SYS2 (NET) Laboratory 8 : Broadcast, Multicast & Routing

These weekly lab scripts have been written as a self-study resource i.e. an activity to be worked on at your own pace, in your own time. The timetabled practical sessions are an opportunity to get advice and guidance. Therefore, you may not always finish each lab during the timetabled session, but do finish each lab's tasks in your own time. Hardware labs are open Mon-Fri, 9:00-17:00.

The aim of this lab is to look in more detail at how packets are routed across a network. Before starting this lab make sure you have watched video **Lecture 4A & 4B**. In the last lab we saw how manual routing rules could be used to define specific routes e.g. routing the internal loopback IP addresses across `tun0` and `tun1`. This type of manual configuration is fine for small systems, but for larger systems becomes impractical. Therefore, we need routing protocols that allow routers to inform other routers about the routes they know about i.e. to allow routers to expand their routing tables beyond what they can "see" on their local network. As you probably guessed routing packets between the network of networks that is the Internet is a little complex i.e. here be dragons :). However, these routing protocols fall into two general categories: Exterior Gateway Protocols (EGP) and Interior Gateway Protocols (IGP). EGPs are used to exchange routing information between Autonomous Systems (AS) i.e. different networks. IGPs are used to exchange routing information within an AS. To keep things simple we are going to limit our investigation to IGPs i.e. to networks like those you are using in the computer science building. Initially we shall jump back up the Internet protocol stack to the Application layer to examine the Routing Information Protocol (RIP) and consider how routing information is transferred between routers. Finally, we will look at some more complex manual routing examples to better understand of how routing rules work. At the end of this practical you will understand how :

- direct and local broadcast addresses are used.
- routing protocols can be used to update a router's routing table.
- to configure the Quagga software suite to implement a router on a Raspberry Pi.
- to configure manual routing and routing protocols on the Mikrotik router.

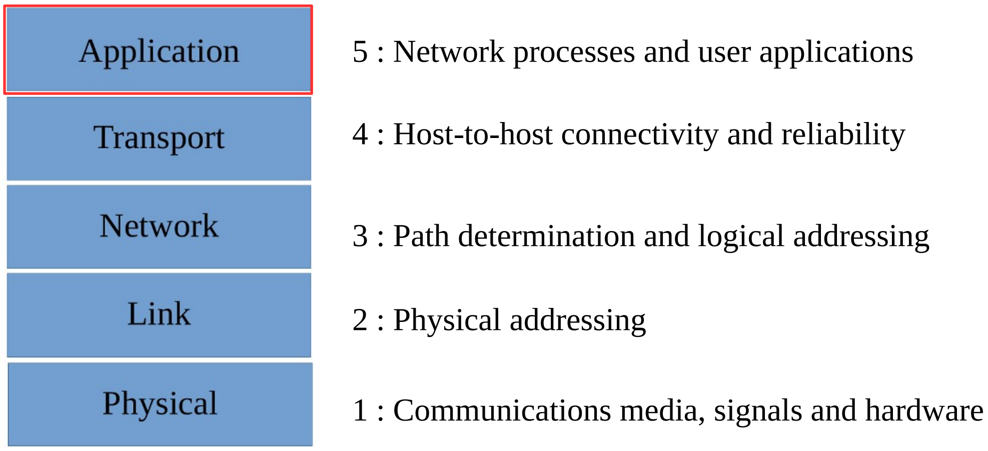

Application
5 : Network processes and user applications
Transport
4 : Host-to-host connectivity and reliability
Network
3 : Path determination and logical addressing
Link
2 : Physical addressing
Physical
1 : Communications media, signals and hardware

Figure 1 : The Internet protocol stack

Task 1

The RIP protocol (RFC 1058: https://datatracker.ietf.org/doc/html/rfc1058) was developed in 1988, using UDP as its transport protocol and port number 520. Routing calculations are performed using a distance-vector routing algorithm i.e. routes are selected on the basis of how many routers a packet has to pass through to get from A to B (number of hops). Neighbouring routers share their routing tables, allowing each router to work out the shortest, lowest cost path. For more information on this algorithm refer to:

http://en.wikipedia.org/wiki/Distance-vector_routing_protocol/

This raises the question: how should routers share (communicate) this routing information? Since a router may not be aware of other routers on a network unicast transmission i.e. peer-to-peer communications may not be possible. Solution: broadcast, as shown in figure 2.

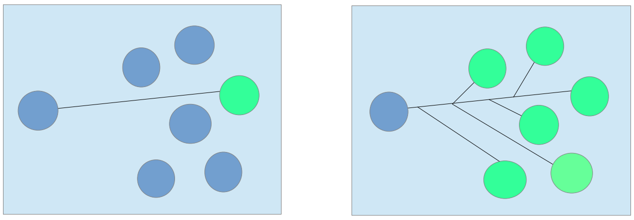


Figure 2 : unicast (left): one-to-one, broadcast (right): one-to-many

Now, rather than using a router's IP address (unicast), the transmitting router uses the network's broadcast IP address. Any router receiving a network packet with this destination address will process this packet as if it was its own IP address.

To calculate the network and direct broadcast address of each NIC, we need to know that network's network mask. The NIC's address is bitwise ANDed with its network mask, then bit positions in the network mask that were 0's are set to 1's in the broadcast address, as shown in figure 3.

NIC addr : 192.168.100.3	11000000 10101000 01100100 00000011
Network mask : 255.255.0.0	11111111 11111111 **00000000 00000000**
Broadcast addr : 192.168.255.255	11000000 10101000 11111111 11111111
Network addr : 192.168.255.255	11000000 10101000 00000000 00000000
Host addr : 172.16.100.5	10101100 00010000 01100100 00000101
Network mask : 255.255.255.252	11111111 11111111 11111111 111111**00**
Broadcast addr : 172.16.100.7	10101100 00010000 01100100 00000111
Network addr : 172.16.100.4	10101100 00010000 01100100 00000100

Figure 3 : direct broadcast network calculation

Note, in addition to the network broadcast address, the IPv4 standard also defines the IP address: 255.255.255.255 as a local broadcast address, a broadcast address independent of the network, a broadcast address that is processed by all hosts on a local network. We have already seen this broadcast addresses in action in lab 3, when we looked at the DHCP protocol.

Connect the Pi system to the lab's network using the **GREEN** network cable connected to the back network sockets. Open a VNC session on Pi-1, then within the VNC session start Wireshark, click on the menu icon and select :

```
Applications -> Internet -> Wireshark
```

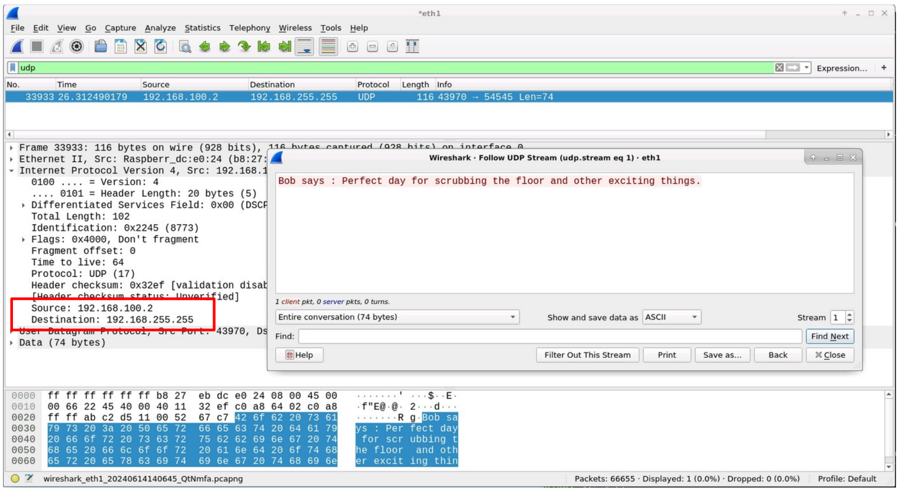
This will open the Wireshark GUI, selecting Ethernet 1 (`eth1`) in the Interface list. To see broadcast packets we will need to remove the network "noise" i.e. unwanted packets, to do this enter the packet protocol filter below :

```
udp
```

and click on the Apply button, then click on the Start icon on the top toolbar in Wireshark. You should now be able to see broadcast packets from a number of different servers :

- NTP - Src: `192.168.X.Y/16`, Dest: `192.168.255.255`
- Bob says - Src: `192.168.100.2/16`, Dest: `192.168.255.255`
- Sensor node - Src: `192.168.100.6/16`, Dest: `192.168.255.255`

An example packet from the Bob-says server is shown in figure 4.



*eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
+

```
udp
```

```
No.   Time             Source          Destination       Protocol Length Info
33933 26.312490179     192.168.100.2   192.168.255.255   UDP      116 43970 → 54545 Len=74
```

```
› Frame 33933: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface 0
› Ethernet II, Src: Raspberr_dc:e0:24 (b8:27:
⌄ Internet Protocol Version 4, Src: 192.168.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  › Differentiated Services Field: 0x00 (DSCP
    Total Length: 102
    Identification: 0x2245 (8773)
  › Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x32ef [validation disab
    [Header checksum status: Unverified]
    Source: 192.168.100.2
    Destination: 192.168.255.255
› User Datagram Protocol, Src Port: 43970, Ds
› Data (74 bytes)
```

Wireshark · Follow UDP Stream (udp.stream eq 1) · eth1

```
Bob says : Perfect day for scrubbing the floor and other exciting things.
```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (74 bytes) Show and save data as ASCII Stream 1

Find:

```
0000  ff ff ff ff ff ff b8 27  eb dc e0 24 08 00 45 00   ·······' ···$··E·
0010  00 66 22 45 40 00 40 11  32 ef c0 a8 64 02 c0 a8   ·f"E@·@· 2···d···
0020  ff ff ab c2 d5 11 00 52  67 c7 42 6f 62 20 73 61   ·······R g·Bob sa
0030  79 73 20 3a 20 50 65 72  66 65 63 74 20 64 61 79   ys : Per fect day
0040  20 66 6f 72 20 73 63 72  75 62 62 69 6e 67 20 74    for scr ubbing t
0050  68 65 20 66 6c 6f 6f 72  20 61 6e 64 20 6f 74 68   he floor  and oth
0060  65 72 20 65 78 63 69 74  69 6e 67 20 74 68 69 6e   er excit ing thin
```

wireshark_eth1_20240614140645_QtNmfa.pcapng Packets: 66655 · Displayed: 1 (0.0%) · Dropped: 0 (0.0%) Profile: Default

Figure 4 : Bob-says server

Task : capture a "Bob says" packet, what words of wisdom does Bob have for you today :). Can you also find a server on the lab's network that is broadcasting weather related facts e.g. broadcasting temperature, humidity and atmospheric pressure data?

Task 2

In its present configuration the Raspberry Pi system only has a single router i.e. the Mikrotik. However, each Raspberry Pi can also be configured as a router using the `Quagga` software suite, (Appendix A for "what is a Quagga"). For more information on these software packages refer to:

https://www.quagga.net/docs.html

Using this software we can now reconfigure our Raspberry Pi system as shown in figure 5. Each Raspberry Pi is now a router that can be configured to use the RIP protocol, to broadcast its routing table to other routers.

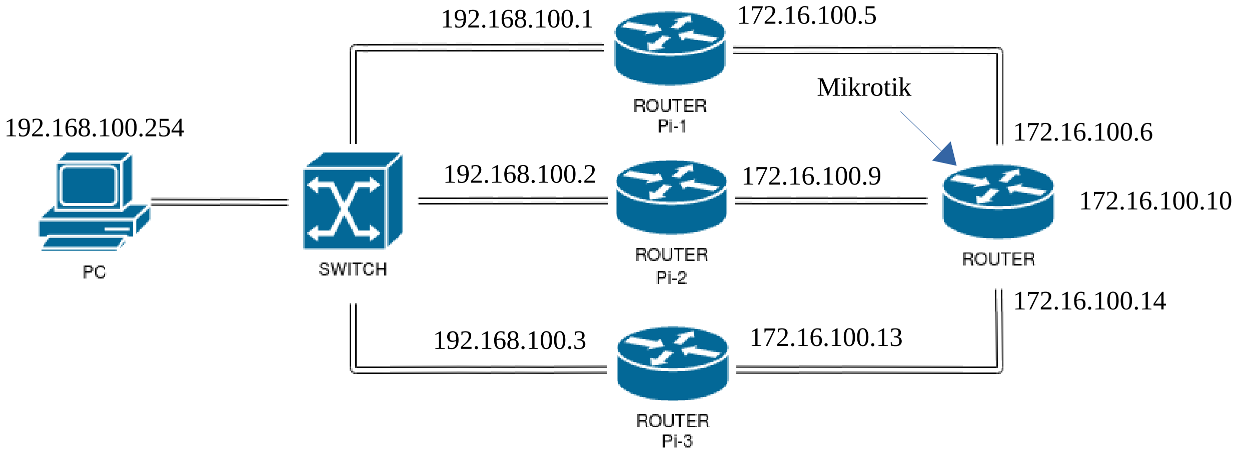


Figure 5 : new “all router” network

To enable the RIP daemon (service) on each Pi we will need to create the following files:

```
/etc/quagga/ripd.conf
/etc/quagga/zebra.conf
```

These files can be accessed from the VLE, or from Appendix B. Download the file : `configs.zip`, unzip this to a directory on the PC. Within this zip there are two folders :

```
version-1
version-2
```

Using Filezilla transfer the `.conf` files from the `version-1` folder to Pi-1, Pi-2 and Pi-3. On the PC click on the start button and select the command prompt.

⊞ `-> Command Prompt`

then SSH into Pi-1 by entering the command below:

```
ssh pi@192.168.X.1         (IP address will vary, X=Box/Desk)
```

In this terminal enter the commands below to move these files into the correct directory:

```
sudo mv ./ripd.conf /etc/quagga/ripd.conf
sudo mv ./zebra.conf /etc/quagga/zebra.conf
```

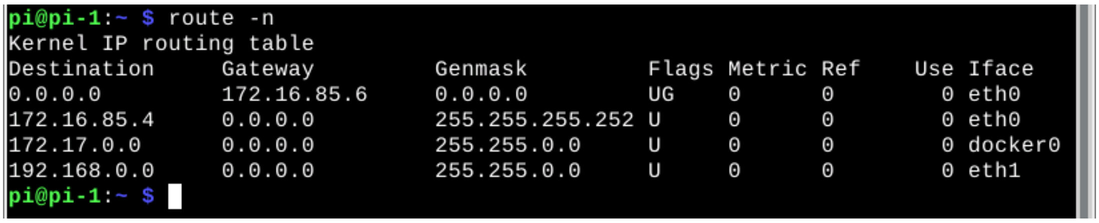
You must use `sudo` owing to directory permissions. Then repeat this process for each Pi i.e. upload and then move the files `ripd.conf` and `zebra.conf` on Pi-2 and Pi-3. For more information on these configuration files refer to:

https://www.nongnu.org/quagga/docs/quagga.html#RIP

Task : before we start the RIP daemons (servers) on each Pi, examine their routing tables. In each SSH terminal i.e. for Pi-1, Pi-2 and Pi-3, enter the following command:

```
route -n
```

An example routing table for Pi-1 is shown in figure 6.



```
pi@pi-1:~ $ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.85.6     0.0.0.0         UG    0      0        0 eth0
172.16.85.4     0.0.0.0         255.255.255.252 U     0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth1
pi@pi-1:~ $
```

