Nov 2001

If DIP==VIP the realserver will receive packets with saddr=local_ip (VIP) from the director. (This is the "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).

3.14.2. What you'll be doing Chris *chrisd (at) better-investing (dot) org* 15 Apr 2002

1. I downloaded the kernel from kernel.org.
2. I downloaded the ipvs patch (use the single patch)
3. I downloaded the hidden patch.
4. I uncompressed the kernel source in /usr/src
5. I uncompressed the hidden patch in /root
6. I uncompressed the ipvs single file.
7. I copied the hidden patch to /usr/src/linux
8. I executed: patch -p1 ¡hidden-2.4.5-1.diff
9. I copied the ipvs patch to /usr/src/linux

10. I executed: patch -p1 ¡linux-2.4.18-ipvs-1.0.2.patch
11. I did a make menu config, and configured my kernel
12. I did the exact same steps you did to compile it:

```
make dep;make clean;make bzImage;make modules;make modules_install
```

I didn't have any glitches.
Things to watch out for:

- Don't apply each patch more than once!
- If it didn't work following these steps and you have RH 7.2, make sure you have all the gcc packages installed. If you must, put the cd back in the drive, and do an "upgrade", and install the tools necessary
- If that isn't it, make sure your gcc is up to date. An easy way of doing this is by downloading red-carpet from Ximian.org. It is an Xwindows updater. You do not have to have gnome to run this (anymore).
- If this doesn't work, track down a *real* guru, cuz I am fresh out of ideas!

3.15. Director kernel - build it yourself

(with suggestions from John Cronin *jsc3 (at) havoc (dot) gtf (dot) org*) Note This is the preferred method. Get a fresh kernel from ftp.kernel.org and get the matching `ip_vs` code from the LVS website (on the software page). Note

You can build the IPVS code into the kernel or as a loadable module, giving two versions of `ip_vs` for each release. The two versions of the code are not always released together (usually there is only a small delay between them). Note

From Horms: If you want to fiddle with the ipvs code, (*e.g.* to change the hash table size, which we don't recommend you do) and you are compiling ipvs as a module, then you only need to recompile and reload the ipvs module for the changes in the module to take effect. You do need to patch and recompile the kernel in order to use ipvs. However for 2.4.x kernels, it is worth observing that this is only so that a symbol that ipvs needs to be able to hook into netfilter is registered. The kernel does not know or care about the code in the ipvs modules when it is compiled (unless you are compiling ipvs straight into the kernel). In the 2.2 kernels ipvs did a lot more patching.

You must start from a clean source tree (just downloaded or run **make mrproper**). If you have previously configured the kernel for your hardware without ipvs, you should reuse your `.config` file. It will be patched, along with the kernel code (giving a new entry in the networking section), and your non-ipvs settings will be maintained. (Keep a backup of your `.config` incase of problems).

Apply the ipvs kernel patch using the instructions in the ipvs tarball. You'll do something like

```
director:/usr/src/linux# patch -p1 <../ipvs-1.0.6-2.2.19/ipvs-1.0.6-2.2.19.patch

or

director:/usr/src/linux# patch -p1 <../ipvs-0.2.7-2.4.2/ipvs-0.2.7-2.4.2.patch
```

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.

3.15.1. Compile instructions The actual kernel compile instructions will vary with kernel patch number. You can build ip_vs directly into the kernel or have it as a loadable module - either will do for an initial setup.

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

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

3.15.2. 2.2.x kernels 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.

```
          [*] 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 LVS 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)
```

Do not change the IP masquerading LVS (hash) table size unless you know a lot more about ip_vs than we do. If you're still thinking of changing the hash table size, at least first read the

HOWTO .

Do all the other kernel stuff - make modules, install the modules, copy the new kernel (and optionally System.map, required by the configure script.) into / or /boot, and edit your boot loader (lilo, grub) files. Make sure you keep your old kernel and kernel config, so you can recover if all does not go well. Reboot to the new kernel. When loading the new kernel (with ip_vs built as modules), make sure the ip_vs* modules get loaded. The README in the kernel source tree has all the necessary info in there. i

3.15.3. 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**. The setup varies with different `ip_vs` patches. Make sure you read the docs. Note

The instructions below are for an early 2.4.x kernel. You can expect each kernel to be a little different. For 2.4.18 (Simon Young *simon-lvs (at) blackstar (dot) co (dot) uk*, 1 Oct 2002)

IP Masquerading lives under:

```
Networking options  --->
```

```
  IP: Netfilter Configuration  --->

  <M> IP tables support (required for filtering/masq/NAT)
  <M>   Full NAT
  <M>     MASQUERADE target support
```

All the LVS stuff should be under:

```
Networking options  --->
  IP: Virtual Server Configuration  --->
```

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)
```

Do not change the IPVS connection (hash) table size unless you know a lot more about ip_vs than we do. If you're still thinking of changing the hash table size, at least first read the HOWTO.

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
```

3.15.4. iptables/ipchains compatability problems Note

I have removed the ipchains option here. This was set to ¡M¿ in previous versions of the mini-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. You should not be compiling ipchains into 2.4 kernels anymore (2.4.x has been out since 1999).

ipchains under 2.4 makes conntrack slow (see HOWTO).

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.

3.15.5. compile kernel Do all the other kernel stuff - make modules, copy the new kernel into / or /boot (giving it a different name to your working kernel *e.g.* bzImage-0.9.4-2.4.12), edit lilo.conf and re-run lilo. Make sure you leave the old kernel info in lilo.conf, so you can recover if all does not go well. Reboot to the new kernel.

3.15.6. If building ip_vs outside 2.4 kernel, have working kernel first. 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 configure script), before building ipvs (if you're doing it separately) and ipvsadm.

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

3.16. How do I check to see if my kernel has the ip-vs patch installed?

Short answer: 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.

3.17. Listing symbols in the kernel

S.W.Seo *ohlyugen (at) yahoo (dot) co (dot) kr*

is there a method to see the symbols in a kernel image? **nm** shows the symbols in a file, but does not work on the kernel image.

Peter T. Breuer *ptb (at) oboe (dot) it (dot) uc3m (dot) es* 8 Mar 2003

In the image, or in the kernel? When it's running, `/proc/ksyms` is what you want. But sure, you can apply **nm** to `vmlinux` and it will show you the symbols. I suspect you are talking about the compressed image plus boot header, `vmlinuz`. You'll have to behead and uncompress it first.

3.18. ipvsadm

ipvsadm is part of the ipvs tarball and is the user interface to LVS. You run ipvsadm on the director.

Before you attempt to build ipvsadm -

- reboot to your newly patched ip_vs kernel

  otherwise you'll get errors about missing ip_vs header files (you'll be using header files from a previous version of the kernel).

- Check that you really are in the new patched kernel.

  Do **uname -a**, or if you have compiled ip_vs as a module, do **lsmod — grep ip_vs**.

Do a **make install** for ipvsadm. Make sure the ipvsadm you are using is from the same ip_vs tarball as the kernel patches (if you just make the new kernel and forget to install the new ipvsadm you'll have the old ipvsadm with the new ipvs code). Since you can't compile/run ipvsadm till the you're running under the new kernel, sometimes you forget to do the make install on ipvsadm after rebooting. Note**ipvsadm** doesn't have strong version testing capabilities. It will run on an incompatible version of ip_vs and it won't be obvious, if you're new to LVS, that the strange output is a version mis-match problem (I've done it and had to ask about it on the mailing list). I usually have several different versions of `ip_vs` around for testing. I install `ipvsadm` as `/sbin/ipvsadm-(ip_vs_version-kernel_version)` *e.g.* `ipvsadm-0.2.12-2.4.4`, and let

rc.system_map relink to the correct ipvsadm on bootup.

3.18.1. compiling ipvsadm with popt You can optionally compile ipvsadm with popt libraries, popt libraries, which allows ipvsadm to handle more complicated arguments on the command line. If your libpopt library is too old, your ipvsadm will segv. The default compile used to link against static popt libraries but now uses dynamic libraries. If you get errors about POPT not found, then you haven't installed popt, or you may not have a recent popt.

Torsten Buslei

I've just compiled and installed a new Kernel (2.4.18 with patch 2.4.19-pre8 and linux-2.4.18-ipvs-1.0.2.patch.gz). After reboot (everything's fine till here), compiling ipvsadm (either ipvs-1.0.2.tar.gz or ipvsadm-1.20.tar.gz) I've get the following error-messages:

```
ipvsadm.c:345: 'POPT_ARGFLAG_OPTIONAL' undeclared (first use in this function)
```

I commented out this part of ipvsadm.c:345

```
{"persistent", 'p', POPT_ARG_STRING/*|POPT_ARGFLAG_OPTIONAL*/, &optarg, 'p'},
```

ipvsadm makes and runs fine now. Was this OK to do?

