Setting up multiple Vservers for home gateway testing

Posted by Excds on Wed 27 Jun 2007 at 11:56

This setup might have other uses which I haven't thought about, but this was the scenario I had. I was working with testing home gateways, DSL/Ethernet/fiber, and had *a lot* of them. In an ideal world I would have one computer for each device acting as a "regular home user's computer", but that would require space/cooling/whatnot. When the number of devices went above ten, that was not really a practical solution anymore. Instead I setup a bunch of vservers.

So instead of this solution:

```
            Test network
                 |
  -----------------------------------
  |     |     |     |    |     |    |
HGW1  HGW2  HGW3  HGW4 HGW5  HGW6  ...
  |     |     |     |    |     |    |
  |     |     |     |    |     |    |
  C     C     C     C    C     C    C
  o     o     o     o    o     o    o
  m     m     m     m    m     m    m
  p     p     p     p    p     p    p
  u     u     u     u    u     u    u
  t     t     t     t    t     t    t
  e     e     e     e    e     e    e
  r     r     r     r    r     r    r
  1     2     3     4    5     6    X
```

I went for this solution:

```
            Test network
                 |
  -----------------------------------
  |     |     |     |    |     |    |
HGW1  HGW2  HGW3  HGW4 HGW5  HGW6  ...
  |     |     |     |    |      |   |
  -----------------------------------
                 |
      VLAN (802.1q) enabled switch
                 |
      ----------------------------
      |    Linux VServer host    |
      | |----------------------| |
      | |      vserver1        | |
      | |      vserver2        | |
      | |      vserver3        | |
      | |      vserver4        | |
      | |      vserver5        | |
      | |      vserver6        | |
      | |      vserverX        | |
      | ------------------------ |
      ----------------------------
                  ^
                  |
           VNC/SSH/...
```

The connection for one home gateway and one vserver is then:

```
vserverX -> tagged VLAN -> Switch -> untagged VLAN -> HGWX -> Internet
```

Basic idea

The basic idea of this setup is that each virtual machine is connected "directly" to the HGW. Each virtual machine will have two network interfaces, one which is tagged with a VLAN and connected to the switch, and one which will be used internally between the virtual machine and the host.

Since the vservers more or less live in the same routing table and network address space, we need to do some source policy routing. My starting point for this was:

http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.rpdb.simple.html

We need to set up source policy routing since all the vservers will use the same subnet and default route (192.168.1.0/24 with 192.168.1.1).

Prerequisites

- Host machine with two network cards and a default Etch install.
- Lots of disk space

Networks

For this I'm going to use VLAN's 3001 - 3120 and the networks:

- 10.200.48.0/24 - Network from which we will access the vserver host
- 192.168.254.0/24 - Network used for iptables forwarding and the Debian mirror
- 192.168.1.0/24 - The client side network.

In the host this will be:

- eth0 - 10.200.48.100
- aliases for eth0
    - 10.200.48.101, 102, 103, ...
    - 192.168.254.101, 102, 103, ...

Example: A single vserver connected to VLAN 3001

It has the following setup:

- eth1.3001: 192.168.1.101, with default route 192.168.1.1
- eth0: 192.168.254.101, to be used internally

In the host we've created an IP alias which will be forwarded to this vserver.

```
10.200.48.101 -> iptables forwarding -> 192.168.254.101
```

Vserver host setup

Network

Add support for VLAN to the kernel, i.e., add 8021q to /etc/modules

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

loop
8021q
```

Host network configuration:

```
# /etc/network/interfaces

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
        address 10.200.48.100
        netmask 255.255.255.0
        gateway 10.200.48.1

# Internally used addresses
auto eth1
iface eth1 inet static
        address 192.168.254.1
        netmask 255.255.255.0
```

Local Debian mirror

First set up a local debian mirror and install Apache2:

```
apt-get install apache2 debmirror

mkdir /mirror
debmirror -v -a i386 -d etch /mirror/debian --method http \
          -h ftp.se.debian.org --ignore-release-gpg
```

In the apache directory /var/www create a symlink to the mirror directory.

```
ln -s /mirror/debian /var/www/debian
```

If you for don't have the disk space or bandwidth for a complete mirror, you should take a look at "apt-proxy".

Restricting ip services

Make sure that apache only listens to addresses not used by the vservers:

```
# /etc/apache2/ports.conf

Listen 10.200.48.100:80
Listen 192.168.254.1:80
```

Do the same for sshd:

```
# /etc/ssh/sshd_config

ListenAddress 10.200.48.100
```

Install required packages

```
apt-get install util-vserver vserver-debiantools \
        linux-image-2.6-vserver-686 vlan
```

And reboot...

Changing default vserverbase

The default vserver directory is "/var/lib/vserver", in my installation i prefer to have it in "/vserver".

