1. Creating a UDP tunnel:

   Shut down your *router ONLY*! Use the following commands on the *host* machine (e.g., thesituation, britishknights, or mclovin). In the commands, replace **ROUTER_NAME** with the name you gave to your *router*!

   ```
   VBoxManage modifyvm ROUTER_NAME --nic4 generic
   ```

   Set the nic to use the generic driver (generally used only for UDP tunneling)

   ```
   VBoxManage modifyvm ROUTER_NAME --nicgenericdrv4 UDPTunnel
   ```

   Set the nic generic driver to be a UDP port.

   ```
   VBoxManage modifyvm ROUTER_NAME --nicproperty4 dest=HOST_IP
   ```

   Where dest is the IP address of the *host* that your *guest*

   mclovin IP: 64.106.46.60
   thesituation IP: 64.106.46.61
   britishknights IP: 64.106.46.62

   Replace **HOST_IP** with one of the above IPs depending on which physical machine your VMs reside on. For example, I created a UDPTunnel from my router VM on *thesituation* (64.106.46.61) with **ROUTER_NAME** = *brolabrouter* to Raj's router VM on *mclovin* (64.106.46.60), with **ROUTER_NAME** = *roust11*

   The specific command I would run on *thesituation* would be:

   ```
   VBoxManage modifyvm brolabrouter --nicproperty4 dest=64.106.46.60
   ```

   The specific command Raj would run on *mclovin* would be:

   ```
   VBoxManage modifyvm roust11 --nicproperty4 dest=64.106.46.61
   ```

   Next, we have to configure a source port **SPORT** with:

   ```
   VBoxManage modifyvm ROUTER_NAME --nicproperty4 sport=SPORT
   ```

   SPORT is the *software* port that will listen on the *host* (e.g., thesituation, britishknights, or mclovin).
   To determine what *YOUR* **SPORT** is, take 40000 and add your student number to it. For example, my student number is 8. I would run the following command on *thesituation*:

   ```
   VBoxManage modifyvm brolabrouter --nicproperty4 sport=40008
   ```

   Raj's student number is 7, therefore, he would run the following command on *mclovin*:

   ```
   VBoxManage modifyvm roust11 --nicproperty4 sport=40007
   ```

   The reason we need to do this, is because each of your sets of 3 VM's is running in the context of your profile on the *hosts*. They are in completely different virtual address spaces and therefore need to use some form of IPC mechanism in order to communicate (which in this case, is a UDP socket). This socket is used to connect one VM to another across different (or the same) physical hosts on which they are running. Please note though that the socket that we are opening is *COMPLETELY INVISIBLE* to the VM. All the VM sees is that you have connected one end of an ethernet cable to it, though keep in mind that this cable is not physical in any way, but simply a *virtual* ethernet cable.

   To complete the UDPTunnel, we need to tell VirtualBox where to connect the other end of your virtual ethernet cable. To do this, we run the following command.

```
VBoxManage modifyvm ROUTER_NAME --nicproperty4 dport=DPORT
```

DPORT is the *software* port that will listen on the *host* (e.g., thesituation, britishknights, or mclovin) that you will be connecting *to*.

For my VM *brolabrouter* running on *thesituation* (60.106.46.61), I would run the following command:

```
VBoxManage modifyvm brolabrouter --nicproperty4 dport=40007
```

And for Raj, whose VM is *roust11* on *mclovin* (64.106.46.61), he would run:

```
VBoxManage modifyvm roust11 --nicproperty4 dport=40008
```

Pay *CAREFUL* attention to the *dport* property. I had to tell VirtualBox that the other end of the virtual ethernet cable should be connected to the port 40007. Similarly, Raj had to tell VirtualBox that the other end of his virtual ethernet cable should be connected to port 40008.

Once all of these virtual hardware configurations have been completed, go a head and power up for VMs as normal.

