Introduction

This lab exercise will introduce RIP, OSPF as internal routing protocols. Furthermore, BGP will be briefly used as exterior routing protocol

The lab is based on a virtual environment as sketched in Figure 1. Please use this figure as reference as this will come in handy.

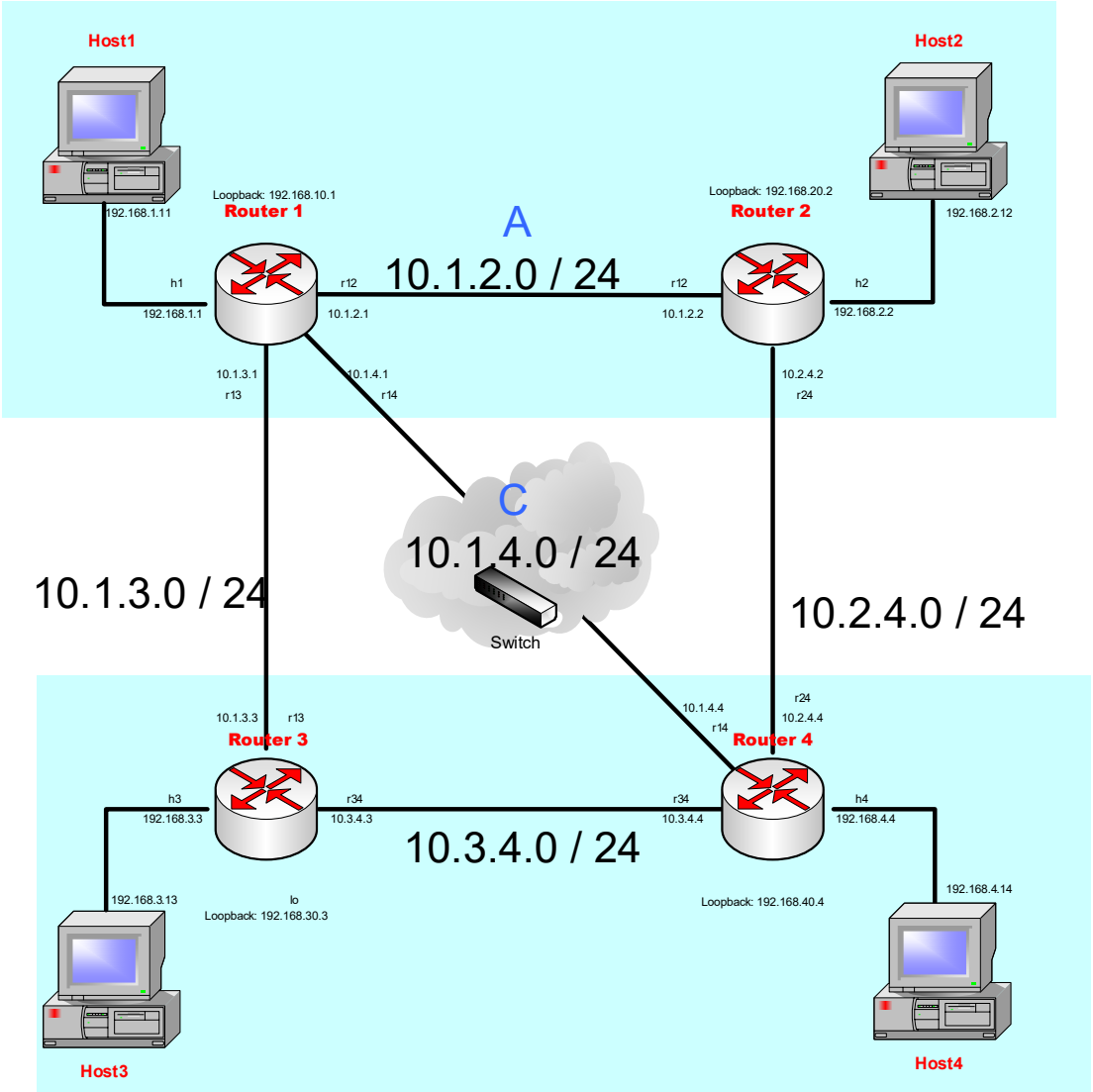

Figure 1: Router network structure

The lab comprises four routers and four hosts with the interfaces and network spaces as given in the Figure. Note that the network interfaces use lower case letters.