Figure 6 : example routing table

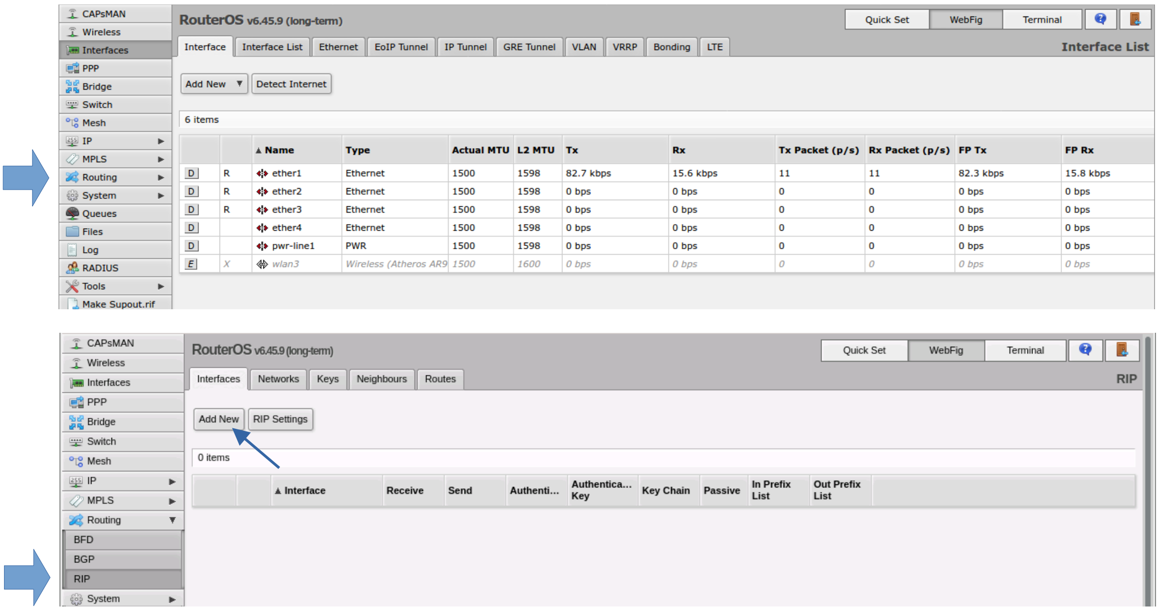
Next, we need to configure the Microtik router. In the Pi-1 VNC remote desktop, open a web browser and connect to the routers web interface via its IP address :

```
http://172.16.X.6                    (X=Box/Desk)
```

This will open the MicroTiks web interface, as shown in figure 7. In the web interface, click on the `WebConfig` tab. Then to enable the RIP routing protocol click on the left hand panel, selecting:

```
Routing -> RIP
```

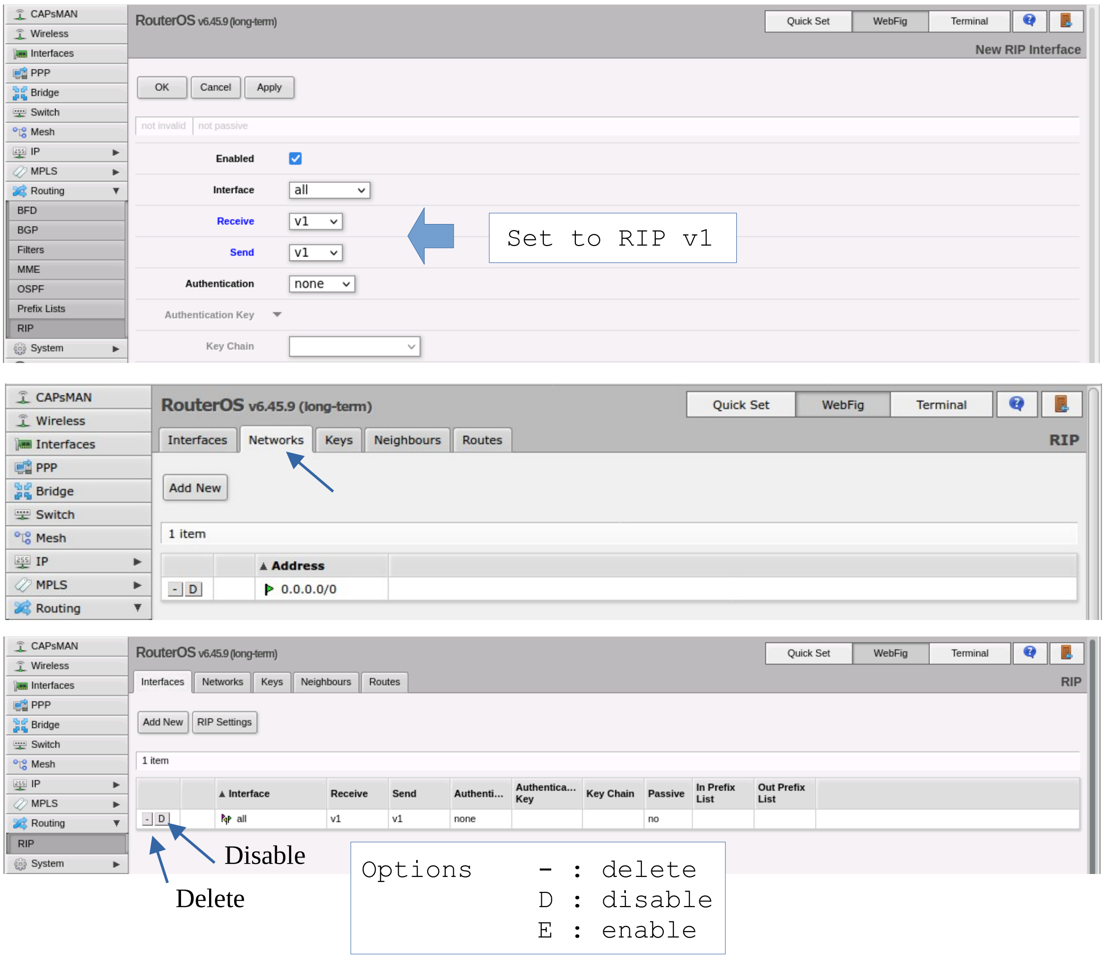
If RIP has already been enabled by a previous person delete this configuration by clicking on the delete - icon, to remove. Then, click on the Add New icon and enter the routing rule shown in figure 8 and click OK. Finally, click on the Network tab and click on the Add New button to update the advertised networks to `0.0.0.0/0` i.e. all interfaces.



		Name	Type	Actual MTU	L2 MTU	Tx	Rx	Tx Packet (p/s)	Rx Packet (p/s)	FP Tx	FP Rx
D	R	ether1	Ethernet	1500	1598	82.7 kbps	15.6 kbps	11	11	82.3 kbps	15.8 kbps
D	R	ether2	Ethernet	1500	1598	0 bps	0 bps	0	0	0 bps	0 bps
D	R	ether3	Ethernet	1500	1598	0 bps	0 bps	0	0	0 bps	0 bps
D		ether4	Ethernet	1500	1598	0 bps	0 bps	0	0	0 bps	0 bps
D		pwr-line1	PWR	1500	1598	0 bps	0 bps	0	0	0 bps	0 bps
E	X	wlan3	Wireless (Atheros AR9	1500	1600	0 bps	0 bps	0	0	0 bps	0 bps

		Interface	Receive	Send	Authenti...	Authentica... Key	Key Chain	Passive	In Prefix List	Out Prefix List

Figure 7 : Mikrotik RIP menu



New RIP Interface

not invalid | not passive

- Enabled: ☑
- Interface: all
- Receive: v1
- Send: v1
- Authentication: none
- Authentication Key:
- Key Chain:

Set to RIP v1

RIP – Networks

1 item

	Address
- D	0.0.0.0/0

RIP – Interfaces

1 item

	Interface	Receive	Send	Authenti...	Authentica... Key	Key Chain	Passive	In Prefix List	Out Prefix List
- D	all	v1	v1	none			no		

Disable

Delete

```
Options    - : delete
           D : disable
           E : enable
```

Figure 8 : Mikrotik RIP configuration

To reduce the “noise” produced by the labs network disconnect the **GREEN** network cable from the switch. Restart Wireshark, select the network interface (NIC): Ethernet 0 (`eth0`), in the Interface list i.e. we need to capture packets on the `172.16.X.4` network. Enter the packet protocol filter :

```
rip
```

and click on the Apply button (this will remove most of the network “noise”), then click on the Start icon on the top toolbar in Wireshark.

