Networking of Xen VMs

0.1 Basic network operations

0.1.1 Listing current networks

We can use command

```
virsh net-list --all
```

to see various networks created for virtualization.

0.1.2 Listing definition of defined networks

We can use command

```
virsh net-dumpxml default
```

to see configuration of network named 'default', that comes predefined with libvirt and xen.

You may see output like:

```
<network>
  <name>default</name>
  <uuid>d99113fb-948a-4ec0-b763-8de5a3c8023b</uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' forwardDelay='0' />
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254' />
    </dhcp>
  </ip>
</network>
```

where:

name → is name of network.

uuid → is unique identifier for this network. Not two networks can have same uuid.

<**forward mode='nat'**> → is used to indicate that packets from this network should be forwarded to local network using NAT. By default NAT will automatically put base machines IP on outgoing packets.

bridge → is used to define name of bridge to be created when this network is started or bridge to be deleted when network is stopped. Hence we do not use 'brctl' to create/delete bridges when we use virsh commands to manage networks. virsh handles bridge creation/deletion for us.

stp='on' → is used to enable Spanning-Tree-Protocol on bridge. This can be useful if we are going to create complex networks with multiple bridges and there is possibility of having network loop among virtual bridges.

forwardDelay='0' → is used to indicate that bridge should forward packets without introducing any artificial delay.

ip → tag is used to define IP address of the bridge. Bridge IP address also becomes IP address of base. This IP address can be used to indicate which IP range which will get used on this network. We can have 'dhcp' sub-tag and 'range' sub-sub tag inside IP tag if we want to assign IP addresses using DHCP on this network. The range of IPs to be assigned by DHCP can be defined in range tag.

0.1.3 Stopping running network

We can stop running network using

```
virsh net-destroy default
```

0.1.4 Undefining already defined network

We can undefine network using

```
virsh net-undefine default
```

0.1.5 Configuration of default network

Configuration of default network which is present by default is present in file '`/usr/share/libvirt/networks/default.xml`'. Hence if we undefine 'default' network during lab and want it back we can use this XML file to re-define network with name default. The command which can be used is

```
virsh net-define default.xml
```

Note that:

1. Network will not get started when it is defined. We need to explicity start network after defining it using '`virsh net-start default.xml`'

2. If we want to start network as soon as it is defined we can use '`virsh net-create default.xml`' in place of '`virsh net-define default.xml`' to ensure that network gets started along with getting defined.

   Most virsh commands have create, start, destroy, define and undefine options where create would combine both define and start operations.

3. Contents of default.xml are:

```
<network>
  <name>default</name>
     <bridge name="virbr0" />
  <forward/>
  <ip address="192.168.122.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.122.2" end="192.168.122.254" />
    </dhcp>
  </ip>
</network>
```

   Hence we can see that:

   (a) 'uuid' gets generated for us if we do not mention one in XML file.

   (b) 'forward' gets created with forward mode='nat'

   (c) Bridge gets created with stp='on' and forwardDelay='0'

0.1.6 Creating and starting custom network

1. We can create and start various custom networks like

```
<network>
<name>host-only</name>
   <bridge name="virbr1" />
<ip address="192.168.123.1" netmask="255.255.255.0">
  <dhcp>
    <range start="192.168.123.2" end="192.168.123.254" />
  </dhcp>
</ip>
</network>
```

   This network would be a host only network and no packets from this network would go outside current host.

2. We can also create and start custom network like

```
<network>
  <name>natted</name>
     <bridge name="virbr2" />
  <forward/>
  <ip address="192.168.124.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.124.2" end="192.168.124.254" />
    </dhcp>
  </ip>
</network>
```

   This network would be a natted network. Outside machines will not be able to initiate connection to VM in this network. But VMs in this network would be able to contact all machines accessible by host using NAT.

0.2 Understaing firewall implications of starting/stopping networks

To enable/disable NAT of packets from virtual network, xen uses default iptables firewall that comes with CentOS. Hence it is important to understand changes that happen in firewall rules when we start/stop a virtual network. Especially what changes in firewall ensure that network is host-only and what changes make it a natted network should be clear.

