# Practical Course Networking Lab

Institute of Computer Science Georg-August-Universität Göttingen

**Lecturer:** Dr. Mayutan Arumaithurai Teaching
**Assistant:** M.Sc. Sameer Kulkarni
**Author:** Hari Raghavendar Rao Bandari
11334055
h.bandari@stud.uni-goettingen.de

Torsten Zielke
21220789
torsten.zielke@stud.uni-goettingen.de

**Date:** September 8, 2015

## Content

# Lab 1

## Pre-Module Question

### What will happen if you type man man in Linux?

You get the manual page for “man” - so the manual for the manual.

### How can you use the command ls to find out about the size of the file?

ls -l file for giving the detailed information about the file. And with “h” as additional parameter you can get the human readable format.

### What happens if you have to files with names file1 and file2 and you type mv file1 fil2?

I suppose it should be “mv file1 file2” because else it would just rename file1 to fil2. But with mv file1 file2 we overwrite the file2.

### Which option of mv issues a warning in this situation?

-i is used for warning. Because it enables the interactive mode.

### What is the command that you issue if you are in directory / and want to copy the file /mydata to directoy /labdata?

cp mydata /labdata/

### What is the command that you issue if you are in directory / and want to copy all files and directories under directory /mydirectory to directory /newdirectory?

cp -r /mydirectory/* /newdirectory/

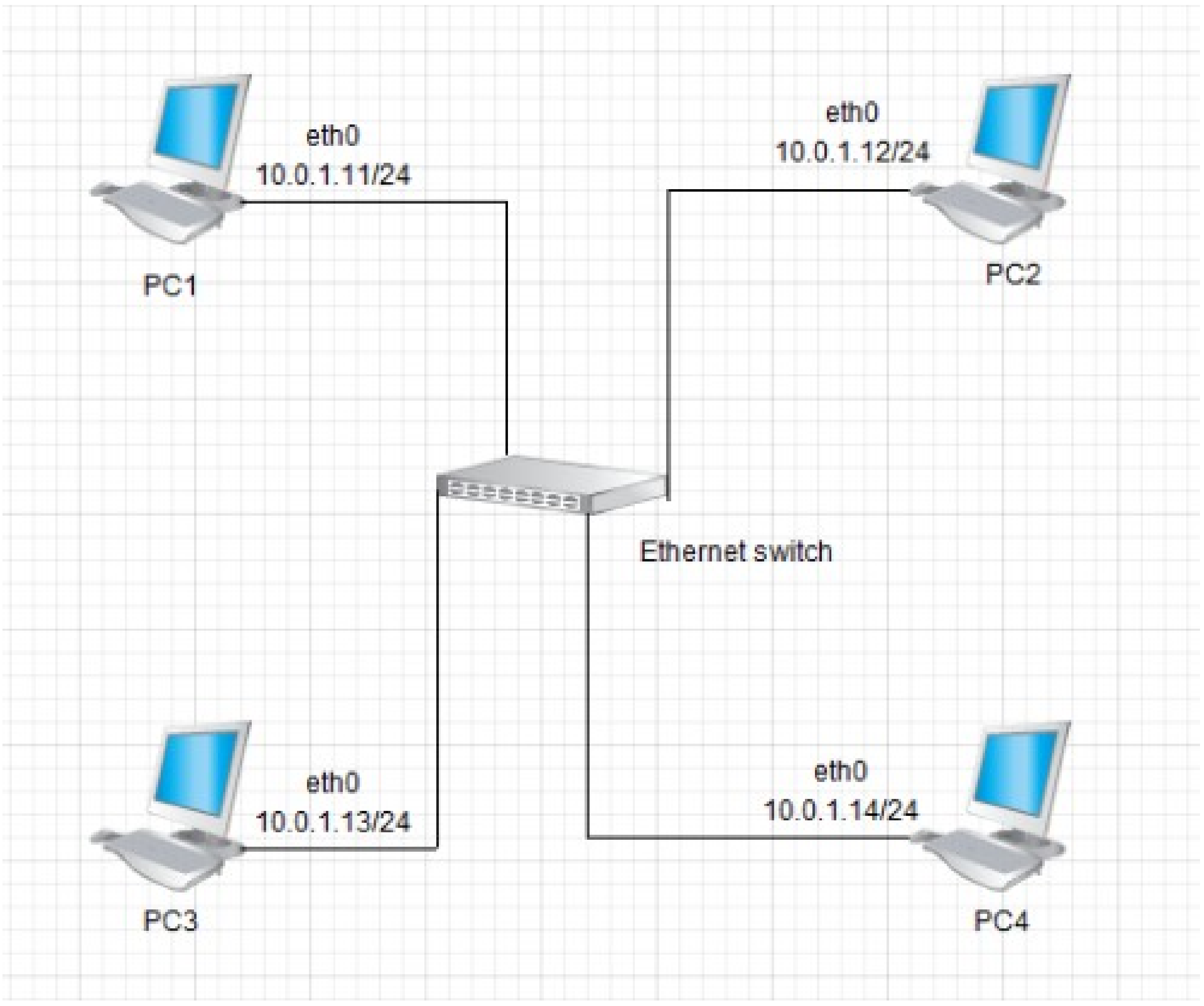
### What happens if you type the command rm * in a directory?

It removes all the files in particular directory, but not the folders.

### What is the command that you issue if you want to delete all files and directories under the directory /mydirectory?

rm -rf /mydirectory/* (in the past this included . and .., but every new system should avoid it now).

## Exercises 1 to 3



In these Exercises we didn't really need to save Output. But we set up the network like in the image above and checked if each connection is working via **telnet** and **ping**.
Additionally we edited and reviewed the function of **/etc/hosts**.

## Exercise 4: "df" output

```
Filesystem       1K-blocks  Used        Available       Use%   Mounted on
/cow         508120             2628    505492          1%     /
udev          494028            4       494024          1%     /dev
tmpfs         101628            1092    100536          2%     /run
/dev/sda1          3863668      927576 2936092          25%    /cdrom
/dev/loop0    894976            894976 0                100%   /rofs
none          4           0     4                 0%    /sys/fs/cgroup
tmpfs         508120            4       508116          1%     /tmp
none          5120              0       5120            0%     /run/lock
none          508120            0       508120          0%     /run/shm
none          102400            16      102384          1%     /run/user
/dev/sdb4          3923840      76      3923764         1%     /media/ubuntu/LIVE
```

*file: /lab_data/lab_01/df_output*

## Exercise 5

In this exercise we had to view /etc/hosts via the more command.

### Which files must be edited to change the name of the Linux PC?

```
/etc/hosts
/etc/hostname
```

### Which files include information that determines whether a LinuxPC performs IP forwarding?

```
/proc/sys/net/ipv4/ip_forward
```

We can access it via cat for viewing and via

```
echo 1/0 > /proc/sys/net/ipv4/ip_forward
```

for editing.

### Attach the standard Ubuntu network interface configuration file:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.0.1.11/24
```

file: /lab_data/lab_01/PC1_interfaces

## Exercise 6

Here we start again with the ping command. This time to see the connection between PC1 and PC2. Additionally we pinged the **loopback interface** (127.0.0.1) which is also called **localhost**.

**Include the output you saved in this exercise:**

```
PING 10.0.1.12 (10.0.1.12) 56(84) bytes of data.
64 bytes from 10.0.1.12: icmp_seq=1 ttl=64 time=0.117 ms
64 bytes from 10.0.1.12: icmp_seq=2 ttl=64 time=0.149 ms
64 bytes from 10.0.1.12: icmp_seq=3 ttl=64 time=0.102 ms
64 bytes from 10.0.1.12: icmp_seq=4 ttl=64 time=0.100 ms
64 bytes from 10.0.1.12: icmp_seq=5 ttl=64 time=0.099 ms

--- 10.0.1.12 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3997ms
rtt min/avg/max/mdev = 0.099/0.113/0.149/0.021 ms
```

file: /lab_data/lab_01/PC1_pingto_PC2

```
PING 10.0.1.11 (10.0.1.11) 56(84) bytes of data.
64 bytes from 10.0.1.11: icmp_seq=1 ttl=64 time=0.154 ms
64 bytes from 10.0.1.11: icmp_seq=2 ttl=64 time=0.130 ms
64 bytes from 10.0.1.11: icmp_seq=3 ttl=64 time=0.104 ms
64 bytes from 10.0.1.11: icmp_seq=4 ttl=64 time=0.098 ms
64 bytes from 10.0.1.11: icmp_seq=5 ttl=64 time=0.099 ms

--- 10.0.1.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.098/0.117/0.154/0.021 ms
```

file: /lab_data/lab_01/PC2_pingto_PC1

Explain the difference between pinging the local ethernet interface and the loopback interface. Specifically, on PC1, what is the difference between typing ping 10.0.1.11 and ping 127.0.0.1.

While **127.0.0.1** is essentially the home adress, which is the own adress for every pc. The 10.0.1.11-14 are the adresses which are reachable for every pc (1-4). It's essentially like saying "I'm living at home", which is true for most people, and saying "I'm living in ExampleStreet in Whatevertown".

## Exsercise 7

Just PC1 and PC2 responded. Because we didn't set the others PC answer to broadcasts requests.

**sudo tcpdump -n host 10.0.1.12**

```
06:03:26.365373 IP 10.0.1.11 > 10.0.1.12: ICMP echo request,id 3236, seq 1, length 64
06:03:26.365535 IP 10.0.1.12 > 10.0.1.11: ICMP echo reply,    id 3236, seq 1, length 64
```

**ping -c 1 10.0.1.12**

```
PING 10.0.1.12 (10.0.1.12) 56(84) bytes of data.
64 bytes from 10.0.1.12: icmp_seq=1 ttl=64 time=0.117 ms

--- 10.0.1.12 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 787ms
rtt min/avg/max/mdev = 0.099/0.113/0.149/0.021 ms
```

**(Additional note: We did this from PC2. But the result is the same, as there are no configuration differences between PC(1..4))**

**ping -c 1 111.111.111.111**

```
PING 111.111.111.111 (111.111.111.111) 56(84) bytes of data.

--- 111.111.111.111 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

**ping -c 2 -b 10.0.1.255**

```
PING 10.0.1.255 (10.0.1.255) 56(84) bytes of data.
64 bytes from 10.0.1.12: icmp_seq=1 ttl=64 time=0.036 ms
64 bytes from 10.0.1.12: icmp_seq=1 ttl=64 time=0.171 ms (DUP!)
64 bytes from 10.0.1.12: icmp_seq=2 ttl=64 time=0.040 ms

--- 10.0.1.255 ping statistics ---
2 packets transmitted, 2 received, +1 duplicates, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.036/0.082/0.171/0.063 ms
```

**tcpdump -n**

```
06:08:30.796602 IP 10.0.1.12 > 10.0.1.255: ICMP echo request, id 3271, seq 1, length 64
06:08:30.796745 IP 10.0.1.11 > 10.0.1.12: ICMP echo reply, id 3271, seq 1, length 64
06:08:31.795594 IP 10.0.1.12 > 10.0.1.255: ICMP echo request, id 3271, seq 2, length 64
06:08:31.795717 IP 10.0.1.11 > 10.0.1.12: ICMP echo reply, id 3271, seq 2, length 64
```

# Lab 2

## Pre-Module Questions

Write an ip command that adds the IP address 10.10.10.10/16 with broadcast address 10.10.255.255 to the interface eth0.

```
sudo ip address add 10.10.10.10/16 broadcast 10.10.255.255 dev eth0
```

Write a tcpdump command that captures packets containing IP datagrams with a source or destination IP address equal to 10.0.1.12.

```
sudo tcpdump -i eth0 host 10.0.1.12 and ip
```

Write a tcpdump command that captures packets containing ICMP messages with 192.168.0.10 as source or destination IP address.

```
sudo tcpdump -i eth0 src 192.168.0.10 or dst 192.168.0.10 and icmp
sudo tcpdump -i eth0 host 192.168.0.10 and icmp
sudo tcpdump -i eth0 icmp and \( 192.168.0.10 or 192.168.0.10\)
```

Write a tcpdump command that captures packets containing IP datagrams between two hosts with IP addresses 134.76.83.11 and 10.0.1.12, both on interface eth0.

```
sudo tcpdump -i eth0 \(host 134.76.83.11 or host 10.0.1.12\) and ip
```

Write a tcpdump filter expression that captures packets containing TCP segments with a source or destination IP address equal to 192.168.0.10.

```
sudo tcpdump -i eth0 host 192.168.0.10 and tcp
```

Write a tcpdump filter expression that, in addition to the constraints of the last question, only captures packets using TCP port number 23.

```
sudo tcpdump -i eth0 host 192.168.0.10 and tcp and port 23
```

Write a wireshark command with capture filter so that all IP datagrams with a source or destination IP address equal to 192.168.0.10 are recorded.

*“capture options” → “capture filter” → from here it is tcpdump syntax: ip and host 192.168.0.10*

Write a wireshark display filter that shows IP datagrams with a destination IP address equal to 192.168.0.50 and frame sizes greater than 400 bytes.

*Filter must be: frame.cap_len >= 400 bytes*

Write a wireshark display filter that shows packets containing ICMP messages with a source or destination IP adress equal to 192.168.0.10 and frame numbers between 15 and 30.

```
ip.addr == 192.168.0.10
  and icmp
    and (frame.number >= 15 and frame.number <= 30)
```

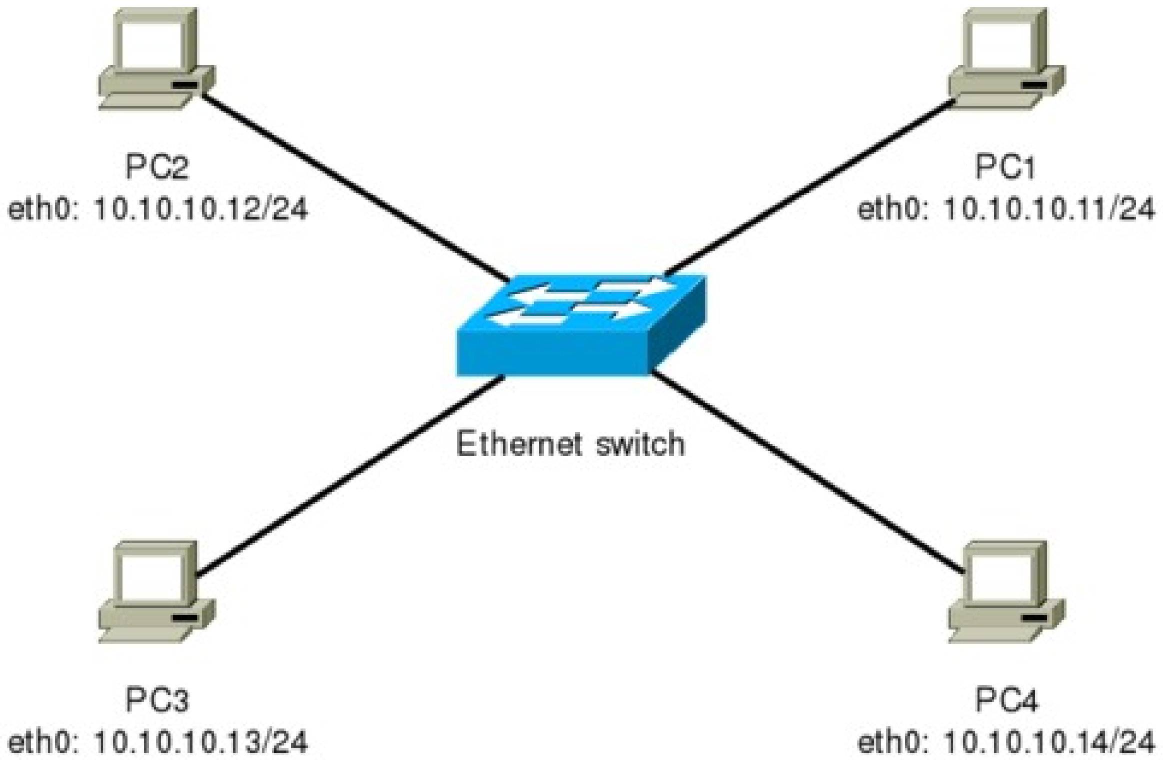
Write a wireshark display filter that shows packets containing TCP segments with a source or destination IP address equal to 192.168.0.10 and using port number 23.

```
ip.addr == 192.168.0.10 and tcp.port == 23
```

Write a wireshark capture filter expression for the last question.

```
host 192.168.0.10 and tcp port 23
```

## Exercises



## Exercise 1

**sudo tcpdump -i eth0 host 10.10.10.12**

```
02:00:48.128185 ARP, Request who-has 10.10.10.12 tell 10.10.10.11, length 28
02:00:48.128350 ARP, Reply 10.10.10.12 is-at 00:e0:4c:00:0a:20 (oui Unknown), length 46
02:00:48.128377 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3130, seq 1, length 64
02:00:48.128475 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3130, seq 1, length 64
02:00:49.129438 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3130, seq 2, length 64
02:00:49.129557 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3130, seq 2, length 64
02:00:50.128451 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3130, seq 3, length 64
02:00:50.128572 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3130, seq 3, length 64
02:00:51.127453 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3130, seq 4, length 64
02:00:51.127568 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3130, seq 4, length 64
02:00:52.126737 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3130, seq 5, length 64
02:00:52.126853 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3130, seq 5, length 64
02:00:53.141593 ARP, Request who-has 10.10.10.11 tell 10.10.10.12, length 46
02:00:53.141618 ARP, Reply 10.10.10.11 is-at 00:e0:4c:00:0a:10 (oui Unknown), length 28
```

## Exercise 2

Here we just had to save the wireshark capture into a file. The file can be found in the Lab 02 folder with the name “exercise2.pcapng”.

## Exercise 3

The wireshark capture for the display filter “***ip.dst == 10.10.10.12***” can be found with the file name ***exercise3dst.pcapng***.

## Exercise 4

Here we were using Wireshark to caputre packets which were send between PC1 and PC2. While we send ping packets from PC1 to PC2 we started a telnet session in the same direction. After logging in we disconnected again and stopped the wireshark capture. The corresponding file can be found as ***“exercise4.1.pcapng”***.

**icmp and ip.addr==10.10.10.12**

```
reading from file exercise4.1.pcapng, link-type EN10MB (Ethernet)
02:08:11.648304 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3237, seq 1, length 64
02:08:11.648458 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3237, seq 1, length 64
02:08:12.647300 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3237, seq 2, length 64
02:08:12.647417 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3237, seq 2, length 64
02:08:13.646740 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3237, seq 3, length 64
02:08:13.646863 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3237, seq 3, length 64
02:08:14.646739 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3237, seq 4, length 64
02:08:14.646858 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3237, seq 4, length 64
02:08:15.646738 IP 10.10.10.11 > 10.10.10.12: ICMP echo request, id 3237, seq 5, length 64
02:08:15.646859 IP 10.10.10.12 > 10.10.10.11: ICMP echo reply, id 3237, seq 5, length 64
```

**tcp and ip.addr==10.10.10.12 (shortened - complete file in folder)**

```
reading from file exercise4.2.pcapng, link-type EN10MB (Ethernet)
02:08:22.354446 IP 10.10.10.11.33755 > 10.10.10.12.telnet: Flags [S], seq 1611333335, win
29200, options [mss 1460,sackOK,TS val 2903629 ecr 0,nop,wscale 7], length 0
02:08:22.354570 IP 10.10.10.12.telnet > 10.10.10.11.33755: Flags [S.], seq 1354508602, ack
1611333336, win 28960, options [mss 1460,sackOK,TS val 895580 ecr 2903629,nop,wscale 7],
length 0
02:08:22.354599 IP 10.10.10.11.33755 > 10.10.10.12.telnet: Flags [.], ack 1, win 229, options
[nop,nop,TS val 2903629 ecr 895580], length 0
02:08:22.354769 IP 10.10.10.11.33755 > 10.10.10.12.telnet: Flags [P.], seq 1:28, ack 1, win 229,
options [nop,nop,TS val 2903630 ecr 895580], length 27 [telnet DO SUPPRESS GO AHEAD, WILL
TERMINAL TYPE, WILL NAWS, WILL TSPEED, WILL LFLOW, WILL LINEMODE, WILL NEW-ENVIRON,
DO STATUS, WILL XDISPLOC]
02:08:22.354868 IP 10.10.10.12.telnet > 10.10.10.11.33755: Flags [.], ack 28, win 227, options
[nop,nop,TS val 895580 ecr 2903630], length 0
02:08:22.360544 IP 10.10.10.12.telnet > 10.10.10.11.33755: Flags [P.], seq 1:13, ack 28, win
227, options [nop,nop,TS val 895582 ecr 2903630], length 12 [telnet DO TERMINAL TYPE, DO
TSPEED, DO XDISPLOC, DO NEW-ENVIRON]
02:08:22.360553 IP 10.10.10.11.33755 > 10.10.10.12.telnet: Flags [.], ack 13, win 229, options
[nop,nop,TS val 2903631 ecr 895582], length 0
02:08:22.360683 IP 10.10.10.12.telnet > 10.10.10.11.33755: Flags [P.], seq 13:52, ack 28, win
227, options [nop,nop,TS val 895582 ecr 2903631], length 39 [telnet WILL SUPPRESS GO AHEAD,
DO NAWS, DO LFLOW, DONT LINEMODE, WILL STATUS, SB TSPEED SEND SE, SB XDISPLOC SEND
SE, SB NEW-ENVIRON SEND SE, SB TERMINAL TYPE SEND SE]
[...]
```

**tcp.port==23 and ip.addr==10.10.10.12 (shortened - complete file in folder)**

```
reading from file exercise4.3.pcapng, link-type EN10MB (Ethernet)
02:08:22.354446 IP 10.10.10.11.33755 > 10.10.10.12.telnet: Flags [S], seq 1611333335, win
29200, options [mss 1460,sackOK,TS val 2903629 ecr 0,nop,wscale 7], length 0
02:08:22.354570 IP 10.10.10.12.telnet > 10.10.10.11.33755: Flags [S.], seq 1354508602, ack
1611333336, win 28960, options [mss 1460,sackOK,TS val 895580 ecr 2903629,nop,wscale 7],
length 0
02:08:22.354599 IP 10.10.10.11.33755 > 10.10.10.12.telnet: Flags [.], ack 1, win 229, options
[nop,nop,TS val 2903629 ecr 895580], length 0
02:08:22.354769 IP 10.10.10.11.33755 > 10.10.10.12.telnet: Flags [P.], seq 1:28, ack 1, win 229,
options [nop,nop,TS val 2903630 ecr 895580], length 27 [telnet DO SUPPRESS GO AHEAD, WILL
TERMINAL TYPE, WILL NAWS, WILL TSPEED, WILL LFLOW, WILL LINEMODE, WILL NEW-ENVIRON,
DO STATUS, WILL XDISPLOC]
02:08:22.354868 IP 10.10.10.12.telnet > 10.10.10.11.33755: Flags [.], ack 28, win 227, options
[nop,nop,TS val 895580 ecr 2903630], length 0

[...]
```

## Exercise 5

### What is the typical destination MAC address of an ARP Request packet?

First: *ff:ff:ff:ff:ff:ff* (the broadcast mac address) but after it nows the mac address it uses this for the ARP request (especially after the ping from PC2 to PC1).

### What are the different values of the Type field in the Ethernet headers that you captured?

*00:e0:4c:00:0a:10* for the PC1, *:20* for PC2 and *ff:ff:ff:ff:ff:ff* for the broadcast address.

### Describe the process which ARP uses to acquire a MAC address for a given IP address.

1. It yells in the network who nows the desired IP (for everyone)
2. Then one who knows which MAC belongs to the IP answers with the MAC address which corresponds to the IP address.

## Exercise 6

**ip neigh show** gives us:

```
PC1(eth0): 00:e0:4c:00:0a:10
PC2(eth0): 00:e0:4c:00:0a:20
PC3(eth0): 00:e0:4c:00:0a:30
PC4(eth0): 00:e0:4c:00:0a:40
```

## Exercise 7

**Use the saved data to determine the interval between each ARP Request. Describe the method used by ARP to determine the time between retransmissions of an unsuccessful ARP Request. Include the relevant parts of your save data in the lab report.**

The interval between the arp requests is 1s. This is due to the fact that it has to wait for 1s by default. It asks for a maximum amount of 5 times.

It uses the broadcast from the ip-protocoll. The broadcast adress is the equivalent to asking all participants in the network.

```
reading from file exercise7.pcapng, link-type EN10MB (Ethernet)
02:15:54.210041 ARP, Request who-has 10.10.10.10 tell 10.10.10.11, length 28
02:15:55.206709 ARP, Request who-has 10.10.10.10 tell 10.10.10.11, length 28
02:15:56.206708 ARP, Request who-has 10.10.10.10 tell 10.10.10.11, length 28
02:15:57.206769 ARP, Request who-has 10.10.10.10 tell 10.10.10.11, length 28
```

**Why are ARP packets not encapsulated like IP packets?**
Every participant in the network (routers, pcs, etc.) needs to understand and work with this request to be able to give a valid answer (if it has one). With this simple and not encapsulated format. The complete data fits in the header of the datagramm.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX: bytes  packets  errors  dropped overrun mcast
    195600  2471      0        0        0        0
    TX: bytes  packets  errors  dropped carrier collsns
    195600  2471      0        0        0        0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
qlen 1000
    link/ether 00:e0:4c:00:0a:10 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    135503  1668      0        0        0        0
    TX: bytes  packets  errors  dropped carrier collsns
    136898  1646      0        0        0        0
3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT
group default qlen 1000
    link/ether 00:18:8b:00:0a:11 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    0       0         0        0        0        0
    TX: bytes  packets  errors  dropped carrier collsns
    0       0         0        0        0        0
```

```
10.0.1.0/24 dev eth0  proto kernel  scope link  src 10.0.1.11
10.10.10.0/24 dev eth0  proto kernel  scope link  src 10.10.10.11
broadcast 10.0.1.0 dev eth0  table local  proto kernel  scope link  src 10.0.1.11
local 10.0.1.11 dev eth0  table local  proto kernel  scope host  src 10.0.1.11
broadcast 10.0.1.255 dev eth0  table local  proto kernel  scope link  src 10.0.1.11
broadcast 10.10.10.0 dev eth0  table local  proto kernel  scope link  src 10.10.10.11
local 10.10.10.11 dev eth0  table local  proto kernel  scope host  src 10.10.10.11
broadcast 10.10.10.255 dev eth0  table local  proto kernel  scope link  src 10.10.10.11
broadcast 10.10.255.255 dev eth0  table local  proto kernel  scope link  src 10.10.10.11
broadcast 127.0.0.0 dev lo  table local  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  table local  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  table local  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  table local  proto kernel  scope link  src 127.0.0.1
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101
```

## Exercise 8

### ip -statistics link

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default
          link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
          RX: bytes  packets  errors  dropped overrun mcast
          195600         2471     0          0          0          0
          TX: bytes  packets  errors  dropped carrier collsns
          195600         2471     0          0          0          0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
group default qlen 1000
          link/ether 00:e0:4c:00:0a:10 brd ff:ff:ff:ff:ff:ff
          RX: bytes  packets  errors  dropped overrun mcast
          135503         1668     0          0          0          0
          TX: bytes  packets  errors  dropped carrier collsns
          136898         1646     0          0          0          0
3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode
DEFAULT group default qlen 1000
          link/ether 00:18:8b:00:0a:11 brd ff:ff:ff:ff:ff:ff
          RX: bytes  packets  errors  dropped overrun mcast
          0          0          0          0          0          0
          TX: bytes  packets  errors  dropped carrier collsns
          0          0          0          0          0          0
```

### ip route list table all

```
10.0.1.0/24 dev eth0  proto kernel  scope link  src 10.0.1.11
10.10.10.0/24 dev eth0  proto kernel  scope link  src 10.10.10.11
broadcast 10.0.1.0 dev eth0  table local  proto kernel  scope link  src 10.0.1.11
local 10.0.1.11 dev eth0  table local  proto kernel  scope host  src 10.0.1.11
broadcast 10.0.1.255 dev eth0  table local  proto kernel  scope link  src 10.0.1.11
broadcast 10.10.10.0 dev eth0  table local  proto kernel  scope link  src 10.10.10.11
local 10.10.10.11 dev eth0  table local  proto kernel  scope host  src 10.10.10.11
broadcast 10.10.10.255 dev eth0  table local  proto kernel  scope link  src 10.10.10.11
broadcast 10.10.255.255 dev eth0  table local  proto kernel  scope link  src 10.10.10.11
broadcast 127.0.0.0 dev lo  table local  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  table local  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  table local  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  table local  proto kernel  scope link  src 127.0.0.1
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101
```

What are the network interfaces of PC1 and what are their respective maxium transmission values?