To start the RIP daemon on each Pi use its SSH terminal i.e. for Pi-1, Pi-2 and Pi-3, and enter the following commands:

```
sudo systemctl restart zebra.service
sudo systemctl restart ripd.service
```

By default RIP routing information is transferred every 30 seconds, but this can vary with different configurations. You will now see some activity in the main window as Wireshark captures the packet associated with routing information transfers. Finally, click on the Stop icon 🟥 on the top toolbar to stop capturing network traffic.

Task : what broadcast address is used by the RIPv1 protocol? How are the routing tables on each Pi updated? Why have these new routes been added?

Hint, remember TCP is a point-to-point protocol so can not use broadcast addresses. What networks are connected to the MikroTik router? What networks does it know about?

Broadcast addresses are limited to a broadcast domain i.e. a subnet, these broadcasts are not forwarded by the router to other subnets (networks). As discussed in the lecture sub-netting is a very important topic in networking. It is a mechanism for dividing up a network, ensuring that only those hosts that need to be visible to each other are, thus improving security and network performance.

In the early days of the Internet subnets were defined by their network class, as shown in figure 9. The `eth1` interface on each Pi uses a class C network address, as defined by its 192.168.X.Y IP address. However, with the adoption of CIDR notation we can update its netmask to make it a 192.168.0.0/16 network i.e. a “class B” network. This approach is also seen on the `eth0` interface, as this interface uses the 172.16.X.0/30 network, a classless network i.e. a CIDR network that does not match one of the predefined classes.

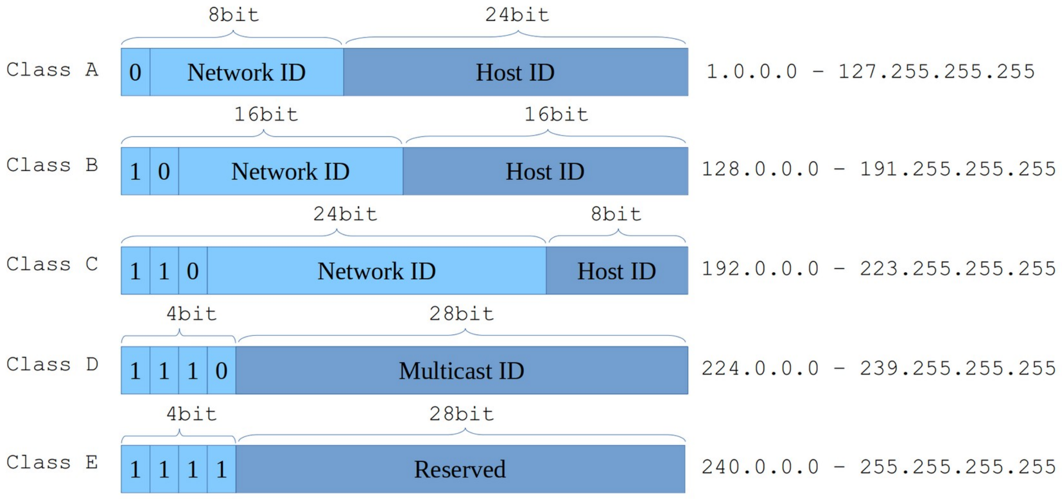


Figure 9 : network classes

Task : if you have not already done so make sure you can answer the lecture’s quick quizzz question: given this subnet mask which of these IP addresses are on the same subnets i.e. have the same network address?

```
CIDR notation:  144.32.50.0/26

Network subnet: 255.255.255.192  11111111.11111111.11111111.11000000
mask

Network address A: 144.32.50.110
Network address B: 144.32.50.120
Network address C: 144.32.50.130
Network address D: 144.32.50.150
Network address E: 144.32.50.190
Network address F: 144.32.50.200
```

Task 3

RIPv1 was introduced when the Internet was relatively “small”, therefore, the amount of data transferred by each broadcast was not significant. However, as networks got bigger this regular 30 second burst of traffic started to become an issue i.e. routers became synchronised, resulting in 100s of routers all transmitting data at the same time.

Note, remember that broadcast packets are received and processed by all hosts/routers on that local network, even if they are not involved in routing packets.

RIPv1 also had a number of limitations e.g. fixed metrics, limited to 15 hops and only supported classful routing i.e. networks defined in figure 9. However, perhaps the most obvious missing functionality is the lack of authentication / passwords, which is a slight security concern :).

To overcome these issues RIPv2 was developed in 1994, again using UDP as its transport protocol, port number 520 (RFC 1723: https://datatracker.ietf.org/doc/html/rfc1723). This protocol supports Classless Inter-Domain Routing (CIDR) and also passwords :). It also switched from broadcast based addressing to multicast addressing i.e. IP address 224.0.0.X, reducing network loads for hosts not involved in network routing.

Multicast addressing uses a Class D address i.e. a multicast ID. Now, only routers monitoring this address will process these packets i.e. you can think of multicast as an application specific “broadcast” address. For more information on what applications use what multicast IDs refer to:

https://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml

To see RIPv2 in action stop the RIP daemon on each Pi using its SSH terminal i.e. for Pi-1, Pi-2 and Pi-3, and enter the following commands:

```
sudo systemctl stop zebra.service
sudo systemctl stop ripd.service
```

Using Filezilla transfer the `.conf` files from the `version-2` folder to Pi-1, Pi-2 and Pi-3. Then enter the commands below to move these files into the correct directory:

```
sudo mv ./ripd.conf /etc/quagga/ripd.conf
sudo mv ./zebra.conf /etc/quagga/zebra.conf
```

Repeat this process for each Pi i.e. upload and then move the files `ripd.conf` and `zebra.conf` on Pi-2 and Pi-3.

In Task 2 each Pi was on its own subnet, therefore, they were unable to see the broadcast packets from other Raspbery Pis. They could only see the broadcast packets from the router i.e. each Raspberry Pi’s routing table was only updated with information from the router’s routing table. Moving the RIP protocol onto `eth1` means that each Raspberry Pi is now in the same broadcast domain, therefore, each Pi will now “see” other Pi routing tables, not just the router’s routing table.

To enable the router to process these new RIPv2 packets update the configuration show in figure 8, changing the receive/send versions from `v1` to `v2`. Restart the Wireshark GUI. Left click on (highlight) the Ethernet 1 (`eth1`) icon eth1 in the Interface list. Next, enter the packet protocol filter :

```
rip
```

and click on the Apply button (this will remove most of the network “noise”). Then click on the Start icon on the top toolbar in Wireshark. Restart the RIP daemon on each Pi using each SSH terminal i.e. for Pi-1, Pi-2 and Pi-3, by entering the following commands:

```
sudo systemctl restart zebra.service
sudo systemctl restart ripd.service
```

You will now see some activity in the main window as Wireshark captures the routing information packets. Finally, click on the Stop icon ■ on the top toolbar to stop capturing network traffic.

Task : can you identify RIPv2 routing packets? What multicast address do they use?

Note, RIP is an example of an early routing protocol and therefore has a number of limitations. For our small system these overheads are not that significant e.g. periodically transferring the full routing table to each router. However, as the size of these network’s grow this becomes impractical. To solve this and other problems the Open Shortest Path First (OSPF) protocol was developed. This is a dynamic routing protocol. Routing calculations are now not just based on the number of hops, but also cost parameters such as bandwidth, delay and load. To perform these calculations this protocol uses link-state routing based on Dijkstra's, or the shortest path first (SPF) algorithm. Each router maintains a link-state database, a network topology map, allowing every router to calculate the “cost” of each route. As with any protocol OSPF has its disadvantages (high initial network load as link-states are transferred, increased router processing and memory requirements) and advantages (once a stable link-state is reached there are minimal network overheads). However, in general it has now replaced RIP.