To understand all this first change default iptables configuration file '`/etc/sysconfig/iptables`' so that it has contents as shown below:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp --icmp-type any -j ACCEPT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Then use '`service iptables restart`' command so that old rules get flushed and firewall uses the configuration mentioned in file '`/etc/sysconfig/iptables`'. You can use commands '`iptables -L -n`' to list firewall rules in filter table. There are also three other tables 'nat', 'mangle' and 'raw'. For purpose of this lab only 'filter' and 'nat' tables are important. To see rules in nat table we can use '`iptables -t nat -L -n`' command.

In case we want to see configuration of start-up file that would lead to same state of firewall as it is now we can use '`iptables-save`' command. This command wont save any configuration, it justs lists current configuration on screen in configuration file format. We can use '`iptables-save > /etc/sysconfig/iptables`' if we really want to make current firewall configuration default start-up firewall configuratin.

Before trying each of below mentioned sections, first destory all currently defined networks so that changes caused when we create a virtual network are removed. Then list firewall rules using command 'iptables-save' and ensure that you see something like:

```
# Generated by iptables-save v1.3.5 on Wed Sep 22 20:27:07 2010
*nat
:PREROUTING ACCEPT [437:78100]
:POSTROUTING ACCEPT [110:7239]
:OUTPUT ACCEPT [111:7621]
COMMIT
# Completed on Wed Sep 22 20:27:07 2010
# Generated by iptables-save v1.3.5 on Wed Sep 22 20:27:07 2010
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [2353:139742]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type any -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Wed Sep 22 20:27:07 2010
```

OR

```
# Generated by iptables-save v1.3.5 on Wed Sep 22 20:27:30 2010
```

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type any -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Wed Sep 22 20:27:30 2010
```

In case you see something different use '`virsh net-list --all`' to see if any networks are still remaining and if all networks are removed, then use '`service iptables restart`' command.

0.2.1 Understanding changes caused by NATTED network on firewall configuration

Create a network named natted using configuration listed above in the same document. Now use '`iptables-save`' command to see the changes that happen in firewall rules when we start a natted network. Destroy the natted network using '`virsh net-destroy natted`' command. Again use '`iptables-save`' command to verify that all rules that were added in firewall got removed.

0.2.2 Understading changes caused by host-only network on firewall configuration

Create a network named host-only using configuration listed above in the same document. Now use '`iptables-save`' command to see the changes that happen in firewall rules when we start a 'host-only' network. Destroy the 'host-only' network using '`virsh net-destroy host-only`' command. Again use '`iptables-save`' command to verify that all rules that were added in firewall got removed.

0.3 Cloning virtual machines

A sample virtual machine named 'vm1' has been created on all lab machines. Use '`virt-clone --prompt`' to make clones 'vm2' and 'vm3' from 'vm1'. Clones save time as we do not have to install complete OS and do all the configuration which we want repeated between virtual machines. Use

'vm2.img' and 'vm3.img' as harddisk for vm2 and vm3 and keep them also in same folder as 'vm1.img'.

0.4 Various network scenarios of VMs

0.4.1 Completely isolated VM

Use 'virsh dumpxml vm1' to see 'vm1' configuration. Now use command 'virsh detach-interface vm1 bridge' to detach bridge interface. Verify that interface actually got removed using 'virsh dumpxml vm1' commmand. Now start 'vm1' using 'virsh start vm1' command. Connect to 'vm1' using 'virt-viewer vm1 &' and login as root user. Verify using 'ifconfig -a' command that there is no ethernet interface available on this VM to contact outside world. Note that this VM can't even contact base machine.

Now without powering of VM use 'virsh attach-interface vm1 bridge xenbr0' to attach a bridged network interface back to 'vm1'. Edit file '/etc/sysconfig/network-scripts/ifcfg-eth0' and remove line containing mac address as part of 'HWADDR' parameter. Now use 'service network restart' to restart networking. Verify that 'vm1' can again connect to outside world using bridged networking.

Now without powering of VM use 'virsh detach-interface vm1 bridge' to detach bridge interface. Now use 'ifconfig -a' command to again verify that interface actually got removed from running OS.

0.4.2 VMs connected only to each other and not to host

Define a network named 'not_even_host' using file 'not_even_host.xml' with following contents