MTU for lo : 65536
MTU for eth0: 1500
MTU for eth1: 1500

The MTU of lo is easy explainable if we know, that the loopback (lo) interface is essentially the “home” of the PC. So it’s easy to accept that there is such a high MTU value.

## Exercise 9

Here we gathered information about the TCP/UDP ports via the "**ss**" command. In detail we used "**ss -a**" to retrieve the information on sockets.
We used "**nstat -z**" to get the information about the statistics from the various networks protocol, while "**nstat -a**" gave us the statistics for the networking protocols since the start of the respective pc.

### How many IP datagrams, ICMP messages, UDP datagrams and TCP segments has PC1 transmitted and received since its last reboot?

| **Protocol** | **received** | **transmitted** |
|---|---|---|
| **IP** | **4244** | **4241** |
| **ICMP** | **404 + 1217 + 222 = 1843** | **222 + 1217 + 438 = 1877** |
| **UDP** | **1174** | **1174** |
| **TCP** | **1224** | **1183** |

The strange answer for ICMP is for the different types. The command "nstat -a" differentiates between Type 0, 3 and 8. So do we.

## Exercise 10

Here we learned the commands "ip address" or respective "ip addr" or even "ip a" which are all the same.

**PC4$ ip addr**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group
default qlen 1000
        link/ether 00:18:8b:00:0a:41 brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default
qlen 1000
        link/ether 00:e0:4c:00:0a:40 brd ff:ff:ff:ff:ff:ff
        inet 10.0.1.14/24 brd 10.0.1.255 scope global eth0
        valid_lft forever preferred_lft forever
        inet 10.10.10.11/24 scope global eth0
        valid_lft forever preferred_lft forever
```

Here we see both ip addresses **..1.14** and **..10.11**. We could have removed the old one with "***ip a d 10.0.1.14 dev eth0***", but decided it's easier to see the concept when we show it in one output.

In general ***ip addr*** or shorter ***ip a*** gives us information about the interfaces, their assigned ip addresses and some additional information, like the MTU, if the interfaces are up and which MAC addresses and parameters belong the them.

## Exercise 11

Here we deleted the arp caches on all PCs via “ip neigh flush all” and started capturing the traffic at PC3(eth0). Additionally we started a telnet connection to 10.10.10.11 which was given to both PCs PC1 and PC4.

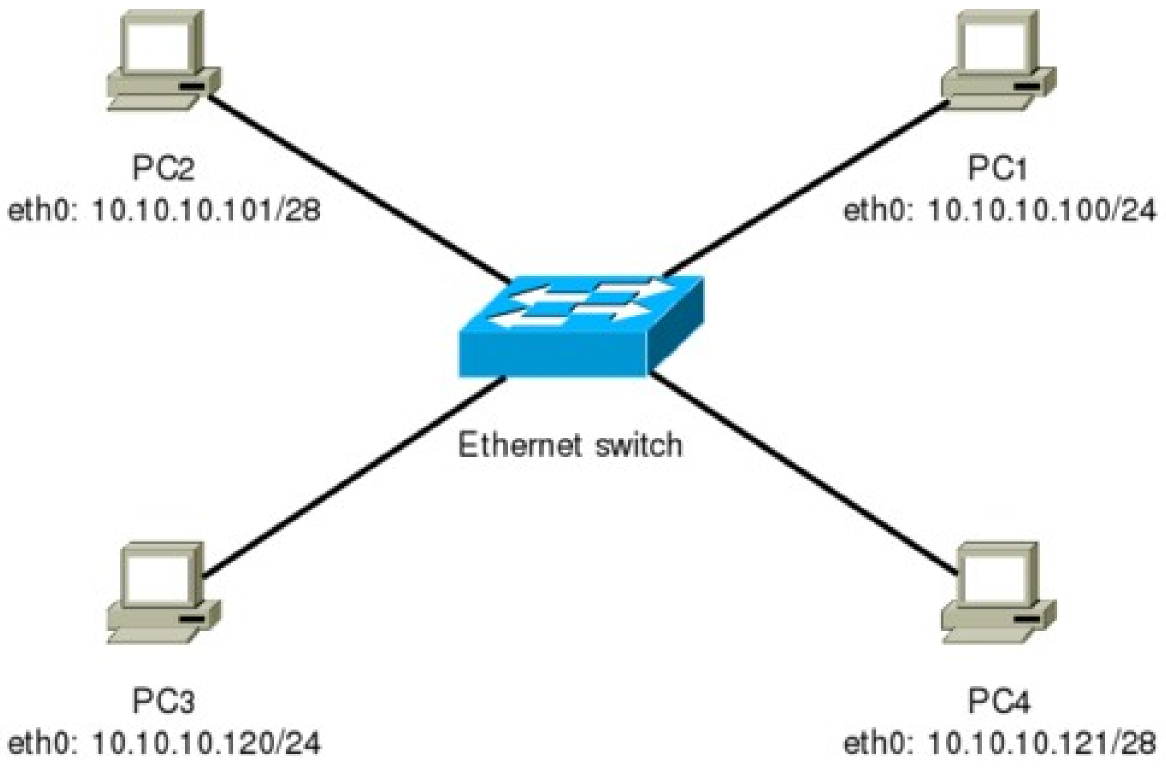
**Give an explanation for the host determined in step 4 and tell wh the telnet session was established at all and did not just return an error.**

PC3 started an ARP request to ask to which PC the IP address 10.10.10.11 belongs. After that, it depends on the other PCs. Which ever answers first with the MAC address gets the connection. The alternative would be throwing an error, but this won’t do good to the stability of larger networks, as he misconfiguration of one administrator could wreck whole parts of the network. This way at least one of the PCs can be reached.

```
reading from file exercise11-7.pcapng, link-type EN10MB (Ethernet)
00:15:51.465498 ARP, Request who-has 10.10.10.11 tell 10.10.10.13, length 28
00:15:51.465642 ARP, Reply 10.10.10.11 is-at 00:e0:4c:00:0a:10 (oui Unknown), length 46
00:15:51.465656 IP 10.10.10.13.53169 > 10.10.10.11.telnet: Flags [S], seq 1928648, win 29200, options
[mss 1460,sackOK,TS val 3841129 ecr 0,nop,wscale 7], length 0
00:15:51.465666 ARP, Reply 10.10.10.11 is-at 00:e0:4c:00:0a:40 (oui Unknown), length 46
00:15:51.465759 IP 10.10.10.11.telnet > 10.10.10.13.53169: Flags [S.], seq 1308138057, ack 1928649, win
28960, options [mss 1460,sackOK,TS val 3844805 ecr 3841129,nop,wscale 7], length 0
00:15:51.465787 IP 10.10.10.13.53169 > 10.10.10.11.telnet: Flags [.], ack 1, win 229, options [nop,nop,TS
val 3841129 ecr 3844805], length 0
```

Here we can see, that PC1 answers first, and in our telnet connection we could see that, we actually connected to PC1.

## Exercise 12



Here we see the configuration. Essential to know is, that PC2 and PC4 are in the Subnets /28, while PC1 and PC3 are still in /24.
/24 is a nice Subnet because it means the last part of the IP address can be flexible from 1 to 254.
But /28 means for PC2 it only can access the IP addresses 10.10.10.97 to 10.10.10.110
For PC4 it means that PC4 can only access the IP addresses 10.10.10.113 to 10.10.10.126.
We can see in the data, that PC1 and PC3 are able to reach each other and send packages to PC2 and PC4.
PC2 is able to complete communication with PC1.
PC4 is able to completely communicate with PC3.

| | PC1 | PC2 | PC3 | PC4 |
|---|---|---|---|---|
| PC1 | - | □ | □ | ✘ |
| PC2 | □ | - | ✘ | ✘ |
| PC3 | □ | ✘ | - | □ |
| PC4 | ✘ | ✘ | □ | - |

## Exercise 13

Here we added the respective hostnames for each PC to the /etc/hosts file. After that it was noch problem to ping to PC1, PC2, etc.

**Explain why the presented method is impractical for networks with a large number of hosts**

Each host needs to know the name for each of the other hosts. If we add one host, **all** other PCs needs to get configured to reach this one host. It’s easier to set up servers which tell the IP-addresses to given names on demand (DNS).

**What will be the result of the host name resolution process if there are more than one IP addresses associated with the same host name in /etc/hosts?**

It takes the first association.

## Exercise 14

In this exercise we capture the traffic of a ftp-connection for sniffing the password and username. We will see, that it’s transmitted in plaintext.

**Use the saved output to identify the port numbers used by ftp client and ftp server and find out the login name and password of the ftp user.**

Th port number of the server is 21, which is the standard port for ftp-servers. Our clientside port is 36253. This can variate between different connections.
The user and passwort are of course ubuntu/lab.

## Exercise 15

Now we repeat E14 with telnet instead of ftp. Like we thought, it was as easy to read as the ftp connection. A huge difference in this case is that there was a package for each character pressed and this package got sent twice. First from the client and then from the server.

We can see the output easier if we “follow” the TCP stream inside of wireshark. This even works for saved captures.

```
.......... ..!.."..'.....#..... ..#..'........!.."..... .....#.....'............P...... .
38400,38400....#.PC1:0.0....'..DISPLAY.PC1:0.0......xterm.............Ubuntu 14.04.1 LTS
PC2 login: uubbuunnttuu .
Password: lab .
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

.]0;ubuntu@PC2: ~.ubuntu@PC2:~$
```

*output of the TCP-follow stream ability by wireshark*

## Exercise 16

We followed a telnet connection via Wireshark. After pressing some keys, we recognized, that we had 2 packets for each key which was pressed. The first one was sent and the second one we received.

Our explanation is, that the telnet server acknowledges each key pressed by sending it back. This is what we can see in our telnet window. Via this method we always get direct confirmation about our actions.

# Lab 3

## Pre-Module Questions

### What is the IOS command to change the Maximum Transmission Unit (MTU) for an interface on a Cisco router?

```
# buffer length length
```

### How does a router determine whether datagrams to a particular host can be directly delivered through one of its interfaces?

It broadcasts it's interfaces (for example eth0, eth1 and eth2). For each interface it has an ARP table where it stores all reachable IPs(or IP ranges) in combination with their MAC addresses or addressranges. This way he knows what to send and where.

### Which systems generate ICMP route redirect messages: routers, hosts or both?

Yes, Routers will generate the ICMP route redirect message to Host. if host routing table is not yet updated. so router will route to the correct router and send the ICMP redirect message to host to update its routing table based on destination IP address.

### What is the default maximum TTL value used by traceroute when sending UDP datagrams?

The default is 30 but can be changed with -m ## where ## is the new max-time.

### Describe the role of a default gateway in a routing table.

The default-gateway is mostly the address of the router and has basically the task to redirect traffic which doesn't belong to the existing subnetwork.

### What is the network prefix of the IP address 10.10.10.10/24

/24 is the prefix length of address. It states that the first 24 bits are network prefix of the address (and the remaining 8bits are available for specific host address).

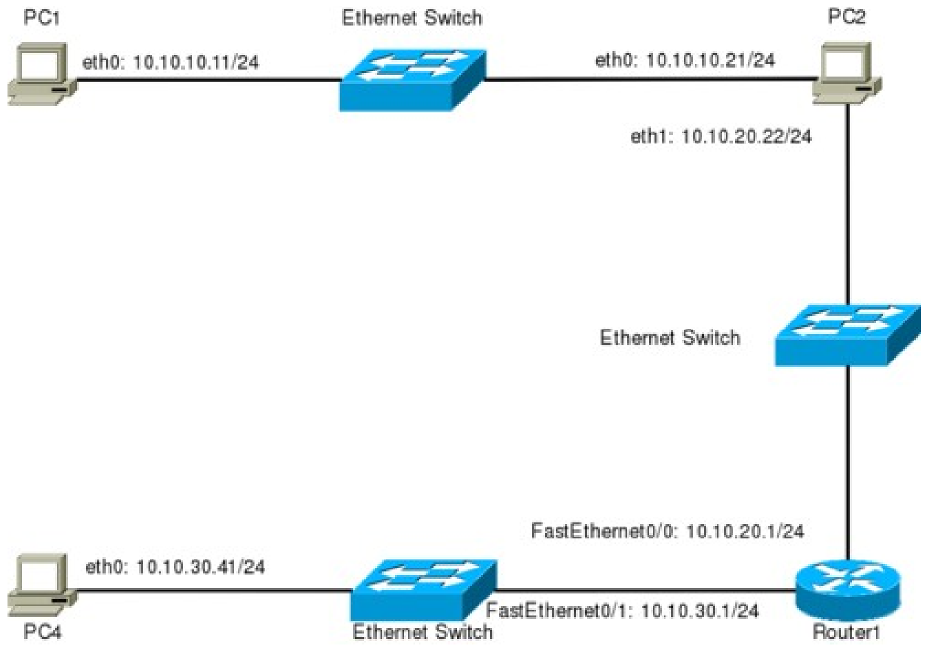
### Explain the difference between a network IP address and network prefix.

The network IP address is a specific address of one participant of the network, wile the network prefix specifies a range of IP addresses.

### An organization has been assigned the network number 134.76.0.0/16 and it needs to create networks that support up to 60 hosts on each IP network. What is the maximum number of networks that can be set up and why?

It has the network range /16 so it has 16 bits (=16384 combinations) for giving IP-Addresses. Of course we need to subtract the ones used for broadcasting and other stuff. Ideally we substract the combinations with .255 and .0 to be sure.

## Exercises



## Exercise 1

### What is the output of PC1 when the ping commands are issued?

When pinging 10.10.10.21 (PC2) we get answers. But the other addresses are (PC4 and the router1) unreachable.

### Which packets, if any, are captured by wireshark?

Just the packets we sent from PC1 to PC2.

### Do you observe any ARP or ICMP packets? If so, what do they indicate?

Yes the ARP packages broadcasted from PC1 and PC2. Then the ICMP packages which are the echo requests issued by the ping command. After that the question from PC2 for MAC address of PC1.

## Which destinations are not reachable and why?

Router1 and PC4 are not reachable because the router1 wasn't configured yet and the PC2 wasn't configured to forward ICMP messages. PC2 should behave as an IP-router.

**ping to the different IPs:**

```
ubuntu@PC1:~$ ping -c 5 10.10.10.21
PING 10.10.10.21 (10.10.10.21) 56(84) bytes of data.
64 bytes from 10.10.10.21: icmp_seq=1 ttl=64 time=0.300 ms
64 bytes from 10.10.10.21: icmp_seq=2 ttl=64 time=0.142 ms
64 bytes from 10.10.10.21: icmp_seq=3 ttl=64 time=0.143 ms
64 bytes from 10.10.10.21: icmp_seq=4 ttl=64 time=0.142 ms
64 bytes from 10.10.10.21: icmp_seq=5 ttl=64 time=0.145 ms

--- 10.10.10.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.142/0.174/0.300/0.063 ms

ubuntu@PC1:~$ ping -c 5 10.10.20.1
connect: Network is unreachable

ubuntu@PC1:~$ ping -c 5 10.10.30.41
connect: Network is unreachable
```

## Exercise 2

Here we just activated ip forwarding via

```
sysctl -w net.ipv4.ip_forward=1
```

## Exercise 3

With the command ip route add we finally connected the PCs. Now it was no problem to reach the different PCs and the router.

**PC1 routing table**

```
10.0.1.0/24 dev eth0  proto kernel  scope link  src 10.0.1.11
10.10.10.0/24 dev eth0  proto kernel  scope link  src 10.10.10.11
10.10.20.0/24 via 10.10.10.21 dev eth0
10.10.30.0/24 via 10.10.10.21 dev eth0
```

**PC2 routing table**

```
10.0.1.0/24 dev eth0  proto kernel  scope link  src 10.0.1.12
10.10.10.0/24 dev eth0  proto kernel  scope link  src 10.10.10.21
10.10.20.0/24 dev eth1  proto kernel  scope link  src 10.10.20.22
10.10.30.0/24 via 10.10.20.1 dev eth1
```

**PC4 routing table**

```
10.0.1.0/24 dev eth0  proto kernel  scope link  src 10.0.1.14
10.10.10.0/24 via 10.10.30.1 dev eth0
10.10.20.0/24 via 10.10.30.1 dev eth0
10.10.30.0/24 dev eth0  proto kernel  scope link  src 10.10.30.41
```

The first part desribes the IP or more often the IP-network-area which shall be routed. After the “via” keyword we can see, through which gateway we have to send the packages, so that they arrive at the given areas. At last we have the device for which this routing table entry is valid.

## Exercise 5

We just had to set up the IP addresses for both interfaces of router1.

## Exercise 6

First we learned some new commands, before we had the actual exercise:

**Display the routing table of router1 before and after adding routing entries to router1. The routing entries should enable router1 to forward datagrams to all networks in this setup.**

```
Router1# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

       10.0.0.0/24 is subnetted, 3 subnets
S      10.10.10.0 [1/0] via 10.10.20.22
C      10.10.20.0 is directly connected, FastEthernet0/0
C      10.10.30.0 is directly connected, FastEthernet0/1
```

The green highlighted area is our new entry. The rest keeps the same. This way router1 not only know the directly connected subnets (marked with an C in the beginning), but also the one behind PC2. After that and our initial configuration of the routing tables at the pcs now each one can access the other hosts.

## Exercise 7

Now the testing of the configuration begins. We pinged each PC and checked if all connections are possible and valid. Essentially the traceroute command itself is enough to verify each connection is working for one pc, because each step gets “pinged”.

```
ubuntu@PC1:~$ traceroute 10.10.30.41
traceroute to 10.10.30.41 (10.10.30.41), 30 hops max, 60 byte packets
 1  10.10.10.21 (10.10.10.21)   0.209 ms  0.200 ms  0.184 ms
 2  10.10.20.1   (10.10.20.1)     1.171 ms  1.446 ms  1.693 ms
 3  10.10.30.41 (10.10.30.41)   0.675 ms  0.669 ms  0.658 ms
```

## Exercise 8

The exercise begins with the explanation how we can show and clear the content of the arp cache from the routers. Additionally we get the command *“(no) arp <ADDRESS>”*.
After we erased the ARP cache on all PCs and router1 we started a Wireshark capture on PC1 and PC4 and sent 5 pings from PC1 to PC4.

### Determine the source and destination addresses in the Ethernet and IP headers for the ICMP Echo Request Messages on PC1 and PC4

| PC# | source / destination |
|---|---|
| PC1 | 10.10.10.11 / 10.10.30.41 |
| PC4 | 10.10.30.41 / 10.10.10.11 |

But important is, how the package progresses. From PC1 at arrives at interface eth0 with the ip address 10.10.10.21 from pc2 and will be forwarded via eth1(..20.22) to Router1 and his FE(0/0) which itself forwards it through FE01 to PC4.

### Use these results to explain how the source and destination ethernet addresses are changed when a datagram is forwarded by a router.

Here we have to differentiate between the final IP address and the next hop for the package. So if we ping from  we have the final IP address 10.10.30.41, but the next hop is 10.10.10.21 (eth0 from PC2). After each hop the next MAC and IP Address progresses until it reaches PC4.
Now it goes the other way around and PC4 wants to answer PC1.

## Exercise 9

|  |  |
|---|---|
| 10.10.30. 9 | Has 2 entries |
| 10.10.30.14 | Has 1 entries |
| 10.10.40. 1 | Has 1 entries |

Our gateway doesn't exist, which is the reason, why we didn't get any answers for our pings. But if the gateways would be configured and existing, we would have get answer for our first question by the 2nd gateway because it has the smaller network.

### ip route

```
10.0.1.0/24 dev eth0  proto kernel  scope link  src 10.0.1.11
10.10.0.0/16 via 10.10.10.71 dev eth0
10.10.10.0/24 dev eth0  proto kernel  scope link  src 10.10.10.11
10.10.20.0/24 via 10.10.10.21 dev eth0
10.10.30.0/24 via 10.10.10.21 dev eth0
10.10.30.9 via 10.10.10.81 dev eth0
```

### Wireshark Capture (only partially)

```
reading from file exercise09.pcapng, link-type EN10MB (Ethernet)
20:59:44.271162 ARP, Request who-has 10.10.10.81 tell 10.10.10.11, length 28
20:59:45.270610 ARP, Request who-has 10.10.10.81 tell 10.10.10.11, length 28
20:59:46.270614 ARP, Request who-has 10.10.10.81 tell 10.10.10.11, length 28
21:00:15.791135 IP 10.10.10.11 > 10.10.30.14: ICMP echo request, id 2961, seq 1, length 64
21:00:20.794609 ARP, Request who-has 10.10.10.21 tell 10.10.10.11, length 28
21:00:20.794725 ARP, Reply 10.10.10.21 is-at 00:e0:4c:00:0a:20 (oui Unknown), length 46
21:01:23.199023 ARP, Request who-has 10.10.10.71 tell 10.10.10.11, length 28
21:01:24.198613 ARP, Request who-has 10.10.10.71 tell 10.10.10.11, length 28
[...]
21:01:40.510613 ARP, Request who-has 10.10.10.71 tell 10.10.10.11, length 28
```

PC1 resolves multiple entries, by using the most exact match. In this case it will try to access 10.10.30.9 via 10.10.10.81 because it is the most exact match.
10.10.30.14 will be accessed via 10.10.10.11 because it's the next fitting exact routing table, while 10.10.40.1 only gets covered by 10.10.10.71 because it's assigned to the subnet 10.10.0.0/16.

## Exercise10

We changed the routes by deleting the old ones and adding default routes from PC1 with PC2 as gateway and added another one from PC2 with FE00 from Router1 as gateway. The next steps were to capture with Wireshark and ping from PC1.

### What is the output of the ping command?

Destination unreachable.

### How far does the ICMP echo message travel?

We get “Destination unreachable” messages by PC2 and Router1. Of course PC1 just gets the message by PC2, while PC2 gets the message by Router1.

### Is an ICMP echo reply returned? If yes, which?

No ICMP echo reply, unless we’re saying that “destination unreachable” is really a reply.

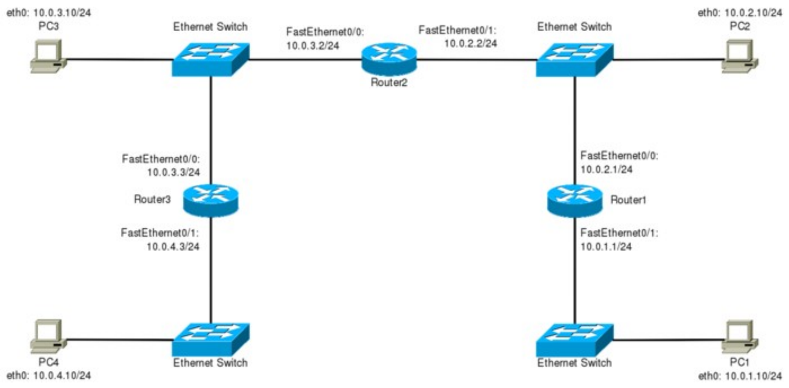
## Exercise 11

After enabled proxy arp an erasing arp cache and routing table of PC4, we set the ip addresses to 10.10.30.41/8. We captured traffic at PCs 1, 2 and 4 only for arp and icmp packets. Now we ping 10.10.10.11 (IP of PC1) from PC4.

The ping is succesful because of the proxy arp which can tell the different PCs exactly where to find which MAC address and under which IP address name.

After we disabled the proxy arp, it wasn’t possible to contact PC1 from PC4 anymore.

## Exercise12



The task was that we provoke and icmp redirect request.
Because the connection wasn’t working properly we fixed it, with static routing information for the routers so PC2 and PC4 can communicat. It was important that we activate the ip redirects for the interface 0/0 of router 1.

Most of the information and the information that we needed an defect connection go on was missing the exercise. So we tried to fix it by:

1. Redirecting traffic for PC4 from R1 → R2.
2. A route table for PC4 from R2 → R3.

Additionally we added (to make a full ping conversation working)

1. A route table for PC4 that PC2 is behind R3.

2. A route table for R3, that PC2 is behind R2.
3. of course R2 already knows, where PC2 is → direct connection.

Is there a difference between the routing table and routing cache immediately after the ICMP route redirect message?

The redirect message is saved only in the cache for static routing.

When you viewed the cache a few minutes later, what did you observer?

The cache will be emptied after a little time - so then it should be same like before the message.

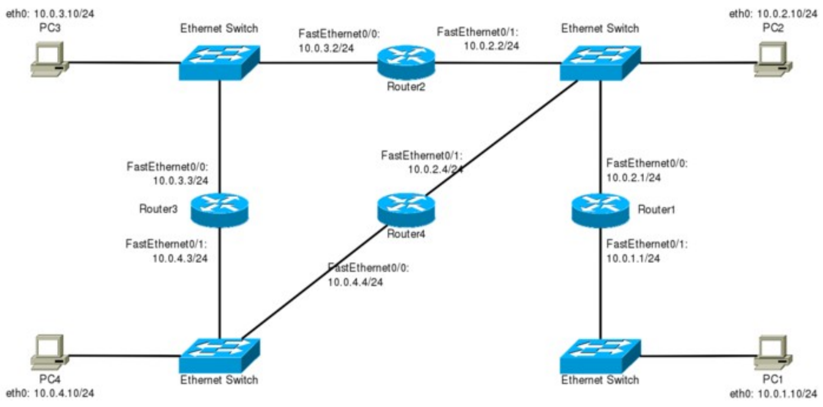
Describe how the ICMP route redirect works using the output you saved.

After PC2 gets the ICMP route redirect message the MAC address for the destination in the headers of PC2s packages changes.

Explain how Router1 know that datagrams destined to 10.0.3.10/24 should be forwarded to 10.0.2.2/24.

Because we told him. We set up a routing table which contains this information.

## Exercise 13



**Routing tables of Router2, 3 and 4**

```
Gateway of last resort is not set
        10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
S       10.0.1.10/32 [1/0] via 10.0.3.3
C       10.0.2.0/24 is directly connected, FastEthernet0/1
C       10.0.3.0/24 is directly connected, FastEthernet0/0
S       10.0.1.0/24 [1/0] via 10.0.3.3
```

```
Gateway of last resort is not set
        10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
S       10.0.1.10/32 [1/0] via 10.0.4.4
```