Horms *horms (at) vergenet (dot) net* 05 May 2002

You would appear to have a version of popt that does not support optional arguments (POPT_ARGFLAG_OPTIONAL). Your fix should be fine, with the side effect that you will have to give an argument to -p / –persistent if you use that option.

3.19. Director: check software

With a bit of skill and luck you will have loaded all the software without any errors.

Check that ipvsadm detects the kernel patches by running

```
director:/etc/lvs:# ipvsadm
```

success will look something like

```
director:/usr/src# ipvsadm
IP Virtual Server version 0.2.7 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
```

If you aren't running a kernel with ipvs patches, you'll get error messages about missing ipvs modules.

3.20. Director kernel: get someone else to do it for you

Despite our continuing admonitions to build at least your first kernel for LVS from the standard kernel, people still try to patch

Some distributions have kernels prepatched with ipvs. RedHat started doing this in late 1999 (but apparently have stopped with kernel 2.4.20 according to Peter Nash *peter (dot) nash (at) changeworks (dot) do (do) uk* 15 May 2003 and with RedHat 9.0 according to James Bourne *james (at) sublime (dot) com (dot) au*). SuSE has talked about doing it (Dec 2000) (and has started doing it May 2003). If you have one of these distributions, follow the instructions provided with the distribution for getting a working LVS director. Since an average of 2 ipvs versions come out for every kernel version, you will likely have an older ipvs version if you choose this route. This is not a problem - ipvs was production quality when first released, however versions more than 6-12months old are not supported on the mailing list. You'll be told to upgrade and come back again.

If you don't know if you have a kernel prepatched with ipvs, either ask your Linux Vendor or look in your kernel source tree for ipvs files (names like /usr/src/linux/net/ipv4/ip_vs_*.c).

If you are not sure, then setup with a fresh kernel from ftp.kernel.org. You can always return to your patched distribution later. If you try to patch a kernel that already has ipvs patches applied, you will get patch errors *e.g.*

```
> 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]
```

People who compile the kernel after a failed second layer of patches wind up on the mailing list with errors that are impossible to fathom. One of the major sources of questions on the mailing list is from people running RedHat who find that their install fails when following the instructions in the mini-HOWTO. They aren't starting with standard kernel and the instructions from RedHat haven't told them what's going on.

Here's advice from Lars*lmb (at) suse (dot) com*

Hey, if you are getting an error message like this, or if your distribution already includes LVS, don't try applying the patch on top of it.

First, try checking with your Linux vendor whether an updated kernel package with full support is already available.

If you think this revision of the LVS patches provides significant improvement over the previous ones, or fixes a critical bug for you, and the vendor doesn't yet provide an updated kernel, please ask them to do so.

If you need a newer revision than what your distribution supplies, please grab a clean kernel tree from kernel.org and start from that. Make sure to apply all other patches you need!

3.20.1. Handling the prepatched RedHat kernel Ramish Patel wrote:

I am using Red Hat Linux 7.2, kernel 2.4.7-10, and I have attempted to apply numerous patches to it for lvs, only two of which have been successfully applied without errors – they came from `ipvs-1.0.4.tar.gz`, which contained the two patches which were successfully applied to the kernel, `linux_kernel_ksyms.c.diff` , `linux_net_netsyms.c.diff` .

I re-compiled the kernel afterwards. I am still getting error messages:

```
"Could not open the /proc/net/ip_masq/vs file
Are you sure that the IP Virtual Server is supported by the kernel?"
```

Horms *horms (at) verge (dot) net (dot) au* 31 Jul 2002

Firstly I would strongly recommend that you start with a kernel from kernel.org (such as 2.4.18) rather than using Red Hat's Kernel unless you know what you are doing. ipvs-1.0.4 does not work out of the box with the Kernel shipped with Red Hat 7.2 and vice versa.

However if you want to do this here is a rough guide.

- You need to apply the linux_kernel_ksyms.c.diff and linux_net_netsyms.c.diff

- You need to remove the existing version of LVS which is present in the 7.2 kernel. You can do this by examining the patches that are provided with the kernel src.rpm and the spec.file.
- Compile the kernel - a few minor things may fail. Again this is the "way of pain".
- Build the kernel modules privieded in the ipvs/ subdirectory of ipvs-1.0.4.tar.gz. You will need to make modifications to the source to get this to work, but they should be minor.

Alex Kramarov has
instructions for building/upgrading the RedHat kernel.

3.21. Director: 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. 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 (at) calderasystems (dot) 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.

Aisha *aayesha83 (at) yahoo (dot) com* Aug 25, 2005

I'm new to LVS and I have done all the steps described in the minimum configuration setup, but when I do modules_install, it tells me that there are "unresolved symboles" in ip_vs.o file.

Horms

These kind of problems usually have one of three causes.

- The you turned on some options, like say netfilter, in a tree that had alredy been built and some object files didn't get rebuilt even though they should have been, and thus some symbols are missing. I remember seeing this a lot with 2.4, it seemed to be a problem with the build system. Fortunately its easy, though a little time consuming to solve.

  From the toplevel directory of your kernel source.

  ```
  mv .config ../saved.config
  make mrproper
  mv ../saved.config .config
  make oldconfig
  make bzImage modules ...
  ```

- You have edited your .config file by hand and have managed to produce an invalid config - for intance you enabled LVS without enabling Netfilter. Don't do this!!! Use make menuconfig or make config unless you really know what is going on.

- The build is fine, but during the course of modules_install, depmod is called, and depmod tries to resolve symbols against the current running kernel, they don't exist, and you get the errror you are seeing. This can be resolved most reasily be rebooting into your new kernel.

3.22. Realservers, Other unices

This list of realservers is from Ratz. About the only thing he hasn't tried yet is Plan 9. Remember to set netmask=/32 for the VIP on LVS-DR and LVS-Tun (for LVS-NAT you can setup with any netmask you like). If you are running non-Linux unix realservers, you can usually handle the arp problem by configuring the device carrying the VIP with the -arp switch.

- 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 Note HPUX arps even if you tell it not to. You'll need to handle the

  arp problem some other way.

Ratz's code to setup non-linux realservers is now in the 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). Note (In the 3yrs the configure script has been out, I've not heard of anyone using this part of the code, so there seems no point in maintaining it.)

Here's the original info from Ratz for realservers with non-Linux OS's. 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
#
```

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.

Gregory Boehnlein

I'm going to be working with a bunch of Solaris 9 boxes in the near future, and I would like to add them to my existing LVS cluster. Does anyone have information on what/how Solaris 9 can be used as the Real Servers in an LVS-DR cluster? On linux, I implement the hidden-arp patch. How is this accomplished on Solaris boxen?

Roberto Nibali *ratz (at) drugphish (dot) ch* 11 Aug 2003

Solaris doesn't have this issue ;).

Chris Kennedy *ckennedy (at) iland (dot) net*

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.

Ramon Kagan *rkagan (at) YorkU (dot) ca* 05 Jun 2002

Just in case anybody is interested. You can do the following on lo0:1 or for paranoid people like me hme1.