```
<network>
  <name>not_even_host</name>
  <bridge name="virbr3" />
</network>
```

Use command 'virsh net-define not_even_host.xml'. Verify using 'virsh net-list --all' that network got defined without any problem. Now use 'virsh net-start not_even_host' to start this network.

Assuming 'vm1' still has no bridged network interface after previous configuration of completely isolated VMs, use command 'virsh attach-interface vm1 bridge virbr3' to connect 'vm1' to 'not_even_host' network. Use commands 'virsh detach-interface vm2 bridge' and 'virsh attach-interface vm2 bridge virbr3' to configure 'vm2' also to become part of 'not_even_host'

network. Edit file '`/etc/sysconfig/network-scripts/ifcfg-eth0`' in both 'vm1' and 'vm2'. In both files remove '`HWADDR`' line, change to '`BOOTPROTO=static`', add line '`NETMASK=255.255.255.0`', configure IP address as '`IPADDR=192.168.200.2`' in 'vm1' and as '`IPADDR=192.168.200.3`' in 'vm2'. Now use '`service network restart`' command in both VMs. Ping 'vm1' from 'vm2' and vice-versa. SSH 'vm1' from 'vm2' and vice-versa to verify both VMs are properly connected.

Note that by not connected to host the meaning is that VMs cant connect to host on its IP addresses and hence normal TCP/IP connnections cannot be established. But since bridge 'virbr3' is created on host, host can always capture packets traversing over this bridge. To understand the point better use command 'ping <vm2_ip>' on 'vm1' so that its pings to 'vm2'. Now on base machine use command '`tcpdump -vn -i virbr3 icmp`' to capture ICMP packets that host can receive on 'virbr3'. You will be able to capture ping requests from 'vm1' to 'vm2' and ping responses from 'vm2' to 'vm1' even though both VMs are not connected to host.

For each virtual interface of VM one corresponding virtual interface gets created on base machine as well. Run '`ifconfig`' command on base machine to see how many interfaces are shown when both 'vm1' and 'vm2' are running and are also connected to '`not_even_host`' network. Use command '`brctl show`' to see which interface is connected to which bridge. You can also use command '`brctl showmacs virbr3`' to see MAC address of virtual interfaces of VM1 and VM2 which are connected to this software bridge. You can verify by running '`ifconfig`' command in 'vm1' and 'vm2' that mac addresses shown when we run '`brctl showmacs virbr3`' are indeed MAC addresses of 'vm1' and 'vm2'.