```
C        10.0.3.0/24 is directly connected, FastEthernet0/0
S        10.0.1.0/24 [1/0] via 10.0.4.4
C        10.0.4.0/24 is directly connected, FastEthernet0/1
```

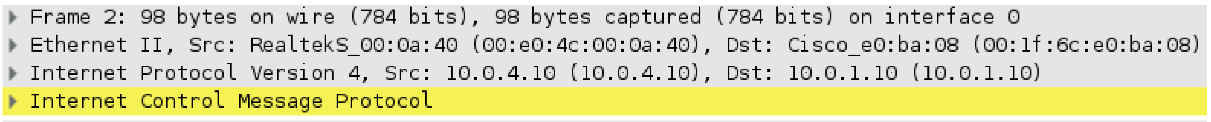
```
Gateway of last resort is not set
         10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
S        10.0.1.10/32 [1/0] via 10.0.2.2
C        10.0.2.0/24 is directly connected, FastEthernet0/1
S        10.0.1.0/24 [1/0] via 10.0.2.2
C        10.0.4.0/24 is directly connected, FastEthernet0/0
```

The ping packets can be found in: “exercise13_pc4_pingpackets.pcapng”

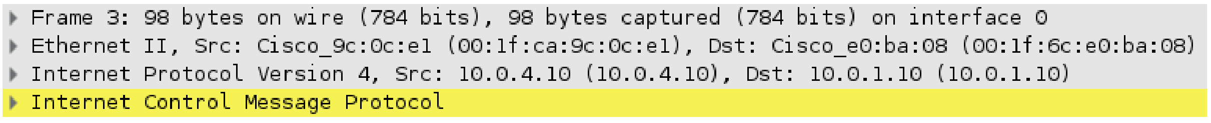
## Do two ICMP packets differ? If yes: How do they differ? Include both packets.

Yes, of course! At first, PC4 is sending to Router4 while in the next round (after the package went PC4 → R4 → R2 → R3) we have the MAC address of FE01 from Router3 in the header.

Unfortunately we can’t find a way to print the MAC addresses to normal text. So here are the screenshots:



```
Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: RealtekS_00:0a:40 (00:e0:4c:00:0a:40), Dst: Cisco_e0:ba:08 (00:1f:6c:e0:ba:08)
Internet Protocol Version 4, Src: 10.0.4.10 (10.0.4.10), Dst: 10.0.1.10 (10.0.1.10)
Internet Control Message Protocol
```



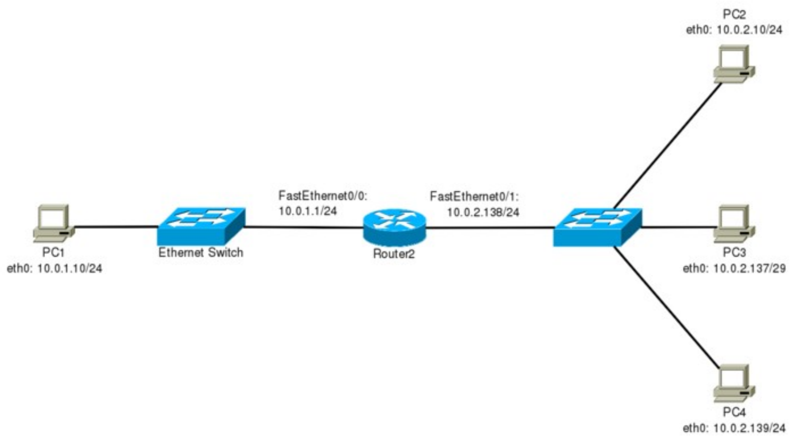
```
Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: Cisco_9c:0c:e1 (00:1f:ca:9c:0c:e1), Dst: Cisco_e0:ba:08 (00:1f:6c:e0:ba:08)
Internet Protocol Version 4, Src: 10.0.4.10 (10.0.4.10), Dst: 10.0.1.10 (10.0.1.10)
Internet Control Message Protocol
```

Here we can see, that the IP destination and source won’t change, but the source MAC addresses.

## Does the ICMP echo request travel forever in the network? Why?

Of course not. The ping command has a given TTL for each package. The default value is 30, but can be changed via command line argument. We can use this behaviour to simulate the traceroute command.

## Exercise 14



The essential part for this setup is, that we have to use router 2 as gateway. Noticable is, that PC3 has a different subnet than PC2 and PC4, even as it lies on the "right" side of the router and is part of the 10.0.2.0/24 subnet.

We could observe, that the normal pings (PC1 → P2-4 and PC2 ⇔ PC4) behaved as expected. Interesting was, that the communication from PC3 to PC2 needed the router as additional hop, because PC3 can't access PC4 from it's own subnet:

| | |
|---|---|
| 10.0.2.137/**29** | 192.0.2.**137** - 192.0.2.**142** |

As we can see, PC4 is within this subnet (10.0.2.139), but PC2 is not.

### If PC3 had no default entry in its table, would you have seen the same results? Explain what would have changed for each *ping*.

We just cover pings covering the connection to or from PC3, as this is the only special host here.

Only the connection between PC2 and PC3 would be affected. PC3 can't access PC2 without the information from the router (which knows, all parts).

# Lab 4

## Pre-Module Question

### Provide the command that configures a Linux PC as an IP router (see Lab 3).

First we have to enable ip_forwarding via one(!) of these two commands:

a. `echo 1 > /proc/sys/net/ipv4/ip_forward`
b. `sysctl -w net.ipv4.ip_forward=1`

This enables the ip forwarding. The next step is to use either a dynamic or static routing. For static routing we need essentially just one command:

a. ip route add HOST/32 via GATEWAY
b. dynamic routing can be realised by ospf, bgp or rip. (We will cover this in Lab 4)

### What are the main differences between a distance vector routing protocol and a link state routing protocol? Give examples for each type of protocol.

RIP and RIP2 are examples for the distance vector routing protocol, while OSPF is used as a link state routing protocol.
One of the differences is the algorithm they are both working with. RIP works with Bellman-Ford-Algorithm, wile OSPF uses the Dijkstra-Algorithm (Shortest Path).
The distance-vectors-protocols (I call them DVP from now on) need far less work than the link-state-protocolls (from now on LSP). LSPs have a faster convergence, while maintaining the simpler implementation (compared to DVP).

### What are the differences between an intradomain routing protocol (also called interior gateway protocol or IGP) and an interdomain routing protocol (also called exterior gateway protocol or EGP)? Give examples for each type of protocol.

IGP protocols are routing protocols which are used inside of autonomous systems (AS), while EGP are used (like the name suggests) outside of autonomous systems.
OSPF and RIP are intradomain routing protocolls. BGP is an EGP which means an (and like wikipedia says the only one which is used) exterior gateway protocol.

The EGP protocolls (so BGP) are used to exchange information between AS. These informations contain which nets are accessable.

## Which routing protocols are supportet by the software package Quagga?

Regarding to the software manual and their website Quagga support RIP, RIPng, OSPFv2/3, Babel and BGP.

## In the Quagga software package, the processes ripd, ospfd and bgpd deal, respectively, with the routing protocols RIP, OSPF and BGP. which role does the process zebra play?

Zebra is the routing manager, which provides kernel routing updates, interface lookups and redistribution of routes between the different routing protocols.

## Describe how a Linux user accesses the processes of Quagga (zebra, ripd, ospfd, bgpd) to configure routing algorithm parameters?

We have to use the terminal and start zebra. After that we can use a syntax which is close to the ones we're using for the CISCO routers. For example: We can use ip address ADDRESS/PREFIX and no ip address ADDRESS/PREFIX to (de)activate an ip address.

## What is the main difference between RIP version 1 (RIPv1) and RIP version 2 (RIPv2)

The main difference is the support for CIDR (Classless Inter Domain Routing) and authentication for RIPv2. RIPv2 is using the zero-fields to store more information with the same space. Especially CIDR is important to make use of the full adressspace of IPv4 which is limited to $2^{32}$ addresses.

## Explain what it means to run RIP in passive mode.

The passive mode is standing for listening and just updating the own routes based on advertisements but not advertising the routes to others, while actually exactly that: It actively advertises the routes to it's neightbors.

## Explain the meaning of triggered updates in RIP.

In the default configuration each RIP-router sends all its routinginformation in given time intervalls to his neightbors. If it has the triggered updates setting, it additionally sends it, when one of its routes changes it's metrik. This may happen with an update it got from it's neighbors.

## Explain the concept of split-horizon in RIP.

The routing table saves the information from which router it got send. This information can be used to just update the other routers. So if we have  router 1,2,...,10 and router 3 is sending an update to router 5, router 5 now sends just updates to the other routers but not 3.
Additionally Split Horizon is used to find broken connections between routers a lot faster than without. This is due to the fact, that it doesn't need to count the metriks up to 16 til the connection will be dropped because of infinite distance.

## What is an autonomous system (AS)? Which roles do autonomous systems play in the Internet?

AS are kind of collections of IP-subnets, which are managed as one entity via an IGP. One AS can consists of multiple sub-nets which can be AS themselfes.
In the Internet AS normally belong to internet providers, huge companies or universities, which are interconnected. The whole is called the internet.

## What is the AS number of your institution? Which autonomous system has AS number 1?

Using ultratools.com I got the following information:
AS1 is US/arin and the owner is LVLT-1 - Level 3 communications, Inc.
AS680 is the ASN of the IP address at the university of G  ttingen.

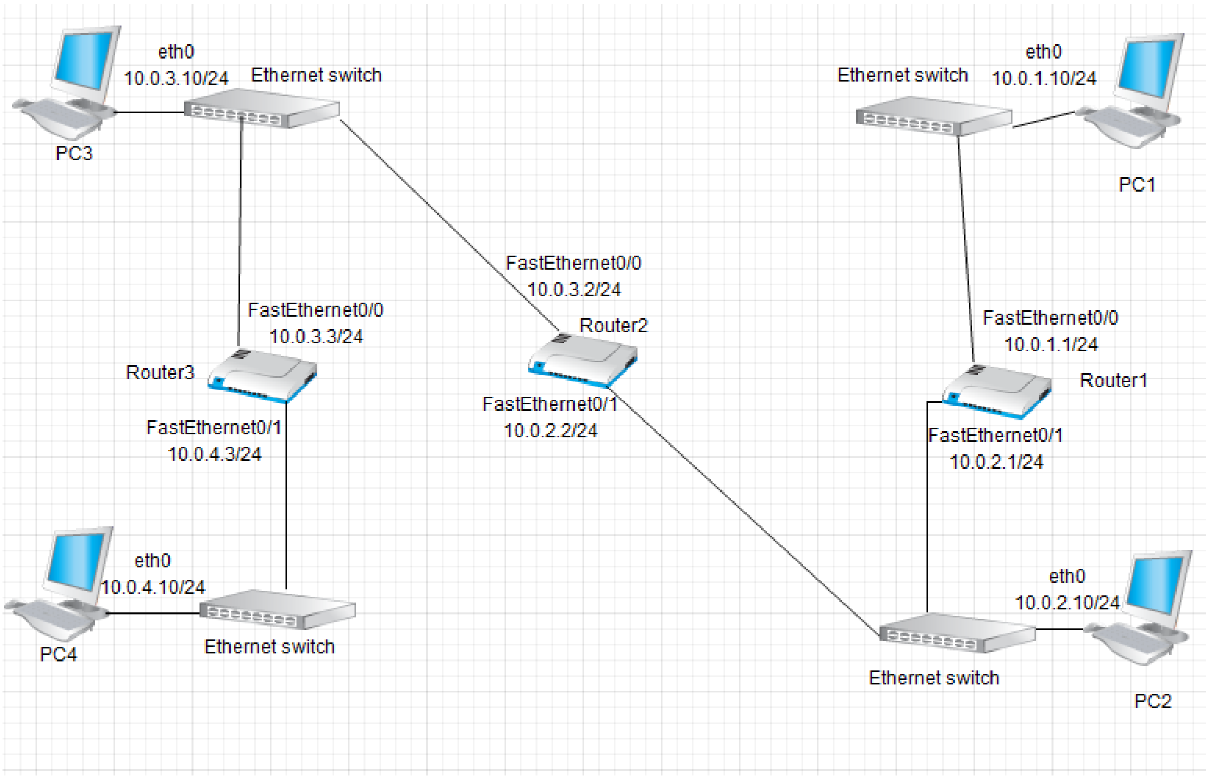
## Explain the terms stub AS, multihomed AS and transit AS.

Stub-AS are connected with exactly one link to a provider (end-vertices). Normally Stub-AS shouldn't exist, because of criteria which just allows AS if it has two providers.
Multihomed-Stub-AS are connected via multiple links to exactly one provider. This is for reliability.

Multihomed-AS are connected to multiple Providers. This provides additional reliability

Transit-AS are connected to other Transit-AS and are the serviceproviders for the preceding 3 AS-types. These are a form of the internet-backbone-networks. A Transit-AS is always a provider for at least one another AS.

## Part 1



### Exercise 1

In this exercise we just configure the routers for rip routing and assign the network 10.0.0.0.

After that we cleared the route tables of the routers.

# Part 2

## Exercise 2

Our first action was to start the quagga process (1) and then connect to it via telnet (2):

```
(1) /etc/init.d/quagga start
(2) telnet localhost 2602
```

Now we're able to set the RIP informations for the PCs.

### What is the destination IP address of RIP packets?

The connected PCs and the special IP address 224.0.0.9.

### Do routers forward RIP packets? In other words, does PC1 receive RIP packets sent by Router3?

Of course not, because each active RIP router processes the RIP packages and sending it's own routing table. Routers only communicate with their direct neighbors and if the neighbors themselves are RIP agents too, they provide their own tables and sending them as RIP packets combined with their own data.

### Which types of routing RIP messages do you observe? The type of a RIP message is indicated by the value of the field *command*. For each packet type that you observed, explain the role that this message type plays in the RIP protocol.

Requests have the number 1 and Responses have the number 2. RIP messages may contain multiple routing entries:

20 bytes

**Adress Family:** Which IP

**Route Tag:** 0

**IP Adress:** 10.0.PC#.0

**Netmask:** 255.255.255.0 (in our case)

**Next hop:** 0.0.0.0 (in our case always)

**Metric:** from 1 to 3 (the request has 1 as static value)

A RIP message may contain multiple routing entries. How many bytes are consumed in a RIP message for each routing table entry? Which information is transmitted for each message?

Each Routing entry consumes 20bytes, which consisting of:

- Adress Family (2B)
- Route Tag (2B)
- IP Address (4B)
- Netmask (4B)
- Next Hop (4B)
- Metrix (4B)

For PC1 include the output of the commands ip route show table all and netstat -rn from steps 4 and quagga# show ip rip from step 5. Discuss the differences in the output of the commands. (unfortunately we just have the output from netstat and quagga show ip rip of PC3)

**ip route show table all**

```
root@PC1:~# ip route show table all
10.0.1.0/24 dev eth0  proto kernel  scope link  src 10.0.1.10
10.0.2.0/24 via 10.0.1.1 dev eth0  proto zebra  metric 2
10.0.3.0/24 via 10.0.1.1 dev eth0  proto zebra  metric 3
10.0.4.0/24 via 10.0.1.1 dev eth0  proto zebra  metric 4
broadcast 10.0.1.0 dev eth0  table local  proto kernel  scope link  src 10.0.1.10
local 10.0.1.10 dev eth0  table local  proto kernel  scope host  src 10.0.1.10
broadcast 10.0.1.255 dev eth0  table local  proto kernel  scope link  src 10.0.1.10
broadcast 127.0.0.0 dev lo  table local  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  table local  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  table local  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  table local  proto kernel  scope link  src 127.0.0.1
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101
```

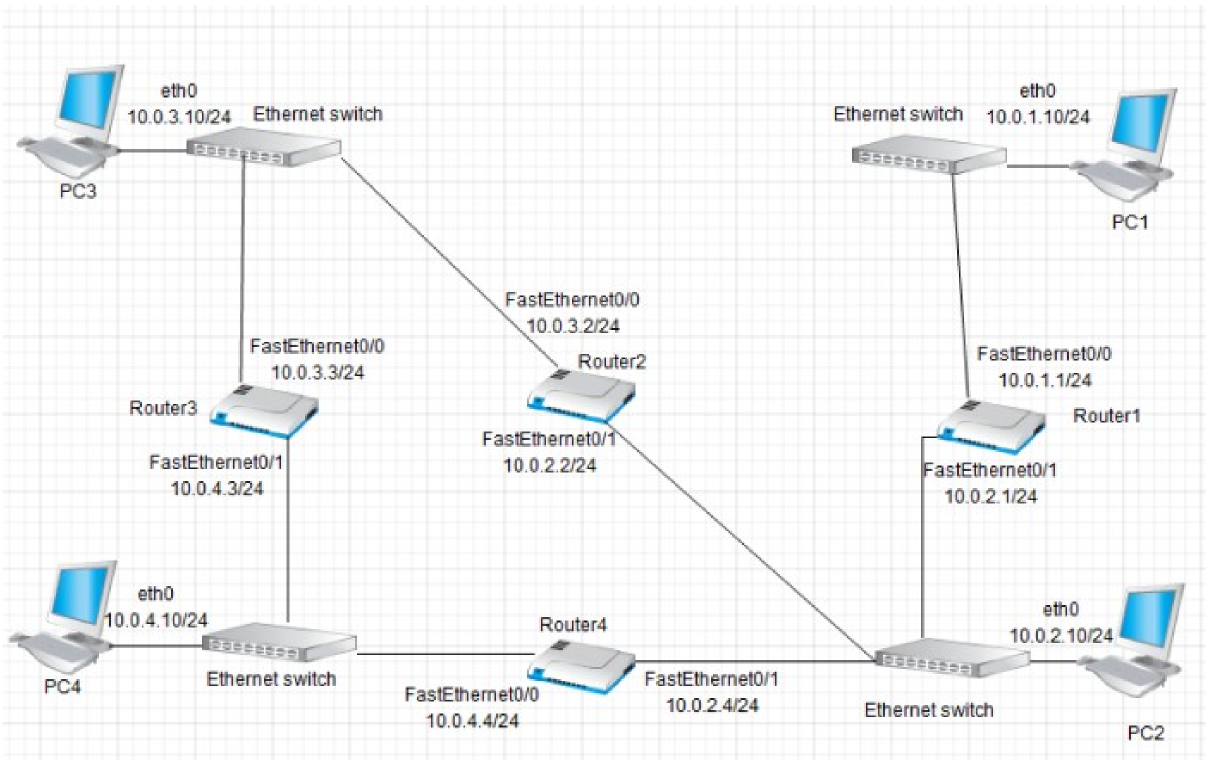
**netstat -rn**

```
ubuntu@PC3:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        10.0.3.2        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        10.0.3.2        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.4.0        10.0.3.3        255.255.255.0   UG        0 0          0 eth0
```

**quagga# show ip rip**

```
ripd# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
        (n) - normal, (s) - static, (d) - default, (r) - redistribute,
        (i) - interface
        Network         Next Hop        Metric From             Tag Time
R(n) 10.0.1.0/24        10.0.3.2        3 10.0.3.2              0 02:50
R(n) 10.0.2.0/24        10.0.3.2        2 10.0.3.2              0 02:50
C(i) 10.0.3.0/24        0.0.0.0         1 self                  0
R(n) 10.0.4.0/24        10.0.3.3        2 10.0.3.3              0 02:52
```

## Part 3



### Exercise 3

In the first part we updated the routing tables and connected router 4 due to the changed setup. After that we enabled RIP on router 4. The new routing tables can be found in the corresponding folder to Lab 4.

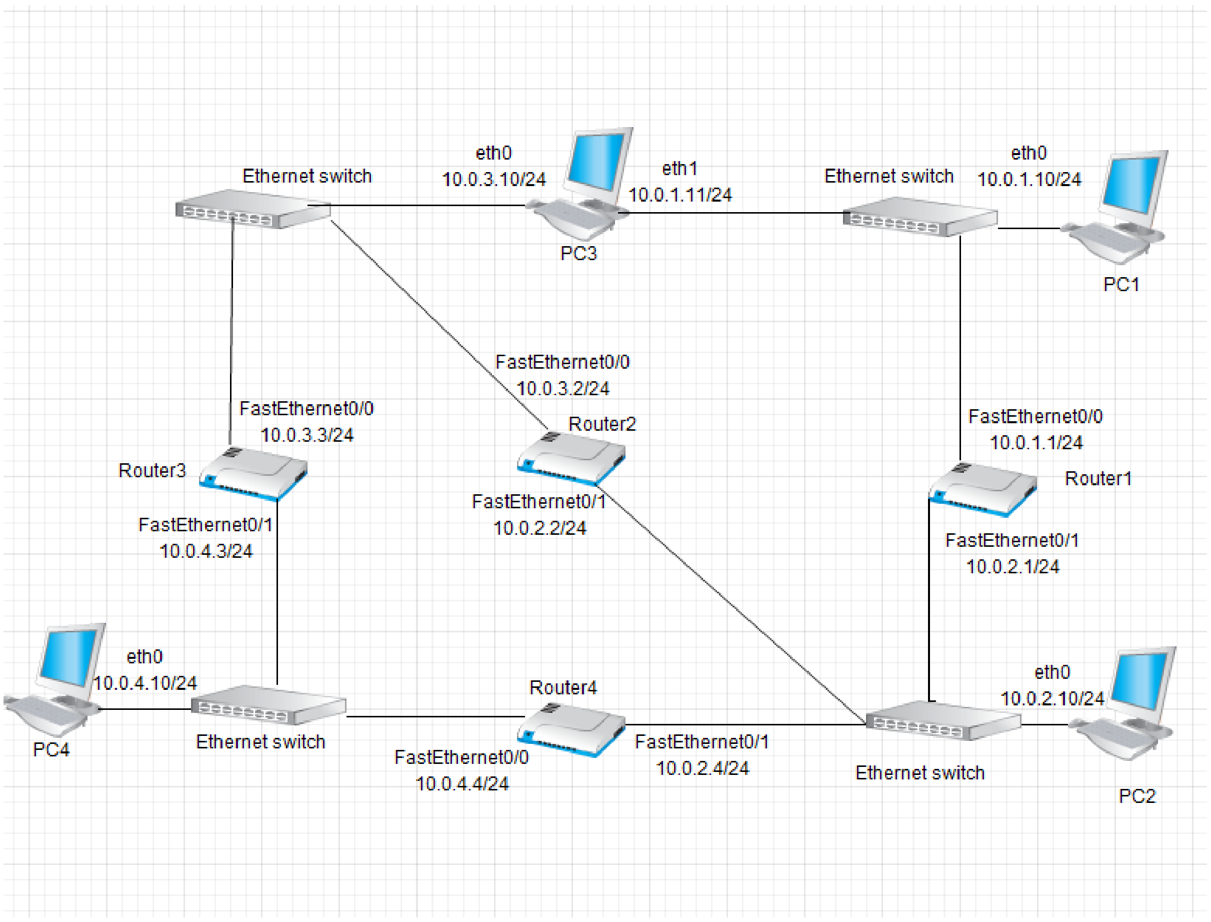
In the second part we pinged from PC4 to PC1. We kept pinging for the following scenario:

We disconnected the ethernet cable which connected router 4 a

**excerpt of ping command (complete file in archive)**

```
ubuntu@PC4:~$ ping 10.0.1.10
PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data.
[...]
64 bytes from 10.0.1.10: icmp_seq=6 ttl=62 time=0.459 ms
From 10.0.1.10 icmp_seq=60 Destination Host Unreachable
[...]
From 10.0.1.10 icmp_seq=185 Destination Host Unreachable
ping: sendmsg: Network is unreachable
[...]
ping: sendmsg: Network is unreachable
64 bytes from 10.0.1.10: icmp_seq=211 ttl=61 time=0.616 ms
64 bytes from 10.0.1.10: icmp_seq=212 ttl=61 time=0.573 ms
[...]
--- 10.0.1.10 ping statistics ---
214 packets transmitted, 10 received, +126 errors, 95% packet loss, time 213116ms
rtt min/avg/max/mdev = 0.442/0.508/0.616/0.066 ms, pipe 3
```

## Part 4



### Exercise 4

We restarted the quagga process on PC3 to configure it as new part of the RIP network. Then we started to capture RIP packets for both interfaces on PC3.

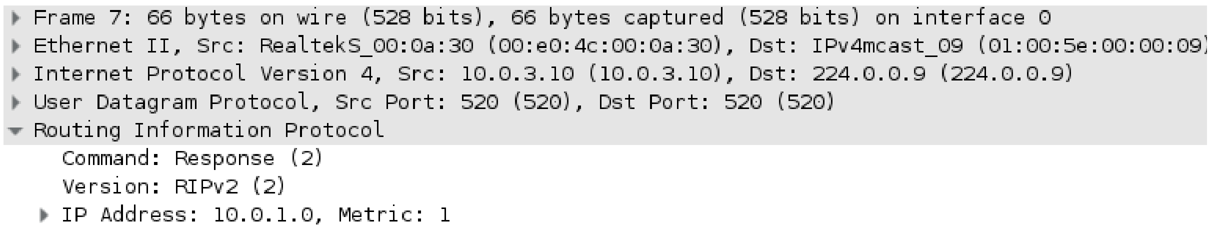
Additionally we checked PC1 and PC4 with netstat -rn if the routing tables have been updated (yes).

In the following order we kept experimenting. First we set the cost metric for both connections of router1 to 10 (10x the normal value). It took a long time until the network readjusted.

After the connection was settled, we shut PC3s eth1 connection down. Now the only way to get to PC1 for the other PCs was the route over router 1 again. The first pings failed until the RIP protocol finally found the correct connection again.

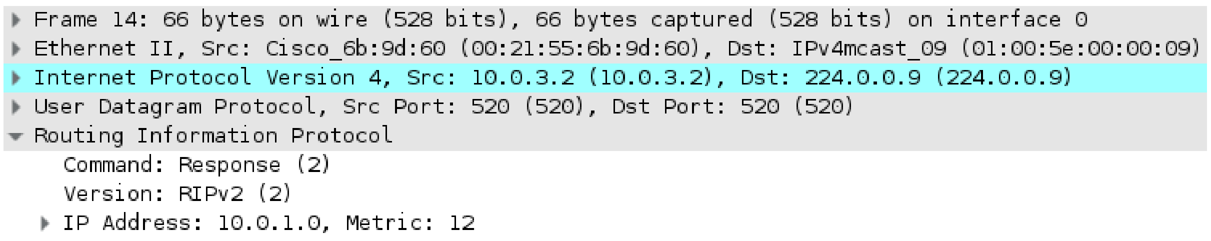
The most essential part we can find on the captured traffic on PC3.

**with connection working**



```
▸ Frame 7: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: RealtekS_00:0a:30 (00:e0:4c:00:0a:30), Dst: IPv4mcast_09 (01:00:5e:00:00:09
▸ Internet Protocol Version 4, Src: 10.0.3.10 (10.0.3.10), Dst: 224.0.0.9 (224.0.0.9)
▸ User Datagram Protocol, Src Port: 520 (520), Dst Port: 520 (520)
▾ Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
  ▸ IP Address: 10.0.1.0, Metric: 1
```

**after disconnecting**



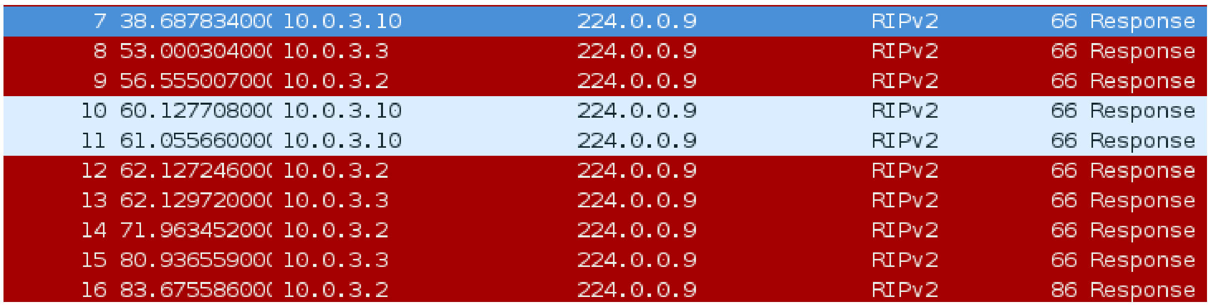
```
▸ Frame 14: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: Cisco_6b:9d:60 (00:21:55:6b:9d:60), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
▸ Internet Protocol Version 4, Src: 10.0.3.2 (10.0.3.2), Dst: 224.0.0.9 (224.0.0.9)
▸ User Datagram Protocol, Src Port: 520 (520), Dst Port: 520 (520)
▾ Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
  ▸ IP Address: 10.0.1.0, Metric: 12
```

As we can see, the connection to PC1 goes different ways with different metrics. The problem here is, that each RIP participant only keeps telling the next neighbors about the routing tables. Additionally it gets the wrong old working information for the connection.

So in simple words: After we disconnected PC3 from PC1, nobody noticed for a long time, because they kept telling each other the old story about how to get to PC1 over PC3.

To avoid the problem we configured the router 1-4 to “timers basic 30 180 60 240” and “flash-update-threshold 0”.

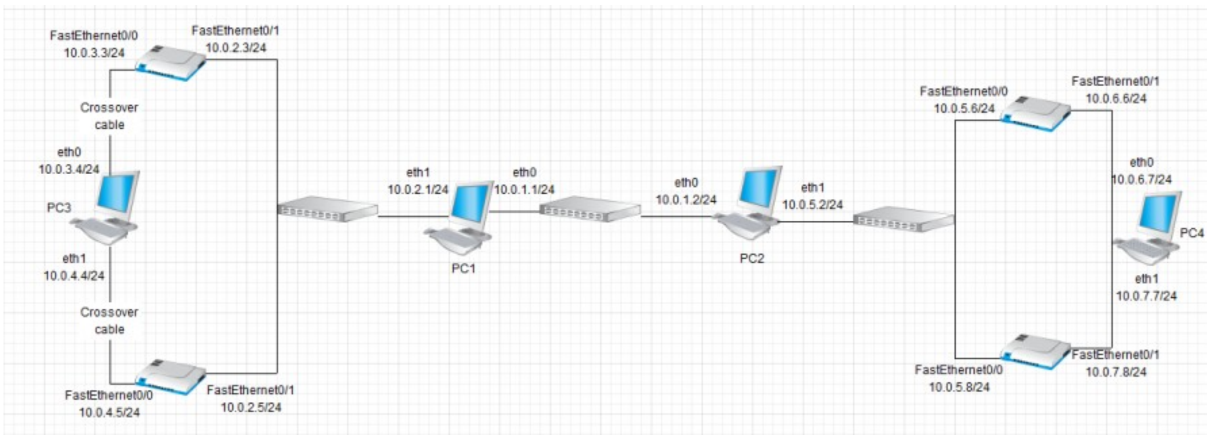
After this we can observe, that after a disconnection of PC3 from PC1 the correct routing table recovers very fast. The interesting aspect here is, that the first valid route wasn’t directly from PC3 → R2 → R1 → PC1, but PC3 → R3 → R4 → R1. In the following Wireshark screenshot we have the first and the last package. The first is still for the direct connection, while the last one is for the connection over R2.