```
ifconfig <intfc> plumb
ifconfig <intfc> <VIP>
ifconfig <intfc> <VIP> <VIP>
ifconfig <intfc> netmask 255.255.255.255
ifconfig <intfc> up
```

This is from the FAQ but I'm adding that this doesn't have to be on lo0.

3.23. Loopback interface on Windows/Microsoft/NT/W2K

Windows is *not* handled by the configure script (which needs **bash** to run).

According to Horms you don't need anything special to handle the ARP problem; presumably Windows honours the noarp flag.

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 mini-HOWTO to find ;-\. There seem to be many ways of doing it. Here are some of the answers. Setting the metric to 254 seems to be critical.

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 (at) virtual-net (dot) 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 (at) nbuster (dot) 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 (at) web2cad (dot) 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 (at) mpoli (dot) 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 though. I've used that trick with Win98SE and it worked.

Brent Knotts *brent (dot) knotts (at) mylocalgov (dot) com* 28 Jan 2003

Concerning Windows 2000 realservers:

It is possible to change the subnet mask to 255.255.255.255 in the registry, and it works fine for my DR setup. This is easier than deleting the extra route to the local interface after each boot.

In Windows 2000, the interfaces are found in:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\
Services\Tcpip\Parameters\Interfaces

Find the interface with the appropriate IP address and change the subnet mask. Rebooting is not necessary, but you should bring the interface down and up.

Foreign Languages for MS

Sebastien *Sebastien.Bonnet (at) experian (dot) fg* 12 Apr 2002

The "MS Lookback Adapter" is called "carte de bouclage"

Malcolm Turnbull

I've set up a simple LVS DR loadbalancer with 4 IIS web servers (win2K) behind it. I've setup a local loopback adapter on each 2K server and set the priority to 254. The loadbalancer works fine... But seems to confuse Windows Networking, SMB Network Browsing no longer seems to work. (Required by RoboCopy).

Martijn Klingens *mklingens (at) ism (dot) nl* 10 Sep 2002. Subject: Re: LVS DR setup with NT2K servers

We're using an almost similar setup, just that we're currently using a simple xcopy and want to migrate to DFS. That doesn't change the basics though. Things that I encountered:

- Windows always adds a route to the subnet on the loopback adapter. If that subnet is also available on the normal LAN your routing table will get confused. In our case we migrated sites with existing IPs to LVS so we had to pick an entire class C subnet as mask. Solution: manually delete the route to the /24 on the loopback interface.

- Disabling the file and print services on the loopback interface usually also helps.

- Check the default gateways and the DNS on the various interfaces. If they are not identical you may find very strange behaviour (although there are cases where it's useful to have them differ, this is not that common).

Hope this helps. If not, please specify a bit more about your setup. If you have problems on the realservers, are they able to resolve another realserver using dns and/or ping another realserver?

Oh, and you mentioned network browsing, which is not entirely the same as opening a share on a computer with a well-known name. Do you really need the browser? We usually have the computer browser service stopped anyway so an intruder isn't instantly able to see all other w2k hosts on the net. A bit moot, since DNS provides the same data, but it doesn't really hurt either.

lstep (at) banquise (dot) org 11 Aug 2004

I'm trying to understand why you set the metric of the localhost interface to 254 (according to the doc found on

LVS with HA for Win2k Terminal Servers when configuring a W2K realserver's loopback in a Direct Routing mode? If I don't do this what problem(s) will I get?

Malcolm Turnbull *malcolm (at) loadbalancer (dot) org*

NT can be a pain in the backside when its routing table gets confused :-) In my experience setting the metric to 254 solves 95% of issues.

Ratz

I'm sorry but this doesn't really convince me. Where can I read about this? What does NT really do when you set such a high metric?

Using the registry to set the mask to 255.255.255.255 does the other 5% Seems to happen more often in NT style domains or servers with multiple NICs. Any NT realserver with a problem is easy to spot as it won't work in DR full stop, and the routing table will be incorrect.

Malcolm Turnbull *malcolm (at) loadbalancer (dot) org* 2005/04/15

We have a small .net based tool to setup the loopback adpater for DR mode on windows 2000/2003 web servers. The idea was to install the loopback and setup as many VIPs as required and then set the netmask to 255.255.255.255. But from testing we've found that windows will sometimes stop responding to the load balancers RIP (for health checks), whereas if you use a netmask of 255.0.0.0 Windows ignores this route because it looks for the smallest subnet in the routing table first *i.e.* you 255.255.255.0 (or whaterver your using for your RIP.)

The small utility requires the .net framework 1.1. When you start iti, click on the red warning 'no adapter found' to install the loopback Then add as many VIPs as you want and click save. Slightly pointless for anyone who knows what they are doing, but someone may find it usefull.

You can

download binary and source http://www.loadbalancer.org/download/nt/. NB. The one marked BETA will set the netmask to 255.255.255.255 (if thats what you want).

Graeme Fowler *graeme (at) graemef (dot) net* 06/11/2005

Adding a WIN NT machine to LVS setup: If you go to "add new hardware" and "select from list", you should find it listed under Network Adapters - Microsoft.

Install the "Microsoft Loopback Adapter" network device, and assign it the relevant address(es). IIRC that it doesn't ARP at all, and I've used it in production for a number of SLB setups - not behind LVS though, these were Cisco IOS or ExtremeWare setups, either using NAT or DR.

3.24. Mac OS X (and Solaris)

Malcolm Turnbull

Has anyone used Mac OS X for a real server in LVS/DR ?

Jerome RICHARD *jrichard (at) virtual-net (dot) fr* 20 Nov 2002

If you wish to configure multiple IP on Mac OS X, you should use the following command :

```
sudo ifconfig lo0 alias 10.0.0.80 netmask 255.255.0.0
```

There is more information in the

AppleCare Documents

Roberto Nibali *ratz (at) drugphish (dot) ch* 20 Nov 2002

Yes, I gave a presentation about a year ago and I remember that one of the attendees had a new shiny G4 with Mac OS X. It worked flawlessly.

Is it OK to use the loopback interface ?

You can but you don't have to. OS X is BSDish, so you need to use BSDish syntax:

```
ifconfig lo0 alias VIP netmask 255.255.255.255 -arp up
```

Malcolm Turnbull *Malcolm.Turnbull (at) crocus (dot) co (dot) uk* 22 Nov 2002

Got it working:

- Solaris:

  ```
  ifconfig lo0:1 plumb
  ifconfig lo0:1 <vip-addr> netmask 255.255.255.255 up
  ```

- Mac OS X:

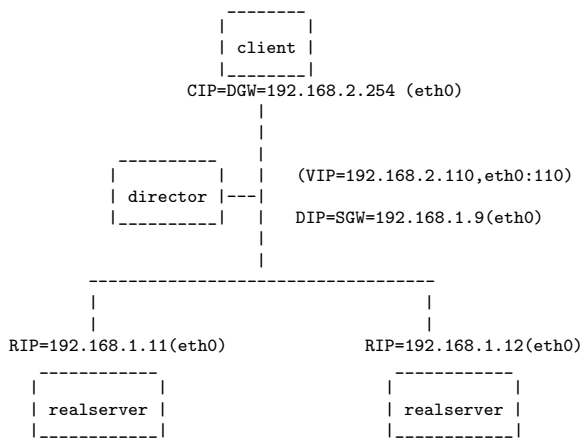
  ```
  ifconfig lo0 alias VIP netmask 255.255.255.255 -arp up
  ```

4. Example 1: LVS using LVS-NAT forwarding

LVS-NAT forwarding was the first method used to setup an LVS. For 2.2 kernels on the director, it gives lower throughput at high load compared to LVS-DR or LVS-Tun, due to the CPU overhead of rewriting packets, but still is useful in some circumstances. For 2.4 (and later) kernels on the director, there is little performance difference between LVS-NAT and the other forwarding methods.

Setup an LVS with two networks (here 192.168.1.0/24 and 192.168.2.0/24) using a single NIC on each machine. (For the test, don't have any other IPs on the machines). The VIP on the ethernet alias (eth0:110) is setup by the configure script, *i.e.*, don't add the VIP. There should be no default routes, and all interfaces on the 192.168.1.0/24 network should be able to ping each other. The client won't be able to connect to anything till the VIP is enabled on the director. Warning

You must have two networks for this setup to work. Here one NIC is used on the director (you can use two if you like, but you'll have to change the setup yourself). If you want all machines on the one network (*192.168.1.0/24*) then read One Network LVS-NAT



4.1. Setup using the configure script

Warning

The `configure` has a bug for LVS-NAT. It removes the default gw for the director (which should be the local router). You'll have to add this back by hand. (The script correctly removes the default gw for LVS-DR and LVS-Tun. For the reason why you shouldn't have a default gw on LVS-DR and LVS-Tun directors see

security concerns: default gw(s) and routing with LVS-DR/LVS-Tun . It would be better for LVS-NAT just to have a route from VIP:80 to 0/0:0 rather than a default route.)

For the v0.9.x series of configure script, use the conf file lvs_nat.conf.one_NIC_two_network

```
#----------lvs_nat.conf-----------------------------------
LVSCONF_FORMAT=1.1
LVS_TYPE=VS_NAT
INITIAL_STATE=on
CLEAR_IPVS_TABLES=yes
#
#VIP line format - device[:alias] IP netmask broadcast
#To help avoid namespace collisions with other VIPs, I set alias=last number of VIP (here 110).
VIP=eth0:110 192.168.2.110 255.255.255.0 192.168.2.255
#
#DIP line format - device[:alias] IP network netmask broadcast
DIP=eth0 192.168.1.9 192.168.1.0 255.255.255.0 192.168.1.255
#
#DIRECTOR_GW - packets with src_addr=VIP, dst_addr=0/0 are sent to DIRECTOR_GW
#to be forwarded to the outside world.
DIRECTOR_GW=192.168.2.254
#
#SERVICE line format - proto port scheduler IP|name:port[,weight] [IP|name:port[weight]]
SERVICE=t telnet rr 192.168.1.11:telnet 192.168.1.12:telnet
#SERVICE=t http rr 192.168.1.11:http,1 192.168.1.12:http,2
#
SERVER_NET_DEVICE=eth0
#VS-NAT real-servers do not have a VIP, i.e. there is no SERVER_VIP_DEVICE
#SERVER_VIP_DEVICE=
#SERVER_GW is not user configurable with LVS-NAT. script sets SERVER_GW = DIP
#SERVER_GW=
#----------end lvs_nat.conf--------------------------------
```

then run the configure script

```
\$./configure lvs_nat.conf
```

this will produce (among other things) a file rc.lvs_nat (or rc.lvs).

Run this rc.lvs_nat file first on the director and then on the two realservers (the file knows whether it's running on a director or realserver and acts appropriately). One of the big traps in setting up a LVS-NAT LVS is that you have to make the director the default route for the realservers (the script does this for you) - it won't work if you don't do this.

4.2. Setup by hand

Run this script on the director.