Now use command '`free -m`' to see how much memory is free on base host after running 'vm1' and 'VM2'. If well over 512 MB RAM is free we can start 'vm3' using '`virsh start vm3`'. If less RAM is free then stop both 'vm1' and 'vm2'. Now use command '`virsh dominfo vm1`' to see memory allocated to 'vm1'. Now use command like '`virsh setmaxmem vm1 400000`' to reduce the memory allocated to 'vm1'. Do same for 'vm3'. Run command '`free -m`' and it will still show the same output as before even after reducing memory. (Why is part of lab queries?) Set RAM for 'vm3' also to be around 400MB using '`virsh setmaxmem vm3 400000`' command. Now start all three VMs 'vm1', 'vm2' and 'vm3'.

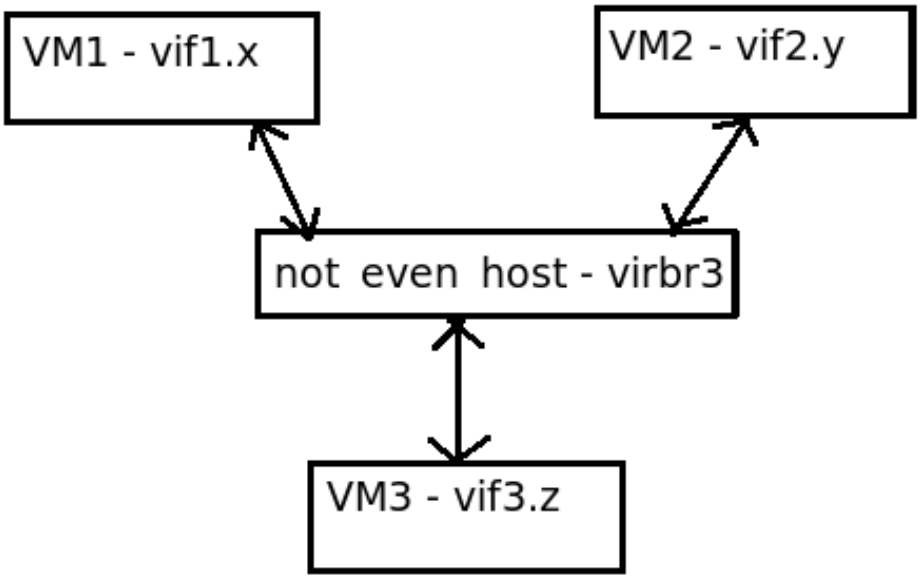
Take necessary steps so that all three VMs 'vm1', 'vm2' and 'vm3' are connected to 'not_even_host' network and have IPs 192.168.200.2, 192.168.200.3 and 192.168.200.4 respectively. Now use '`brctl`' commands to find interfaces for each VM. Now that virtual interfaces are named in format '`vif<p>.<q>`' where 'p' is the domain ID of the virtual domain for which the interface is created. Hence using combination of '`brctl`' and '`virsh`' commands you can

exactly identify which interface is for which VM.

Now ping from 'vm1' to 'vm2' and try to capture packets on 'virbr3', VM1 interface, VM2 interface and then VM3 interface. Note that when we ping from VM1 to VM2 we can capture packets either on 'virbr3', on VM1 interface or on VM2 interface but not on VM3 interface. Hence, our software bridge 'virbr3' acts very similar to bridges or switches and avoids sending packets to VM3 which are not intended for it.

Hence all the packets processed by bridge can be captured at virbr3. If we are interested in packets as seen/sent by a particular VM then we can capture packets only on that VMs virtual interface. Note that tcpdump/wireshark etc. are written for normal PC interface but they can be used on virtual interfaces as well. Hence even other tools like snort can be used on bridge interface or virtual network interface to detect intrusion and take corresponding action.

Figure 1: Connectivity diagram when using 'not_even_host' network



0.4.3 VMs connected to host using private LAN but not to outside world

Change VM1, VM2, VM3 configuration so that they are connected to 'host-only' network as described below. Disconnect them from 'not_even_host' network.