```
 7 38.68783400( 10.0.3.10   224.0.0.9   RIPv2   66 Response
 8 53.00030400( 10.0.3.3    224.0.0.9   RIPv2   66 Response
 9 56.55500700( 10.0.3.2    224.0.0.9   RIPv2   66 Response
10 60.12770800( 10.0.3.10   224.0.0.9   RIPv2   66 Response
11 61.05566000( 10.0.3.10   224.0.0.9   RIPv2   66 Response
12 62.12724600( 10.0.3.2    224.0.0.9   RIPv2   66 Response
13 62.12972000( 10.0.3.3    224.0.0.9   RIPv2   66 Response
14 71.96345200( 10.0.3.2    224.0.0.9   RIPv2   66 Response
15 80.93655900( 10.0.3.3    224.0.0.9   RIPv2   66 Response
16 83.67558600( 10.0.3.2    224.0.0.9   RIPv2   86 Response
```

file: lab_data/lab_04/exercise4c_pc3_count_to_infinity_wireshark.pcapng

## Part 5



We unset all the RIP settings and use OSPF now. This includes the PCs and their quagga daemons. For this setup to work we needed to setup ip forwarding for the PCs. After that all hosts and routers were able to communicate via ping and traceroute.

### Exercise 5A

There was nothing to be saved here, but we set up the routers for the OSPF protocol for area 1 and network 10.0.0.0/8. As router id we used the IP address of eth0 as router-id.

### Exercise 5B

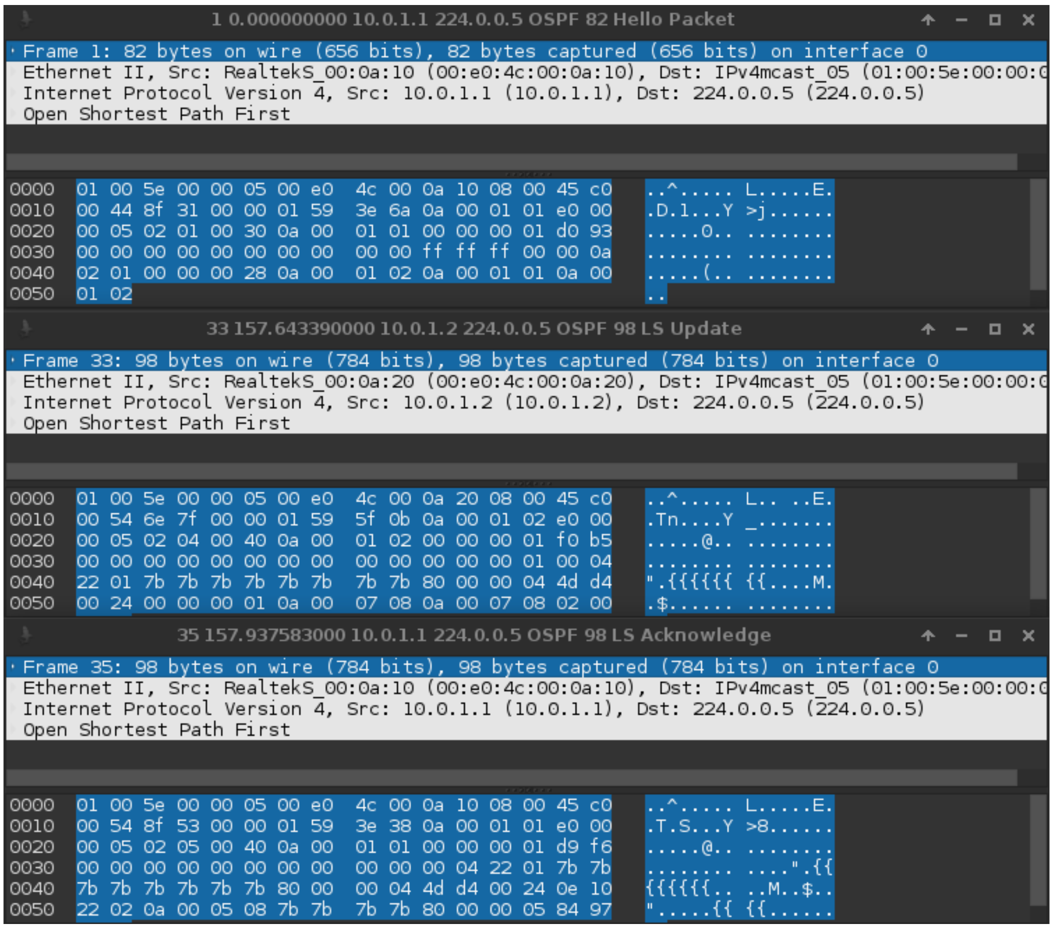
We reconfigured quagga on the PCs for the OSPF protocol. Of course we need to enable ip forwarding for all PCs. Again there is nothing to save here.

### Exercise 5C

Finally we got to the part, where we observe the convergence of the OSPF protocol. In this case we did unplug the cable from PC4 to Router 3. For the lab report we had to save the statistics from the `ping` command:

```
--- 10.0.7.7 ping statistics ---
107 packets transmitted, 62 received, +7 errors, 42% packet loss, time 106012ms
rtt min/avg/max/mdev = 0.785/0.965/1.191/0.143 ms
```

We had to save the different types of captured OSPF packets as well:



```
1 0.000000000 10.0.1.1 224.0.0.5 OSPF 82 Hello Packet
Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
Ethernet II, Src: RealtekS_00:0a:10 (00:e0:4c:00:0a:10), Dst: IPv4mcast_05 (01:00:5e:00:00:0
Internet Protocol Version 4, Src: 10.0.1.1 (10.0.1.1), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First

0000  01 00 5e 00 00 05 00 e0  4c 00 0a 10 08 00 45 c0   ..^..... L.....E.
0010  00 44 8f 31 00 00 01 59  3e 6a 0a 00 01 01 e0 00   .D.1...Y >j......
0020  00 05 02 01 00 30 0a 00  01 01 00 00 00 01 d0 93   .....0.. ........
0030  00 00 00 00 00 00 00 00  00 00 ff ff ff 00 00 0a   ........ ........
0040  02 01 00 00 00 28 0a 00  01 02 0a 00 01 01 0a 00   .....(.. ........
0050  01 02                                              ..

33 157.643390000 10.0.1.2 224.0.0.5 OSPF 98 LS Update
Frame 33: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: RealtekS_00:0a:20 (00:e0:4c:00:0a:20), Dst: IPv4mcast_05 (01:00:5e:00:00:0
Internet Protocol Version 4, Src: 10.0.1.2 (10.0.1.2), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First

0000  01 00 5e 00 00 05 00 e0  4c 00 0a 20 08 00 45 c0   ..^..... L.. ..E.
0010  00 54 6e 7f 00 00 01 59  5f 0b 0a 00 01 02 e0 00   .Tn....Y _.......
0020  00 05 02 04 00 40 0a 00  01 02 00 00 00 01 f0 b5   .....@.. ........
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 04   ........ ........
0040  22 01 7b 7b 7b 7b 7b 7b  7b 7b 80 00 00 04 4d d4   ".{{{{{{ {{....M.
0050  00 24 00 00 00 01 0a 00  07 08 0a 00 07 08 02 00   .$...... ........

35 157.937583000 10.0.1.1 224.0.0.5 OSPF 98 LS Acknowledge
Frame 35: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: RealtekS_00:0a:10 (00:e0:4c:00:0a:10), Dst: IPv4mcast_05 (01:00:5e:00:00:0
Internet Protocol Version 4, Src: 10.0.1.1 (10.0.1.1), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First

0000  01 00 5e 00 00 05 00 e0  4c 00 0a 10 08 00 45 c0   ..^..... L.....E.
0010  00 54 8f 53 00 00 01 59  3e 38 0a 00 01 01 e0 00   .T.S...Y >8......
0020  00 05 02 05 00 40 0a 00  01 01 00 00 00 01 d9 f6   .....@.. ........
0030  00 00 00 00 00 00 00 00  00 00 00 04 22 01 7b 7b   ........ ....".{{
0040  7b 7b 7b 7b 7b 7b 80 00  00 04 4d d4 00 24 0e 10   {{{{{{.. ..M..$..
0050  22 02 0a 00 05 08 7b 7b  7b 7b 80 00 00 05 84 97   ".....{{ {{......
```

In our case we got 3 different kind of packets: “Hello”, “LS Update” and “LS Acknowledge” (wikipedia knows about 11 types[1].

Additionally we had to add the link state database. In this case the output of “show ip ospf database” from PC2(quagga):

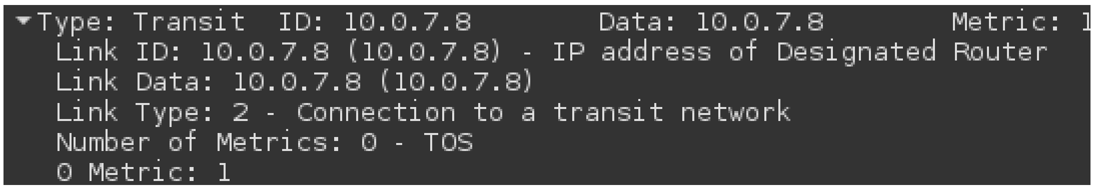
```
ospfd# show ip ospf database
      OSPF Router with ID (10.0.1.2)
              Router Link States (Area 0.0.0.1)
Link ID        ADV Router     Age  Seq#       CkSum  Link count
10.0.1.1        10.0.1.1        778 0x8000000b 0x2aaf 2
10.0.1.2        10.0.1.2        198 0x8000000a 0x11bc 2
10.0.3.4        10.0.3.4        583 0x80000006 0x17ac 2
10.0.4.5        10.0.4.5        582 0x80000003 0x793b 2
10.0.5.1        10.0.5.1       1046 0x80000003 0xfcc5 2
10.0.6.6        10.0.6.6        205 0x80000004 0x2677 2
10.0.6.7        10.0.6.7       1106 0x80000007 0x544a 2
123.123.123.123 123.123.123.123  237 0x80000004 0x4dd4 1

              Net Link States (Area 0.0.0.1)

Link ID        ADV Router     Age  Seq#       CkSum
10.0.1.2        10.0.1.2        778 0x80000002 0x61c4
10.0.2.3        10.0.5.1        753 0x80000004 0xa341
10.0.3.3        10.0.5.1       1000 0x80000003 0xa155
10.0.4.5        10.0.4.5        584 0x80000001 0x945b
10.0.5.6        10.0.6.6        205 0x80000001 0x519a
10.0.6.6        10.0.6.6       1075 0x80000003 0xc915
10.0.7.8        123.123.123.123 1075 0x80000003 0x88a3
```

1https://de.wikipedia.org/wiki/Open_Shortest_Path_First#Link_State_Advertisements_LSA

We chose the LS Update packages we received. In our opinion the important parts here are the “Transit” parts in the packages. These can only be found with the update packages containing the Router LSA (not the Network LSA)



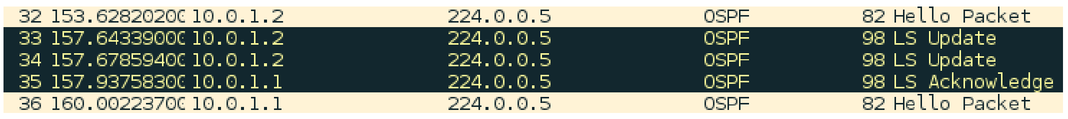
```
▾Type: Transit  ID: 10.0.7.8        Data: 10.0.7.8        Metric: 1
   Link ID: 10.0.7.8 (10.0.7.8) - IP address of Designated Router
   Link Data: 10.0.7.8 (10.0.7.8)
   Link Type: 2 - Connection to a transit network
   Number of Metrics: 0 - TOS
   0 Metric: 1
```

The names for the Options and their values are self explanatory.

Following are the disired answers to the questions of the 5th step:

## How quickly are OSPF messages sent after the cable is disconnected?

We had the impression that it happened almost immediately. Looking at the packages we can see, that every 3-4 seconds an packet (mostly “hello”) will be send. The same pattern was to be seen with the LSUs:



```
32 153.6282020C 10.0.1.2      224.0.0.5     OSPF      82 Hello Packet
33 157.6433900C 10.0.1.2      224.0.0.5     OSPF      98 LS Update
34 157.6785940C 10.0.1.2      224.0.0.5     OSPF      98 LS Update
35 157.9375830C 10.0.1.1      224.0.0.5     OSPF      98 LS Acknowledge
36 160.0022370C 10.0.1.1      224.0.0.5     OSPF      82 Hello Packet
```

## How many OSPF messages are sent?

We suppose that are only the non “Hello” packets where meant, because else it would just rely on our speed of pressing the stop button.

In our case we had 3 ACKs and 6 LSUs.

## Which type of OSPF packet is used for flooding link state information?

Most definetely this is the LS Update package.

## Describe the flooding of LSAs to all routers.

Each router sends forth the packages it gets to all subrouters. Important to know is, that in the case of OSPF, we have a tree dedicated to the forwarding to other routers/hosts. The tree is build in a way similar to the Dijkstra-Algorithm[2]. This way each router gets it’s messages only from one location, while giving it to all locations “below” it.

2https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Which type of encapsulation is used for OSPF packets (TCP, UDP, etc.)?

The OSPF packets are only encapsulated within the IP packet.

What is the destination address of OSPF packets?

The destination is always 224.0.0.5.

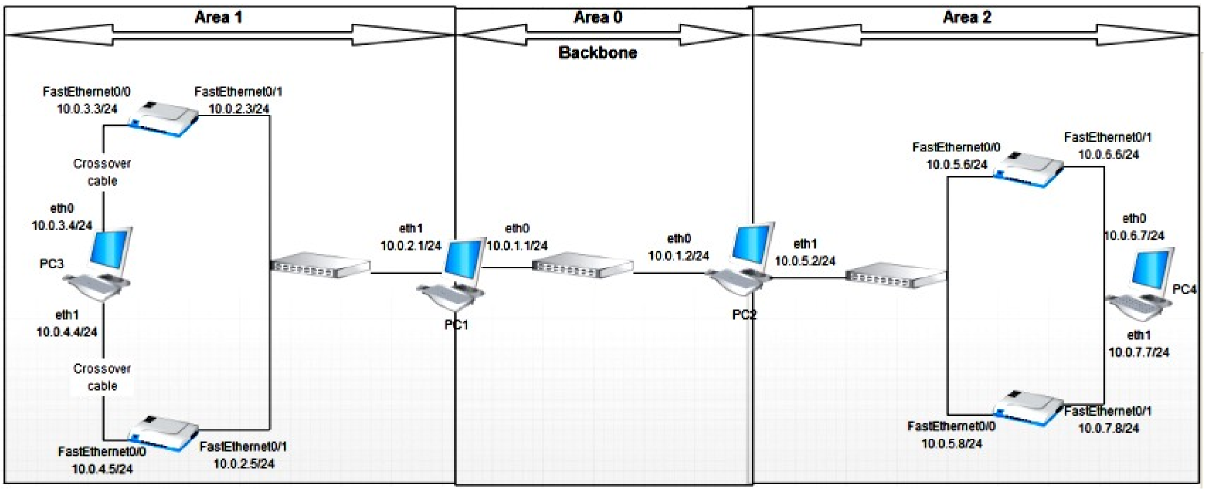
Count the number of lost packets and calculate the time it took OSPF to update the routing tables.

It took our setup ~43s (=packets) to update the routing tables.

Can you confirm, that the link state databases are indentical?

In our case each database only were different in the age of the entries, which is a very good indicator, for identical databases.

## Part 6



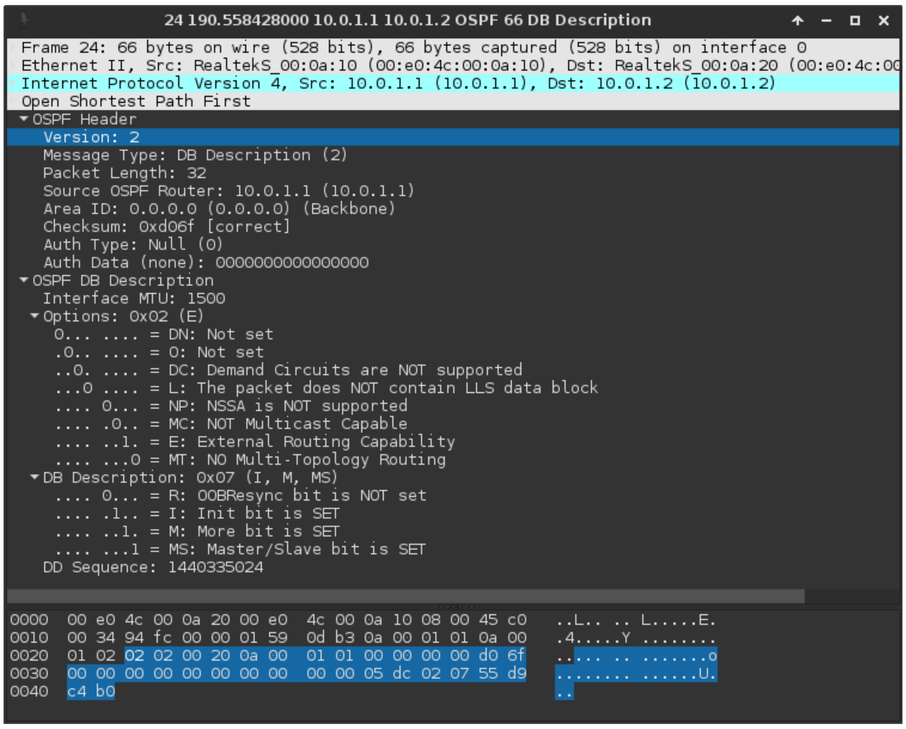
### Exercise 6

To reset the quagga settings, we made, we restarted the quagga services. For the new configuration we divide the network in the 3 areas 0, 1 and 2. The border hosts PC1 and PC2 are configured as ospf routers belonging to the areas 0/1 (PC1) and 0/2 (PC2).

**Include the wireshark output in your report showing, if any, the different types of OSPF packets that you did not observe in part 5.**

We could only find one essential packet which is different:

The DB Description file.



```
24 190.558428000 10.0.1.1 10.0.1.2 OSPF 66 DB Description
Frame 24: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: RealtekS_00:0a:10 (00:e0:4c:00:0a:10), Dst: RealtekS_00:0a:20 (00:e0:4c:00
Internet Protocol Version 4, Src: 10.0.1.1 (10.0.1.1), Dst: 10.0.1.2 (10.0.1.2)
Open Shortest Path First
 OSPF Header
   Version: 2
   Message Type: DB Description (2)
   Packet Length: 32
   Source OSPF Router: 10.0.1.1 (10.0.1.1)
   Area ID: 0.0.0.0 (0.0.0.0) (Backbone)
   Checksum: 0xd06f [correct]
   Auth Type: Null (0)
   Auth Data (none): 0000000000000000
 OSPF DB Description
   Interface MTU: 1500
  Options: 0x02 (E)
    0... .... = DN: Not set
    .0.. .... = O: Not set
    ..0. .... = DC: Demand Circuits are NOT supported
    ...0 .... = L: The packet does NOT contain LLS data block
    .... 0... = NP: NSSA is NOT supported
    .... .0.. = MC: NOT Multicast Capable
    .... ..1. = E: External Routing Capability
    .... ...0 = MT: NO Multi-Topology Routing
  DB Description: 0x07 (I, M, MS)
    .... 0... = R: OOBResync bit is NOT set
    .... .1.. = I: Init bit is SET
    .... ..1. = M: More bit is SET
    .... ...1 = MS: Master/Slave bit is SET
   DD Sequence: 1440335024

0000  00 e0 4c 00 0a 20 00 e0  4c 00 0a 10 08 00 45 c0   ..L.. .. L.....E.
0010  00 34 94 fc 00 00 01 59  0d b3 0a 00 01 01 0a 00   .4.....Y ........
0020  01 02 02 02 00 20 0a 00  01 01 00 00 00 00 d0 6f   ...... .. .......o
0030  00 00 00 00 00 00 00 00  00 00 05 dc 02 07 55 d9   ........ ......U.
0040  c4 b0                                              ..
```

Include the output of the link state databases (saved in step 5).

We included all reports from step 5, but decided against including them all here, because it wouldn’t be much easier to read. The files can be found in the corresponding folder to lab04.

But essenital is the part, where PC1 and PC2 have their databases sorted by area from 0 to 2.

Due to the similarity we include here just the report from PC2:

```
ospfd# show ip ospf database
        OSPF Router with ID (10.0.1.2)


        Router Link States (Area 0.0.0.0)
Link ID         ADV Router      Age  Seq#       CkSum  Link count
10.0.1.1        10.0.1.1        248 0x8000000a 0x4fbf 1
10.0.1.2        10.0.1.2        253 0x8000000a 0x4dbe 1


        Net Link States (Area 0.0.0.0)
Link ID         ADV Router      Age  Seq#       CkSum
10.0.1.2        10.0.1.2        253 0x80000001 0x63c3


        Summary Link States (Area 0.0.0.0)
Link ID         ADV Router      Age  Seq#       CkSum  Route
10.0.2.0        10.0.1.1        1411 0x80000002 0xcd6c 10.0.2.0/24
10.0.3.0        10.0.1.1        383 0x80000002 0xcc6b 10.0.3.0/24
10.0.4.0        10.0.1.1        227 0x80000003 0xbf76 10.0.4.0/24
10.0.5.0        10.0.1.2        432 0x80000002 0xa68f 10.0.5.0/24
10.0.6.0        10.0.1.2        276 0x80000001 0x0c1f 10.0.6.0/24
10.0.7.0        10.0.1.2        276 0x80000002 0x9a98 10.0.7.0/24


        Router Link States (Area 0.0.0.2)
Link ID         ADV Router      Age  Seq#       CkSum  Link count
10.0.1.2        10.0.1.2        283 0x80000007 0xd32f 1
10.0.5.6        10.0.5.6        433 0x80000002 0x06d7 1
10.0.5.7        10.0.5.7        284 0x80000005 0x08b7 2
10.0.5.8        10.0.5.8        277 0x80000004 0x880c 2


        Net Link States (Area 0.0.0.2)
Link ID         ADV Router      Age  Seq#       CkSum
10.0.5.6        10.0.5.6        284 0x80000004 0xcd02
10.0.7.7        10.0.5.7        284 0x80000001 0x9d61


        Summary Link States (Area 0.0.0.2)
Link ID         ADV Router      Age  Seq#       CkSum  Route
10.0.1.0        10.0.1.2        433 0x80000001 0xd466 10.0.1.0/24
10.0.2.0        10.0.1.2        238 0x80000001 0x2e02 10.0.2.0/24
10.0.3.0        10.0.1.2        238 0x80000001 0x2d01 10.0.3.0/24
10.0.4.0        10.0.1.2        226 0x80000002 0x200c 10.0.4.0/24
```

Include the answers to the questions from step 4. Refer to the saved link state databases in your answers.

There were no questions in step 4. We suppose we need to answer the question from step 5 “Which differences do you note?” (in comparison to E5).

The main difference is that we have multiple areas, but also the new part of “Summaray Link states” for each area.

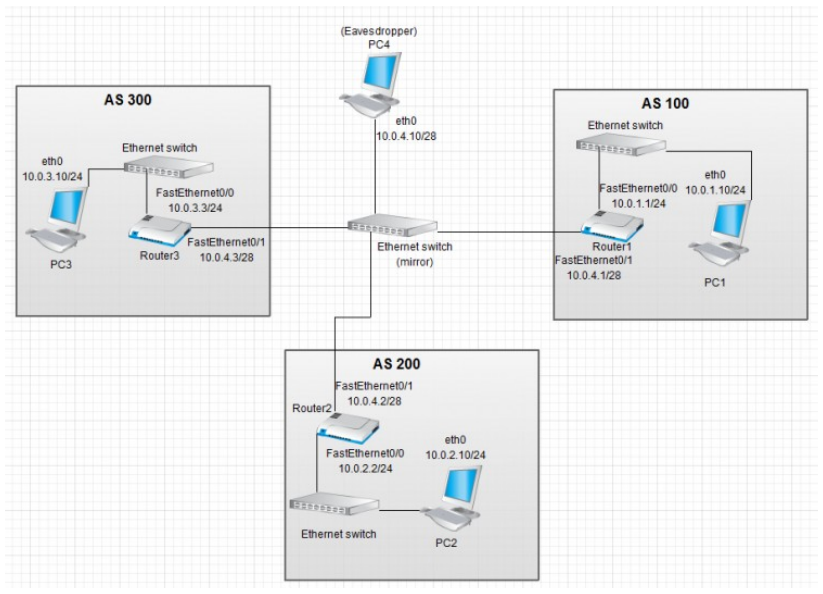
**Explain the output of the command `show ip ospf border-routers` from step 5.**

It is conveniant, that the output is rather self explanatory. Even the indicator of i or I for intra/inter routings is useful. Additionally we get the information, where the Area 1 is and how we can get there. Not only by the IP address, but also via which interface.