```
mkdir /vservers
rm /etc/vservers/.defaults/vdirbase
ln -s /vservers /etc/vservers/.defaults/vdirbase
setattr --barrier /vservers
```

Enable creation/destruction of VLAN devices for vservers

```
mkdir /etc/vservers/.defaults/interfaces
touch /etc/vservers/.defaults/interfaces/vlandev
```

Creating a vserver for use as a template

```
vserver vserver-template build -n vserver-template --hostname vserver-template.localdomain \
        --interface eth0:192.168.1.106/24 -m debootstrap -- \
        -d etch -m http://192.168.254.1/debian
```

If everything went fine, we should be able to start the vserver and see it running.

```
vserver vserver-template start
vserver vserver-template enter
```

Add all the default packages we want to use.

```
apt-get update
apt-get install ssh lighttpd wmaker wmaker-data vncserver xbase-clients \
        xfonts-base firefox locales xterm xutils menu locales

# Generate locales
dpkg-reconfigure locales

echo 'LANGUAGE="en_US:en"' > /etc/environment
echo 'LANG=en_US.UTF-8' >> /etc/environment
echo 'LC_PAPER="en_US.UTF-8"' >> /etc/environment
echo 'LC_ALL="en_US.UTF-8"' >> /etc/environment


# Change password for root
passwd
```

```
# Add a default user
adduser theuser

# DNS
echo "nameserver 192.168.1.1" > /etc/resolv.conf

# Set a vnc password
vncpasswd
```

Create a custom script to start vnc

```
#!/bin/bash
# /etc/init.d/vncstart

/bin/su - -c "/usr/bin/vncserver -geometry 1024x768 -depth 16 :0 > /dev/null 2>&1" &
```

And add it to our virtual machine startup

```
chmod 755 /etc/init.d/vncstart
ln -s /etc/init.d/vncstart /etc/rc3.d/S99vncstart
```

The last part is to clean up our template vserver, so we won't duplicate all base packages. And also to shut it down, in preparation for cloning.

```
apt-get clean
exit
vserver vserver-template-stop
```

Add our custom vserver up/down scripts

First we create the directories for pre/post-run scripts.

```
mkdir /etc/vservers/vserver-template/scripts
mkdir /etc/vservers/vserver-template/scripts/pre-start.d
mkdir /etc/vservers/vserver-template/scripts/post-stop.d
```

And then add our custom scripts for source routing and iptables forwarding.

```
#!/bin/bash
# /etc/vservers/vserver-template/scripts/pre-start.d/10configure_network.sh

# IP address for the vserver primary network interface
ADDRESS=`cat root/address.txt | sed 's/[^[:alnum:].]//g'`

# IP address which will be forwarded to the vserver with iptables
FORWARD_ADDRESS=`cat root/forward_address.txt | sed 's/[^[:alnum:].]//g'`

# VLAN tag
VLAN=`cat root/vlan.txt | sed 's/[^[:alnum:].]//g'`

# Number of the virtual machine
NUMBER=`cat root/vserver-number.txt | sed 's/[^[:alnum:].]//g'`

# Which routing table to use
TABLE=`cat etc/hostname | sed 's/[^[:alnum:].]//g'`

DATE=`date "+%Y-%m-%d %H:%M:%S : "`

echo "$DATE ip addr add $FORWARD_ADDRESS dev eth0" >> root/start.log
ip addr add $FORWARD_ADDRESS dev eth0 2>&1 >> root/start.log

echo "$DATE ip rule add from $ADDRESS table $TABLE" >> root/start.log
ip rule add from $ADDRESS table $TABLE 2>&1 >> root/start.log

echo "$DATE ip route add default via 192.168.1.1 dev eth1.$VLAN table $TABLE" >> root/start.log
ip route add default via 192.168.1.1 dev eth1.$VLAN table $TABLE 2>&1 >> root/start.log

echo "$DATE ip route flush cache" >> root/start.log
ip route flush cache 2>&1 >> root/start.log

echo "$DATE iptables -t nat -A PREROUTING -d $FORWARD_ADDRESS -j DNAT --to-destination 192.168.254.$NUMBER" >>
iptables -t nat -A PREROUTING -d $FORWARD_ADDRESS -j DNAT --to-destination 192.168.254.$NUMBER 2>&1 >> root/sta
```

```
#!/bin/bash
# /etc/vservers/vserver-template/scripts/post-stop.d/10deconfigure_network.sh

# IP address for the vserver primary network interface
ADDRESS=`cat root/address.txt | sed 's/[^[:alnum:].]//g'`

# IP address which will be forwarded to the vserver with iptables
FORWARD_ADDRESS=`cat root/forward_address.txt | sed 's/[^[:alnum:].]//g'`