```
 <network>
<name>host-only</name>
   <bridge name="virbr1" />
<ip address="192.168.123.1" netmask="255.255.255.0">
  <dhcp>
    <range start="192.168.123.2" end="192.168.123.254" />
```

```
    </dhcp>
  </ip>
</network>
```

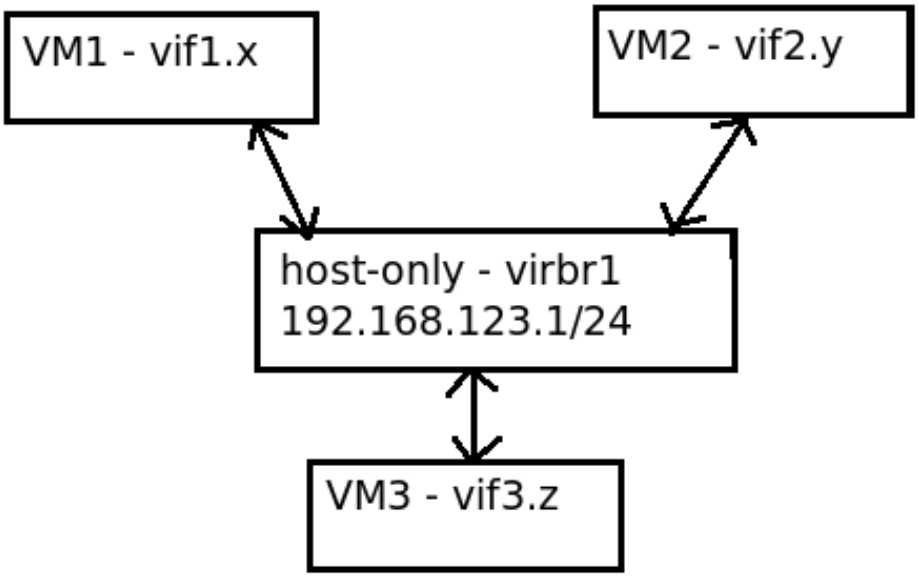
Restart network on all three VMs so that they get IPs from DHCP. To enable getting IPs from DHCP you might have to edit file '`/etc/sysconfig/network-scripts/ifcfg-eth0`' in all three hosts and change '`BOOTPROTO=dhcp`'. You should remove/comment lines that try to supply values for '`IPADDR`' and '`NETMASK`' as now these parameters will be given by DHCP server.

Find IPs of all three VMs and ping from one VM to other two VMs to ensure they are all able to connect to each other. Run '`ifconfig`' command on base machine and you should see IP address 192.168.123.1 assigned to bridge. Try to ping 192.168.123.1 from all three VMs and all three VMs from base machine to verify that VMs are all connected to base machine via 'virbr1' network named 'host-only'.

Now ping from base machine to VM1 and try to capture packets on 'virbr1' and then on virtual interfaces of VM1, VM2 and VM3. You should see ping requests and replied only on 'virbr1' and on VM1 virtual interface. While capturing VM2 or VM3 virtual interfaces you should not see any ping requests or replies.

Now ping from VM1 to VM2 and capture packets on 'virbr1' and then on virtual interfaces of VM1, VM2 and VM3. You should see ping requests and replies on 'virbr1', on VM1 and VM2 interfaces. Only VM3 virtual interface will not show any ping requests and responses. Hence there is no interface where we can capture packets that are meant only for host machine. We can capture all packets received by bridge on 'virbr1' which will include packets for host, but the packets of VM1, VM2 and VM3 will also get included in capture.

Figure 2: Connectivity diagram when using 'host-only' network



0.4.4 VMs connected to host using private LAN and to outside world using NAT

Create network named natted using