```
Router1#show ip ospf border-routers
OSPF Process 1 internal Routing Table
Codes: i - Intra-area route, I - Inter-area route
i 10.0.1.1 [1] via 10.0.2.1, FastEthernet0/1, ABR, Area 1, SPF 15
```

Thought the output is very nice to read, we wondert, why the Area 2 is missing. Our solution was, that area 0 might act kind of like containing area 1/2 and is responsible for the further connection. This way the routing tables won’t be flooded with information about networks, we might never want to reach.

## Part 7



### Exercise 7A

After we tested out RIP and OSPF, we're now about to see how BGP behaves in networks. For this, we adjusted the routers 1-3 for "routing" to the internet and all the remote-as (other automated systems, like AS100-300).

show ip route

```
Gateway of last resort is not set
        10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
B       10.0.2.0/24 [20/0] via 10.0.4.2, 00:02:30
B       10.0.3.0/24 [20/0] via 10.0.4.3, 00:02:30
C       10.0.1.0/24 is directly connected, FastEthernet0/0
C       10.0.4.0/28 is directly connected, FastEthernet0/1
```

show ip bgp

```
  Network       Next Hop        Metric LocPrf Weight Path
*> 10.0.1.0/24  0.0.0.0   0     32768              i
*  10.0.2.0/24  10.0.4.2 0     300      200        i
*>              10.0.4.2 0     0        200        i
*  10.0.3.0/24  10.0.4.3 0     200      300        i
*>              10.0.4.3 0     0        300        i
```

show ip bgp paths

```
Address  Hash Refcount Metric Path
0x44A398F0    0       5       0               i
0x44A395BC    1       2       0               i
0x44A39660 1510       1       0 300 200       i
0x44A39704 1510       1       0 200 300       i
0x44A3984C 1737       2       0 200           i
0x44A397A8 1837       2       0 300 i
```

Describe the different types of BGP messages you observe in the wireshark window on PC.

- **OPEN**

  *This will be sent once at the beginning of a connection and needs to be answered with a KEEPALIVE message. It contains at least BGP Version, AS number, the hold timer and the bgp identifier.*

- **KEEPALIVE**

  *This acknowledges the* **OPEN** *message. This will be send regularly, to see if the connected routers are still online. Important is the "hold time" given by the* **OPEN** *message, because it tells* ***BGP*** *how long it waits for the* **KEEPALIVE** *message.*

- **UPDATE**

  *As the name suggests, this contains a path-change. Each* **UPDATE** *message can contain multiple paths to be added and/or to be removed.*

- **NOTIFICATION**

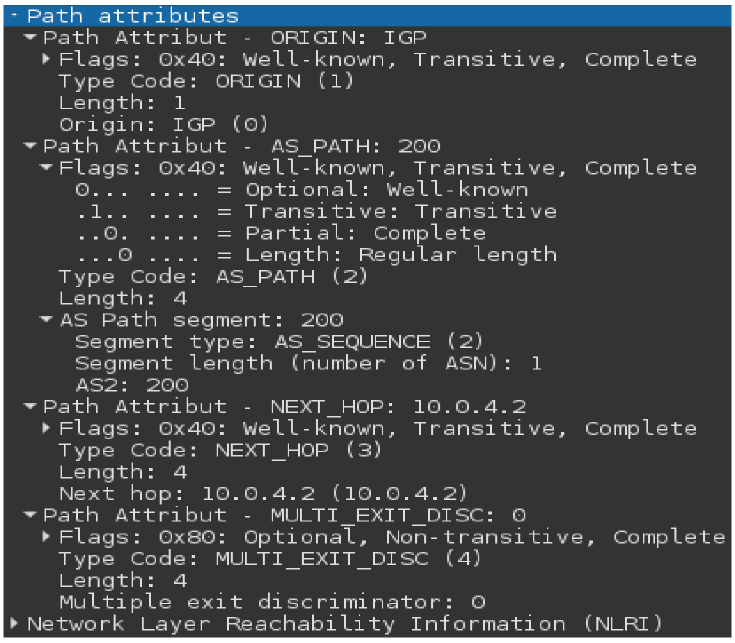
  *Via the* **NOTIFICATION** *we get a message which contains error and statuscodes. All paths we (as router) got over this connection are now to be removed. Via BGP* **UPDATE** *now we spread the message about the lost connections.*

- (TCP Messages)

  TCP Messages is in brackets because they are mostly just 60byte packets containing FIN/ACKs.

Use the saved output to provide a brief explanation of how the routers find the proper path between the autonomous systems.

Essentially we get information about how to get where and via who. Additionally we get the information about the Location and which AS number.



```
- Path attributes
  ▾Path Attribut - ORIGIN: IGP
    ▸Flags: 0x40: Well-known, Transitive, Complete
     Type Code: ORIGIN (1)
     Length: 1
     Origin: IGP (0)
  ▾Path Attribut - AS_PATH: 200
    ▾Flags: 0x40: Well-known, Transitive, Complete
       0... .... = Optional: Well-known
       .1.. .... = Transitive: Transitive
       ..0. .... = Partial: Complete
       ...0 .... = Length: Regular length
     Type Code: AS_PATH (2)
     Length: 4
    ▾AS Path segment: 200
       Segment type: AS_SEQUENCE (2)
       Segment length (number of ASN): 1
       AS2: 200
  ▾Path Attribut - NEXT_HOP: 10.0.4.2
    ▸Flags: 0x40: Well-known, Transitive, Complete
     Type Code: NEXT_HOP (3)
     Length: 4
     Next hop: 10.0.4.2 (10.0.4.2)
  ▾Path Attribut - MULTI_EXIT_DISC: 0
    ▸Flags: 0x80: Optional, Non-transitive, Complete
     Type Code: MULTI_EXIT_DISC (4)
     Length: 4
     Multiple exit discriminator: 0
▸Network Layer Reachability Information (NLRI)
```

# Lab 5

## Pre-Module Questions

### What do the protocols covered in this lab use port numbers for?

They're used to establish connections. Each port number can transmit messages between two devices. Multiple connections are possible with the port approach.

### Provide the syntax of the nttcp command for both the sender and receiver, which executes the following scenario:

*A TCP server has IP address 10.0.2.6 and a TCP client has IP address 10.0.2.7,. The TCP server is waiting on port number 2222 for a connection request. The client connects to the server and transmits 2000 bytes to the server, which are sent as four write operations of 500 bytes each.*

Local: nttcp -l 500 -x 2000 -p 2222 10.0.2.7

Remote: nttcp -r -i

(This way the remote/passive client will get the information by the sending-part.)

### Answer the following questions on Path MTU Discovery:

***1. How does TCP decide the maximum size of a TCP segment?***

***2. How does UDP decide the maximum size of a UDP datagram?***

***3. What is the ICMP error generated by a router when it needs to fragment a datagram with the DF bit set? Is the MTU of the interface that caused the fragmentation also returned?***

***4. Explain why a TCP connection over an Ethernet segment never runs into problems with fragmentation.***

**to 1)** It takes the largest IP Datagram, the host can handle - (IP and TCP header sizes combined)

**to 2)** It’s a given length by the IPv4/v6 protocol. It’s maximum 65.507 (which comes from $2^{16}$-8bytes UDP Header and minus 20 IP Header). And it’s 65527 with ipv6.

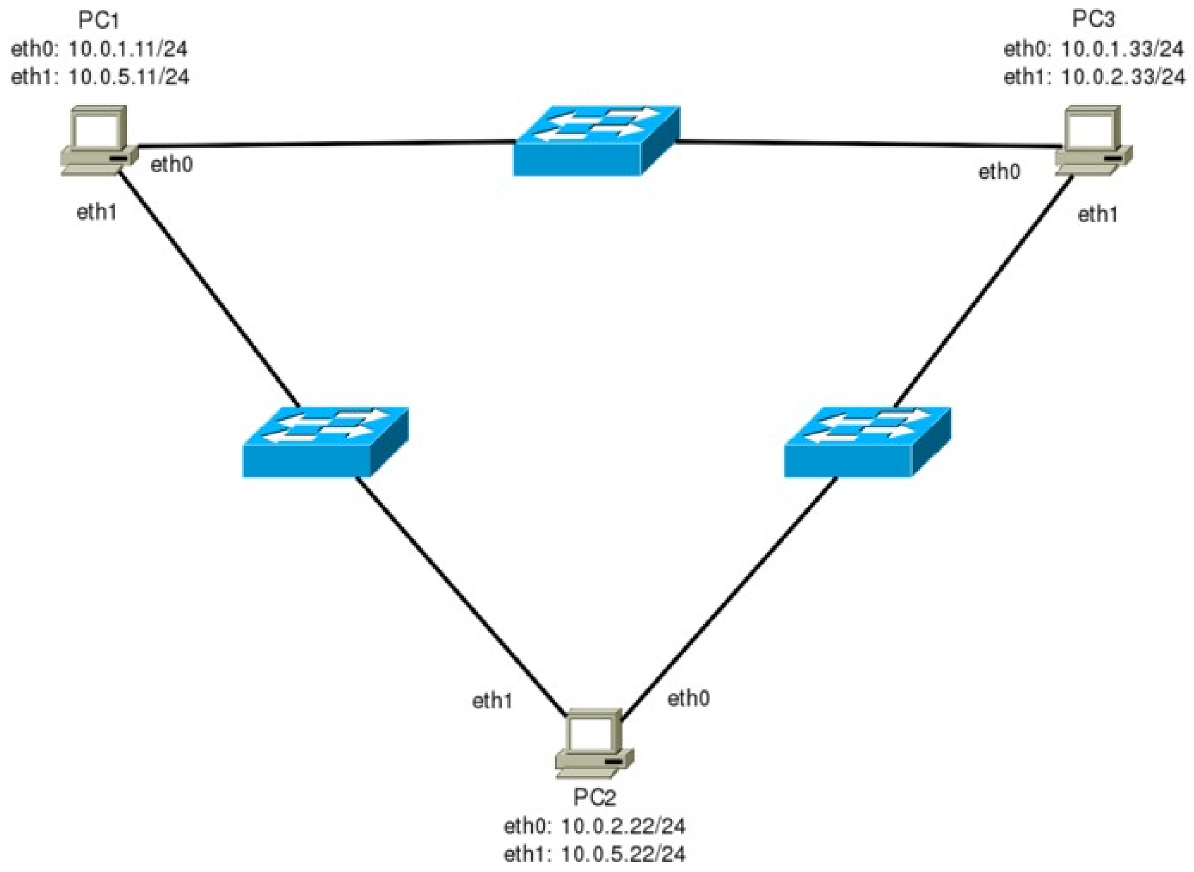
**to 3)** It contains the code 4 which stands for “Fragmentation need, but DF bit is set”. Of couse DF means “Don’t Fragment”.

**to 4)** We get immediate acknowledgement (ACK) from the receiver everytime and for every frame. Problematic frames will be send again.

## Assume a TCP sender receives an ACK in which the acknowledgment number is set to 34567 and the window size is set to 2048. Which sequence numbers can the sender transmit?

34567+2048 Are acknowledged. So all packets after that. This means 36616 and following.

## Exercise 1



In the first exercise we did setup the network as shown above. IP forwarding from on all PCs but PC3 should be disabled and so we did it.
Finally we set PC3 as default gateway for PC1 and PC2. We veriefied the connection with ping and went on to:

## Exercise 2

For this exercise we captures packages on eth1 of PC1. we did only capture packets, that have 10.0.5.22 (PC2-eth1) as source or destination. Additionally we set the another capture on PC2-eth1.

After the setup we executed:

```
PC2$ nttcp -r -s -l 1024 -n 10 -p 4444 -i -u
```

```
PC1$ nttcp -t -s -l 1024 -n 10 -p 4444 -u 10.0.5.22
```

The commands containing following options:

**nttcp**

| | |
|---|---|
| **-r** | *Set this for the receiver* |
| **-t** | *Set this for the transmitter* |
| **-s** | *This is used to force a pattern by which UDP packets can be compared* |
| **-l** | *Length of the buffer* |
| **-n** | *Number of buffers* |
| **-p** | *Specifies the Port for the first TCP connection* |
| **-i** | *Sets nttcp for a receiver so that it behaves as a daemon* |
| **-u** | *Use UDP packets* |
| **10.0.5.22** | *The destination (for transmitter)* |

| | sending | receiving |
|---|---|---|
| total | 10752 | 10780 |
| payload | 10248 | 10344 |
| ratio | 0,95…* | 0,96…* |
| biggest datagram total | 1066 | 1066 |
| biggest datagram payload | 1024 | 1032 |
| biggest datagram ratio | 0,96... | 0,97... |

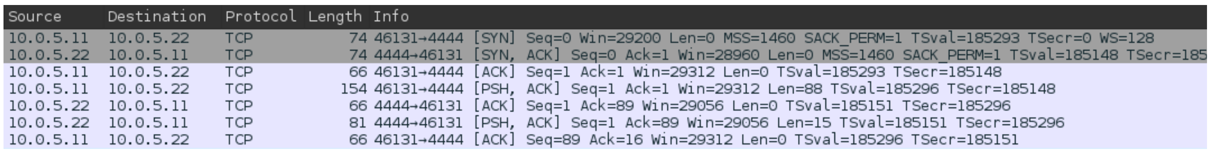
*(*exact ratio in lab folder)*

Have a look at the UDP headers. Which header fields change between transmission and is there any difference in the UDP headers sent by PC1 and received by PC2?

The length and the checksum changes. The checksum changes for each packet for for source and destination. We think it's because PC3 is reconfiguring the transmission length.

How did nttcp choose the source port number? Support your answers with excerpts from your captured data.

The passive/receiving partner tells via TCP (even if the testing protocol is UDP) to the sending/active partner which port is free. This is the Port PC1 is using here.



```
Source      Destination  Protocol Length Info
10.0.5.11   10.0.5.22    TCP          74 46131→4444 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=185293 TSecr=0 WS=128
10.0.5.22   10.0.5.11    TCP          74 4444→46131 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=185148 TSecr=185
10.0.5.11   10.0.5.22    TCP          66 46131→4444 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=185293 TSecr=185148
10.0.5.11   10.0.5.22    TCP         154 46131→4444 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=88 TSval=185296 TSecr=185148
10.0.5.22   10.0.5.11    TCP          66 4444→46131 [ACK] Seq=1 Ack=89 Win=29056 Len=0 TSval=185151 TSecr=185296
10.0.5.22   10.0.5.11    TCP          81 4444→46131 [PSH, ACK] Seq=1 Ack=89 Win=29056 Len=15 TSval=185151 TSecr=185296
10.0.5.11   10.0.5.22    TCP          66 46131→4444 [ACK] Seq=89 Ack=16 Win=29312 Len=0 TSval=185296 TSecr=185151
```

## Exercise 3

In change to the setup in Exercise 2, this time we did set it up, so that it uses not UDP but TCP.

### Count the number of packets transferred in the whole exercise, divide this into packets sent by PC1 or PC2.

PC1: has 20. Because it requests ACKS

PC2: has 18. Because he’s just listening

(We have two additional ARP requests)

### Count the number of packet transferred for each TCP segment.

PC1 sends 1024 bytes in 10 buffers, but the TCP/IP protocol (including fragmentation) makes 18 out of it.

### Count the number of packets which do not carry a payload.

21 packets in whole have just the minimum size of 66 bytes, which means they just contain the headers.

### What are the sizes of the different TCP segments you observe?

66, 74, 81, 151, 1090

## What is the ratio of transmitted application data to overhead data, including all headers down to Ethernet? Also, take the biggest TCP datagram and compare its payload to its total size. Compare this to the analog ratio you measured in the last exercise.

When we take the largest packages we have following values:

| | | | |
|---|---|---|---|
| total: | 1090bytes containing of | | |
| data: | 1024bytes with | | |
| headers: | 66bytes | (0.06% of total; 0.0644 ratio to data) | |
| - | TCP | 32bytes | (0.48 of headers) |
| - | IP | 20bytes | (0.30 of headers) |
| - | Ethernet | 14bytes | (0.22 of headers) |

So the TCP packets need more space than UDP, which is obvious, as we need to carry more information (Sequence numbers etc.) and additionally have a more complex checksum.

## What different TCP flags do you observe? Explain what they are used for and their effect in this context.

There’re 9 different TCP flags:

| | |
|---|---|
| **NS*** | - *ECN-nonce concealment protection* |
| **CWR** | - *Congestion Window reduced (for congestion control mechanisms)* |
| **ECE** | - *Dual role: depends on* **SYN***-flag. syn==1: TCP peer is* **ECN** *capable. syn==0: packet with congestion experienced flag in IP haeder received during normal transmission.* |
| **URG** | - *Urgent pointer field is important* |
| **ACK** | - *Indicates that we have to look at the acknowledgement field.* |
| **PSH** | - *Push function. Asks for pushing the buffered data to the receiving application.* |
| **RST** | - *Reset the connection.* |
| **SYN** | - *Synchronize the sequence numbers. The essential part for our TCP Connections.* |
| **FIN** | - *No more data from server.* |

*(source: https://en.wikipedia.org/wiki/Transmission_Control_Protocol)*

**(didn’t observe)*

## Exercise 4

In this exercise we capture and observe the differences between FTP (TCP based) and TFTP(UDP based). For this we create a file filed with randomized values, transmit it and observe from there on.

### How long did it take for each of the transfers to complete? Explain why, using the capturedn data.

| | | |
|---|---|---|
| TFTP: | 10567318 bytes in 4.3s | (2399,92kbyte/s) |
| FTP: | 10485760 bytes in 0.89s | (11505,62kbyte/s) |

The speed and data difference is obvious easily to explain in the nature of the udp packet sizes: 558 with TFTP compared to 1514 with FTP. Additionally TFTP uses it's own kind of ACK for acknowledging the packages.

### How many TCP connections were created for the FTP session and why?

2 Main connections: One for the FTP connection itself and one for the FTP Data. This used even another port.

## Exercise 5

The purpose of this exercise is to find the maximum Bufferlenght until a fragmentation occurs. We're using nttcp again and of course the wireshark captures.

The first border ist the MTU of PC3 (which is used to transport packages over it's interfaces). In our case it's at 1500. The packagse weren't possible to send anymore after 505676. Which seemed rather random to us, as we expected that it would be possible to send till the full 16bits for the offset would be used. In our case this was even reproducable (not exactly after 505676, but at different values around 500000 and 510000).

We have several interesting parts of the headers. For the TCP protocol we're using the **more fragments** bit. To see which fragment we're getting we need the Fragment offset and the information where (in which frame) we can find the reassembled packet.

## Exercise 6

We changed the MTU values for the PCs:

**PC1:** eth0.mtu == 1500
**PC2:** eth0.mtu == 500
**PC3:** eth0.mtu == 1500; eth1.mtu == 1500

After that we recheck the fragmentation and inspect the packages. Then we change the MTU PC2 to 1500 and of PC3 to 500 (Case 2).

### Case 1

#### Where do you observe fragmentation, if any? Explain!

We did not see any fragmentation. As the traffic arrives at PC2 last, there is no PC which can split the traffic in smaller packages. The MTU of PC1 and PC3 are the same (at this stage).

#### Do you observe any ICMP error message? Why?

ICMP error messages didn't appear because there was not reason for it.

#### Check the DF flag.

DF flag was set.

## Case 2

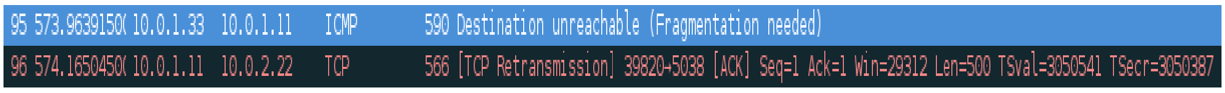
### Where do you observe fragmentation, if any? Explain!

We did not get any fragmentation, but (see next Question). Additionally we set the DF flag.

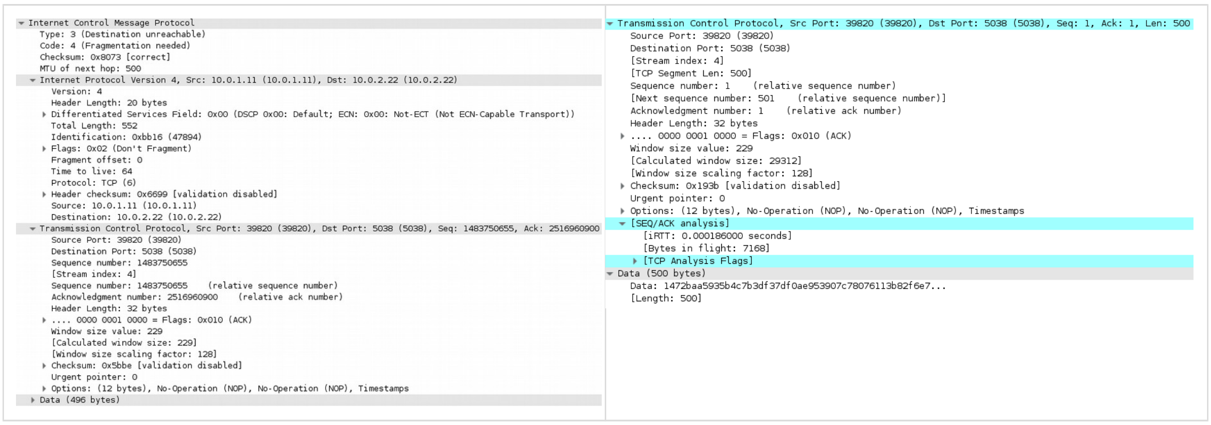
### Do you observe any ICMP error message? What are they used for?

ICMP gave us the message “Destination unreachable (Fragmentation needed). This is due to the reason that DF flag was set and also the MTU of PC3 was set to 500 bytes. This way the 1090byte large packets can’t arrive.

### If any ICMP error messages were captured, include at least one and the following TCP segment in the report.



```
95 573.9639150( 10.0.1.33  10.0.1.11   ICMP   590 Destination unreachable (Fragmentation needed)
96 574.1650450( 10.0.1.11  10.0.2.22   TCP    566 [TCP Retransmission] 39820→5038 [ACK] Seq=1 Ack=1 Win=29312 Len=500 TSval=3050541 TSecr=3050387
```



```
▾ Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 4 (Fragmentation needed)
    Checksum: 0x8073 [correct]
    MTU of next hop: 500
  ▾ Internet Protocol Version 4, Src: 10.0.1.11 (10.0.1.11), Dst: 10.0.2.22 (10.0.2.22)
      Version: 4
      Header Length: 20 bytes
    ▸ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
      Total Length: 552
      Identification: 0xbb16 (47894)
    ▸ Flags: 0x02 (Don't Fragment)
      Fragment offset: 0
      Time to live: 64
      Protocol: TCP (6)
    ▸ Header checksum: 0x6699 [validation disabled]
      Source: 10.0.1.11 (10.0.1.11)
      Destination: 10.0.2.22 (10.0.2.22)
  ▾ Transmission Control Protocol, Src Port: 39820 (39820), Dst Port: 5038 (5038), Seq: 1483750655, Ack: 2516960900
      Source Port: 39820 (39820)
      Destination Port: 5038 (5038)
      Sequence number: 1483750655
      [Stream index: 4]
      Sequence number: 1483750655    (relative sequence number)
      Acknowledgment number: 2516960900    (relative ack number)
      Header Length: 32 bytes
    ▸ .... 0000 0001 0000 = Flags: 0x010 (ACK)
      Window size value: 229
      [Calculated window size: 229]
      [Window size scaling factor: 128]
    ▸ Checksum: 0x5bbe [validation disabled]
      Urgent pointer: 0
    ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ Data (496 bytes)
```

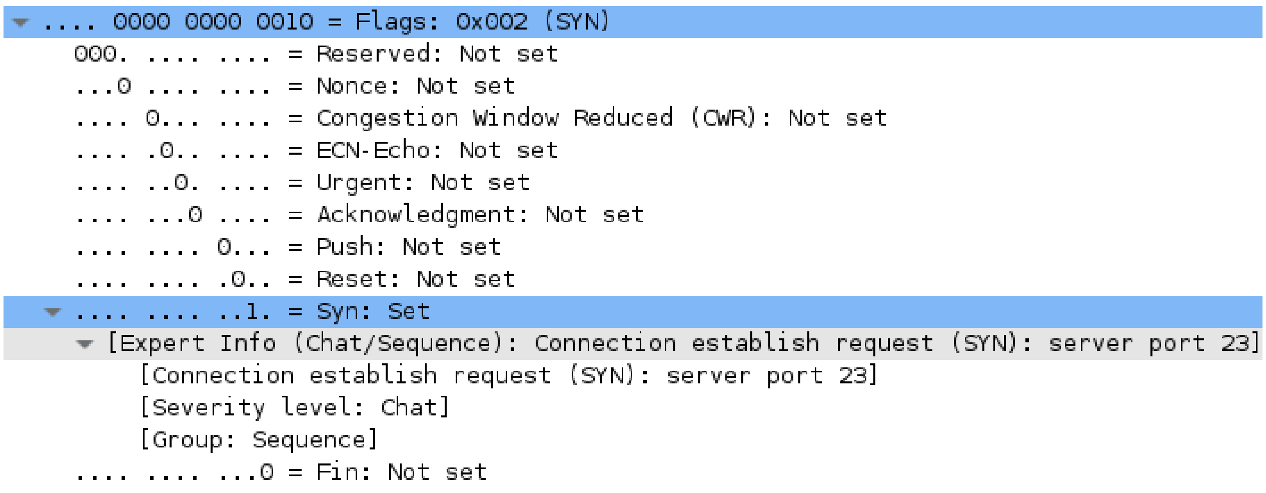
```
▾ Transmission Control Protocol, Src Port: 39820 (39820), Dst Port: 5038 (5038), Seq: 1, Ack: 1, Len: 500
    Source Port: 39820 (39820)
    Destination Port: 5038 (5038)
    [Stream index: 4]
    [TCP Segment Len: 500]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 501    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 32 bytes
  ▸ .... 0000 0001 0000 = Flags: 0x010 (ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
  ▸ Checksum: 0x193b [validation disabled]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▾ [SEQ/ACK analysis]
      [iRTT: 0.000186000 seconds]
      [Bytes in flight: 7168]
    ▸ [TCP Analysis Flags]
▾ Data (500 bytes)
    Data: 1472baa5935b4c7b3df37df0ae953907c78076113b82f6e7...
    [Length: 500]
```

## Exercise 7

As first step we resetted the MTU values on all interfaces and set up a wireshark capture on eth0/1 for PC1 and PC2 to analyse a telnet connection we started but close via telnet command. We reconnected until PC2 automatically closes the telnet session.

### First telnet Session

**What are the packets of the handshake? Which flag is set in each of them?**



```
▾ .... 0000 0000 0010 = Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  ▾ .... .... ..1. = Syn: Set
    ▾ [Expert Info (Chat/Sequence): Connection establish request (SYN): server port 23]
        [Connection establish request (SYN): server port 23]
        [Severity level: Chat]
        [Group: Sequence]
    .... .... ...0 = Fin: Not set
```

The SYN-flag was set in all of the handshakes. Interesting here is that Wireshark gives us more Information based on the context.

For each flag explain how a server or client interprets it.

We already described the TCP flags in Exercise 3.

Determine the initial SEQ numbers.

0, 1, 13, 28, 52, 100, ...

What is the sequence number of the TCP segment that contains data and is transmitted from PC1 to PC2?

1

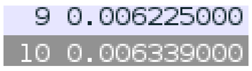
What are the initial window sizes that are exchanged?

229

What is the MSS value exchanged between the TCP client and the TCP server?

1460

How long does it take for the TCP connection to be established?