```
#!/bin/sh
#------mini-HOWTO-setup-LVS-NAT-director----------

#set ip_forward ON for vs-nat director (1 on, 0 off).
cat /proc/sys/net/ipv4/ip_forward
echo "1" >/proc/sys/net/ipv4/ip_forward

#director is gw for realservers
#turn OFF icmp redirects (1 on, 0 off)
echo "0" >/proc/sys/net/ipv4/conf/all/send_redirects
cat       /proc/sys/net/ipv4/conf/all/send_redirects
echo "0" >/proc/sys/net/ipv4/conf/default/send_redirects
cat       /proc/sys/net/ipv4/conf/default/send_redirects
echo "0" >/proc/sys/net/ipv4/conf/eth0/send_redirects
cat       /proc/sys/net/ipv4/conf/eth0/send_redirects
```

```
#setup VIP
/sbin/ifconfig eth0:110 192.168.2.110 broadcast 192.168.2.255 netmask 255.255.255.0

#set default gateway
/sbin/route add default gw 192.168.2.254 netmask 0.0.0.0 metric 1

#clear ipvsadm tables
/sbin/ipvsadm -C

#install LVS services with ipvsadm
#add telnet to VIP with rr sheduling
/sbin/ipvsadm -A -t 192.168.2.110:telnet -s rr

#first realserver
#forward telnet to realserver 192.168.1.11 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t 192.168.2.110:telnet -r 192.168.1.11:telnet -m -w 1
#check that realserver is reachable from director
ping -c 1 192.168.1.11

#second realserver
#forward telnet to realserver 192.168.1.12 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t 192.168.2.110:telnet -r 192.168.1.12:telnet -m -w 1
#checking if realserver is reachable from director
ping -c 1 192.168.1.12

#list ipvsadm table
/sbin/ipvsadm
#------mini-HOWTO-setup-LVS-NAT-director----------
```

Run this script on the realservers

```
#!/bin/sh
#---------mini-HOWTO-setup-LVS-NAT-realserver-------
#installing default gw 192.168.1.9 for vs-nat'
/sbin/route add default gw 192.168.1.9
#show routing table
/bin/netstat -rn

#checking if DEFAULT_GW is reachable
ping -c 1 192.168.1.9

#looking for VIP on director from realserver
ping -c 1 192.168.2.110

#set_realserver_ip_forwarding to OFF (1 on, 0 off).
echo "0" >/proc/sys/net/ipv4/ip_forward
cat       /proc/sys/net/ipv4/ip_forward

#---------mini-HOWTO-setup-LVS-NAT-realserver-------
```

4.3. Test LVS-NAT operation

On the director run the command

```
\$ipvsadm
```

The output should be something like

```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
```

```
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.2.110:telnet rr
  -> 192.168.1.11:telnet          Masq    1      0          0
  -> 192.168.1.12:telnet          Masq    1      0          0
```

Now telnet from the client to 192.168.2.110. You will get the login prompt from one of the realservers (note which one). Then on the director re-run ipvsadm

```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.2.110:telnet rr
  -> 192.168.1.11:telnet          Masq    1      1          0
  -> 192.168.1.12:telnet          Masq    1      0          0
```

Note the ActiveConn counter has incremented.

Open another window on the client and telnet to 192.168.2.110. You should get the login prompt for the other realserver. Re-run ipvsadm.

```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.2.110:telnet rr
  -> 192.168.1.11:telnet          Masq    1      1          0
  -> 192.168.1.12:telnet          Masq    1      1          0
```

There are now 2 connections active.

You are now connected to the realservers by a telnet session. Congratulations - you have set up a LVS-NAT LVS for the service telnet.

4.4. Setup LVS-NAT for the service http

Make sure the realservers are serving http with each realserver listening to port 80 on its own IP (the RIP, *e.g.* 192.168.1.11, 192.168.1.12) *i.e.* on a different IP for each realserver. Make the 2 webpages a little different so that you can tell which machine you have connected to.

Change the conf file by uncommenting the http line and commenting out the telnet line in the conf file.

```
#SERVICE=t telnet rr 192.168.1.11telnet 192.168.1.12:telnet
SERVICE=t http rr 192.168.1.11:http 192.168.1.12:http
```

Rerun the configure script on the new conf file, rerun the `rc.lvs`files on the director and then the realservers.

or

For configuration by hand, substitute http for telnet in the director script - you don't need to run the realserver script again.

If you get no errors then run ipvsadm

```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.2.110:www rr
  -> 192.168.1.11:www             Masq    1      0          0
  -> 192.168.1.12:www             Masq    1      0          0
```

Direct your http client to 192.168.2.110. If you get the expected webpage, shift-reload a few times (for netscape), watching the webpages alternate between realservers. In some circumstances, presumably due to http persistance, the connection will persist on one realserver. In this case use lynx, curl or telnet to 192.168.2.110 80 and at the blank prompt type

```
GET /
```

to use lynx do

```
\$ lynx -dump http://192.168.2.110/
```

Re-run ipvsadm (on the director)

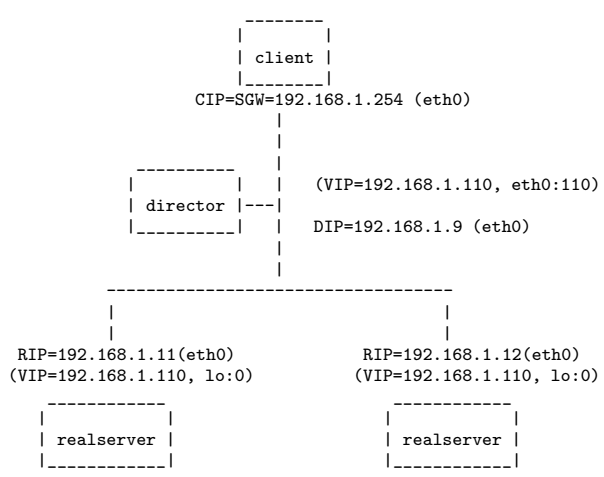
```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.2.110:www rr
  -> 192.168.1.11:www             Masq    1      0          13
  -> 192.168.1.12:www             Masq    1      0          12
```

Unlike telnet there are no ActiveConn seen even though you have a webpage displayed on the client. The http protocol disconnects after 15secs changing the connections to InActConn. These expire in about 2 mins (re-run ipvsadm after 2mins and see that InActConn drops to 0).

The template timeout can be seen by invoking ipchains -L -M -n (for 2.2 kernels, for 2.4 kernels and more info see the HOWTO).

5. Example: Setup LVS using LVS-DR forwarding

Setup an LVS with a single network (here 192.168.1.0/24) on machines with a single NIC. The IPs on ethernet aliases (eg eth0:110, lo:0) are setup by the configure script. There should be no default routes and all machines should be able to ping each other.



5.1. VS-DR for the service telnet

For the v0.9.x series of configure script, use the conf file lvs_drt.conf.one_NIC_one_network