# VLAN tag
```

```
VLAN=`cat root/vlan.txt | sed 's/[^[:alnum:].]//g'`

# Number of the virtual machine
NUMBER=`cat root/vserver-number.txt | sed 's/[^[:alnum:].]//g'`

# Which routing table to use
TABLE=`cat etc/hostname | sed 's/[^[:alnum:].]//g'`

DATE=`date "+%Y-%m-%d %H:%M:%S : "`

echo "$DATE ip addr del $FORWARD_ADDRESS/32 dev eth0" 2>&1 >> root/stop.log
ip addr del $FORWARD_ADDRESS/32 dev eth0

echo "$DATE ip rule delete from $ADDRESS/32" 2>&1 >> root/stop.log
ip rule delete from $ADDRESS/32

echo "$DATE ip route flush cache" 2>&1 >> root/stop.log
ip route flush cache

echo "$DATE iptables -t nat -D PREROUTING -d $FORWARD_ADDRESS -j DNAT --to-destination 192.168.254.$NUMBER" 2>&
iptables -t nat -D PREROUTING -d $FORWARD_ADDRESS -j DNAT --to-destination 192.168.254.$NUMBER
```

Don't forget to set executable permissions.

```
chmod 755 /etc/vservers/vserver-template/scripts/pre-start.d/10configure_network.sh
chmod 755 /etc/vservers/vserver-template/scripts/post-stop.d/10deconfigure_network.sh
```

For these scripts to work, we need the following files in "/root/" on the virtual machine:

- address.txt - This is the address on the "local" network, i.e., 192.168.1.x.
- forward_address - This is the "public" address, which we will have forwarded services to.
- vlan.txt - Which virtual network this machine will belong to.
- vserver-number.txt - Which of our vservers we're running on.

If you've got all those settings working, then it's time to shutdown our template vserver and prepare it for copying to non-templates.

Preparing the template vserver for copying

Before going any further, we might want to backup our template vserver.

```
cd /vservers
cp -a vserver-template vserver_backup
cd /etc/vservers
cp -a --no-dereference vserver-template vserver_backup
```

First of all we have to change the network interface settings, so they'll be usable with an ugly script I will provide later.

```
# /etc/vservers/vserver-template/interfaces/0/dev
echo "eth1.VLANNUMBER" > /etc/vservers/vserver-template/interfaces/0/dev

# /etc/vservers/vserver-template/interfaces/0/ip
echo "192.168.1.VSERVERNUMBER" > /etc/vservers/vserver-template/interfaces/0/ip

# Add directory for the second network device
mkdir /etc/vservers/vserver-template/interfaces/1

# /etc/vservers/vserver-template/interfaces/1/dev
echo "eth0" > /etc/vservers/vserver-template/interfaces/1/dev

# /etc/vservers/vserver-template/interfaces/1/ip
echo "192.168.254.VSERVERNUMBER" > /etc/vservers/vserver-template/interfaces/1/ip

# /etc/vservers/vserver-template/name
echo "vserverVSERVERNUMBER" > /etc/vservers/vserver-template/name

# /etc/vservers/vserver-template/uts/nodename
echo "vserverVSERVERNUMBER.yourdomain.com" > /etc/vservers/vserver-template/uts/nodename

# /vservers/vserver-template/etc/hostname
echo "vserverVSERVERNUMBER" > /vservers/vserver-template/etc/hostname

# Then create the directory "/etc/vserver/.routing", which will store information
# about our additional routing tables.
mkdir /etc/vservers/.routing
```

Adding the last hack...

Since we've now prepared our template server, it's time to start cloning. For this I've used this dirty hack, which I put in "/usr/local/bin/".

```
#!/bin/bash
# vserver-from-template.sh

# /etc/vservers/vserver-template/interfaces/0/dev - VLANNUMBER
# /etc/vservers/vserver-template/interfaces/0/ip - IPADDRESS
# /etc/vservers/vserver-template/interfaces/1/ip - IPADDRESS
# /etc/vservers/vserver-template/name - VSERVERNUMBER
# /etc/vservers/vserver-template/uts/nodename - VSERVERNUMBER

if [ $# -ne 3 ]; then
    echo "You must supply parameters:"
    echo "$0   "
    exit 0
fi