```
 9 0.006225000
10 0.006339000
```

6.339 x $10^3$s

What are the packets used for connection termination? Which flag is set in each of them?

FIN is always set. In our case we This happened in 19/20 and 40/41.

### Second telnet Session

#### How does this differ from the other connection termination?

In the second case the PC2 is terminating the connection after about 60s.

#### How long is the timeout before the telnet server closes the connection?

60s

## Exercise 8

For this we just had to add a static ARP cache entry to PC1, that associates 10.0.5.100 to 00:18:8b:00:0a:11 on eth1:

### What is the pattern in which the client tries to connect?

1,2,4,8 seconds and doubling the time until 32s are reached then it drops the connection

### Does the TCP client try to reset the connection once it gives up?

No reset is happening.

### In which configuration would the ARP ache entry be unnecessary?

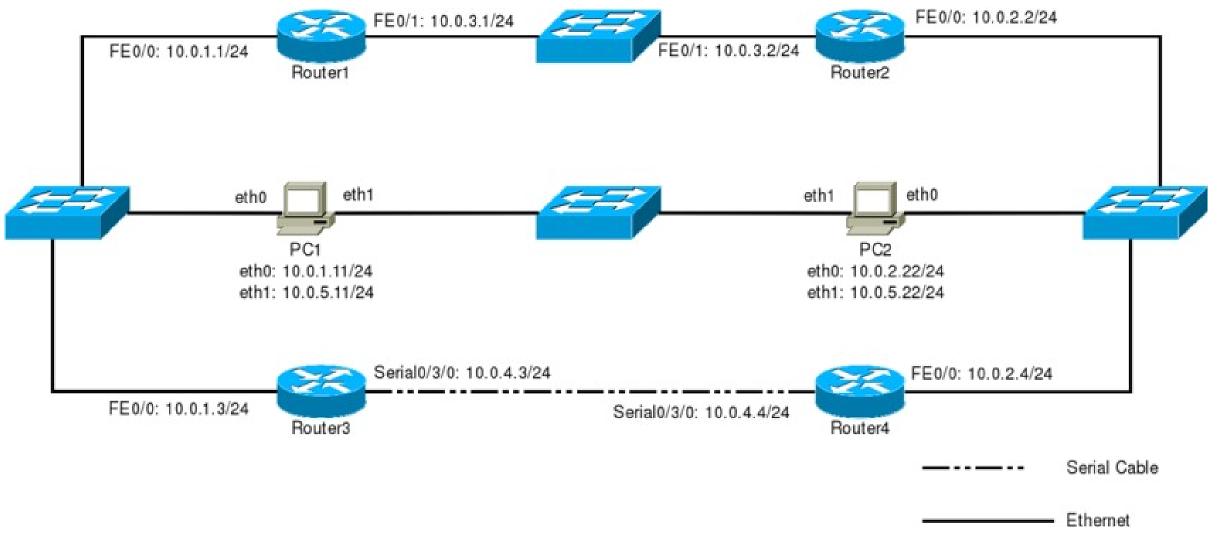
With a default gateway the pc1 would try to get to 10.0.5.22 through the default gateway we specified.

## Exercise 9

We captchured just the first connection attempt by pc1 and the refuse from pc2. So we captured first the tcp flag SYN by pc1 and then the RST, ACK by PC2.

It took about 11 ms.

## Advanced TCP



## Pre-Module Question

### Describe the following heuristics used in TCP and explain why they are used: Nagle’s algorithm, Karn’s algorithm.

***Nagle’s algorithm*** is used to stop the hosts from sending to many little packets, but instead send less packets with more information.

It’s pretty simple and uses following principle: If a package is full, send it. If it’s not full, send it only if no other un-acknowledged packages are on the way.

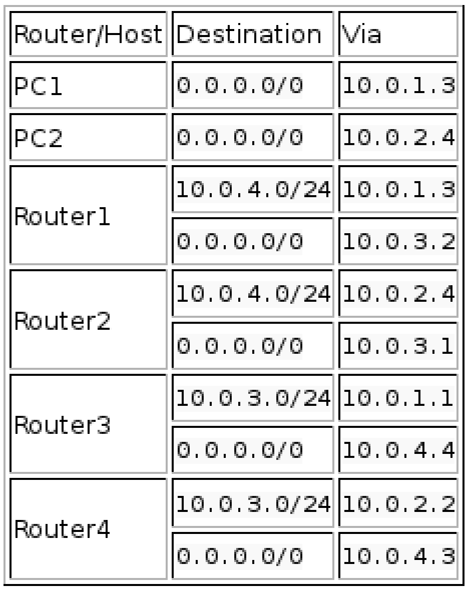
***Karn’s algorithm*** is essentially about getting accurate estimates of the round-trip time (RTT). For this it ignores retransmitted and ambiguous segments in general segments. The actual estimation is a smoothed value based on previous RTT and the new arrived segments.

Both algorithms main purpose is to handle congestion in a better way than it would be possible with the plain Ethernet/IP and even TCP communication.

## Exercise 1

Here we just set up the network like in the picture shown. Additionally we learn how to connect the routers via serial-cable and use it as a network connection.

We set the routing information like in the table shown right here ⇒



| Router/Host | Destination | Via |
|---|---|---|
| PC1 | 0.0.0.0/0 | 10.0.1.3 |
| PC2 | 0.0.0.0/0 | 10.0.2.4 |
| Router1 | 10.0.4.0/24 | 10.0.1.3 |
| | 0.0.0.0/0 | 10.0.3.2 |
| Router2 | 10.0.4.0/24 | 10.0.2.4 |
| | 0.0.0.0/0 | 10.0.3.1 |
| Router3 | 10.0.3.0/24 | 10.0.1.1 |
| | 0.0.0.0/0 | 10.0.4.4 |
| Router4 | 10.0.3.0/24 | 10.0.2.2 |
| | 0.0.0.0/0 | 10.0.4.3 |

## Exercise 2

Here we check the typing of characters in a telnet connection.

### How many packets are exchanged for each character?

Each character gets send 2 times. The first time

### Why does the client delay the ACK for a character echo and how long is this delay?

In Exercise 2 we just got a delay of less than 0.1s and each character was directly confirmed.

### What is the time delay associated with the transmission of ACKs from the telnet server on PC2?

The time delay is changing constantly. And of course none of the packages contain the value as it is a calculated value. It should be possible to determine this via *Karns algorithm*.

### Which TCP flags are set in segments that are used for transmitting keystrokes?

ACK and PSH = 0000 0001 1000

### Why do segments that have an empty payload carry a sequence number? Why does this not result in confusion at the TCP receiver?

They may have no payload, but still carry information in the headers. There're different reasons for this. It can be just for KEEPALIVE or as ACK for previous packages.

### How does the value of the window size progress as you type?

It's changing between 227 and 229.

### Explain what has changed as you increased your typing speed

It packed multiple characters in one package. these were the packages with the window size of 229.

## Exercise 3

The setup and procedure is the same as in Exercise 2 but we're connecting via the slower Serial Interface.

### For each character you typed, how many packets are exchanged?

Mostly less than 1 packet. There were parts where 5-6 characters were send in just one package.

### What happens if you increase the typing speed?

More characters will be send together in one package.

### When do you observe delayed ACKs, if any?

When we typed slower, we got more delayed ACKs.

### Are you able to observe that more than one character is transmitted in the same packet?

Yes constantly. Higher speed in typing resulted in more characters for each package.

## Exercise 4

With the same setup we're now using nttcp to check the connection instead of telnet.

```
(PC2) $ nttcp -i -r -s -l 1000 -n 500 -p 4444
(PC1) $ nttcp -t -s -D -l 1000 -n 500 -p 4444 10.0.5.22
```

### Estimate the average amount of data that is ACKed at once.

The whole 1000byte length gets ACKed by PC2 4 times. In avarage this is about 250bytes which gets ACKed at once.

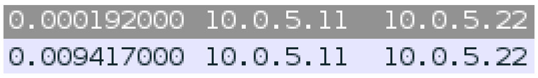
### What is the maximum amount of data that is ACKed at once?

About 86 packages get ACKed in the beginning. This makes about 90% of the data as the last ACK came for package 96.

### Explain how the receiver window size changes over time.

It starts with about 28 $10^4$ but goes down very fast.

### Select a matching transmission and its ACK: What is the time difference between data transmission and reception of the ACK?



```
0.000192000 10.0.5.11  10.0.5.22
0.009417000 10.0.5.11  10.0.5.22
```

Which is a difference of about 9.3 $10^{-3}$s

### Why or why not is the sender always using the full advertised window?

In our case it didn't use the full advertised window. As the connection is fast it just holds the packages for a little while and then sends them out.

### Examine and explain the pattern that occurs when the sender has sent all its data and closes the connection.

The only pattern is, that the sender sends the FIN, ACK which gets answered by an ACK. After that the connection closed in our capture.

## Exercise 5

We’re redoing exercise 4 over the slow serial connection but let the wireshark capture open.

### Compare the pattern to the last exercise.

In general we have the same pattern with delayed (slower connection) appearance.

Additionally we have more data per package.

### Is the ACK frequency any different?

We couldn’t observe any huge differences in the ACK frequency.

### Is the window size progression different?

Yes we get larger window sizes, which is a great example for reducing the overhead via headers and so on.

### Why or why not is the sender always using the full advertised window?

In our experience it didn’t use the full advertised window, but it uses more of it.

## Exercise 6

Same task like in Exercise 5 but with a faster serial cable.

### Is the “SACK Permitted” Option included in the handshake?

Yes. In the 2nd package there is the “SACK permitted” option in the TCP header.

### About what you observed after the next disconnects?

#### How is what you observe different from the first disconnects?

The connection recovers faster.

#### Do you observe SACKs or fast retransmits?

In our capture we couldn’t observe both of it.

## Exercise 7

Here we reconfigure the routing tables for the communication between PC1 and PC2. Again we decrease the serial cable clock rate to 9600.

After we wart a capture and observing the communication between nttcp sender and receiver.

### Estimate the size of the ssthresh parameter when the congestion window is small.

There actually was no ssthresh parameter. We even tried fullsearch

### Where is the sender performing slow start? Match the congestion window pattern to the rules of slow start.

Again the Window size just fluctuates between 29k and 29.3k. We can’t see the real patern here

## Where does fast recovery take place?

Right after replugging the cable. The purpose is, that the connection gets

# Lab 6

## Pre-Module Questions

### Describe the differences between a LAN switch/bridge and a router.

The main difference is that routers are for more intelligent in forwarding traffic. In general routers are able to join together multiple LANs and WANs while switches are commonly used inside LANs. Mostly just for the LANs themselfes.

### What is the difference between an Ethernet switch and an Ethernet hub? Which is more suitable for a network with a high traffic load, a switch or a hub? Explain.

A switch only delivers to the one who shall get the package while a hub just “spams” the whole network on every and to each host. So the answer is of course: Use a switch. This protects for flooding and overheading the network.

### What motivates the use of the term *transparent* in transparent bridges?

Transparent bridges can’t be seen from outside. They won’t appear as hop in traceroutes for examples.

### Which role does the spanning tree protocol play when interconnecting LAN switches/bridges?

It prevents multiple connections and messages over different ways. The spanning tree is build such that each host only gets its messages from one host.

### In the context of the IEEE 802.1d specification of the spanning tree protocol, define the following terms:

Root bridge The “root” of the spanning tree

Root port The port from where the information comes.

Designated bridge After blocking connections, each part of the network is only reachable through it's designated bridge.

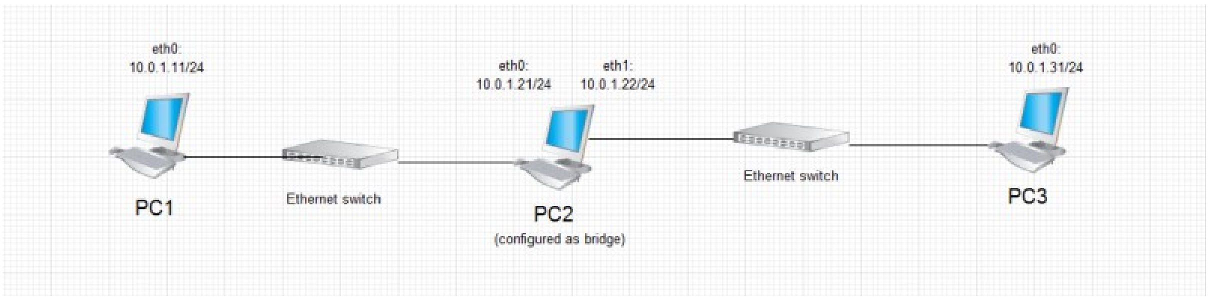
Designated port is the port of the designated bridge, which is used for this part of the network.

Blocked port This is a port which gets blocked, because it's path is redundant for the tree

### In the spanning tree protocol, how does a LAN switch/bridge decide which ports are in a blocking state?

All the ports which are not part of the forwarding and connection. So in general the blocked ports are redundant, as the information can travel better ways.

## Part 1



### Exercise 1

In this exercise we configure the PCs as shown above. PC2 was set up as bridge. Then we observe the PC2 as bridge in action while issuing ping/traceroute from PC1 → PC3.

#### Does PC2 modify the source and destination MAC and IP addresses?

No, PC2 just forwards it.

#### If PC2 was configured as an IP router, how would the output differ?

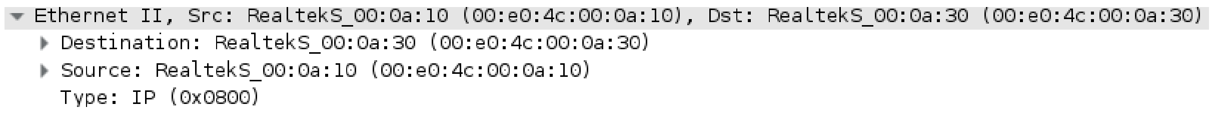
PC1 would know about it. PC1 would send the package to the MAC address of PC2 but with the IP address to PC3.

Does the bridge manipulate any of the fields in the MAC and IP headers?

The bridge doesn’t manipulate at all. It just forwards.

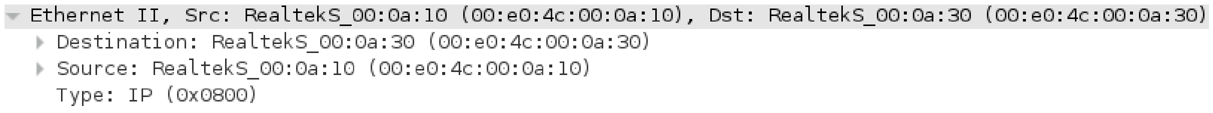
Do the source and destination MAC/IP addresses change when a packet traverses a bridge? Provide an explanation and include an example from the captured data.

PC1:



```
▾ Ethernet II, Src: RealtekS_00:0a:10 (00:e0:4c:00:0a:10), Dst: RealtekS_00:0a:30 (00:e0:4c:00:0a:30)
  ▸ Destination: RealtekS_00:0a:30 (00:e0:4c:00:0a:30)
  ▸ Source: RealtekS_00:0a:10 (00:e0:4c:00:0a:10)
    Type: IP (0x0800)
```

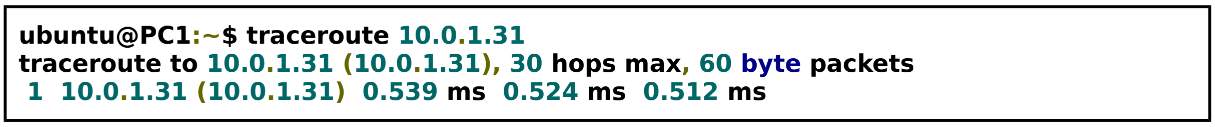
PC3:



```
▾ Ethernet II, Src: RealtekS_00:0a:10 (00:e0:4c:00:0a:10), Dst: RealtekS_00:0a:30 (00:e0:4c:00:0a:30)
  ▸ Destination: RealtekS_00:0a:30 (00:e0:4c:00:0a:30)
  ▸ Source: RealtekS_00:0a:10 (00:e0:4c:00:0a:10)
    Type: IP (0x0800)
```

As we can see, there’s no difference.

Include the output of the traceroute command from Step 5.

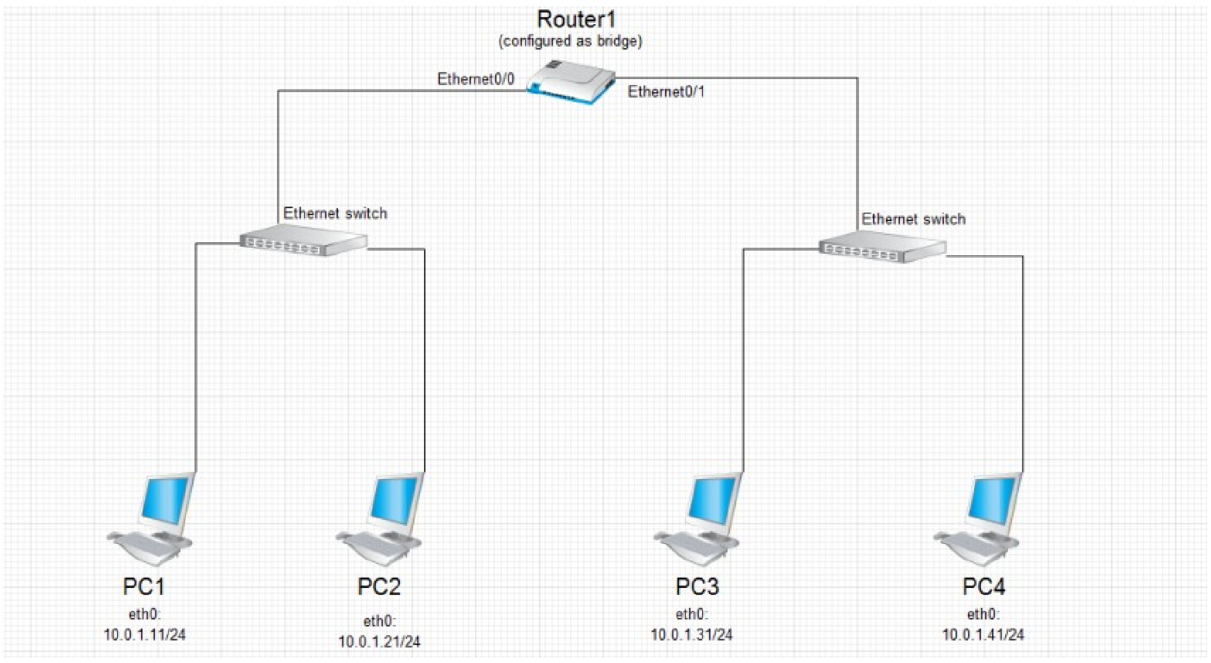


```
ubuntu@PC1:~$ traceroute 10.0.1.31
traceroute to 10.0.1.31 (10.0.1.31), 30 hops max, 60 byte packets
 1  10.0.1.31 (10.0.1.31)  0.539 ms  0.524 ms  0.512 ms
```

Provide an explanation why PC2 does not appear in the output of the traceroute command in Step 6.

As PC2 is acting as a bridge it just forwards. It’s not supposed to interact directly but only to make the interaction between PC1 and PC3 possible.

## Part 2



Here we reconfigure the system. This time we're not using a PC as a bridge but a router.

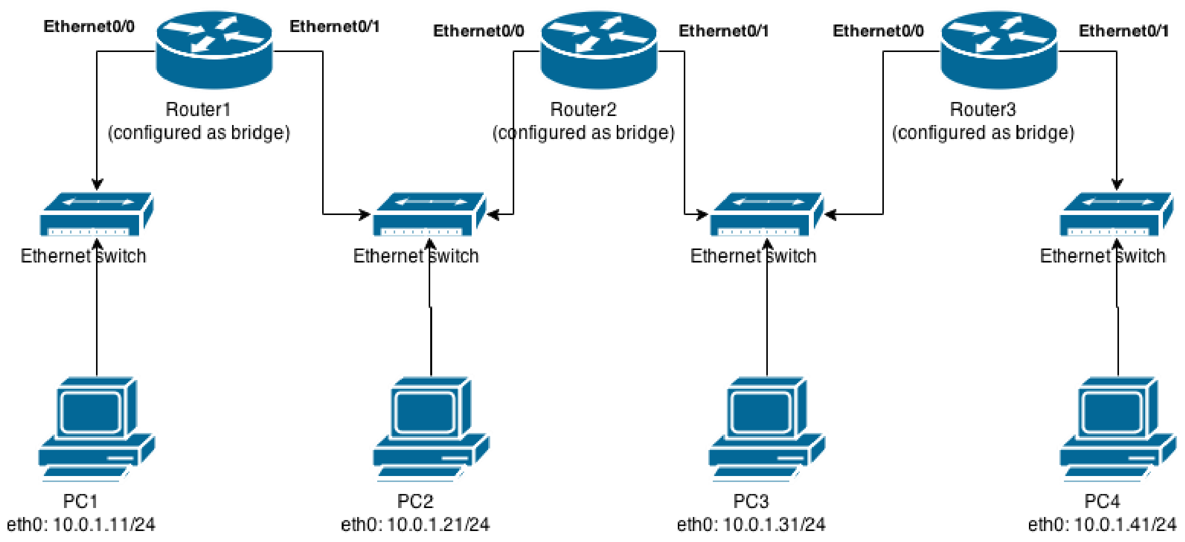
Compare the results to the outcome of the traceroute command in Exercise 1C.

We didn't see any differences.

Why is it not possible to issue a *ping* command to Router1?

Because Router1 has no IP address. And even if it had one, it would try to forward the message.

## Part 4

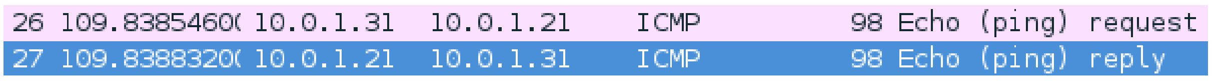


### Exercise 4A

Here we studied the bridges and how they set up their MAC forwarding tables from the network traffic.
After that, we had to issue a set of **ping** commands. After each command, we saved the MAC forwarding table on all bridges.

#### How far the ICMP Echo Request and Reply packets travel?

PC1 gets the ping from PC3→PC2:



```
26 109.8385460( 10.0.1.31   10.0.1.21     ICMP          98 Echo (ping) request
27 109.8388320( 10.0.1.21   10.0.1.31     ICMP          98 Echo (ping) reply
```

PC2 gets the pings from PC2 → PC4 (obvious) as well as the ping from PC1 and PC3.

PC4 gets only the pings its related to.

PC4 can't see the communication between PC1 and PC2 as well as the communication between PC3 and PC2.

#### Use the captured data to illustrate the algorithm used by bridges to forward packets.

Each bridge knows (after it got a message) through which interface it can get to the desired IP-Address.

For each of the transmitted packets, explain if the learning algorithm results in changes to the MAC forwarding table. Describe the changes.

After the first ping router 1 and 2 know, where the PCs 1 and 2 are. This goes on through all the pings. Each communication start gives the bridges the side on which the desired IP address can be found.

## Exercise 4B

We reconnected the PC2 in a way it’s connected to the PC4. After that we ping (without -c parameter) from PC1 to PC2.

Explain the outcome after steps 4-7 and compare it to the outcome of step 3.

Unfortunately the explanation was already written down by step 3. After we reconnected PC3 to a new loaction, the routers wouldn’t know about it, until they get packages by the PC3 itself or drop the MAC-entries for this PC because it wasn’t refreshed for too long.
When we ping from PC3 we actually first renew the MAC table of router 3, which itself sends it forward, so that router 2 gets a notice and finally router 1 gets noticed as well.

# Part 7

## Exercise 7

Describe which of the ping commands are successful and which fail. For each route, provide an explanation why the path is taken for each of the ping commands.

**(PC1)$ ping -c 1 10.0.4.31**

This ping went the simple way and passed first router 1 and then the switch between router 1 and PC3 which didn’t noticed any of the other packages. Important is, that PC1 has router2 as default gateway. So the order is exactly: **PC1→R1→R2→PC3**!

**(PC1)$ ping -c 1 10.0.4.41**

This route goes first to the default gateway of PC1 (router 2) and will go to router 3 after that, as this is the default gateway of router 2. Router 3 finally knows PC 4 and can access PC4 directly (over switch).
In short: **PC1→R1→R2→R3→PC4**

**(PC4)$ ping -c 1 10.0.1.11**

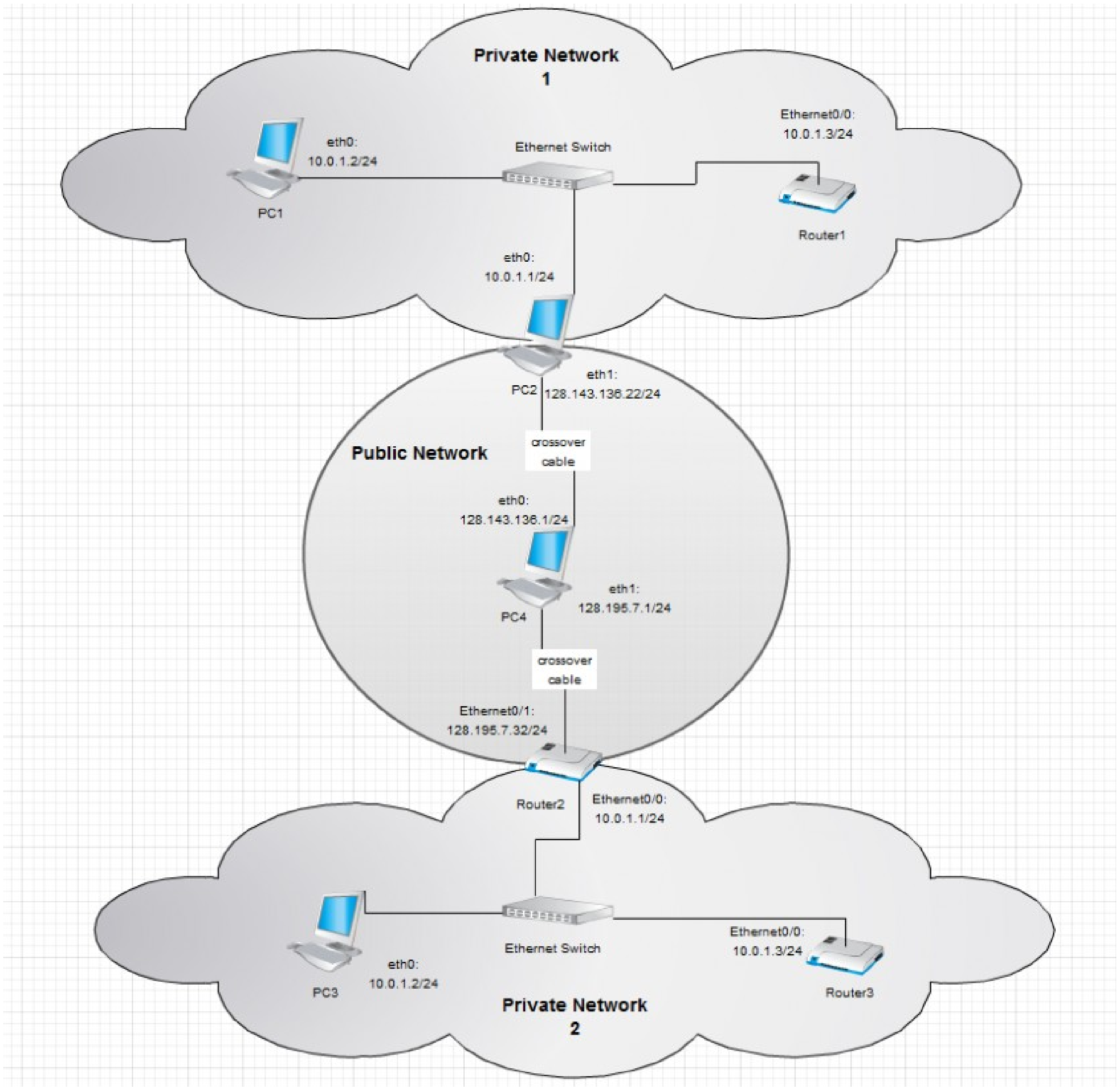
PC4 has Router3(0/1) as default gateway and addresses its ping to it. Router 3 itself has 10.0.3.2(R2) as default gateway. These are the two hops which are missing from the packages (See wireshark).

**(PC1)$ ping -c 1 10.0.3.21**

As predescribed reasons state this is really easy: **PC1 → R2 → PC2**.

# Lab 7

## Part 1


Private Network 1
eth0: 10.0.1.2/24
PC1
Ethernet Switch
Ethernet0/0: 10.0.1.3/24
Router1
eth0: 10.0.1.1/24
PC2
eth1: 128.143.136.22/24
Public Network
crossover cable
eth0: 128.143.136.1/24
PC4
eth1: 128.195.7.1/24
crossover cable
Ethernet0/1: 128.195.7.32/24
Router2
Ethernet0/0: 10.0.1.1/24
Ethernet Switch
Ethernet0/0: 10.0.1.3/24
PC3
eth0: 10.0.1.2/24
Router3
Private Network 2

## Exercise 1A/B

First we set up the whole network like shown in the figure. This includes setting up the Router2 for NAT.

**NAT before setup**