```
#----------lvs_dr.conf----------------------------------------
LVSCONF_FORMAT=1.1
LVS_TYPE=VS_DR
INITIAL_STATE=on
CLEAR_IPVS_TABLES=yes
#VIP line format - device[:alias] IP netmask broadcast
#To help avoid namespace collisions with other VIPs, I set alias=last number of VIP (here 110).
#note: for LVS-DR, LVS-Tun, the IP is in a /32 network
VIP=eth0:110 192.168.1.110 255.255.255.255 192.168.1.110
#DIP line format - device[:alias] IP network netmask broadcast
DIP=eth0:9 192.168.1.9 192.168.1.0 255.255.255.0 192.168.1.255
#no DIRECTOR_GW for LVS-DR or LVS-Tun
#DIRECTOR_GW=
#SERVICE line format - proto port scheduler IP[,weight] [IP[,weight]]
SERVICE=t telnet rr 192.168.1.11 192.168.1.12
#SERVICE=t http rr 192.168.1.11 192.168.1.12
SERVER_VIP_DEVICE=lo:110
SERVER_NET_DEVICE=eth0
#SERVER_GW - packets with src_addr=VIP, dst_addr=0/0 are sent to SERVER_GW
#to be forwarded to the outside world.
#For standard LVS-DR,VS-Tun, this must _NOT_ be the director.
#For Julian's martian modification (see the HOWTO), it will be the director.
#If you don't know about the martian modification, you aren't using it.
#The script will not neccesarily set up the SERVER_GW as the real-servers's default gw.
SERVER_GW=192.168.1.254
#----------end lvs_dr.conf-------------------------------------
```

then

```
\$./configure lvs_dr.conf
```

this will produce (among other things) a file rc.lvs_dr

Run this rc.lvs_dr file first on the director and then on the two realservers (the file knows whether it's running on a director or realserver and acts appropriately).

5.2. Setup by hand

On the director run this script

```
#!/bin/bash
#---------------mini-rc.lvs_dr-director------------------------
#set ip_forward OFF for lvs-dr director (1 on, 0 off)
#(there is no forwarding in the conventional sense for LVS-DR)
cat       /proc/sys/net/ipv4/ip_forward
echo "0" >/proc/sys/net/ipv4/ip_forward

#director is not gw for realservers: leave icmp redirects on
echo 'setting icmp redirects (1 on, 0 off) '
echo "1" >/proc/sys/net/ipv4/conf/all/send_redirects
cat       /proc/sys/net/ipv4/conf/all/send_redirects
echo "1" >/proc/sys/net/ipv4/conf/default/send_redirects
cat       /proc/sys/net/ipv4/conf/default/send_redirects
echo "1" >/proc/sys/net/ipv4/conf/eth0/send_redirects
cat       /proc/sys/net/ipv4/conf/eth0/send_redirects

#add ethernet device and routing for VIP 192.168.1.110
/sbin/ifconfig eth0:110 192.168.1.110 broadcast 192.168.1.110 netmask 255.255.255.255
/sbin/route add -host 192.168.1.110 dev eth0:110
#listing ifconfig info for VIP 192.168.1.110
/sbin/ifconfig eth0:110

#check VIP 192.168.1.110 is reachable from self (director)
/bin/ping -c 1 192.168.1.110
#listing routing info for VIP 192.168.1.110
/bin/netstat -rn

#setup_ipvsadm_table
#clear ipvsadm table
/sbin/ipvsadm -C
#installing LVS services with ipvsadm
#add telnet to VIP with round robin scheduling
/sbin/ipvsadm -A -t 192.168.1.110:telnet -s rr

#forward telnet to realserver using direct routing with weight 1
/sbin/ipvsadm -a -t 192.168.1.110:telnet -r 192.168.1.11 -g -w 1
#check realserver reachable from director
ping -c 1 192.168.1.12

#forward telnet to realserver using direct routing with weight 1
/sbin/ipvsadm -a -t 192.168.1.110:telnet -r 192.168.1.12 -g -w 1
#check realserver reachable from director
ping -c 1 192.168.1.12

#displaying ipvsadm settings
/sbin/ipvsadm

#not installing a default gw for LVS_TYPE vs-dr
#---------------mini-rc.lvs_dr-director------------------------
```

Pat Villani *Pat (dot) Villani (at) hp (dot) com* 09 Oct 2004 I'm using the LVS-DR example out of the mini-HOWTO. One of the first steps is to turn off ip_forward. Why? It seems to work without that step and doing so interferes with my NAT service.

I only turned it off because you don't need it on for LVS-DR, and turning it off makes the machine more secure. You can turn it on if you have other reasons to do so.

On the realservers run this script

```
#!/bin/bash
#----------mini-rc.lvs_dr-realserver-----------------
#installing default gw 192.168.1.254 for vs-dr
/sbin/route add default gw 192.168.1.254
#showing routing table
/bin/netstat -rn
#checking if DEFAULT_GW 192.168.1.254 is reachable
ping -c 1 192.168.1.254

#set_realserver_ip_forwarding to OFF (1 on, 0 off).
echo "0" >/proc/sys/net/ipv4/ip_forward
cat       /proc/sys/net/ipv4/ip_forward

#looking for DIP 192.168.1.9
ping -c 1 192.168.1.9

#looking for VIP (will be on director)
ping -c 1 192.168.1.110

#install_realserver_vip
/sbin/ifconfig lo:110 192.168.1.110 broadcast 192.168.1.110 netmask 0xffffffff up
#ifconfig output
/sbin/ifconfig lo:110
#installing route for VIP 192.168.1.110 on device lo:110
/sbin/route add -host 192.168.1.110 dev lo:110
#listing routing info for VIP 192.168.1.110
/bin/netstat -rn

#hiding interface lo:110, will not arp
echo "1" >/proc/sys/net/ipv4/conf/all/hidden
cat       /proc/sys/net/ipv4/conf/all/hidden
echo "1" >/proc/sys/net/ipv4/conf/lo/hidden
cat       /proc/sys/net/ipv4/conf/lo/hidden

#----------mini-rc.lvs_dr-realserver-----------------
```

5.3. Test LVS-DR operation

Then on the director run the command

```
\$ipvsadm
```

The output should be something like

```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.110:telnet rr
  -> 192.168.1.11:telnet          Route    1      0          0
```

```
  -> 192.168.1.12:telnet          Route    1      0          0
```

Note that the forwarding method has changed from Masq to Route.

Now telnet from the client to 192.168.1.110. You will get the login prompt from one of the realservers (note which one). Then re-run ipvsadm

```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.110:telnet rr
  -> 192.168.1.11:telnet          Route    1      1          0
  -> 192.168.1.12:telnet          Route    1      0          0
```

If you have tcpwrappers running around telnet on the realservers, the login will be delayed by an interval between 7secs(Slackware) and 3mins(RedHat). You can fix this by changing the line in inetd.conf

```
from
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd

to
telnet  stream  tcp     nowait  root    /usr/sbin/in.telnetd  in.telnetd
```

and re-HUPing inetd. If you have a pam-ified telnetd and login, you may never be able to fix this delay (RedHat 6.2 man pages tell you to change entries in /etc/pam.conf, a file that doesn't exist).

After logging in, the telnet session behaves normally. This delay is only a minor nuisance, but it is difficult to differentiate from a hung connection due to an LVS misconfiguration (which hopefully would have been picked up when running the configure script). This delay doesn't occur with LVS-NAT (see the section on identd in the HOWTO for an explanation).

Open another window on the client and telnet again to 192.168.1.110. You should get the login prompt for the other realserver. Re-run ipvsadm.

```
director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.110:telnet rr
  -> 192.168.1.11:telnet          Route    1      1          0
  -> 192.168.1.12:telnet          Route    1      1          0
```

There are now 2 connections active.

You are now connected to the realservers be a normal telnet session. Congratulations - you have set up a LVS-DR LVS for the service telnet.

5.4. Setup LVS-DR for the service http

IMPORTANT: Make sure the httpd on the realservers are listening to VIP:80 (here 192.168.1.110:80) *i.e* both realservers are listening to the **same IP**. The connection from the lvs client will arrive at VIP:80 on the realserver. You will need the VIP setup on lo:0 before the httpds will start. (You also can have the

httpd listening on the RIP, here 192.168.1.11 or 192.168.1.12, if you like, but these IPs are not involved in the LVS.) Make the webpage on each realserver a little different so that you can tell which realserver you have connected to.

Change the conf file by adding an http line and commenting out the telnet line.