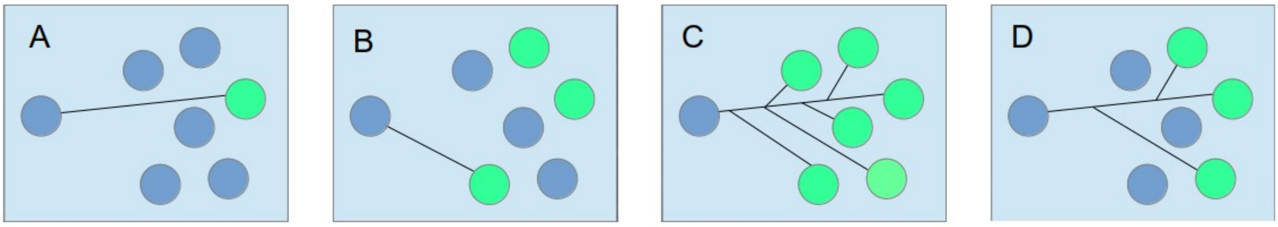


Figure 10 : communication types: unicast (A), anycast (B), broadcast (C) and multicast (D)

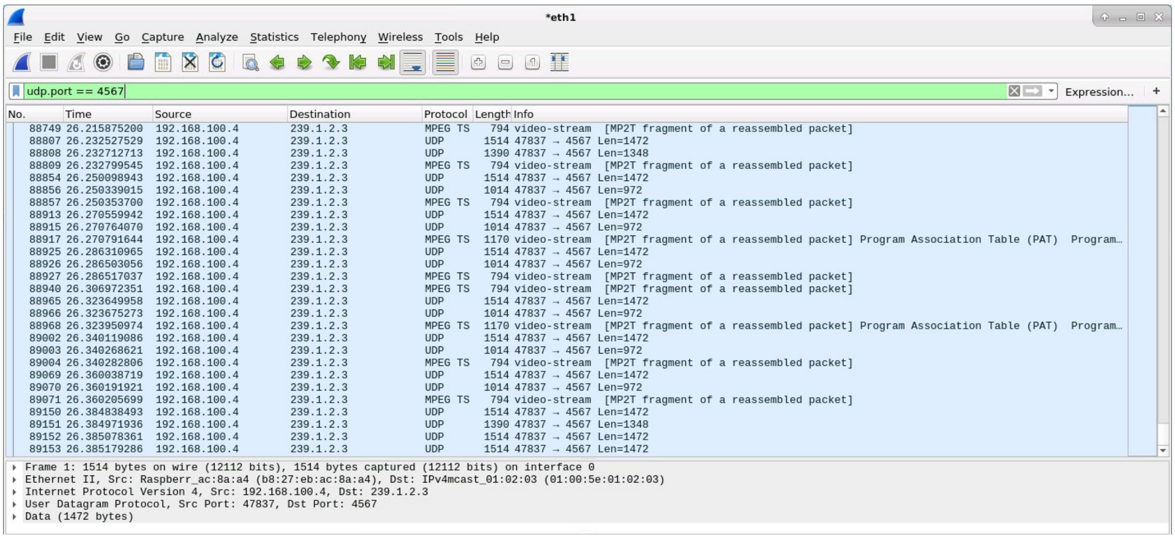
Task 4

RIP is a good example of an application layer protocol that needs to communicate the same data to multiple destinations. To achieve this we could implement multiple-unicast communication links, however, this will result in an increased processing load for the transmitting host. We could communicate this data using a broadcast, however, this will result in an increased processing load for all receiving hosts i.e. every host has to process a broadcast packet, even if the data is not for it. Therefore, the preferred solution used by RIPv2 and OSPF is multicast i.e. the transmitting host transmits the data once and only those hosts “signed up” to receive this data will process it, as shown in figure 10. Another application that makes use of multicast communications is video streaming i.e. one transmitter, multiple receivers. To see this in action the FTP server is also configured to stream video data on multicast address 239.1.2.3, port 4567.

Reconnect your system to the lab’s network using the **GREEN** network cable. On a Pi of your choice open the Wireshark GUI, selecting the network interface Ethernet 1 (`eth1`) in the Interface list. Next enter the packet protocol filter :

```
udp.port == 4567
```

and click on the Apply button (this will remove most of the network “noise”), then click on the Start icon on the top toolbar in Wireshark. You should now be able to capture multicast packets from the FTP server i.e. source IP address 192.168.100.4, destination IP address 239.1.2.3. As shown in figure 11.



udp.port == 4567

```
No.   Time         Source          Destination  Protocol Length Info
88749 26.215875200 192.168.100.4   239.1.2.3    MPEG TS   794 video-stream  [MP2T fragment of a reassembled packet]
88807 26.232527529 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
88808 26.232712713 192.168.100.4   239.1.2.3    UDP      1390 47837 → 4567 Len=1348
88809 26.232799545 192.168.100.4   239.1.2.3    MPEG TS   794 video-stream  [MP2T fragment of a reassembled packet]
88854 26.250098943 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
88856 26.250339015 192.168.100.4   239.1.2.3    UDP      1014 47837 → 4567 Len=972
88857 26.250353700 192.168.100.4   239.1.2.3    MPEG TS   794 video-stream  [MP2T fragment of a reassembled packet]
88913 26.270559942 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
88915 26.270764070 192.168.100.4   239.1.2.3    UDP      1014 47837 → 4567 Len=972
88917 26.270791644 192.168.100.4   239.1.2.3    MPEG TS  1170 video-stream  [MP2T fragment of a reassembled packet] Program Association Table (PAT)  Program…
88925 26.286310965 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
88926 26.286503056 192.168.100.4   239.1.2.3    UDP      1014 47837 → 4567 Len=972
88927 26.286517037 192.168.100.4   239.1.2.3    MPEG TS   794 video-stream  [MP2T fragment of a reassembled packet]
88940 26.306972351 192.168.100.4   239.1.2.3    MPEG TS   794 video-stream  [MP2T fragment of a reassembled packet]
88965 26.323649958 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
88966 26.323675273 192.168.100.4   239.1.2.3    UDP      1014 47837 → 4567 Len=972
88968 26.323950974 192.168.100.4   239.1.2.3    MPEG TS  1170 video-stream  [MP2T fragment of a reassembled packet] Program Association Table (PAT)  Program…
89002 26.340119086 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
89003 26.340268621 192.168.100.4   239.1.2.3    UDP      1014 47837 → 4567 Len=972
89004 26.340282806 192.168.100.4   239.1.2.3    MPEG TS   794 video-stream  [MP2T fragment of a reassembled packet]
89069 26.360038719 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
89070 26.360191921 192.168.100.4   239.1.2.3    UDP      1014 47837 → 4567 Len=972
89071 26.360205699 192.168.100.4   239.1.2.3    MPEG TS   794 video-stream  [MP2T fragment of a reassembled packet]
89150 26.384838493 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
89151 26.384971936 192.168.100.4   239.1.2.3    UDP      1390 47837 → 4567 Len=1348
89152 26.385078361 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
89153 26.385179286 192.168.100.4   239.1.2.3    UDP      1514 47837 → 4567 Len=1472
```

```
▸ Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▸ Ethernet II, Src: Raspberr_ac:8a:a4 (b8:27:eb:ac:8a:a4), Dst: IPv4mcast_01:02:03 (01:00:5e:01:02:03)
▸ Internet Protocol Version 4, Src: 192.168.100.4, Dst: 239.1.2.3
▸ User Datagram Protocol, Src Port: 47837, Dst Port: 4567
▸ Data (1472 bytes)
```

Figure 11 : multicast packets

In theory all we should have to do to view this video is open a terminal and enter the command:

```
ffplay udp://239.1.2.3:4567
```

Task : on your selected Pi enter this command. However, you should observe that even though you have started `ffplay` no video is playing :(. Can you identify why this is so and make the required changes to the network to allow you to play the video shown in figure 12?

Hint, what is the default network interface `ffplay` will be listening on, do you need to add a route to the routing table?

The issue here is that when you start `ffplay` it has to decide which network interface to use. As the IP address 239.1.2.3 is not listed in its routing table it will default to the one attached to the GW i.e. NIC `eth0`. However, as we can see from figure 11 the multicast packets are arriving on NIC `eth1`, therefore, we need to add a new route to the routing table :

```
sudo ip route add 239.1.2.3/32 dev eth1
```

Restart `ffplayer` so that this new route is used by the application and after a couple of seconds you should see some jellyfish :).