```
Router2$ show ip nat translations
Pro Inside global      Inside local        Outside local       Outside global
--- 200.0.0.2          10.0.1.2            ---                 ---
```

**NAT after setup**

```
Router2$ show ip nat translations
Pro       Inside global                Inside local       Outside local       Outside global
icmp      200.0.0.1:3209 10.0.1.1:3209      128.195.7.1:3209    128.195.7.1:3209
---       200.0.0.1                    10.0.1.1           ---                 ---
tcp       200.0.0.2:23                 10.0.1.2:23          128.195.7.1:38490  128.195.7.1:38490
icmp      200.0.0.2:3211 10.0.1.2:3211      128.195.7.1:3211    128.195.7.1:3211
---       200.0.0.2                    10.0.1.2           ---                 ---
icmp      200.0.0.3:3210 10.0.1.3:3210      128.195.7.1:3210    128.195.7.1:3210
---       200.0.0.3                    10.0.1.3           ---                 ---
```

**For each of the preceding ping commands, provide an explanation of why the command succeeds or fails.**

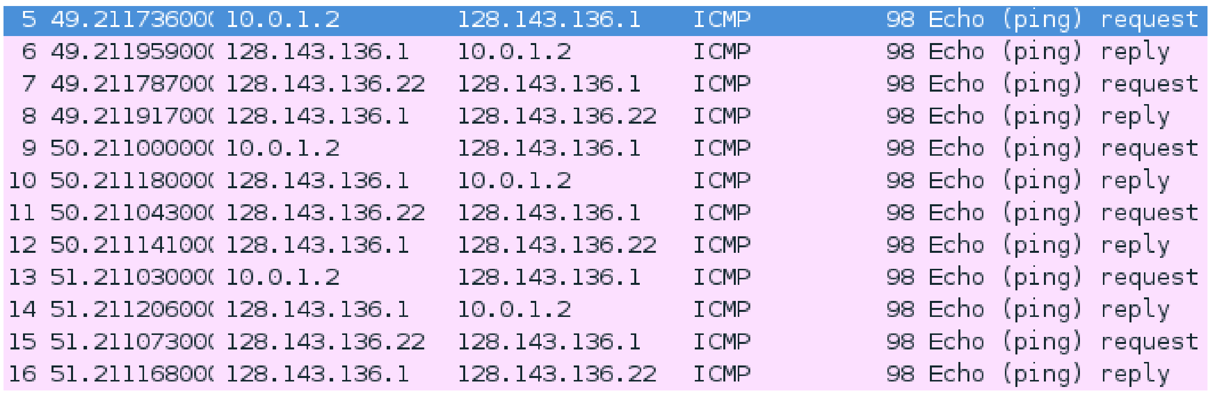
All of the pings but the ping from PC4 to PC3 via the intern IP address (10.0.1.2) are working. The last one is obvious, as this only counts for inside the Private Network 2.

## Exercise 1C

We modified the NAT table of PC2 such that all outgoing IP-datagrams are set to IP address 128.143.136.22. After that we observed the traffic at the NAT device via inpsecting telnet-connections.

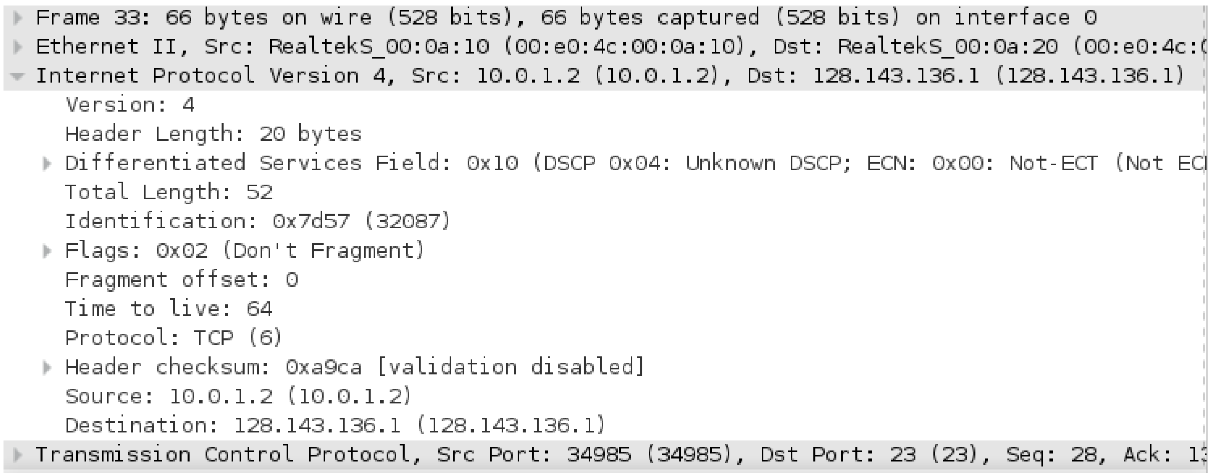
For each of the preceding *telnet* and *ping* commands, provide an explanation why a command succeeds or fails.

Again the ping to 10.0.1.2 fails. The others ones are successful, as the PC2 ‘renames’ the IP addresses so the communication works.

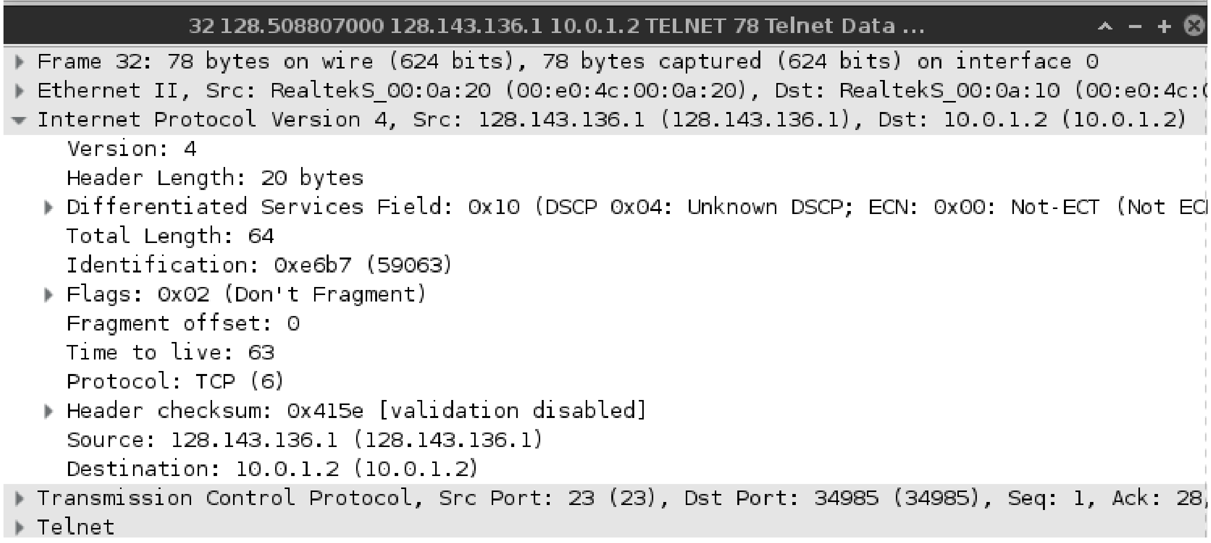


```
 5 49.21173600( 10.0.1.2       128.143.136.1   ICMP        98 Echo (ping) request
 6 49.21195900( 128.143.136.1  10.0.1.2        ICMP        98 Echo (ping) reply
 7 49.21178700( 128.143.136.22 128.143.136.1   ICMP        98 Echo (ping) request
 8 49.21191700( 128.143.136.1  128.143.136.22  ICMP        98 Echo (ping) reply
 9 50.21100000( 10.0.1.2       128.143.136.1   ICMP        98 Echo (ping) request
10 50.21118000( 128.143.136.1  10.0.1.2        ICMP        98 Echo (ping) reply
11 50.21104300( 128.143.136.22 128.143.136.1   ICMP        98 Echo (ping) request
12 50.21114100( 128.143.136.1  128.143.136.22  ICMP        98 Echo (ping) reply
13 51.21103000( 10.0.1.2       128.143.136.1   ICMP        98 Echo (ping) request
14 51.21120600( 128.143.136.1  10.0.1.2        ICMP        98 Echo (ping) reply
15 51.21107300( 128.143.136.22 128.143.136.1   ICMP        98 Echo (ping) request
16 51.21116800( 128.143.136.1  128.143.136.22  ICMP        98 Echo (ping) reply
```

**For each successful telnet session, include the IP header data of an outgoing and an incoming packet header (with respect to the private network).**



```
Frame 33: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: RealtekS_00:0a:10 (00:e0:4c:00:0a:10), Dst: RealtekS_00:0a:20 (00:e0:4c:(
Internet Protocol Version 4, Src: 10.0.1.2 (10.0.1.2), Dst: 128.143.136.1 (128.143.136.1)
    Version: 4
    Header Length: 20 bytes
    Differentiated Services Field: 0x10 (DSCP 0x04: Unknown DSCP; ECN: 0x00: Not-ECT (Not EC
    Total Length: 52
    Identification: 0x7d57 (32087)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xa9ca [validation disabled]
    Source: 10.0.1.2 (10.0.1.2)
    Destination: 128.143.136.1 (128.143.136.1)
Transmission Control Protocol, Src Port: 34985 (34985), Dst Port: 23 (23), Seq: 28, Ack: 13
```



```
32 128.508807000 128.143.136.1 10.0.1.2 TELNET 78 Telnet Data ...
Frame 32: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
Ethernet II, Src: RealtekS_00:0a:20 (00:e0:4c:00:0a:20), Dst: RealtekS_00:0a:10 (00:e0:4c:(
Internet Protocol Version 4, Src: 128.143.136.1 (128.143.136.1), Dst: 10.0.1.2 (10.0.1.2)
    Version: 4
    Header Length: 20 bytes
    Differentiated Services Field: 0x10 (DSCP 0x04: Unknown DSCP; ECN: 0x00: Not-ECT (Not EC
    Total Length: 64
    Identification: 0xe6b7 (59063)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 63
    Protocol: TCP (6)
    Header checksum: 0x415e [validation disabled]
    Source: 128.143.136.1 (128.143.136.1)
    Destination: 10.0.1.2 (10.0.1.2)
Transmission Control Protocol, Src Port: 23 (23), Dst Port: 34985 (34985), Seq: 1, Ack: 28,
Telnet
```

How does a PC know that a packet coming from the public network is destined to a host in the private network?

Via the MAC-address.

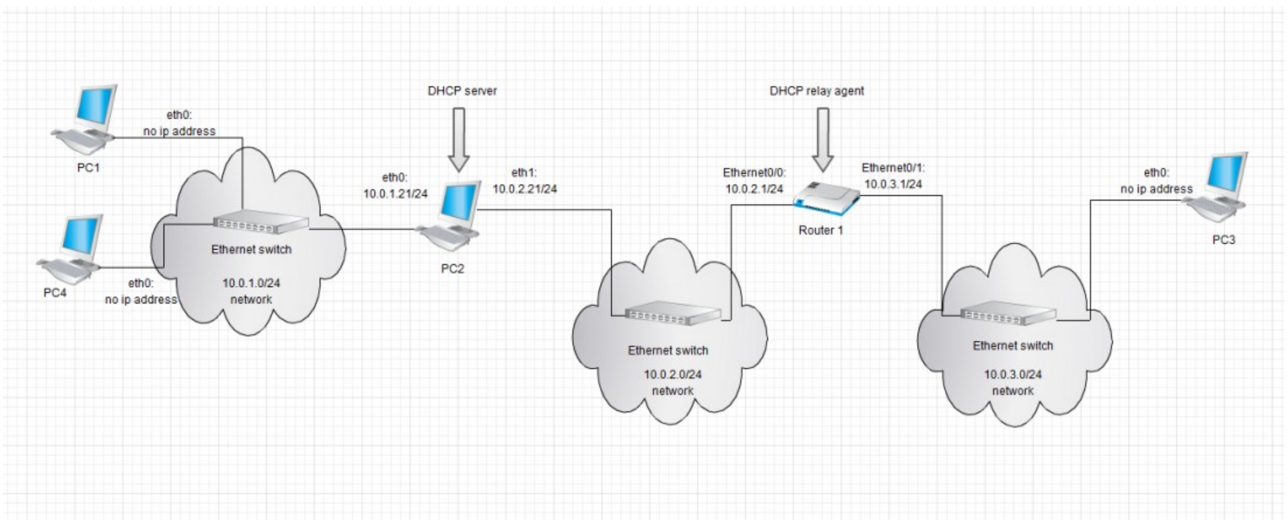
Explain the steps performed by the Linux kernel during IP address translation.

The address gets translated via a mapping of MAC addresses to IP-addresses. In this case PC2 also needs to know on which side the respective MAC address is.

## Exercise 1D

In this exercise we observe the problems of NAT with FTP. First we connected to PC2 from PC4 and downloaded a file (successfully). We tried something similar (from PC3) but didn’t succeed. As we used *128.143.136.22* for connecting to FTP the router couldn’t know, which one of the hosts behind the NAT we are trying to access. This could be solved via port-mapping.

## Part 2



### Exercise 2A-C

We changed the setup and configured PC2 as DHCP server.

Use a figure to explain the packets that were exchanged by the DHCP client and the DHCP server as part of the process of acquiring an IP address.

PC1 asks via Broadcast for an IP address. Due to the broadcasting nature, every PC (so PC2 too) gets the message. PC2 answers with an offering of a new address. IT uses already the new address to assign this. PC1 then answers to PC2 (again via Broadcast) that it wants to request this very address. PC2 finally ACKnowledges the IP address.

Additionally it’s important to know, that PC2 sends additional information via the **offer** package. From here we get the following information:

- assigned IP
- DHCP Server Identifier (IP of the DHCP server)
- subnet mask
- default router
- (boot device and file name)

Explain the entries in the lease file dhcpd.leases. How is the content of the lease file used when a DHCP client cannot contact the DHCP server?

> Each part of the lease file contains an IP address and it's start and endpoint for the connection. Additional information are (among other for us non-relevant informations) the MAC address and the client-hostname a.

In most client-server applications, the port number of a server is a well-known number (e.g. an FTP server uses port number 21, the Telnet server uses port number 23, etc.), while the client uses a currently available (ephemeral) port number. DHCP is different. Here, both the client and the server use a well-known port: UDP port 67 for the DHCP server and UDP port 68 for the DHCP client. Refer to RFC951 and provide an explanation for this protocol design choice.

> First and formost the PC is not accessible via IP without an IP address (of course it can be accessed directly via MAC and listen to traffic). But to be accassible at anytime (to get/provide a new IP address) it's important to have a standard.
>
> Additionally it would confuse client and server, if there was a disconnect and the configuration changes (for example because of a random port).
>
> Via a static port and the server listening to broadcasts there can be a way to provide ip addresses in most cases.

## Exercise 2D

For this we just added a DHCP relay agent (in this case it was router 1).

Does the DHCP relay agent modify DHCP packets or the IP header? If so, what are the modifications?

> When a router configured as a Relay Agent receives a DHCP broadcast, it converts it to unicast packet with destination MAC/IP address of the configured DHCP server, and source MAC/IP of the router itself. So yes, it changes the packages.

Is there a difference in the response of the DHCP server as compared to the DHCP configuration of PC1? If so, explain the difference.

> No we couldn't see a major difference.

How does the DHCP server (PC2) know on which network PC3 is located when it receives the DHCP request?

> In the DHCP discover we can find the information of the Relay agent IP address, which gives us the subnet where the ip-searching host is.

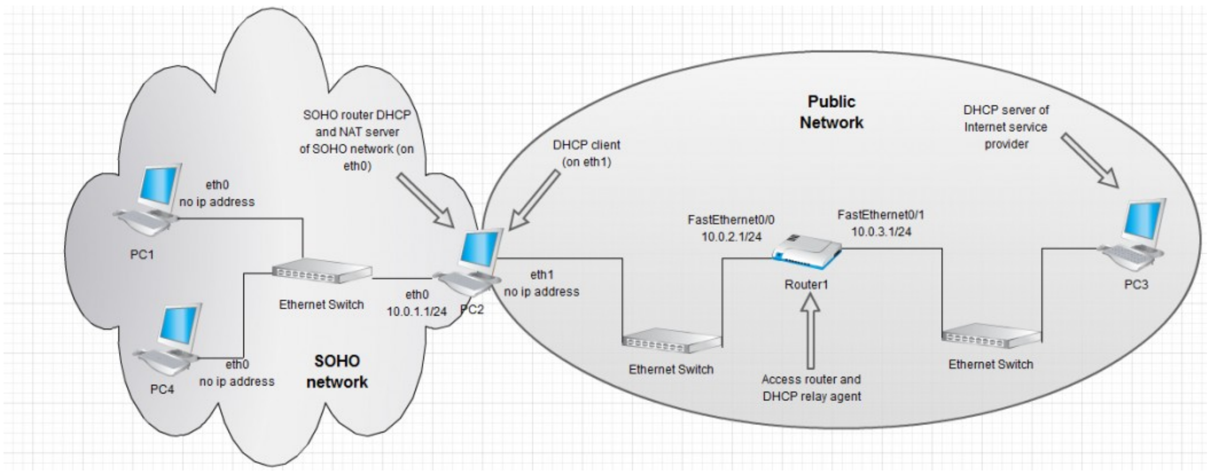
What is the destination IP address of the first DHCP packet that the DHCP server sends to PC3?

The IP address of the Ethernet0/1 interface of router 1. This is the interface which is on the side of the PC3.

Include the Wireshark data of the first three DHCP packets that are exchanged between PC3 and PC2.

```
113 244.5232350( 10.0.3.1       10.0.2.21      DHCP      342 DHCP Discover - Transaction ID 0x3d989522
114 244.5235120( 10.0.2.21      10.0.3.1       DHCP      342 DHCP Offer    - Transaction ID 0x3d989522
115 244.5274530( 10.0.3.1       10.0.2.21      DHCP      342 DHCP Request  - Transaction ID 0x3d989522
116 244.5276230( 10.0.2.21      10.0.3.1       DHCP      342 DHCP ACK      - Transaction ID 0x3d989522
```

## Part 3



### Exercise 3

Here we're experimenting with a "SOHO" setup, which stands for *small office, home office*. This setup is representative for local area networks working as NAT connected to the www.

**First ping request including it's reply.**

```
15 205.7990710( 10.0.1.1        10.0.1.6        ICMP        62 Echo (ping) request  id=0x70c3, seq=0/0,
16 205.7991140( 10.0.1.6        10.0.1.1        ICMP        62 Echo (ping) reply    id=0x70c3, seq=0/0,
```

**Routing Table PC1**

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.1.21       0.0.0.0         UG        0 0          0 eth0
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

## Routing Table PC2

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 eth1
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
```

## Routing Table PC3

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.3.1        0.0.0.0         UG        0 0          0 eth0
10.0.3.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

## Routing Table PC4

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.1.21       0.0.0.0         UG        0 0          0 eth0
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

## ifconfig PC1

```
eth0      Link encap:Ethernet  HWaddr 00:e0:4c:00:0a:10
          inet addr:10.0.1.6  Bcast:10.0.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:51 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11444 (11.4 KB)  TX bytes:1860 (1.8 KB)
          Interrupt:16

eth1      Link encap:Ethernet  HWaddr 00:18:8b:00:0a:11
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1220 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1220 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:96520 (96.5 KB)  TX bytes:96520 (96.5 KB)
```

**ifconfig PC2**

```
eth0      Link encap:Ethernet  HWaddr 00:e0:4c:00:0a:20
          inet addr:10.0.1.1  Bcast:10.0.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:158 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40292 (40.2 KB)  TX bytes:16828 (16.8 KB)
          Interrupt:16

eth1      Link encap:Ethernet  HWaddr 00:18:8b:00:0a:21
          inet addr:10.0.2.2  Bcast:10.0.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:191 errors:3 dropped:6 overruns:0 frame:0
          TX packets:106 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:21700 (21.7 KB)  TX bytes:13128 (13.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1330 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1330 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:110652 (110.6 KB)  TX bytes:110652 (110.6 KB)
```

**ifconfig PC3**

```
eth0      Link encap:Ethernet  HWaddr 00:e0:4c:00:0a:30
          inet addr:10.0.3.23  Bcast:10.0.3.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:379 errors:0 dropped:0 overruns:0 frame:0
          TX packets:287 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28817 (28.8 KB)  TX bytes:95842 (95.8 KB)
          Interrupt:21 Memory:fe9e0000-fea00000

eth1      Link encap:Ethernet  HWaddr 00:18:8b:00:0a:31
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1940 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1940 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:153832 (153.8 KB)  TX bytes:153832 (153.8 KB)
```

**ifconfig PC4**

```
eth0      Link encap:Ethernet  HWaddr 00:e0:4c:00:0a:40
          inet addr:10.0.1.5  Bcast:10.0.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:190 errors:0 dropped:0 overruns:0 frame:0
          TX packets:171 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:37886 (37.8 KB)  TX bytes:40282 (40.2 KB)
          Interrupt:16

eth1      Link encap:Ethernet  HWaddr 00:18:8b:00:0a:41
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2025 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2025 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:160760 (160.7 KB)  TX bytes:160760 (160.7 KB)
```

**NAT table PC2**

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  10.0.1.0/24          anywhere
```

# Lab 8

## Pre-Module Questions

### Briefly explain what the command host does.

With host we can lookup DNS entries. There're flags (options) we can add to the command to give us more/less information. It's possible to search for an IP address and for a domain name.

### Describe the following terms that are used in the Domain Name System.

Top-level domain is the part of the url which is in the very end (like .com and .de). It's the highest form auf name resolving.
CNAME(canonical name) is used to give a domain another names. It's used to connect exampledomain.net to the domain exampledomain.com which itself is connected to the IP address itself.
Resolvers are simple software modules which are running on DNS participating hosts and are able to ask for informations from nameservers.
Name server is the server which knows all the belongings of names and servers/services. In many cases our ISP are running one name server. But services like opendns and googles name server are popular as well.
Label is the technical expression for the domain names.
FQDN(fully qualified domain name) The full address which consists of root, tld, 1-3-level labels. And example is www.example.net. .
BIND means Berkeley Internet Name Domain Server which is the whole of Client and testingapplications. It's the most often used server application for DNS.
Inverse lookup gives the name to an IP address.
RR(resource record) is the most common information unit in DNS. It consists of "<name> [<ttl>] [<class>] <type> <rdata>", where rdata describe the Resource record in a detailed way and class gives the Protocol-Group to which the record belongs.
SOA(start of authority) consists of some important information about the DNS record like the refresh timer, serial number and when the DNS expires.

### Explain the following types of DNS queries:

Inverse queries see inverse lookup.
Iterative queries A client asks a DNS server for information. The server answers like expected.

Recursive queries Like an iterative query, but the DNS server itself might ask other DNS servers (like the root DNS server)

## What is the difference between a DNS domain and a DNS zone?

While DNS domain is a name for one domain (like the name suggests) the DNS zone is a distinct and contiguous portion of the domain name space in the domain name system. In the internet the domain space is organized into a hierarchical layout of subdomains below the DNS root Domain.

## What are some of the top-level domains in the DNS namespace?

.de, .com, .net, .berlin, ...

## Are domain names case sensitive? What, if any, is the constraint on the length of domain names?

The full domain name may not exceed a total length of 253 ASCII characters which are case-insensitive.

## Provide a list of the names and IP addresses of all root servers of the Internet.

1. Verisign, Inc. - 198.41.0.4
2. Information Sciences Institute - 192.228.79.201
3. Cogent Communications - 192.33.4.12
4. University of Maryland - 199.7.91.13
5. NASA Ames Research Center - 192.203.230.10
6. Internet Systems Consortium, Inc - 192.5.5.241
7. U.S. DOD Network Information Center - 192.112.36.4
8. U.S. Army Research Lab - 128.63.2.53
9. Netnod - 192.36.148.17
10. Verisign, Inc. - 192.58.128.30
11. RIPE NCC - 193.0.14.129
12. ICANN - 199.7.83.42
13. WIDE Project - 202.12.27.33

## **What is the purpose of the top-level domain *arpa*?**

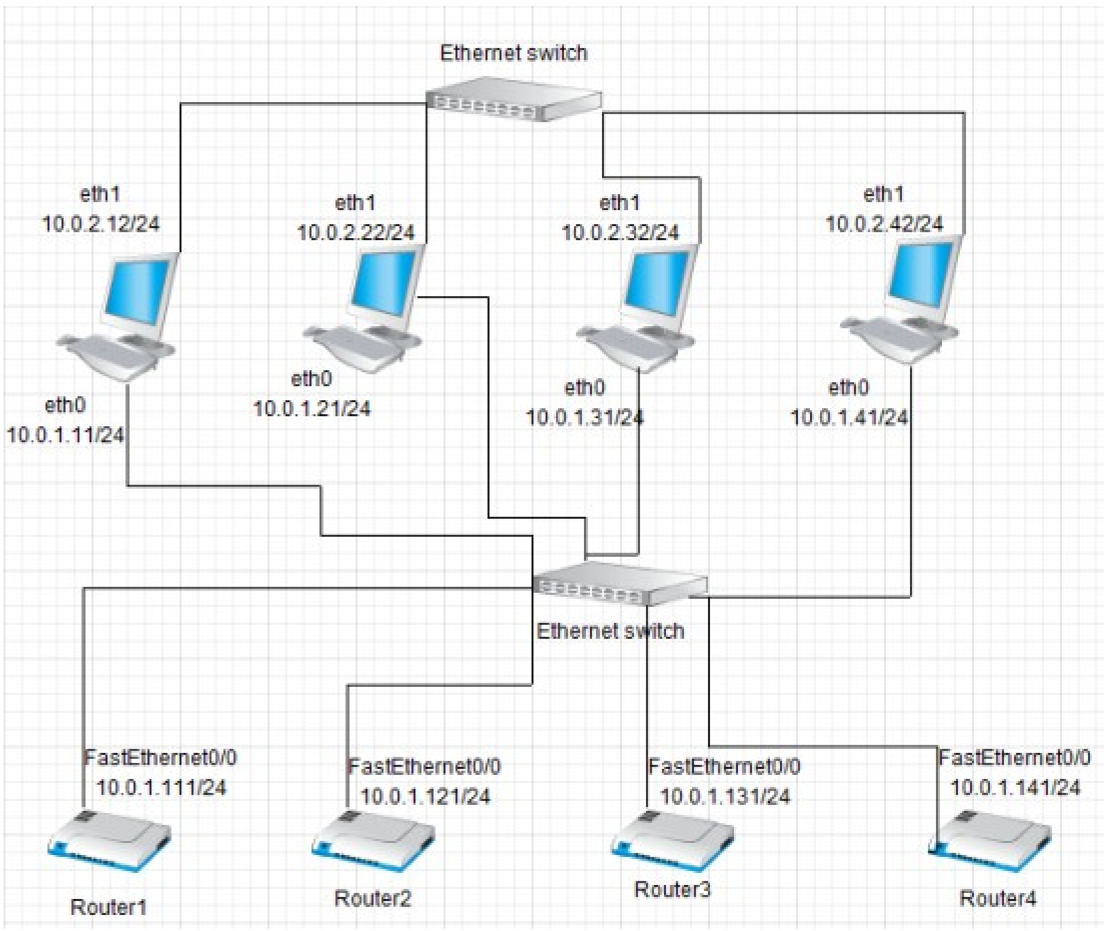
This is used mainly for administrative purposes. It was the first step to the internet like we have it today. Back in the days it was called arpa.net and was primary to connect universities and later the military.

From a command prompt on a Unix or Windows machine, run the command nslookup www.cnn.com, which shows the IP addresses that are associated with the domain name www.cnn.com. When you access www.cnn.com using a web browser on your computer, which IP address is chosen by your computer?