Lab environment

The lab environment is built on the virtual machine (VM) used previously in the course The lab uses docker containers and namespaces to establish lightweight virtual machines (within the virtual machine) and to connect them, respectively.

Start the VM from Oracle VirtualBox and log in using username “cyber” and password “34334”.

Open a terminal windows and start the lab with:

```
./lab-start.sh
```

This will prepare the lab first time. Later executions can be done a bit faster using following command from the 34334 folder:

```
sudo python lab_webapp.py
```

The first time the lab is started it can take a bit of time and it will require internet access to obtain an Ubuntu image and all the needed extra packages. You can follow the establishment in the terminal console. Building the “base” image can be time consuming. When complete a web page will open and look like Figure 2. If no webbrowsers open, then open one yourself and go to URL `localhost:5000`. Do not click any of the buttons before the note “Ready to serve” in the terminal window.

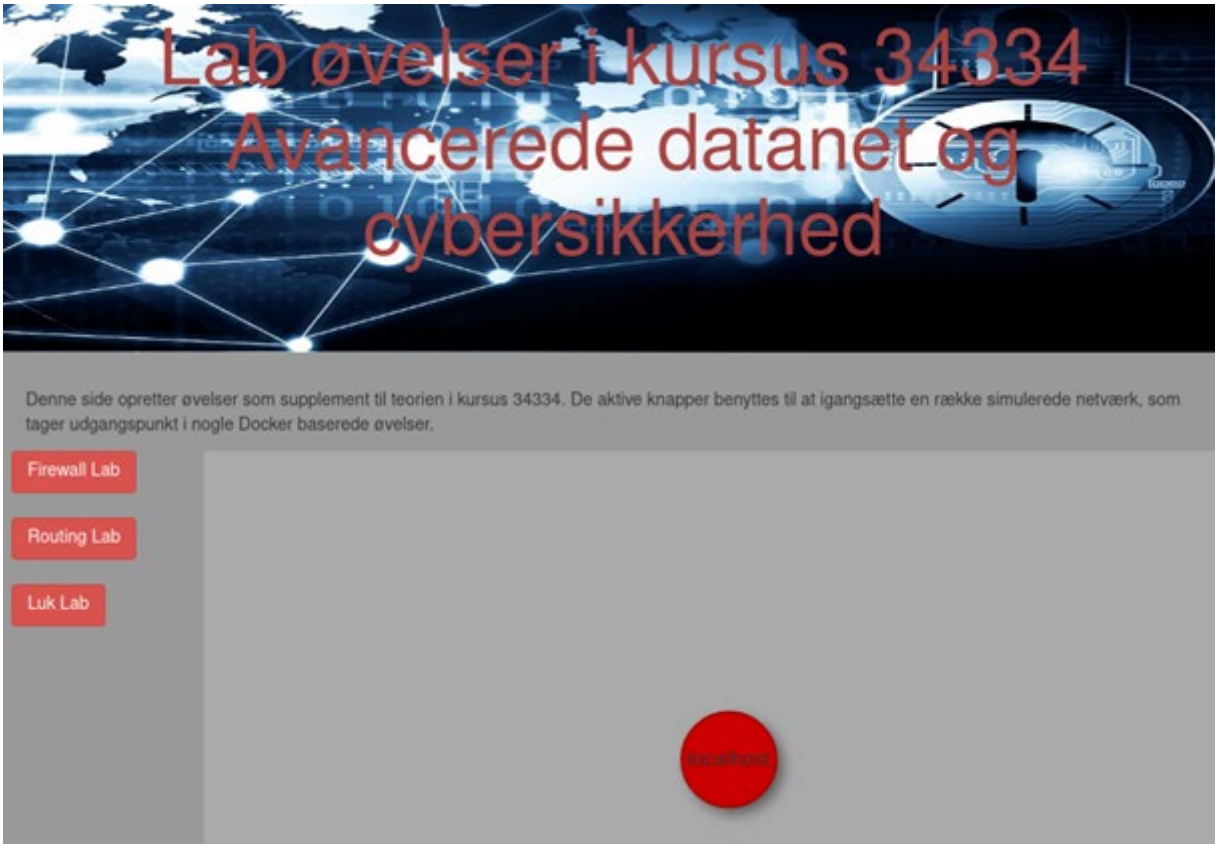

Figure 2: Start screen lab environment

The web page shows the current network setup, which for now is empty except from the localhost, which is the root namespace.