```
#----------lvs_dr.conf-----------------------------------
#SERVICE=t telnet rr 192.168.1.11 192.168.1.12
SERVICE=t http rr 192.168.1.11 192.168.1.12
#----------end lvs_dr.conf-------------------------------
```

Rerun the configure script on the conf file, rerun the `rc.lvs`files on the director and then the realservers.

or

if you are setting up by hand, change telnet to http and rerun the script on the director. You don't need to rerun the script on the realservers.

If you get no errors then run ipvsadm

```
>director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.110:www rr
  -> 192.168.1.11:www             Route    1      0          0
  -> 192.168.1.12:www             Route    1      0          0
```

Connect your http client to 192.168.1.110. If you get the expected webpage, shift-reload a few times, watching the webpages alternate between realservers. In some circumstances, presumably due to http persistance, the connection will persist on one realserver. In this case use lynx or telnet to 192.168.1.110 80 and at the blank prompt type

```
GET /
```

to use lynx do

```
\$ lynx -dump http://192.168.1.110/
```

Re-run ipvsadm (on the director)

```
>director:/lvs/conf# ipvsadm
IP Virtual Server version 0.9.14 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.110:www rr
  -> 192.168.1.11:www             Route    1      0          5
  -> 192.168.1.12:www             Route    1      0          4
```

Unlike telnet there are no ActiveConn seen even though you have a webpage displayed on the client. The http protocol disconnects after 15secs (max). The connections change to InActConn and expire in about 2 mins. If you then re-run ipvsadm the InActConn will drop to 0.

Congratulations, you're done. Go have fun. For more information look at the HOWTO and docs on the LVS website and scan the mail archives. If you have any questions, join the mailing list.

6. How users have done it

6.1. Dan Browning, LVS-DR on Red Hat 7.2: handling ARP, other misc problems

The original posting is at

lvs mailing list archives

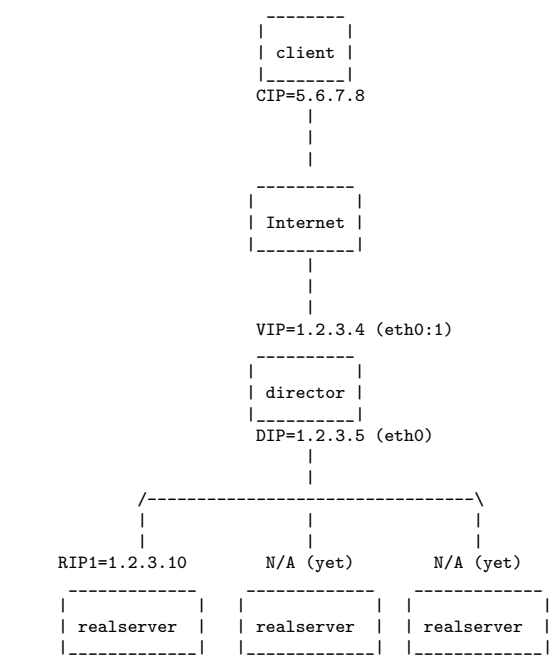
Dan Browning *db (at) kavod (dot) com* 2002-01-05

Okay, this is the last time I post on this, I promise. :-) My trigger finger was a little quick on the last mail I sent out, so this one includes *all* the instructions (more or less) the way I did it to setup a LVS-DR on Red Hat 7.2.

I hope that it may be of use to someone sometime. Next project: automatic failover to backup LVS director...

```
mkdir ~/download/piranha
cd ~/download/piranha
wget \

ftp://ftp.linux.org.uk/pub/linux/piranha/7.2/piranha/piranha-0.6.0-15.i386.rpm \
ftp://ftp.linux.org.uk/pub/linux/piranha/7.2/ipvsadm/ipvsadm-1.18-8.i386.rpm \
ftp://ftp.linux.org.uk/pub/linux/piranha/7.2/scsi_reserve/scsi_reserve-0.7-6.i386.rpm \
        -c
rpm -Uvh *.rpm

chkconfig piranha-gui on
service piranha-gui restart

piranha-passwd homelast

# If you will be using two directors (that need to sync seemlessly)
# Setup keyless scp on all the nodes:
ssh-keygen -t rsa
cat .ssh/id_rsa.pub | ssh SERVERNAME 'cat >>~/.ssh/authorized_keys2'

# Helpful Documentation
http://ha.redhat.com/docs/high-availability/index.html
http://www.linuxvirtualserver.org/docs/arp.html
http://www.linux-vs.org/~julian/hidden.txt

# Enabling IP Encapsulation
# On each real server, establish a tunnel between it and each virtual
server address. For example, these commands establish two tunnels (tunl0
and # # tunl1) to two virtual server addresses...
# To prevent real servers, rather than the active router,
# from intercepting ARP broadcasts, you also need to hide
# tunnels from ARP broadcasts. For example, these commands
# hide tunnels tunl0:

# Insert the ipip module, if not statically compiled into the kernel
already
insmod ipip
# Make the tunl0 device up
ifconfig tunl0 0.0.0.0 up
# Start the hiding interface functionality
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
# Hide all addresses for this tunnel device
echo 1 > /proc/sys/net/ipv4/conf/tunl0/hidden
# Configure a VIP on an alias of tunnel device
ifconfig tunl0:0 1.2.3.4 up

# Testing
lynx --dump http://VIP/test
ab -n 100 -c 10 http://VIP/index.html
```

Environment: Red Hat 7.2, Piranha 0.6.0-15, RH stock kernel (2.4.7-10)




```
###############
##   DETAILS:
###############
Setup the Director:

Install Piranha, lvsadm
Configure like so:

serial_no = 38
primary = 1.2.3.4
service = lvs
backup = 0.0.0.0
heartbeat = 1
heartbeat_port = 539
keepalive = 6
deadtime = 18
network = direct
reservation_conflict_action = preempt
debug_level = NONE
virtual http {
     active = 1
     address = 1.2.3.5 eth1
     port = 80
     send = "GET / HTTP/1.0\r\n\r\n"
     expect = "HTTP"
     load_monitor = none
     scheduler = wlc
     protocol = tcp
     timeout = 6
     reentry = 15
     quiesce_server = 0
     server www4real {
         address = 1.2.3.10
```

```
         active = 1
         weight = 1
     }
}

        IP Virtual Server version 0.8.1 (size=65536)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port             Forward Weight ActiveConn InActConn
TCP  1.2.3.5:80 rr
  -> 1.2.3.10:80               Route   1      0          0

CURRENT LVS PROCESSES
root      1992  0.0  0.0  1604  600 ?        S    15:45   0:00 pulse
root      2295  0.0  0.0  1604  600 ?        S    15:45   0:00
/usr/sbin/lvs --nofork -c /etc/sysconfig/ha/lvs.cf
root      2299  0.0  0.0  1640  648 ?        S    15:45   0:00
/usr/sbin/nanny -c -h 1.2.3.10 -p 80 -s GET / HTTP/1.0\r\n\r\n -

## Notes for recompiling 2.4.17 with ipvs and hidden patches on Red Hat 7.2 ##
## (On both real server and the director)

# Download directory
export D=/tmp/download

mkdir \$D
cd \$D

#kernel
wget http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.17.tar.gz

#hidden patch
wget http://www.linux-vs.org/~julian/hidden-2.4.5-1.diff

#IPVS patch
wget
http://www.linuxvirtualserver.org/software/kernel-2.4/linux-2.4.12-ipvs-
0.8.2.patch.gz

#net filter module - if you want to do just the module instead of the
big kernel patch above.
wget
http://www.linuxvirtualserver.org/software/kernel-2.4/ipvs-0.8.2.tar.gz

#ipvs admin
wget ftp://rpmfind.net/linux/redhatbeta/ha/i386/ipvsadm-1.17-2.i386.rpm

# Unpack new kernel
tar zxvf linux-2.4.17.tar.gz

# Unpack ipvs patch
gunzip linux-2.4.12-ipvs-0.8.2.patch.gz

# Unpack kernel
mv linux /usr/src/linux-2.4.17
cd /usr/src

# Recreate symlink
rm -f linux-2.4
ln -s linux-2.4.17 linux-2.4
ln -s linux-2.4.17 linux

# Apply "hidden" patch
cd linux-2.4.17
patch -p1 < \$D/hidden-2.4.5-1.diff

Should see:
################################
patching file include/linux/inetdevice.h
```

```
patching file include/linux/sysctl.h
Hunk #1 succeeded at 334 (offset 9 lines).
patching file net/ipv4/arp.c
Hunk #3 succeeded at 754 (offset -1 lines).
patching file net/ipv4/devinet.c
Hunk #1 succeeded at 756 (offset 20 lines).
Hunk #2 succeeded at 1013 (offset -4 lines).
Hunk #3 succeeded at 1079 (offset 20 lines).
patching file Documentation/filesystems/proc.txt
Hunk #1 succeeded at 1583 (offset 5 lines).
patching file Documentation/networking/ip-sysctl.txt
################################

# Apply ipvs patch
patch -p1 < \$D/linux-2.4.12-ipvs-0.8.2.patch

# ipvsadm 1.18-8, which is newer, is already installed (from piranha
project)

make clean
make mrproper
make menuconfig
make bzImage
make modules
make modules_install
make install #doesn't support GRUB yet.  - or can copy the
arch/i386/boot/bzImage file manually
vi /boot/grub/grub.conf:
title 2.4.17_ipvs
        root (hd0,0)
        kernel /boot/vmlinuz-2.4.17 ro root=/dev/sda1