```
<network>
  <name>natted</name>
  <bridge name="virbr2" />
  <forward/>
  <ip address="192.168.124.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.124.2" end="192.168.124.254" />
    </dhcp>
```

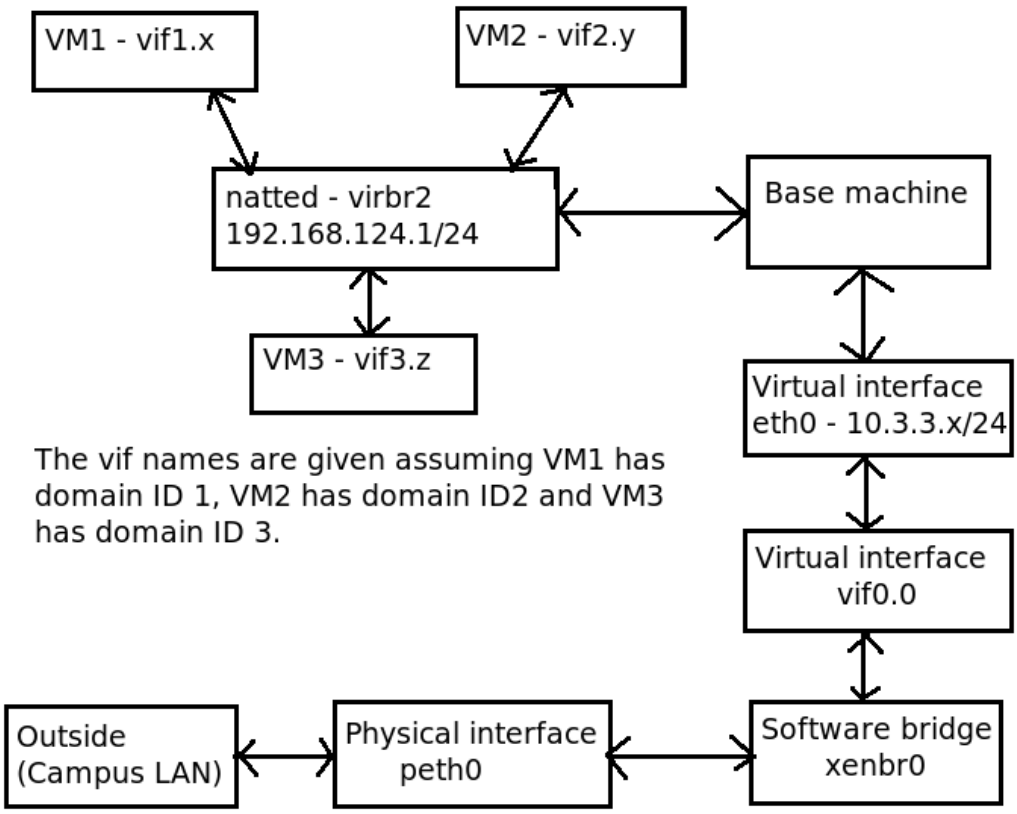
```
  </ip>
</network>
```

Disconnect all three VMs from network 'host-only' and connect them to network 'natted'. Now restart network on all three VMs so that they can take IPs related to new network to which they are connected.

Ping 'www.google.co.in' or 'www.yahoo.com' from VM1. Now capture packets on 'virbr2', VM1 virtual interface, 'eth0' and 'peth0'. You should see packets on all the four interfaces. Try to capture packets on VM2, VM3 virtual interface and you should not see any ping requests/replies to/from google server.

Now '`ping 10.3.3.1`' from your local machine. Capture packets on both 'eth0' and '`peth0`'. You should still see packets on both interface. See below diagram of how things are connected to understand the above output.

Figure 3: Connectivity diagram when using 'natted' network



You should understand that for VM in natted network if it has to send packet to Outside (Campus LAN) then the packet must go to VMs virtual

interface, then to 'virbr2', then to 'eth0', then to 'xenbr0', then to 'peth0' and then finally it will leave physical interface of host and go to campus LAN. The reverse path would also involve so many virtual interfaces, software bridges and physical interface.

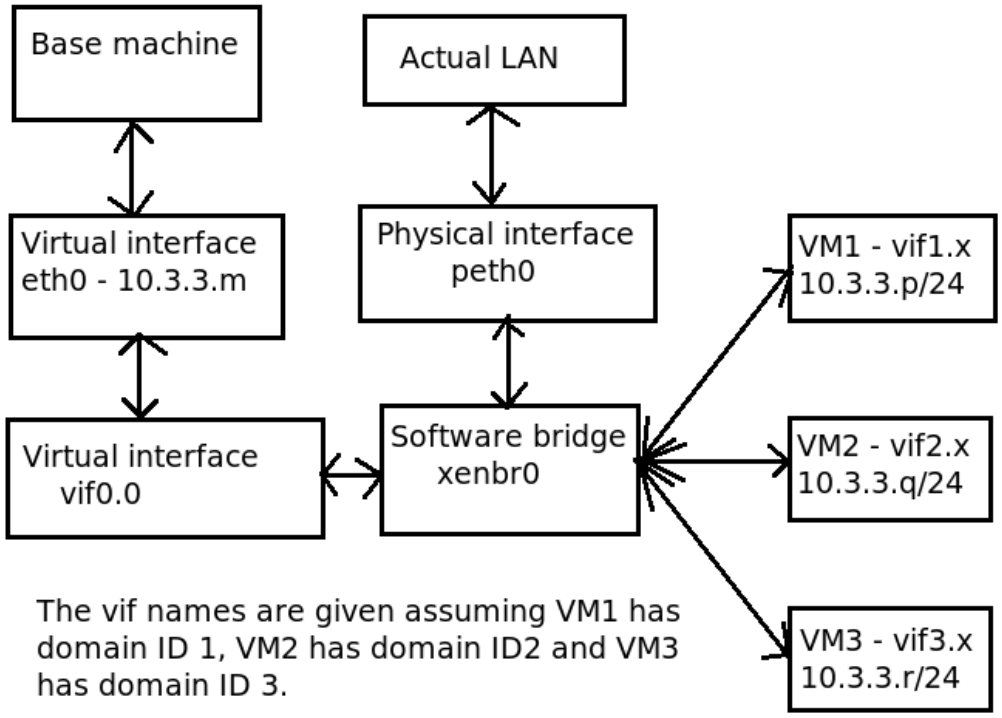
You can use '`iptables-save`' command verify the iptables rules that cause host to put its own IP when it receives packets from 'virbr2', before it sends them outside via 'eth0'.

0.4.5 VMs directly connected to outside LAN using bridging

Disconnet all three VMs (VM1, VM2 and VM3) from natted network and connect them to 'xenbr0' bridge. Restart network if VMs do not automatically get 10.3.3.0/24 series IP.

'`ping www.google.co.in`' from VM1. You should be able to capture these packets on virtual interface of VM1, 'xenbr0' and peth0. But the same packets should not be visible on 'vif0.0', 'eth0' or virtual interfaces of other VM.

Figure 4: Connectivity diagram when using 'xenbr0' bridge



0.5 Lab queries

1. When we add a bridge interface to running VM, why do we need edit file '`/etc/sysconfig/network-scripts/ifcfg-eth0`' before we can start using the interface?

2. Why do we assign static IPs to hosts connected to 'not_even_host' network. What is the problem in assigning IPs usind DHCP? What can be done so that IPs in 'not_even_host' network are also assigned using DHCP?

3. If we have VM1 and VM2 running and we use command '`free -m`' to see available memory we will get output that some X MB RAM is available for base machine. Now if we poweroff VM1 and VM2 and even reduce the maximum RAM allocated to both VMs then also '`free -m`' command will show that X MB RAM is available for base machine. Why this X has not increased? How can we increase the amount of RAM available to base machine if both VM1 and VM2 are configured to use less RAM then before?