Click “Routing Lab” to establish the Routing Lab Environment with the four routers and four host in addition to two switches. When the console window relaxes, you might have to press F5 to see

the generated environment including some of the IP addresses assigned to the hosts and router1 and router1. You might have to drag the devices to make them look nice as in Figure 3.

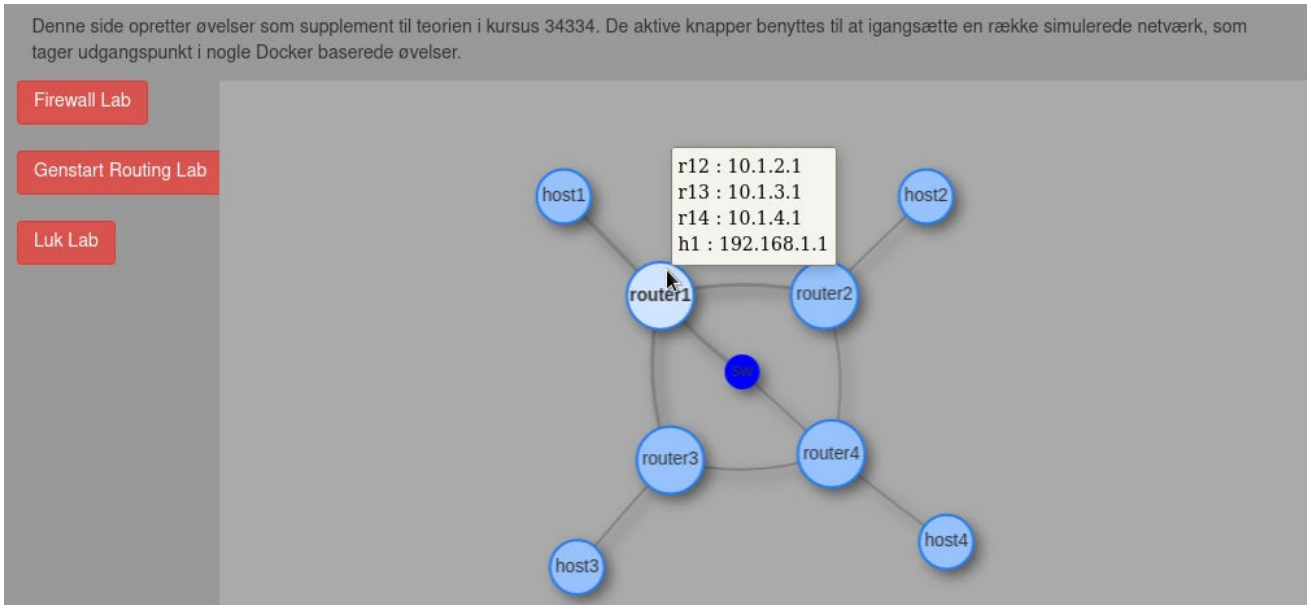

Figure 3: Router lab environment

Quagga

Quagga is routing platform software, which can be used to upgrade a standard linux to a router. It, among others, support RIP, OSPF and BGP. The setupe is handled via a set of configuration files named daemons.conf, zebra.conf, ripd.conf, ospfd.conf, bgpd.conf etc.

In this lab, Quagga is installed on some Ubuntu docker containers and it is mostly left for the user – you – to configure the configurations files according to the purpose.

Router configurations

The lab environment is preestablished with the RIP protocol working on router1 and router2. This can be verified by pinging host1 and host2 from a new terminal window utilizing network namespaces as discussed during the course.

If you need to change configuration files etc in the routers (or hosts) you can access the docker container with a bash terminal using the following command (here for router1) from a terminal

```
sudo docker exec -ti router1 bash
```

You also need to access the docker container to start and stop services etc. Here you can use exactly the applications, which are installed in the docker container Note that Wireshark is not installed and graphical applications need a correctly set display variable. However, you will only need Wireshark accessing the routers via namespaces, and here Wireshark works as it should

If you need to copy a file from the docker container (her router1 ripd.conf file) to you local Kali you can use the following

```
sudo docker cp router1:/etc/quagga/ripd.conf ripd-kopi.conf
```

As previously described, in most cases you can just use the network namespaces . E.g., if you would like to ping router2 (r12 interface) from router1, you access router1 namespace and use the ping command. Here you use the applications installed in your Debian virtual machine.

```
sudo ip netns exec router1 ping 10.1.2.2
```

Using the namespace you can also run graphical applications like wireshark without considering the display parameter

```
sudo ip netns exec router1 wireshark
```

You can close the lab clicking "Luk Lab". *Be aware that this means that all changes to configurations files in the docker containers will be lost*. So make sure to copy them to the localhost VM.