Figure 12 : video stream

Task 5

Routing protocols are the preferred solution when setting up multiple hosts. However, there are situations were you need to manually setup routing rules as the hosts or routers on the network may not support these routing protocols. Consider the lab “network” shown in figure 13. This diagram is a representation of the hosts connected to the lab’s internal network. The network connected to the `eth1` NIC is a “class B” network a `/16` network :

```
192.168.0.0/16     :    subnet 0
```

When you connect the network switch on the Raspberry Pi base system to the lab’s network via the **GREEN** cable you will be able to connect to any of the desks in the lab and the lab’s servers :

192.168.100.1/16	:	NTP time server
192.168.100.2/16	:	SMTP mail server
192.168.100.3/16	:	DNS name server
192.168.100.4/16	:	FTP file server
192.168.100.5/16	:	Ping server
192.168.100.6/16	:	Sensor node server

All of these hosts are on the same subnet i.e. they are all on the same broadcast domain. This means that any broadcasts will be received and processed by ALL hosts.

IMPORTANT, remember for two host to communicate across a local network they must be on the same subnet, they must be in the same broadcast domain.

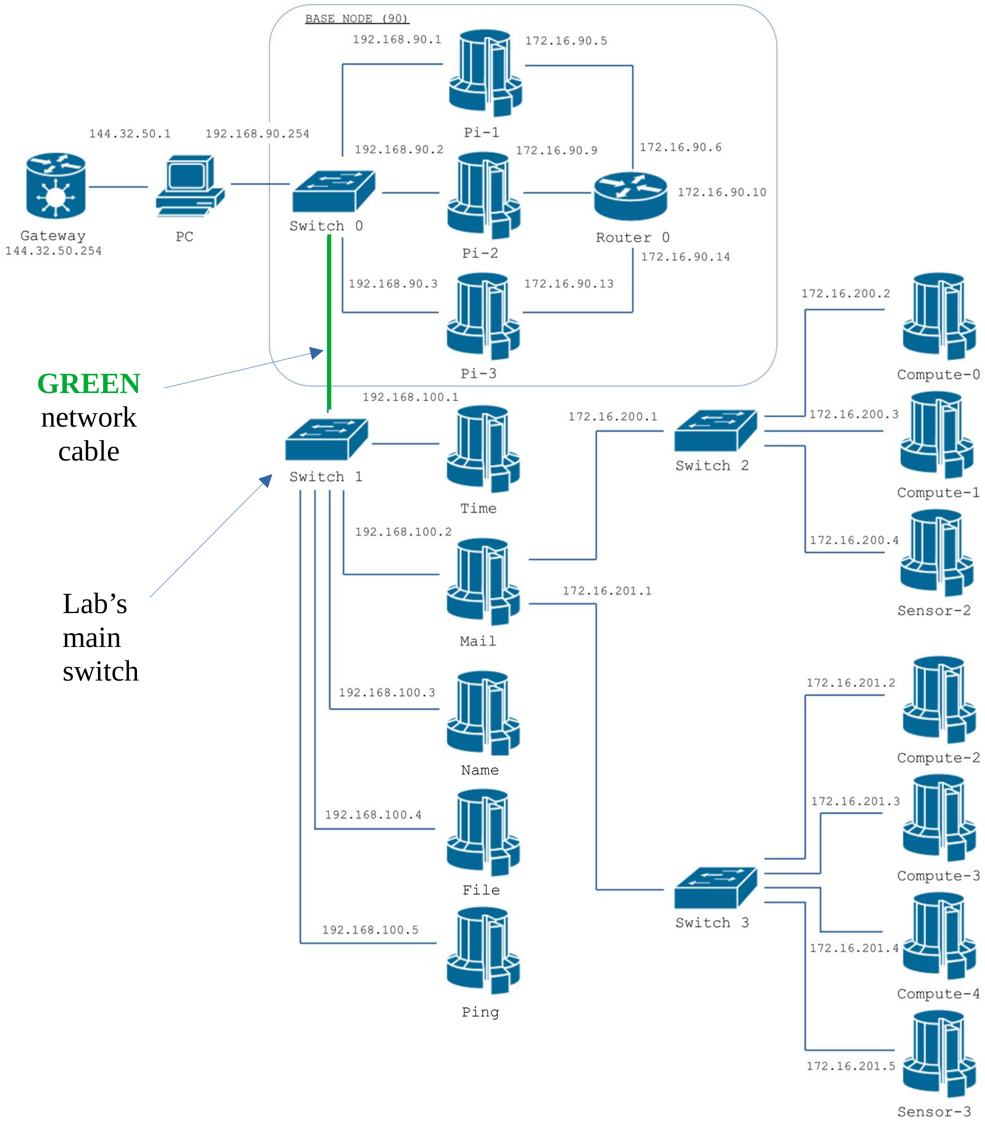
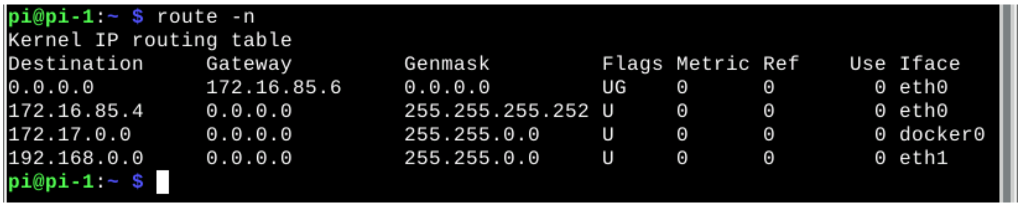


Figure 13 : lab networks and hosts



```
pi@pi-1:~ $ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.85.6     0.0.0.0         UG    0      0        0 eth0
172.16.85.4     0.0.0.0         255.255.255.252 U     0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth1
pi@pi-1:~ $ █
```

Figure 14 : example routing table

If you look closely at figure 13, we can see that the mail-server is also connected to two class C networks:

```
172.16.200.0/24     :    subnet 1
172.16.201.0/24     :    subnet 2
```

Task : consider Pi-1's routing table shown in figure 14. What will happen if Pi-1 was to ping a host on the 172.16.200.0/24 network? Will this packet be routed correctly?

Using this routing table packets will be sent to the default gateway (GW) i.e. the microTik router. Unfortunately this router does not know about these other networks, or hosts, so the packets will be dropped.

To manually configure our network we will first need to stop the RIP daemon on each Pi and the MikroTik. Using the MikroTik's web interface delete the RIP service on the router, as shown in figure 8. Next stop the RIP daemon running on each Pi i.e. SSH into Pi-1, Pi-2 and Pi-3 and enter the following commands:

```
sudo systemctl stop ripd.service
sudo systemctl stop zebra.service
```

Alternatively, you can reboot the system by pressing the big **RED** button i.e. perform a power cycle. This will automatically remove the previous Quagga configuration files. To manually configure our network we will need to consider the route from Pi-1 to Compute-0, as shown in figure 15.