4. Try to ping 'www.google.co.in' or 'www.yahoo.com'. Or try to use '`nslookup www.rediff.com`' on VM1, VM2 or VM3 when they are connected to host-only network. Is there something wrong with the output and name 'host-only'? If yes, what is the problem and how can you prevent it?

5. When we use 'host-only' or 'natted' networks, Is it possible to host a webserver on VM that can be accessed from Outside (that is campus LAN)? If yes, how? What are the limitations of the approach that you have mentioned?

6. Create a setup where we have VM1 and VM2 such that:

   (a) All packets from VM2 go directly to outside network via virtual interface to 'xenbr0' to 'peth0'

   (b) All packets from VM1 which are not for host or other VMs in same network go to VM2. Then from VM2 these packets should go to outside network.

   Try traceroute from VM1 to 'www.google.co.in' and you should see VM2 IP address on the route which verifies that packets from VM1 to outside world do actually go through VM2.

   Mention detailed steps used to achieve the above setup including:

(a) Virtual networks that you have created

(b) Number of interfaces on each VM and to which virtual network they are connected.

(c) Detailed IP address, netmask, etc. network configuration of both VMs.

(d) Any special changes done on VM1 or VM2 to make things work.

In case you are not able to achieve the output at least mention what all you tried, use '`tcpdump`' to verify till what point packet is reaching and where it is getting dropped.

7. *You do not have to try to create the below mentioned setup. Just explain how it can be created based on learning/experiences from previous question.*

   Setup a network with two VMs - VM1 and VM2 where:

   (a) All packets of VM2 go to outside network via NAT using base machines IP address. That is, all packets originating from VM2 first go to VM2's virtual interface then to virtual bridge IP of base machine. Then host NATs the packets and puts its on LAN IP and sends the packets via 'eth0' to 'vif0.0' to 'xenbr0' to 'peth0' to outside.

   (b) All packets from VM1 which are not for host or other VMs in same network go to VM2. Then from VM2 these packets go to outside world in same way as packets originating from VM2 do.

   What are the differences in steps required to achieve this in comparison to steps required to solve previous problem.

8. Setup a virtual network of VM1 and VM2 such that:

   (a) Both VM1 and VM2 can contact outside world using natted network.

   (b) VM1 and VM2 are on different networks and hence belong to different IP ranges.

   Mention how did you achieve the setup. Also mention what is required to ensure that both VMs VM1 and VM2 can contact each other. Also mention what is required to ensure that both VMs cant contact each other.