It might be beneficial to develop small scripts for copying configurations files between the host machine and the relevant docker containers.

Enjoy the exercise!

RIP

Please refer to Figure 1 for the network structure

First, we have to configure the routers to use a protocol to learn about the network. In this case the RIP protocol.

Q 1. For each router, list the networks, which should be announced by the RIP protocol to know all the information of the network?

Go to Router 1 and start bash command prompt `sudo docker exec -ti router1 bash`

Go to the location of Quagga configuration files: `cd /etc/quagga`

Check the content of the "daemons" file: `nano daemons`. Note that "zebra" and "ripd" are enabled.

Zebra is responsible for overall Quagga configuration incl. interface configuration and static routes. Try `nano zebra.conf` to check the current configuration.

Zebra.conf can be used to define interfaces and interface properties. It can be ignored if the links are already defined via ip addr or if no special properties are needed.

Note that in the preconfigured zebra files you should remove the underscores in the interface names in order to enable the link-detect property. We apologise this minor error in the setup.

Check the configuration of RIP on router 1: `nano ripd.conf`.

Take a look at the configuration of RIP and check that it is done in accordance with Figure 1

Quagga service is already running, but you can later restart e.g. after change of configuration files:

```
service quagga restart
```

Go to Router 2 and start bash command prompt `sudo docker exec -ti router2 bash`

Repeat the steps done for Router1.

Go to Router 3 and start bash command prompt `sudo docker exec -ti router3 bash`

Go to the location of Quagga configuration files: `cd /etc/quagga`

Edit zebra.conf (`nano zebra.conf`) and and ripd.conf (`nano ripd.conf`) to become in accordance with Figure 1.

Restart Quagga when configuration is complete : `service quagga restart`

Go to Router 4 and start bash command prompt `sudo docker exec -ti router4 bash`

Repeat the steps done for Router3.

Check the content of the router table in Router 1: `ip route`

Repeat the steps above for Router 2-4.

Start wireshark between router 1 and 2 and check the content of RIP response messages.

Q 2. Find a RIP response message and explain the included routing information?

Start a ping from host 1 to host 3: `ip netns exec host1 ping 192.168.3.13`

The route taken can be verified with traceroute: `traceroute dest_ip -n`

Q 3. Report the route taken using traceroute?

To see that the network can recover a loss of a connection pull the "cable" (router1-router3):

E.g. shutting down the link between router1 and router3

```
sudo ip netns exec router1 ip link set r13 down
```

Q 4. Explain and document the route taken from router1 to router3?

To see how quick the recovery is, reconnect the cable `ip link set r13 up`. Then, use the command prompt to ping from host 1 to host 3 (both ways). Use the command:

```
fping [IP address of the computer] -c 200 -t 6
```

Take out the cable (shutdown link) again and see what happens. Press *Ctrl+C* when it starts receiving again.

You can see how many packets were lost and in error, the packets were sent with a timeout of 6ms so you can calculate the approximate time to recover a route.

Q 5. How long did it take to recover the route?

Now start fping again and see what happens when the cable is connected again.

Q 6. How many packets are lost? Explain your result?

Start fping without the –t 6 command.

Now have a close look on connection between Router 1 and Router 4. We will disconnect cables going to the Switch between router 1 and 4 to learn on RIP operation.

Ping from host4 to host1. Disconnect the first cable (router1-sw) and wait for some time.

Q 7. How long time does it take for ping to work again? Explain the difference in recovery compared to Q5.

Connect again the first cable (router 1- SW) and check that ping is again working as before.

Disconnect the first cable (router 1- SW) and the second cable (SW – router 4) shortly after.

Q 8. What happens when the second link is disconnected? Explain the mechanisms for recovery?

OSPF

Again, refer to the network structure in Figure 1

Go to Router 1 and start bash command prompt `sudo docker exec -ti router1 bash`

Go to the location of Quagga configuration files: `cd /etc/quagga`

Change the content of the "daemons" file such that we will run OSPF instead of RIP: `nano daemons`. You should now change the lines with ripd and ospfd to: `ripd=no`, `ospfd=yes`.