FORWARDADDRESS=$1
VSERVERNUMBER=$2
VLANNUMBER=$3


cd /vservers
cp -a vserver-template "vserver$VSERVERNUMBER"

cd /etc/vservers

cp -a --no-dereference vserver-template "vserver$VSERVERNUMBER"

cd "/etc/vservers/vserver$VSERVERNUMBER"
rm vdir
ln -s "/etc/vservers/.defaults/vdirbase/vserver$VSERVERNUMBER" vdir
rm run
ln -s "/var/run/vservers/vserver$VSERVERNUMBER" run

echo "default" > "/etc/vservers/vserver$VSERVERNUMBER/apps/init/mark"

cat /etc/vservers/vserver-template/interfaces/0/dev | sed -e "s/VLANNUMBER/$VLANNUMBER/g" > /tmp/vservercopy.$$
mv /tmp/vservercopy.$$ "/etc/vservers/vserver$VSERVERNUMBER/interfaces/0/dev"

cat /etc/vservers/vserver-template/interfaces/0/ip | sed -e "s/VSERVERNUMBER/$VSERVERNUMBER/g" > /tmp/vserverco
mv /tmp/vservercopy.$$ "/etc/vservers/vserver$VSERVERNUMBER/interfaces/0/ip"

cat /etc/vservers/vserver-template/interfaces/1/ip | sed -e "s/VSERVERNUMBER/$VSERVERNUMBER/g" > /tmp/vserverco
mv /tmp/vservercopy.$$ "/etc/vservers/vserver$VSERVERNUMBER/interfaces/1/ip"

cat /etc/vservers/vserver-template/name | sed -e "s/VSERVERNUMBER/$VSERVERNUMBER/g" > /tmp/vservercopy.$$
mv /tmp/vservercopy.$$ "/etc/vservers/vserver$VSERVERNUMBER/name"

cat /etc/vservers/vserver-template/uts/nodename | sed -e "s/VSERVERNUMBER/$VSERVERNUMBER/g" > /tmp/vservercopy.
mv /tmp/vservercopy.$$ "/etc/vservers/vserver$VSERVERNUMBER/uts/nodename"

cat "/vservers/vserver$VSERVERNUMBER/etc/hostname" | sed -e "s/VSERVERNUMBER/$VSERVERNUMBER/g" > /tmp/vserverco
mv /tmp/vservercopy.$$ "/vservers/vserver$VSERVERNUMBER/etc/hostname"

echo "$VSERVERNUMBER    vserver$VSERVERNUMBER" >> /etc/vservers/.routing/tables
cat /etc/vservers/.routing/tables | sort | uniq > /tmp/vservercopy.$$
mv /tmp/vservercopy.$$ /etc/vservers/.routing/tables

cat /etc/iproute2/rt_tables.original > /etc/iproute2/rt_tables
cat /etc/vservers/.routing/tables >> /etc/iproute2/rt_tables

echo "$FORWARDADDRESS" > "/vservers/vserver$VSERVERNUMBER/root/forward_address.txt"
echo "192.168.1.$VSERVERNUMBER" > "/vservers/vserver$VSERVERNUMBER/root/address.txt"
echo "$VLANNUMBER" > "/vservers/vserver$VSERVERNUMBER/root/vlan.txt"
echo "$VSERVERNUMBER" > "/vservers/vserver$VSERVERNUMBER/root/vserver-number.txt"

# You might want to comment this out if you're creating _a lot_ of vservers...
vserver "vserver$VSERVERNUMBER" start

exit 0
```

The final touch before running this script is to save the original routing table file and to set the correct permissions for our cloning script.

```
cp /etc/iproute2/rt_tables /etc/iproute2/rt_tables.original
chmod 755 /usr/local/bin/vserver-from-template.sh
```

When testing this, I used this information as test data:

```
vserver-from-template.sh 10.200.48.120 101 3001
vserver-from-template.sh 10.200.48.121 102 3002
vserver-from-template.sh 10.200.48.122 103 3003
vserver-from-template.sh 10.200.48.123 104 3004
vserver-from-template.sh 10.200.48.124 105 3005
vserver-from-template.sh 10.200.48.125 106 3006
vserver-from-template.sh 10.200.48.126 107 3007
vserver-from-template.sh 10.200.48.127 108 3008
vserver-from-template.sh 10.200.48.128 109 3009
vserver-from-template.sh 10.200.48.129 110 3010
```

And after a while I'm able to ssh to the different virtual machines, and to connect to the home gateway with for instance Firefox (Iceweasel?).

Using more than 50 vservers

I discovered that there's a problem when using 2.6.18-4-vserver-686 and trying to run more than 50 vservers. This is due to some bug concerning vserver in that kernel version. Thanks to daniel_hozac and harry on #vserver for helping me find out what the problem actually was.

If you want to run more than 50 vservers, you will have to either upgrade to Debian unstable or compile a newer kernel and the vserver utils.

Also, if you already have a number of vservers and you want to upgrade to a newer version of vserver, there will be problems running the vservers after the upgrade. This blog entry by Christian Schenk solves that problem:

http://www.christianschenk.org/blog/upgrading-vserver-from-20-to-22/

So that's basically my setup. Please tell me if you find it useful or if you might have other ideas of usage.