In my case:

```
VBoxManage startvm brolabrouter --type headless
```

2. NIC Configuration:

After your router has booted back up and you can connect back to it, we need to configure the IP address of the newly created network interface card (NIC). For this to work, both routers will need to be on the *SAME* subnet! Virtual box will allow at most 8 NICs to be added to a virtual machine. For this lab, you will only be adding the above NIC (nic4). This will be reflected in the virtual machine as *eth3* if

```
 $ ifconfig -a
```

is run. Similarly, if you had run just

```
$ ifconfig
```

You would only see lo, eth0, eth1, and eth2 and no eth3. That is because eth3 has not been given an IP address yet and so *NO* entry will exist for it in */etc/network/interfaces*. We will need to configure these entries statically, similar to how we did in the previous lab. The following is what my */etc/network/interfaces* file looks like.

```
$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 192.168.8.1
```

```
netmask 255.255.255.192

auto eth2
iface eth2 inet static
address 192.168.8.65
netmask 255.255.255.192

auto eth3
iface eth3 inet static
address 192.168.8.129
netmask 255.255.255.252
```

Here is the corresponding */etc/network/interfaces* file for Raj.

```
$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 192.168.7.1
netmask 255.255.255.192

auto eth2
iface eth2 inet static
address 192.168.7.65
netmask 255.255.255.192

auto eth3
iface eth3 inet static
address 192.168.8.130
netmask 255.255.255.252
```

The thing to note is that *eth3* is on a different subnet than either *eth1* or *eth2*. It also has a different netmask. More specifically, we will be specifying much smaller subnets (/30s to be exact) that only allow 2 hosts per subnet. The first is 192.168.8.129, and the second will be 192.168.8.130. Note that 192.168.8.129 will be on one of either you or your partner's VMs (in this case, 192.168.8.129 is on my router), and the other partner will obviously have 192.168.8.130 (Raj in this case).

More generally, the two IP addresses that you and your partner's *eth3* interfaces should have the form 192.168.A.B/30

Where A is a student number of *either YOU or YOUR partner ONLY*!

B is an a number that is greater than 128. Look in the references section for links to some useful sites.

And the /30 specifies a subnet with the netmask 255.255.255.252.

3. Quagga configuration:

   Next, we are going to use the network protocol suite *Quagga* to configure routes dynamically between *your* network and your *partner's*.

```
$ sudo apt-get -y install quagga
```

   Edit the file */etc/quagga/daemons*. It should look like below:

```
$ cat /etc/quagga/daemons
# This file tells the quagga package which daemons to start.
#
# Entries are in the format: <daemon>=(yes|no|priority)
#   0, ``no''  = disabled
#   1, ``yes'' = highest priority
#   2 .. 10  = lower priorities
# Read /usr/share/doc/quagga/README.Debian for details.
#
# Sample configurations for these daemons can be found in
# /usr/share/doc/quagga/examples/.
#
# ATTENTION:
#
# When activation a daemon at the first time, a config file, even if it is
# empty, has to be present *and* be owned by the user and group ``quagga'', else
# the daemon will not be started by /etc/init.d/quagga. The permissions should
# be u=rw,g=r,o=.
# When using ``vtysh'' such a config file is also needed. It should be owned by
# group ``quaggavty'' and set to ug=rw,o= though. Check /etc/pam.d/quagga, too.
#
# The watchquagga daemon is always started. Per default in monitoring-only but
# that can be changed via /etc/quagga/debian.conf.
#
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
isisd=no
babeld=no
```

   Note that *ANY* line beginning with # is a comment and simply ignored. The two important lines are *zebra=yes* and *ripd=yes*. This will tell quagga that it should start the zebra and ripd daemons when the system boots.