We do not need to change zebra.conf that specifies the interfaces, as there is no change to the configuration.

Check the configuration of OSPF on router 1: `nano ospfd.conf`. The configuration must be finalised according to our network structure.

The Quagga service is already running, but we need to restart due to change of configuration files: `service quagga restart`

Repeat the steps above for Router 2-4

Now all the routers run OSPF. Now the routing table should have the network topology.

Check the content of the router table in Router 1: `ip route`

Q 9. Report the router table.

Repeat the steps above for Router 2-4.

Go to Host 1 and start bash command prompt: `sudo docker exec -ti host1 bash.`

Start a ping from host 1 to host 3: `ping 192.168.3.13`

Q 10. Report the route taken using traceroute from a netns: `traceroute 192.168.3.13 -n`

Now have a close look on connection between Router 1 and Router 4. We will disconnect cables going to the Switch between router 1 and 4 to learn on OSPF operation compared to RIP.

Start wireshark between router 1 and 2 and explain the different OSPF protocol messages observed then the following steps are taken:

Ping from host 4 to host 1. Disconnect the first cable (router 1- SW) and wait for some time.

Q 11. Compare recovery time with measured recovery time for similar scenario in RIP (Q5)? Explain the differences!

Connect again the first cable (router 1- SW) and check that ping is again working as before.

Disconnect the first cable (router 1- SW) and the second cable (SW – router 4) shortly after.

Q 12. Explain the recovery time comparing with Q7?

In OSPF you can manipulate the metrics (cost of using the link).

Start a ping from host 1 to host4 and from host4 to host1 and note the TTL

Add the following lines to ospfd.conf in router 1 only

```
interface r14
   ip ospf cost 100
```

This will add a cost of 100 for the 1_4 link.

Remember to restart Quagga: `service quagga restart`

Q 13. Notice TTL again. What is the difference for each session compared to the situation before changing the metric? What is the current metric for the path to host4 and how was this value created?

In OSPF you have something called “areas”.

Q 14. What do you use areas for?

Make router 1 and 2 belong to area 1 and router 3 and 4 belong to area 2.

Q 15. Ping from host1 to any other host and explain the results?

Border Gateway Protocol (BGP)