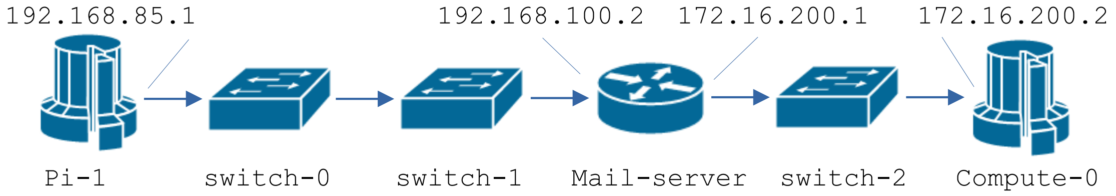
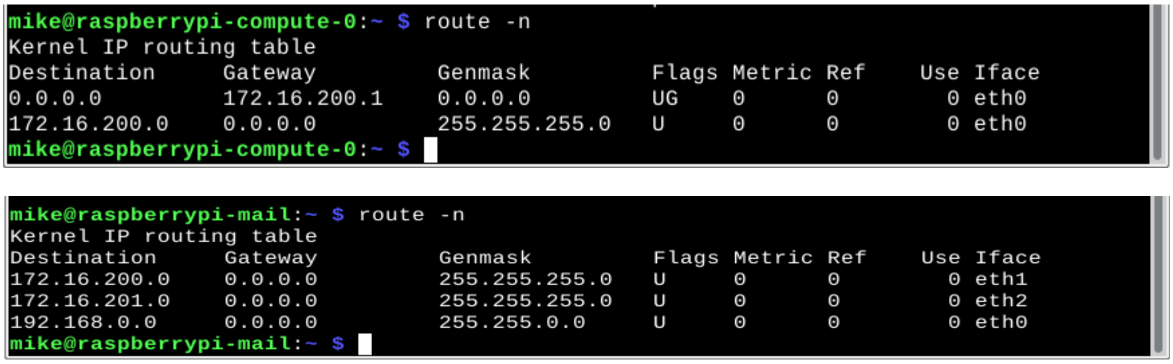


Figure 15 : path from Pi-1 to Compute-0



```
mike@raspberrypi-compute-0:~ $ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.200.1    0.0.0.0         UG    0      0        0 eth0
172.16.200.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
mike@raspberrypi-compute-0:~ $
```

```
mike@raspberrypi-mail:~ $ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.200.0    0.0.0.0         255.255.255.0   U     0      0        0 eth1
172.16.201.0    0.0.0.0         255.255.255.0   U     0      0        0 eth2
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
mike@raspberrypi-mail:~ $
```

Figure 16 : Compute-0 (top), Mail-server (bottom) routing tables

Task : examine figures 14, 15 and 16. Can you identify the different networks i.e. the different subnets, their network IP addresses and associated net masks. Why don't we need to configure switch-0, switch-1 and switch-2?

Hint, what addresses do switches speak? Do switches work on layer-2 or layer-3?

For the moment all we need to consider is how do we stop Pi-1 sending these packets to the MikroTik router. To do this on we can define the mail-server Pi as the gateway to this new network by entering the command below on Pi-1 :

```
sudo ip route add 172.16.200.0/24 via 192.168.100.2
```

Note, if needed you can delete incorrect route using the `del` option e.g. to remove the previous routing rule we would enter the command:

```
sudo ip route del 172.16.200.0/24 via 192.168.100.2
```

Task : test that you can now ping Compute-0 from Pi-1? Why don't we need to configure the routing rules on Compute-0 to correctly route packets from Compute-0 to Pi-1?

Hint, examine the network diagram and IP addresses used in figure 15 and the routing tables shown in figure 16. Can you see how these routing rules will work on this network?

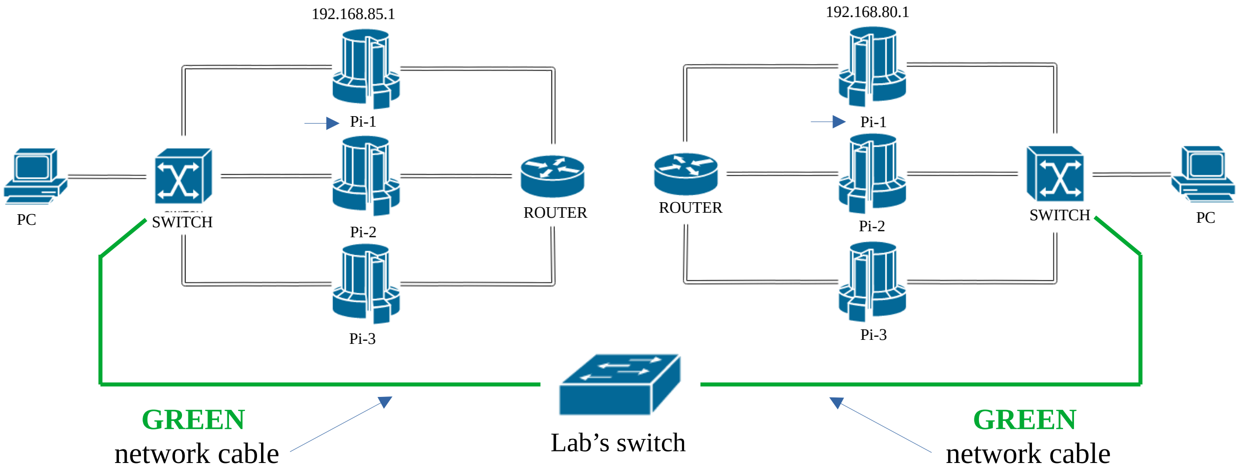


Figure 17 : connecting systems via the lab's switch

Task 6

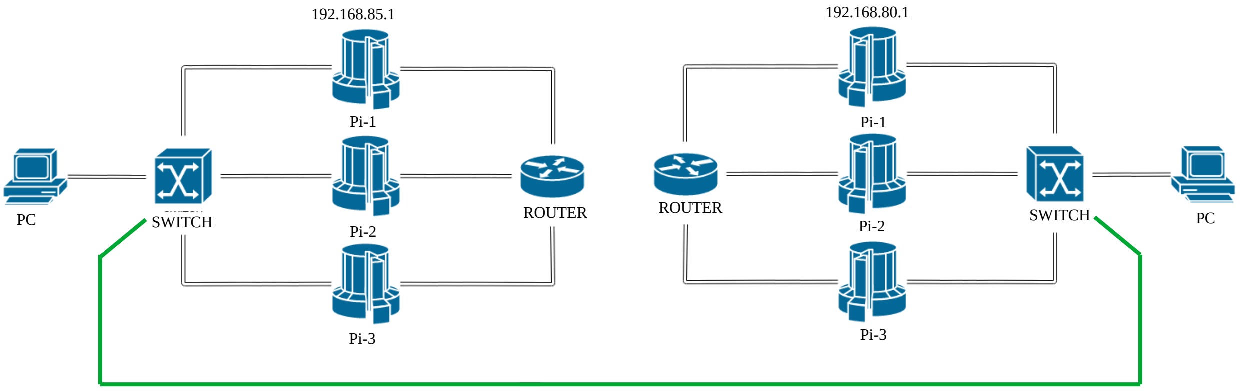
In addition to connecting to core network resources i.e. the labs NTP, SMTP, FTP & DNS servers, we can also connect base systems together e.g. desks 85 and 80, as shown in figure 17. In this example the IP addresses for Pi-1 on each of these systems will be:

- Desk 85 : 192.168.85.1/16
- Desk 80 : 192.168.80.1/16

Task : are these Raspberry Pis on the same subnet? If you connect these base systems to the lab's network switch as shown in figure 17, will Pi-1 be able to ping the other Pi-1? Will these Raspberry Pis be on the same broadcast domain?

Alternatively, rather than connecting these systems via the lab's network we could connect these systems directly, as shown in figure 18.

Task : are the networks shown in figures 17 and 18 functional equivalent? How do the networks shown in figures 17 and 18 differ in terms of broadcast and collision domains?



GREEN network cable direct connection

Figure 18 : direct connection via the base system switches

The final method of connecting the Raspberry Pi system to other hosts is via the Mikrotik router, as shown in figure 19. In this case we will need to setup appropriate routing rules within this router and each Pi to correctly route packets through this network path.

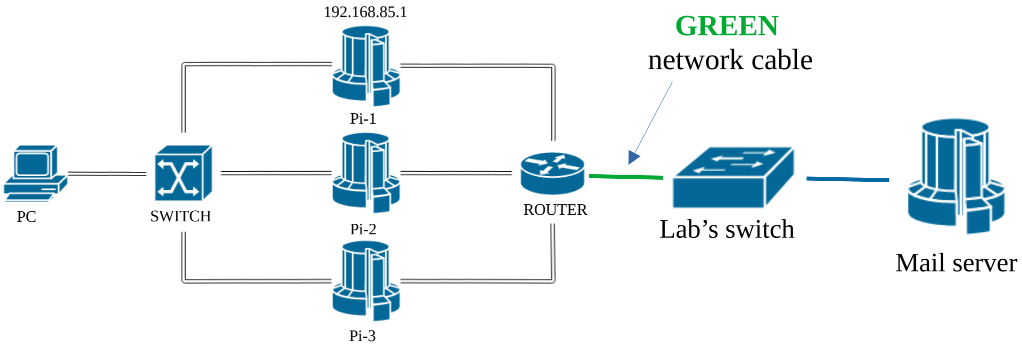


Figure 19 : connecting systems via the router

Task : can you setup the Mikrotik router and routing rules on the Pi, such that the Pi can route its mail packets through the router rather than the switch?