   Quagga comes with example configuration files that we will use to configure both zebra and ripd. Specifically, the files */etc/quagga/zebra.conf* and */etc/quagga/ripd.conf* will be used by quagga to configure the daemons.

```
$ sudo cp /usr/share/doc/quagga/examples/zebra.conf.sample /etc/quagga/zebra.conf

$ sudo cp /usr/share/doc/quagga/examples/ripd.conf.sample /etc/quagga/ripd.conf

$ sudo chown quagga:quaggavty /etc/quagga/*.conf

$ sudo chmod 640 /etc/quagga/*.conf
```

```
$ sudo /etc/init.d/quagga start
```

Now that the zebra and ripd servers are running, we can configure them by connecting to them using telnet.(The default password can be seen in */etc/quagga/zebra.conf*. Pro-tip, it's *zebra*)

```
$ telnet localhost zebra
```

Enter the default password.

```
Router> en
```

Enter the default password...again...

```
Router# conf t
Router(config)# hostname MyRouter
MyRouter(config)# log file /var/log/quagga/zebra.log
MyRouter(config)# debug zebra events
MyRouter(config)# debug zebra packet
MyRouter(config)# password 2l33t4u
MyRouter(config)# enable password 2l33t4u
MyRouter(config)# interface eth1
MyRouter(config-if)# ipv6 nd suppress-ra
MyRouter(config-if)# exit
MyRouter(config)# interface eth2
MyRouter(config-if)# ipv6 nd suppress-ra
MyRouter(config-if)# exit
MyRouter(config)# interface eth3
MyRouter(config-if)# ipv6 nd suppress-ra
MyRouter(config-if)# exit
ip forwarding
MyRouter(config)# do wr
Configuration saved to /etc/quagga/zebra.conf
MyRouter(config)# exit
Connection closed by foreign host.
```

Now that zebra is configured, we can check out the change made to the */etc/quagga/zebra.conf*:

```
$ sudo cat /etc/quagga/zebra.conf
!
! Zebra configuration saved from vty
!   2015/09/28 19:35:50
!
hostname MyRouter
password 2l33t4u
enable password 2l33t4u
log file /var/log/quagga/zebra.log
!
debug zebra events
debug zebra packet
!
```

```
interface eth0
 ipv6 nd suppress-ra
!
interface eth1
 ipv6 nd suppress-ra
!
interface eth2
 ipv6 nd suppress-ra
!
interface eth3
 ipv6 nd suppress-ra
!
interface lo
!
ip forwarding
ipv6 forwarding
!
!
line vty
!
```

Next, we need to configure ripd:

```
$ telnet localhost ripd
```

Enter default password.

```
Router> en
```

One last time with the default password...

```
Router# conf t
Router(config)# hostname MyRouter
Router(config)# password 2l33t4u
Router(config)# log file /var/log/quagga/ripd.log
Router(config)# debug rip events
Router(config)# debug rip packets
Router(config)# router rip
Router(config)# version 2
Router(config)# network eth1
Router(config)# network eth2
Router(config)# network eth3
Router(config)# do wr
Configuration changes saved to /etc/quagga/ripd.conf
```

Now that the */etc/quagga/ripd.conf* file has been updated, we can check it out.

```
$ sudo cat /etc/quagga/ripd.conf
!
! Zebra configuration saved from vty
!   2015/09/28 19:38:15
!
```

```
hostname MyRouter
password 2l33t4u
enable password 2l33t4u
log file /var/log/quagga/ripd.log
log stdout
!
debug rip events
debug rip packet
!
router rip
 version 2
 network eth1
 network eth2
 network eth3
!
line vty
!
```

Now restart quagga:

```
$ sudo /etc/init.d/quagga restart
```

The last order of business is to make sure the firewall doesn't interfere.

```
$ sudo ufw allow 520/udp
```

4. References:

   Quagga configuation with screen captures:

   https://rbgeek.wordpress.com/2012/05/01/rip-routing-between-ubuntu-and-centos-using-quagga/

   RIP Protocol:

   https://en.wikipedia.org/wiki/Routing_Information_Protocol

   https://tools.ietf.org/html/rfc2453

   Subnet's & Masking:

   http://www.aelius.com/njh/subnet_sheet.html