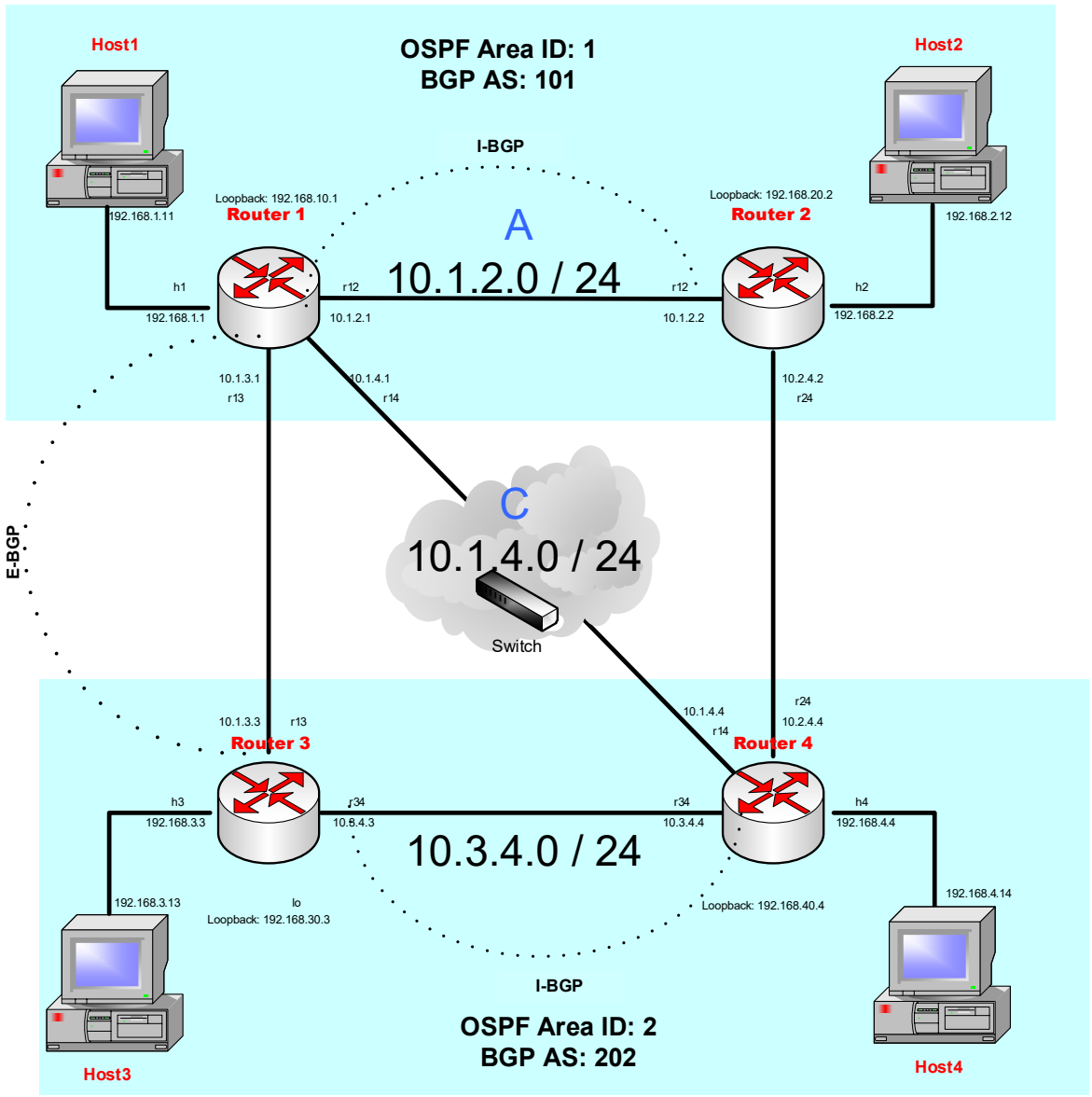

Figure 4: BGP Routing network structure

Use the configuration from the OSPF part. Remember that Router 1 and Router 2 resides in Area 1 and that Router 3 and Router 4 resides in Area 2. We will assign AS number 101 to Area 1 and AS number 202 to Area 2

BGP is used to exchange information between the networks. There are two types of BGP, External (E-BGP) and internal (I-BGP).

In the following, you will set up Router 1 (AS101) and Router 3 (AS202) and exchange information with E-BGP.

Start with Router 1:

Go to Router 1 and start bash command prompt `sudo docker exec -ti router1 bash`

Go to the location of Quagga configuration files: `cd /etc/quagga`

Change the content of the "daemons" file such that we will add BGP: `nano daemons`. You should now change the line with bgpd to: `bgpd=yes`. We do not need to change zebra.conf that specifies the interfaces, as there is no change to the configuration.

Create a new BGP configuration file: **nano bgpd.conf.** Add the following content:

```
hostname router1
password cyber

router bgp 101
  bgp router-id 10.10.0.1
  redistribute connected
  neighbor 10.1.3.3 remote-as 202
  redistribute ospf
```

The Quagga service is already running, but we need to restart due to change of configuration files:

```
service quagga restart
```

Do the same steps for Router 3, only change is the configuration file:

```
hostname router3
password cyber

router bgp 202
  bgp router-id 10.10.0.3
  redistribute connected
  neighbor 10.1.3.1 remote-as 101
  redistribute ospf
```

Remember to restart Quagga.

Ping from host 1 to host3 and from host 2 to host 4

Q 16. Are replies observed from host 3? Are replies observed from host 4? Explain why?

In the configuration files for BGP, we redistribute routes learned from OSPF. Consider how we can do the opposite and redistribute BGP learned routes. Use the following redistribute command in the relevant files:

```
redistribute bgp
```

Q 17. Report the bgp and ospf configuration files for router 1 and router 2?

Ping again from host2 to host4 and vice versa.

Q 18. Which path (indicated in sequence of routers) is now taken for the ping messages?

Q 19. Print the routing table for Router 4 `ip route -n` and check that all networks can be reached.

Q 20. Capture with wireshark BGP messages between router 1 and router 3? How often are Keep Alive messages transmitted? What is needed in order to receive other BGP messages? Report the wireshark trace or screenshot?

You have now completed the exercise and gained hands-on experience with the routing protocols BGP, OSPF and RIP. We used the Open Source solution Quagga, but the principles are completely similar for real Juniper and Cisco routers.