IMPORTANT : when you have finished, restore the Mikrotik configuration back to its original settings, such that it's good to go for the next person using that desk. If you are not sure how to do this do ask and I can show you how.

Appendix A : Quagga

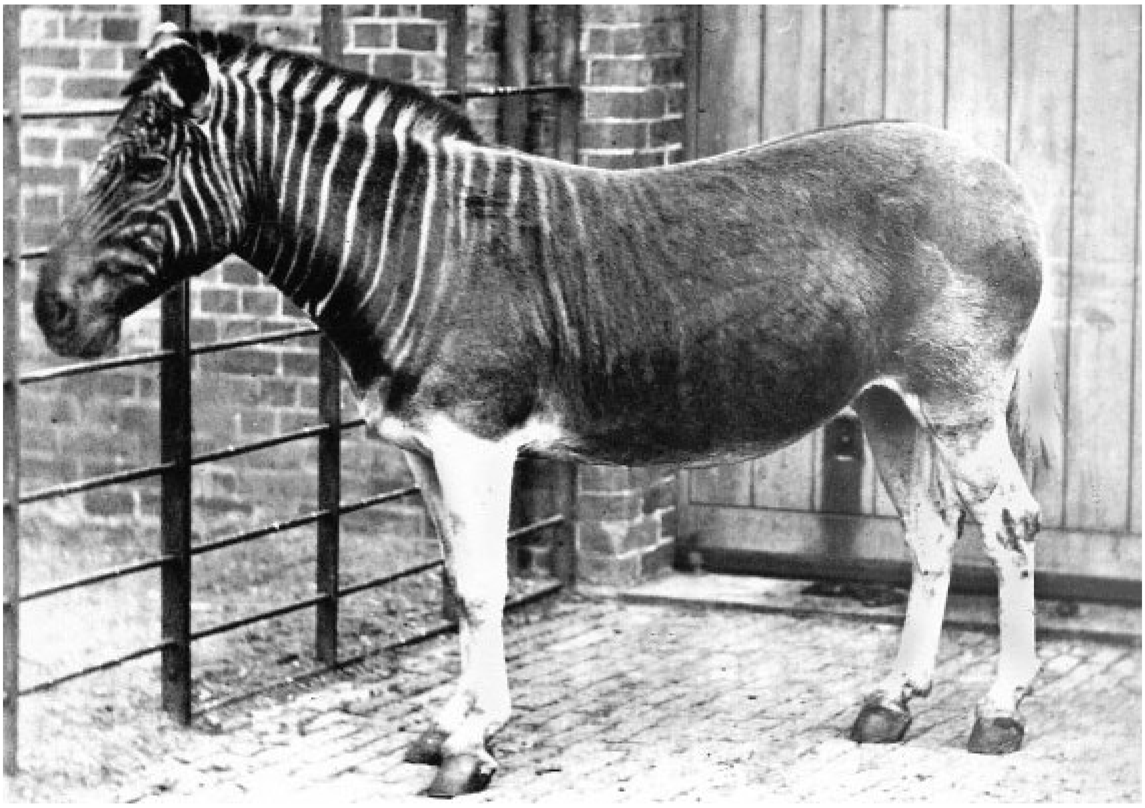


Figure A1 : Quagga (https://en.wikipedia.org/wiki/Quagga)

```
! -*- rip -*-
hostname ripd
!password zebra
!
router rip
      version 1
      network eth0
      redistribute static
      redistribute kernel
      redistribute connected
      timers basic 10 15 30
      no passive-interface eth0
!
log file ripd.log
```

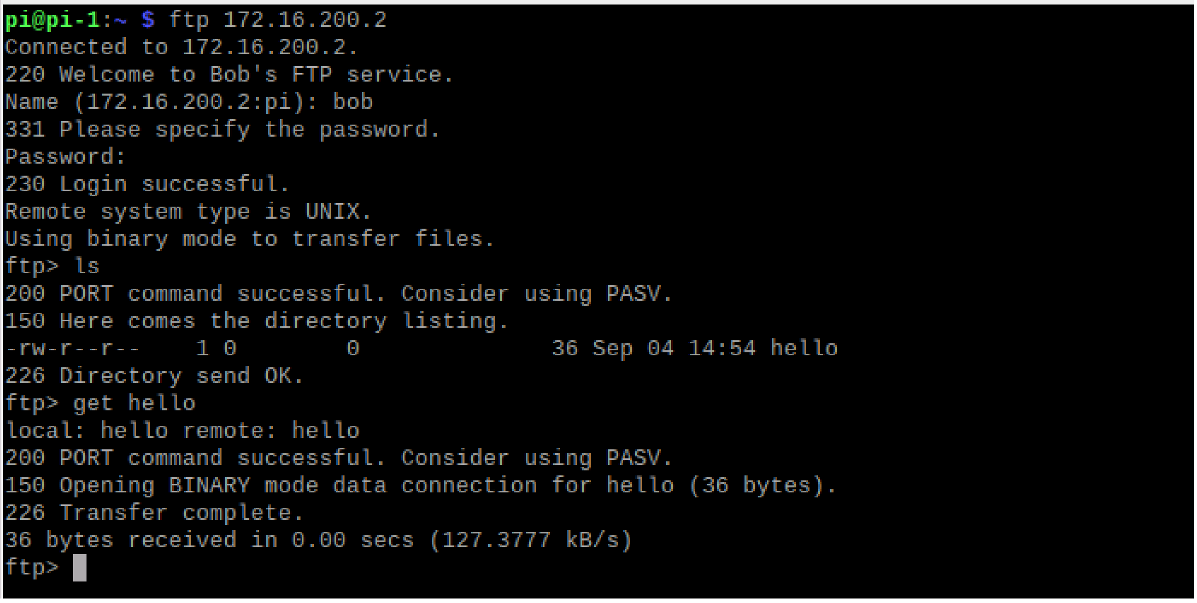
Figure 20 : ripd.conf version 1

```
! -*- zebra -*-
hostname Router
password zebra
enable password zebra
!
log file ripd.log
```

Figure 21 : zebra.conf version 1

Note, comments are indicated by a “!”. Indentations can be spaces or tabs

Appendix B : command line FTP client



```
pi@pi-1:~ $ ftp 172.16.200.2
Connected to 172.16.200.2.
220 Welcome to Bob's FTP service.
Name (172.16.200.2:pi): bob
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0              36 Sep 04 14:54 hello
226 Directory send OK.
ftp> get hello
local: hello remote: hello
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for hello (36 bytes).
226 Transfer complete.
36 bytes received in 0.00 secs (127.3777 kB/s)
ftp> 
```

Figure 22 : command line FTP client

To connect to the FTP servers running on Compute-0, Compute-1, Compute-2 and Compute-3 we can use a command line FTP client. To connect to the FTP server on Compute-0, open a terminal on Pi-1. Next, enter the command :

```
ftp 172.16.200.2
```

Login using the username and password from lab 4 :

```
username : bob
password : asdzxc
```

To see what files are available in the home directory enter the command :

```
ls
```

This will list the files in that directory. If needed to change directory use the `cd` command.

To download the text file “`hello`” enter the command :

```
get hello
```

This will download this file to the directory on Pi-1 in which you launched the FTP client from. This complete process is shown in figure 22. To close the connection to the FTP server enter the command:

```
bye
```

For more information on the FTP commands supported by the command line FTP client refer to the web page below, or the man page on Pi-1.

https://www.commandlinux.com/man-page/man1/ftp.1.html