#now copy the /usr/src/linux-2.4.17 to the next linux box:
tar czf linux-2.4.17-dir.tgz /usr/src/linux-2.4.17/

scp linux-2.4.17-dir.tgz SERVER_TWO:/usr/src

#now unpack in SERVER_TWO
tar zxvf linux-2.4.17-dir.tgz
cd linux-2.4.17
make modules_install
make install
# do grub config again.
title 2.4.17_ipvs
        root (hd0,0)
        kernel /boot/vmlinuz-2.4.17 ro root=/dev/sda1
# reboot!
```

6.2. Jezz Palmer, setup of LVS squid

Jezz Palmer *J (dot) D (dot) F (dot) Palmer (at) swansea (dot) ac (dot) uk* 29 May 2002, to set up a squid LVS.

Directors are running 2.4.18 Kernel RealServers 2.4.7 (standard RH 7.2 but recompiled). Squid did not work with 2.4.18 or 2.4.17 for some reason.

Firstly I created the director. I used instructions written by Dan Browning (above) to configure the kernel Note, I only used the kernel config bit, I've edited it for 2.4.18 and the later ipvsadm and ipvsadm patch and a few other bits.

Notes for recompiling 2.4.18 with ipvs and hidden patches on Red Hat 7.2

```
# Download directory
export D=/tmp/download
```

```
mkdir \$D
cd \$D

#kernel
wget http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.18.tar.gz

#hidden patch
wget http://www.linux-vs.org/~julian/hidden-2.4.5-1.diff

#IPVS patch
wget
http://www.linuxvirtualserver.org/software/kernel-2.4/linux-2.4.18-ipvs-1.0.
0.patch.gz

#ipvs admin
wget
http://www.linuxvirtualserver.org/software/kernel-2.4/ipvsadm-1.20.tar.gz

#Unpack and install ipvsadm admin
tar xvzf  ipvsadm-1.20.tar.gz
cd ipvsadm
make install

# Unpack new kernel
tar zxvf linux-2.4.18.tar.gz

# Unpack ipvs patch
gunzip linux-2.4.18-ipvs-1.0.0.patch.gz

# Unpack kernel
mv linux /usr/src/linux-2.4.18
cd /usr/src

# Recreate symlink
rm -f linux-2.4
ln -s linux-2.4.18 linux-2.4
ln -s linux-2.4.18 linux

# Apply "hidden" patch
cd linux-2.4.18
patch -p1 < \$D/hidden-2.4.5-1.diff

Should see:
################################
patching file include/linux/inetdevice.h
patching file include/linux/sysctl.h
Hunk #1 succeeded at 334 (offset 9 lines).
patching file net/ipv4/arp.c
Hunk #3 succeeded at 754 (offset -1 lines).
patching file net/ipv4/devinet.c
Hunk #1 succeeded at 756 (offset 20 lines).
Hunk #2 succeeded at 1013 (offset -4 lines).
Hunk #3 succeeded at 1079 (offset 20 lines).
patching file Documentation/filesystems/proc.txt
Hunk #1 succeeded at 1583 (offset 5 lines).
patching file Documentation/networking/ip-sysctl.txt
################################

# Apply ipvs patch
patch -p1 < \$D/ linux-2.4.18-ipvs-1.0.0.patch

make clean
make mrproper

# At this stage I copied the default config (matching my system) from the
original RH 7.2 kernel  to .config so save me time selecting modules.
```

```
cp /usr/src/linux-2.4.7-10/configs/kernel-2.4.7-i686-smp.config ./.config

make menuconfig
```

Leave all other settings the same, but go to ...

```
Networking Options -> IP: Virtual Server Configuration and build in [*] virtual server support [EXPERIMENTAL]
```

and accept the defaults.

```
make bzImage
make modules
make modules_install
make install #doesn't support GRUB yet.  - or can copy the
arch/i386/boot/bzImage file manually
vi /boot/grub/grub.conf:
title 2.4.18_ipvs
root (hd0,0)
kernel /boot/vmlinuz-2.4.18 ro root=/dev/****

#If you have two (or more) directors with the same hardware then copy to the
new kernel to them,

tar czf linux-2.4.18-dir.tgz /usr/src/linux-2.4.18/
scp linux-2.4.18-dir.tgz SERVER_TWO:/usr/src

#now unpack in SERVER_TWO
tar zxvf linux-2.4.18-dir.tgz
cd linux-2.4.18
make modules_install
make install
# do grub config again.
title 2.4.18_ipvs
root (hd0,0)
kernel /boot/vmlinuz-2.4.17 ro root=/dev/****
# reboot!
```

If the hardware is substantially different then repeat the process on the next server using the correct default config file from /usr/src/linux-2.4.7-10/configs/

Real Servers.

For the real servers I just repeated the above process but with the default kernel (2.4.7) supplied with RH 7.2. The ipvsadm patch isn't necessary but what the hell, that's what I did. It would probably work without the ipvsadm patch and just the hidden patch.

Getting it to Work.

I'm using LVS_DR. Initially I decided to set up an http LVS using 1 director and 3 http realservers. Http purely because it's easier to ascertain as to whether it's working or not. So I setup apache on the three machines, making the root index page to just say the name of the machine. Then I used Joe Mack's configure script with my config to generate the rc.lvs script to run on the machines in the LVS. I downloaded configure-lvs_0.9.2 from http://www.linuxvirtualserver.org/Joseph.Mack/configure-lvs_0.9.2.tar.gz

It requires Net-DNS-0.19, which I downloaded from http://www.perl.com/CPAN-local/authors/id/B/BB/BBB/Net-DNS-0.19.tar.gz

So after untaring and installing Net-DNS-0.19, I untarred the configure-lvs_0.9.2.tar.gz file

```
tar xvzf configure-lvs_0.9.2.tar.gz
cd configure-lvs_0.9.2
```

I then chose the closest match to my network of the 12 or so example configs and edited it to look like this. The one I chose was lvs_dr.conf.one_NIC_one_network. (IPs etc deleted for convenience).

```
#----------lvs_dr.conf----------------------------------------
LVSCONF_FORMAT=1.1
LVS_TYPE=VS_DR
INITIAL_STATE=on
CLEAR_IPVS_TABLES=yes
#VIP line format - device[:alias] IP netmask broadcast
#To help avoid namespace collisions with other VIPs, I set alias=last number
of VIP (here 110).
#note: for LVS-DR, LVS-Tun, the IP is in a /32 network
VIP=eth0:110 my_vip 255.255.255.255 my_vip
#DIP line format - device[:alias] IP network netmask broadcast
DIP=eth0:9 my_dip my_dip_network 255.255.255.0 my_dip_broadcast
#no DIRECTOR_GW for LVS-DR or LVS-Tun
#DIRECTOR_GW=
#SERVICE line format - proto port scheduler IP[,weight] [IP[,weight]]
SERVICE=t http rr rip1,1 rip2,1 rip3,1
SERVER_VIP_DEVICE=lo:110
SERVER_NET_DEVICE=eth0
#SERVER_GW - packets with src_addr=VIP, dst_addr=0/0 are sent to SERVER_GW
#to be forwarded to the outside world.
#For standard LVS-DR,VS-Tun, this must _NOT_ be the director.
#For Julian's martian modification (see the HOWTO), it will be the director.
#If you don't know about the martian modification, you aren't using it.
#The script will not neccesarily set up the SERVER_GW as the real-servers's default gw.
SERVER_GW=my_server_gw
#----------end lvs_dr.conf-------------------------------------
```

I used the configure script to generate the rc.lvs

./configure lvs_dr.conf.one_NIC_one_network

The `rc.lvs`was ran on the local machine then copied on to all of the remaining servers in the LVS, IE and the 3 http realservers.

At this point by typing ipvsadm at the prompt you should get something like this showing that your LVS is running.

```
[root@director1 root]# ipvsadm
IP Virtual Server version 1.0.0 (size=65536)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  vip:80 rr
  -> rip1:80                      Route   1      0          0
  -> rip2:80                      Route   1      0          0
  -> rip3:80                      Route   1      0          0
```

Then to test it go to a browser on another box and type http://vip (or whatever your VIP is), you should get a screen showing the name of the first server in the list, keep hitting refresh and the name should change as it rotates around the list of realservers. This will only happen with RR scheduling but it proves the point nicely.

Next step, applying this to squid. Eek!
Firstly I changed the config file to the squid servers and squid port.

```
#----------lvs_dr.conf----------------------------------------
LVSCONF_FORMAT=1.1
LVS_TYPE=VS_DR
INITIAL_STATE=on
CLEAR_IPVS_TABLES=yes
#VIP line format - device[:alias] IP netmask broadcast
#To help avoid namespace collisions with other VIPs, I set alias=last number
of VIP (here 110).
#note: for LVS-DR, LVS-Tun, the IP is in a /32 network
VIP=eth0:110 vip 255.255.255.255 vip
#DIP line format - device[:alias] IP network netmask broadcast
DIP=eth0:9 dip dip_network 255.255.255.0 dip_broadcast
#no DIRECTOR_GW for LVS-DR or LVS-Tun
#DIRECTOR_GW=
#SERVICE line format - proto port scheduler IP[,weight] [IP[,weight]]
SERVICE=t squid rr rip1,1 rip2,1
SERVER_VIP_DEVICE=lo:110
SERVER_NET_DEVICE=eth0
#SERVER_GW - packets with src_addr=VIP, dst_addr=0/0 are sent to SERVER_GW
#to be forwarded to the outside world.
#For standard LVS-DR, LVS-Tun, this must _NOT_ be the director.
#For Julian's martian modification (see the HOWTO), it will be the director.
#If you don't know about the martian modification, you aren't using it.
#The script will not neccesarily set up the SERVER_GW as the real-servers's default gw.
SERVER_GW=server_gw
#----------end lvs_dr.conf-------------------------------------
```

Then re-ran

```
./configure lvs_dr.conf.one_NIC_one_network
```

And ran the resulting `rc.lvs` script on all the servers in the LVS, not I only have 2 realservers in the squid setup.
This gives an ipvsadm output like this

```
[root@director1 root]# ipvsadm
IP Virtual Server version 1.0.0 (size=65536)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  vip:3128 rr
  -> rip1:3128                    Route   1      0          0
  -> rip2:3128                    Route   1      0          0
```

(from Joe - at this stage the realservers, setup by the 0.9.x configure script, were blocked from accessing webservers on the internet. This problem is explained in the 3-Tier section of the HOWTO and led to the next version of the configure script.)

In the meantime I discovered that if I restarted the network on the realservers after the `rc.lvs` script had been executed then the realservers could access the outside world. Also during this time whilst Joe was working on the alterations to the configure script I set about testing the Squid LVS and writing scripts to handle realserver failure and set about introducing a backup director.

At this stage I had a squid LVS which was using rr scheduling, it soon became apparent that rr scheduling wasn't going to work on it's own. Some HTTPS

sites (*e.g.* banking facilities) won't accept requests for a client coming from multiple IP address, so I tried the various schedulers and the use of persistence to get the LVS to balance satisfactorily for all sites. (Joe - this is described more in the HOWTO in the Services section on squids.)

I have found that wlc with a persistence of 360 seconds works beautifully. This ensures that requests from a particular client are sent to the same realserver for 6 minutes (that's a rolling 6 minutes). The timeout can be increased, but most banking facilities will timeout a connection if inactivity exceeds a few minutes.

```
[root@director1 root]# ipvsadm
IP Virtual Server version 1.0.0 (size=65536)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  vip:3128 wlc -p 360
  -> rip1:3128                    Route   1      0          0
  -> rip2:3128                    Route   1      0          0
```

6.3. Two Box LVS

It's possible to setup two boxes as a fully functional failover LVS. When one box is the director, it is also serving as a realserver with localnode and the other box is another realserver. The two boxes run failover code to allow them to swap roles as directors. This is more complicated than you'd want for one of your first setups, and in practice there are problems handling the failover. These are discussed in the

Localnode section of the HOWTO http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/LVS-HOWTO.localnode.html#two_box_lvs

7. What if there are problems

7.1. can't patch kernel, can't compile patched kernel, can't compile ipvsadm

- Are you using the standard kernel?
- Does your version of ip_vs match the kernel
- does your ipvsadm come from the tarball for your version of ip_vs.

7.2. Can't compile ipvsadm

You (probably) installed a kernel binary and when you try to compile **ipvsadm**, you get error messages about missing header files.

Problem: You don't have the kernel header files (in `/usr/src/linux/include/linux/`) linked from `/usr/include/linux/`.

Solution: Install your kernel header files and make the link.

7.3. ip_tables doesn't work

Symptom: You're running 2.4.x and get wierd non-informative error messages from **ip_tables** (**ip_tables** doesn't have useful error messages) like

```
ip_tables.o: init_module:
Device or resource busy
```

Problem: you have ipchains loaded

Solution: remove ipchains (*i.e.* unload with **rmmod ipchains.o**). You should also remove ipchains support from your kernel so that no-one else can fall into the same trap.

7.4. ipvsadm gives wierd erroneous output

Symptom: you get inpropriate characters, numbers for IPs, ports, services in the output of **ipvsadm**

Problem: your ipvsadm doesn't match the version of ip_vs. (You may have forgotten to compile the new version of ipvsadm after rebooting the computer following loading the LVS-patched kernel and you're using an old ipvsadm).

Solution: get the ipvsadm that matches your ip_vs.

7.5. Help! My LVS doesn't work

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 translate your posting into our language and then figure out what you've done wrong. We haven't seen any new problems in setting up a basic LVS since the early days, but replying to the same requests does take a lot of time. (There are still plenty of postings about how an LVS does or doesn't work that no-one has thought about, which we are interested in.) Occasionally after 10 exchanges with 3 people on the mailing list trying to help figure out what's wrong, the original poster will suddenly remember that they have filter rules (which they'd "checked out thoroughly" and were sure were working) that they "forgot to tell us about".

Since 1999, substantial effort has been put into the HOWTO, the LVS-mini-HOWTO and the configure script to enable people with minimal or no understanding of LVS to setup a working LVS. If you don't want to use a configure script, there is enough information here for you to setup a working LVS from the command line.

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 1 or 2 realservers serving telnet (or httpd), then rather than getting us to figure out what you've done wrong, how about you use the tools we've provided, to setup the basic LVS. Once you have a

working basic LVS, then you can branch out and try to set it up your own way. If you can't get the scripts to work, then we'll try to fix those.

If you need to contact us, first look at the problem report information (HOWTO) needed to solve problems.

7.6. The client says "connection refused"

The machine that receives the request packet is replying that it is not listening for that service. Possible reasons for this are (with help from Ratz)

- 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 on the VIP (for LVS-DR and LVS-Tun) or the RIP (for LVS-NAT).
- a packet filter rule
- service dest is unavailable
- service is not running on RS
- he's misspelling the IP address
- network cable unplugged (you should get icmp "host down" type replies here)

7.7. connection hangs; 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.

If you are using LVS-NAT and one network, you must also
handle the ICMP redirects.
Note
Billy *ntadmin (at) reachone (dot) com* 18 Mar 2004

If your LVS **is working** and you have connections that disconnect (like with http), by the time you've downloaded your page, the connection will be in the InActConn state. See

httpd is stateless and normally closes connections. Connections being in InActConn state is a diagnostic only if the connection is hanging (*i.e.* the client is not receiving webpages from the LVS).

7.8. initial connection is delayed, but once connected everything is fine

Usually you have problems with

authd/identd. Simplest thing is to stop your service on the realserver from calling the identd server on the client (*i.e.*disconnect your service from identd).

Another problem, which causes slowness in any situation, is DNS timing out on a call. Ideally you shouldn't have DNS running on any machine in an LVS - you will need to run sendmail etc off `/etc/hosts`. However if you have DNS, make sure there are entries for all IPs on that machine (if you have multiple interfaces) and any machine that you'll be calling. To check, run commands that need name resolution *e.g.* **netstat -a** (will need entries in `/etc/services` as well) or **route** (will need entries in `/etc/networks` as well). If either of these pause, rerun them in the form which doesn't need name resolution (**netstat -an**, **route -n**) to locate the entries forcing the pause - these don't have name resolution.

7.9. connection takes several seconds to connect

I've had trouble with this once when my routing was wrong and ICMP had to figure it out for me. Another possibility is a delay from DNS lookup. Try connecting to the name of the VIP and to the IP.

7.10. My LVS doesn't loadbalance, the client always goes to the same realserver

This is for LVS-DR or LVS-Tun. The most likely explanation is that you haven't solved the arp problem. To check that you've handled the arp problem you can

test the interface on the realserver.

7.11. My LVS still doesn't work: what do I do now?

If you've setup the simple telnet LVS from the LVS-mini-HOWTO and now are trying your own LVS, and it doesn't work, you'll have to debug your setup.

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). Noteno 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 configure script, see mini-HOWTO) 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).

If your connection requests aren't connecting, then you need to see where the packets are being stopped. You can watch packets with **tcpdump** or **netstat -an** (to see whether the request has left one node or arrived at another and to see if replies have been made).