```
Non-authoritative answer:
www.cnn.com    canonical name = turner.map.fastly.net.
Name:   turner.map.fastly.net
Address: 23.235.43.73
```

And ping uses the one address I see here.

## Part 1



Part one is only used to configure the network. Nothing to save here.

## Part 2

Part 2 consists of a little bit experimenting but only one question:

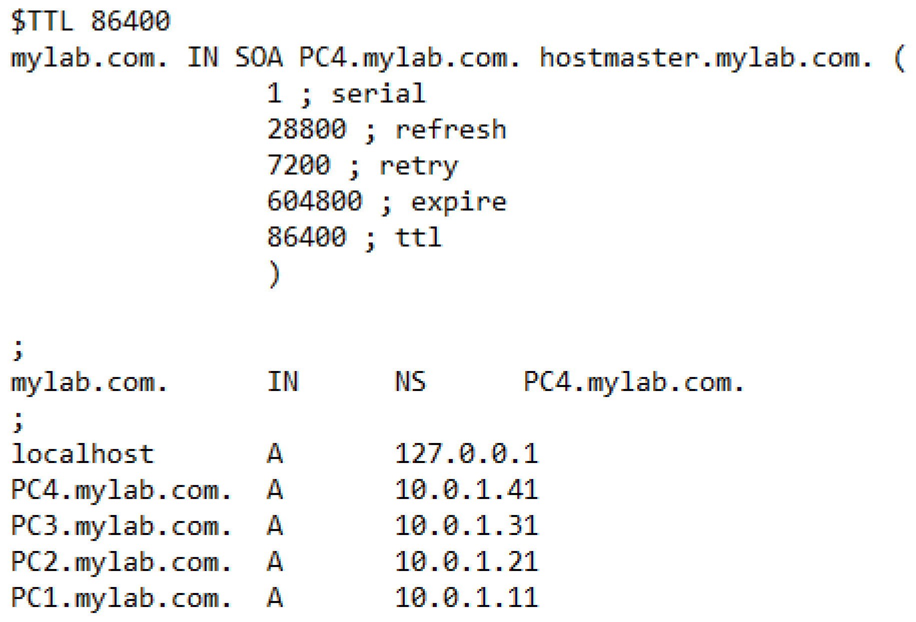
What happens if the same name is assigned to different IP addresses?

It takes the first IP address named in the host file.

## Part 3

### Exercise 3A

**Explain each record:**



```
$TTL 86400
mylab.com. IN SOA PC4.mylab.com. hostmaster.mylab.com. (
                1 ; serial
                28800 ; refresh
                7200 ; retry
                604800 ; expire
                86400 ; ttl
                )

;
mylab.com.      IN      NS      PC4.mylab.com.
;
localhost       A       127.0.0.1
PC4.mylab.com.  A       10.0.1.41
PC3.mylab.com.  A       10.0.1.31
PC2.mylab.com.  A       10.0.1.21
PC1.mylab.com.  A       10.0.1.11
```

The line "$TTL 86400"  in the beginning gives us the TTL of the file. After this time (1 day) the records are not valid anymore.

The entries are very simple: We have a FQDN with it's A record connection to an IP address.

### Exercise 3C/C

We tried multiple times to do, what the command says, but the command

```
$ /etc/init.d/bind9 start
```

never worked. Neither bind(9), nor named were started as a service. We also tried

```
$ /etc/init.d/bind9 status
```

which always told us that the service isn't running.

What worked was "named":

```
$ named -g
```

It is to be noticed, that /etc/rc.d/init.d/network was not existent, but /etc/init.d/networking was. We used it from now on, because an executable script named network was not existent on PC1-PC4 (we used find and locate).

Now everything worked like it should. In the folder **lab_08** we have all the files.

But unfortunately we couldn't get Exercise 3D to run. Because it never found the **PCX.lab8.net**. We added all the configuration files to the given folder lab_08, just in case we might have overseen something. We even tried from the beginning and some google solutions. We think the systems might have evolved too much after the book was published and there are more options to be checked and changed. Even intense research in man resolv.conf/named/named.conf couldn't help us.

As we checked the lab8 domain is not mandatory for the next tasks, so we moved on.

## Exercise 3E

```
ubuntu@PC1:~$ host -v PC3.mylab.com
Trying "PC3.mylab.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36165
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;PC3.mylab.com.                 IN      A

;; ANSWER SECTION:
PC3.mylab.com.          86400   IN      A       10.0.1.31

;; AUTHORITY SECTION:
mylab.com.          86400   IN      NS      PC4.mylab.com.

;; ADDITIONAL SECTION:
PC4.mylab.com.          86400   IN      A       10.0.1.41

Received 81 bytes from 10.0.1.41#53 in 1 ms
Trying "PC3.mylab.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8400
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;PC3.mylab.com.                 IN      AAAA

;; AUTHORITY SECTION:
mylab.com.          86400   IN      SOA     PC4.mylab.com. hostmaster.mylab.com. 1 28800
7200 604800 86400

Received 82 bytes from 10.0.1.41#53 in 0 ms
Trying "PC3.mylab.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16650
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
```

```
;PC3.mylab.com.                 IN      MX

;; AUTHORITY SECTION:
mylab.com.          86400   IN      SOA     PC4.mylab.com. hostmaster.mylab.com. 1 28800
7200 604800 86400

Received 82 bytes from 10.0.1.41#53 in 0 ms
```

## Part 4

We finally observe traffic between DNS resolvers and DNS servers. for this we use wireshark.

### Do all commands generate a DNS message?

The first and the last message generate DNS message. The first only if we don't hold the hostname "PC3.mylab.com" in the host file. This is a neat mechanism which allows easy blocking of ads and maleware by adding the respective domains to the host file and point them to the loopback device.

### Determine how domain names and IP addresses are encoded in DNS messages.

UDP→DNS→Queries: Here we have 1 or more entries. The domain name is ascii coded. The IP address consists exactly of 4 octets of binary numbers.

### What happens if a DNS query that cannot be resolved is issued?

The server answers with the flags "Recursion desired" and "server failure".

### If you repeat one of the commands, does PC1 issue another request or does PC1 cache the previous response?

It's not asking again until an entry is not valid or the TTL is down.

### DNS queries are either recursive or iterative. Use the data captured by Wireshark to determine if DNS queries are generated by issuing the ping commands.

Our commands were all iterative. We couldn't get a recursive command running, as the DNS server needs to send us to a root server which itself would give us DNS server responsible for handling our request.

## Part 5

We now run caching only servers. The main difference is, that the PCs are caching the requests for a short amount of time.

### Exercise 5

This exercise is about running a caching-only DNS server and observing the difference between a name resolution of this name serving solution and the last one.

#### Which of the commands generate a DNS message?

After the host file was cleared we get DNS messages for the first and second command. The third one doesn't generate because of the caching part. Localhost is already saved in the host file

#### Are any DNS queries issued when you repeat the query to resolve PC3.mylab.com?

As answered in the first question: no. Because it already has the records.

#### What happens on PC2 if a DNS query that cannot be resolved is issued? Is there any difference as compared to PC1?

We couldn't observe any differences except that PC2 now asks PC1 after it can't find the right answer in it's own database.

#### **Suggest an advantage and a disadvantage of running a caching-only DNS server at each host?**

**A simple pro reason** is almost immediate answer. Especially if we want more from one domain. Additionally we can imagine a setup like we have at home, where one router connects several devices to the internet. If one of the people is searching via duckduckgo.com, and another ones desires the services, the router can use the local information and answer immediately.
Of course the caching only approach has **disadvantages**. One of them is clearly the interval it gets updates. While the ROOT servers and even in most cases the servers from our ISPs are on track with the latest changes. Our router might not know if some domain is now available under a new destination. This can easily happen for dynamic IP addresses which can be connected to domain names via services like ddns or other dynamic IP services.

## Part 6

Now we’re about to discover the hierarchy of the DNS servers.

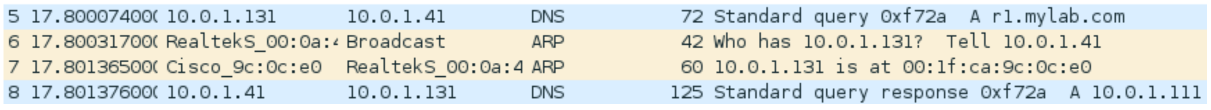
### Exercise 6C

#### For each command explain how the observed DNS queries are resolved.

(We removed the lab8 outputs, because we had the problems earlier with it)

**Router1$ ping R2.mylab.com**

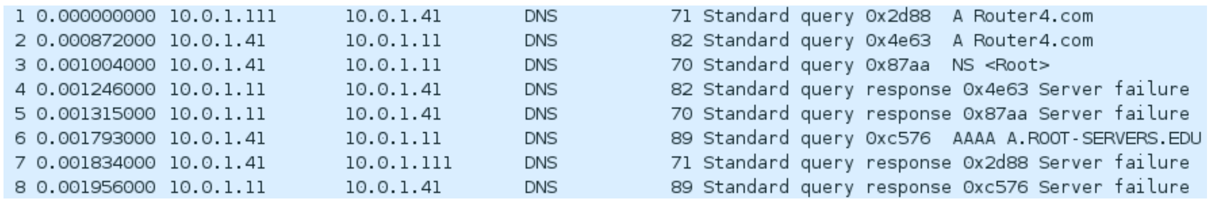
PC4 answers for this one and gives the right solutions.



```
5 17.80007400( 10.0.1.131       10.0.1.41        DNS          72 Standard query 0xf72a  A r1.mylab.com
6 17.80031700( RealtekS_00:0a:4 Broadcast        ARP          42 Who has 10.0.1.131?  Tell 10.0.1.41
7 17.80136500( Cisco_9c:0c:e0   RealtekS_00:0a:4 ARP          60 10.0.1.131 is at 00:1f:ca:9c:0c:e0
8 17.80137600( 10.0.1.41        10.0.1.131       DNS         125 Standard query response 0xf72a  A 10.0.1.111
```

**Router1$ ping Router4.com**

For this, we should get answers from PC2 which is responsible for the .com tlds. But unfortunately:

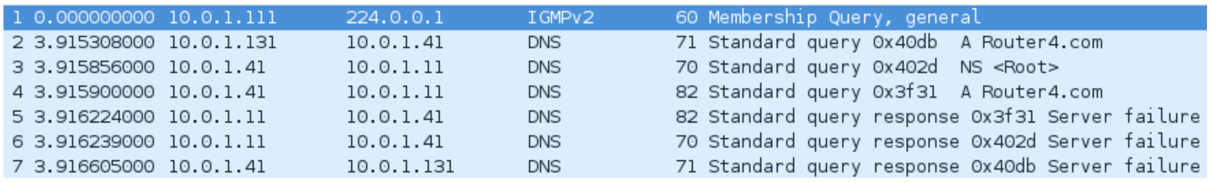


```
1 0.000000000  10.0.1.111       10.0.1.41        DNS          71 Standard query 0x2d88  A Router4.com
2 0.000872000  10.0.1.41        10.0.1.11        DNS          82 Standard query 0x4e63  A Router4.com
3 0.001004000  10.0.1.41        10.0.1.11        DNS          70 Standard query 0x87aa  NS <Root>
4 0.001246000  10.0.1.11        10.0.1.41        DNS          82 Standard query response 0x4e63 Server failure
5 0.001315000  10.0.1.11        10.0.1.41        DNS          70 Standard query response 0x87aa Server failure
6 0.001793000  10.0.1.41        10.0.1.11        DNS          89 Standard query 0xc576  AAAA A.ROOT-SERVERS.EDU
7 0.001834000  10.0.1.41        10.0.1.111       DNS          71 Standard query response 0x2d88 Server failure
8 0.001956000  10.0.1.11        10.0.1.41        DNS          89 Standard query response 0xc576 Server failure
```

But we can see how, .111 asks .41 first, as this one is resposible for this tld. But as .41 (PC4) deosn’t know the answer which causes it to ask .11(PC1).

**Router3$ ping Router4.com**

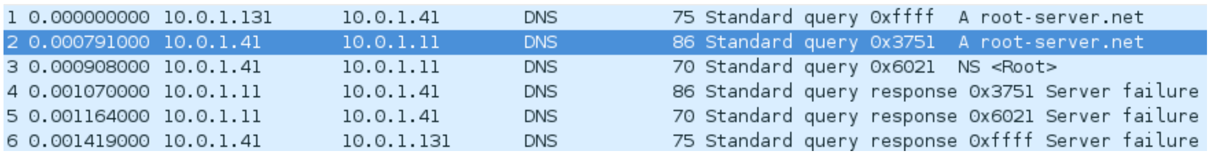
Here we have a simlilar answer like in the question from Router1:



```
1 0.000000000  10.0.1.111       224.0.0.1        IGMPv2       60 Membership Query, general
2 3.915308000  10.0.1.131       10.0.1.41        DNS          71 Standard query 0x40db  A Router4.com
3 3.915856000  10.0.1.41        10.0.1.11        DNS          70 Standard query 0x402d  NS <Root>
4 3.915900000  10.0.1.41        10.0.1.11        DNS          82 Standard query 0x3f31  A Router4.com
5 3.916224000  10.0.1.11        10.0.1.41        DNS          82 Standard query response 0x3f31 Server failure
6 3.916239000  10.0.1.11        10.0.1.41        DNS          70 Standard query response 0x402d Server failure
7 3.916605000  10.0.1.41        10.0.1.131       DNS          71 Standard query response 0x40db Server failure
```

**Router3$ ping root-server.net**

Another failure for the root-server domain. But at least the PC4 asks PC1 first (as this one is for ROOT).



```
1 0.000000000  10.0.1.131       10.0.1.41        DNS          75 Standard query 0xffff  A root-server.net
2 0.000791000  10.0.1.41        10.0.1.11        DNS          86 Standard query 0x3751  A root-server.net
3 0.000908000  10.0.1.41        10.0.1.11        DNS          70 Standard query 0x6021  NS <Root>
4 0.001070000  10.0.1.11        10.0.1.41        DNS          86 Standard query response 0x3751 Server failure
5 0.001164000  10.0.1.11        10.0.1.41        DNS          70 Standard query response 0x6021 Server failure
6 0.001419000  10.0.1.41        10.0.1.131       DNS          75 Standard query response 0xffff Server failure
```

All of these responses where we can see that one server is asking another one for “help” have the “recursive desired” flag set.

List the authoritative servers for the .net domain and .com domain.

**.net** and **.com** are both managed by VeriSign Global Registry Services. We covered the IP above.

Do you observe recursive or iterative DNS queries, or both? What is the main advantage/disadvantage of recursive DNS queries? What is the main advantage(disadvantage of iterative DNS queries?

We could see a recursive query after Router 3 asked PC4 for the IP address for the domain “root-server.net”. The request gets recursively transferred to PC1 which is responsible for the ROOT DNS.
**ping R2.mylab.com** on the other hand is an iterative request.
The **advantage** of recursive requests is obviously that we get information about one domain (if it exists) even if the information about it wasn’t distributed (yet), while it takes a significantly longer time to get the answer. The last point is the **disadvantage**. As the alternative is, that there will be no answer at all, the weight of this disadvantage is fairly low.

# Lab 9

## Pre-Module Questions

### Define the following SNMP terms:

SNMP Manager The one controlling and receiving/editing the information with SNMP
SNMP agent The one providing snmpd. Typically a host which can be configured via SNMP.
Management Information Base is an object which contains a collection of information organized hierarchically.
Object Identifier Different objects can be accessed via the OI. It uniquely identifies a managed object in the MIB hierarchy.

### Describe the purpose of the following SNMP tools:

snmpd The daemon which runs on the SNMP agents.
snmpget A tool to retrieve information from the agents.
snmpgetnext Gets the next information. Very nice to iterate through a set of informations.
snmpwalk Essentially a script which uses snmpgetnext to iterate through a subset of informations.
snmptrapd This receives and logs SNMP trap messages
snmptranslate Gives us information, what an OID really means. It “translates” the given OID and reverse.
snmpset Set information on an SNMP agent. (very powerful!)

### Which ports are used by snmpd and snmptrapd

snmpd: 161 and snmptrapd 162

### The OID 1.3.6.1.2.1.4.2 describes which object?

“IP-MIB::ipDefaultTTL”

### What numerical OID describes tcpRetransSegs in the group tcp of MIB-II?

.iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).tcp(6).tcpRetransSegs(12)
⇒ .1.3.6.1.2.1.6.12

### Give the OID of the Internet.

.iso(1).org(3).dod(6).internet(1) ⇒ .1.3.6.1

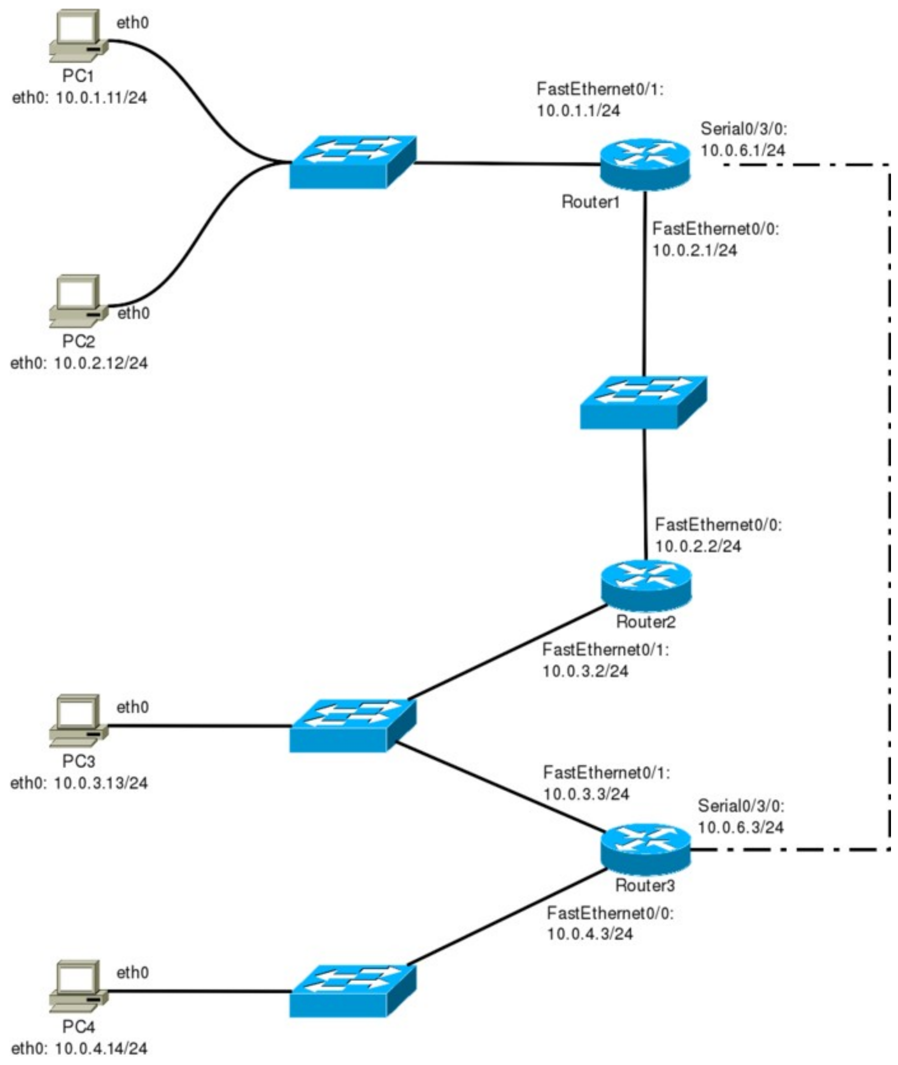
#### Describe the purpose of SNMP traps.

SNMP traps can listen passively for the agents for any useful messages (new IP for example). It’s basically a monitoring tool.

## Give a list of the main differences between SNMPv1 and SNMPv2 and a list of the main differences between SNMPv2 and SNMPv3.

SNMP v2 adds little improvements in security (via groups) but additional improvements in performance.
SNMP v3 finally introduced encryption to securely modify clients. This way network-sniffers won’t get cleartext data anymore.

## First Setup



## Exercise 1

Here we set up the whole network and assign default routers. After that we configured all interfaces of the routers and set their ospf cost to 10.

## Exercise 2

| MIB | OID# | description | children | data type |
|---|---|---|---|---|
| **IF-MIB** | MIB-2 2<br><br>(MIB-2 = 1.3.6.1.2.1)<br><br>⇒ 1.3.6.1.2.1.2 | The MIB module to describe generic objects for network interface sub-layers. This MIB is an updated version of MIB-II's ifTable, and incorporates the extensions defined in RFC 1229. | ifNumber(1)<br>ifTable(2) | OCTET STRING (SIZE(0..255)) |
| **INET-ADDRESS-MIB** | MIB-2 76 | This MIB module defines textual conventions for representing Internet addresses. An Internet address can be an IPv4 address, an IPv6 address or a DNS domain name. | - | IPv4:<br>OCTET STRING (SIZE (4))<br><br>IPv6:<br>OCTET STRING (SIZE (16\|20)) |
| **IP-FORWARD-MIB** | ip 24<br><br>(ip = 1.3.6.1.2.1.4) | IPv4/v6 version-independent revision. Minimal changes were made to the original RFC 2096 MIB to allow easy upgrade of existing IPv4 implementations to the version-independent MIB. These changes include: Adding inetCidrRouteDiscards as a replacement for the deprecated ipRoutingDiscards and ipv6DiscardedRoutes objects. Adding a new conformance statement to support the implementation of the IP Forwarding MIB in a read-only mode. The inetCidrRouteTable replaces the IPv4-specific ipCidrRouteTable, its related objects, and related conformance statements. | - | (not) enabled |
| **IP-MIB** | MIB-2 48 | The MIB module for managing IP and ICMP implementations, but excluding their management of IP routes. | ipMIBConformance(2) | many for the options |
| **TCP-MIB** | MIB-2 49 | The MIB module for managing TCP implementations. | tcpMIBConformance(2) | - |
| **UDP-MIB** | MIB-2 50 | The MIB module for managing UDP implementations. Copyright (C) The Internet Society (2005). This version of this MIB module is part of RFC 4113; see the RFC itself for full legal notices. | udpMIBConformance(2) | - |

## Definitions:

ipForwarding: 1/2 which stands for "forwarding enabled" and "disabled".
sysName: Name of the system. In our cases we had mostly router1-4 and PC1-4.
ipRouteTable: The routing table we already know from earlier exercises.
tcpRtoAlgorithm: The algorithm used to determine the timeout value used for retransmitting unacknowledged octets.
tcpConnState: "*The state of this TCP connection.*

> *The only value which may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a 'badValue' response if a management station attempts to set this object to any other value.*
>
> *If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node, resulting in immediate termination of the connection.*
>
> *As an implementation-specific option, a RST segment may be sent from the managed node to the other TCP endpoint (note however that RST segments are not sent reliably).*"

## What routing protocols are defined?

other(1), local(2), netmgmt(3), icmp(4), egp(5), ggp(6), hello(7), rip(8), is-is(9), es-is(10), ciscoIgrp(11), bbnSpfIgp(12), ospf(13), bgp(14), idpr(15)

## What TCP states are defined?

closed(1), listen(2), synSent(3), synReceived(4), established(5), finWait1(6), finWait2(7), closeWait(8), lastAck(9), closing(10), timeWait(11), deleteTCB(12)

## What are the states of ipForwarding?

Values: forwarding(1), notForwarding(2)

## Exercise 3

```
$ snmpget -v1 -c public localhost system.sysDescr.0
RFC1213-MIB::sysDescr.0 = STRING: "Linux PC1 3.13.0-37-generic #64-Ubuntu SMP Mon
Sep 22 21:30:01 UTC 2014 i686"

$ snmpget -v1 -c public localhost udp.udpOutDatagrams.0
RFC1213-MIB::udpOutDatagrams.0 = Counter32: 567

$ snmpget -v1 -c public localhost udp.udpInDatagrams.0
RFC1213-MIB::udpInDatagrams.0 = Counter32: 9

$ snmpget -v1 -c public localhost system.sysUpTime.0 system.sysName.0
RFC1213-MIB::sysUpTime.0 = Timeticks: (75331) 0:12:33.31
RFC1213-MIB::sysName.0 = STRING: "PC1"

$ snmpget -v1 -c public localhost interfaces.ifTable.ifEntry.ifDescr.1
RFC1213-MIB::ifDescr.1 = STRING: "lo"

$ snmpwalk -v1 -c public localhost interfaces.ifTable.ifEntry.ifDescr
RFC1213-MIB::ifDescr.1 = STRING: "lo"
RFC1213-MIB::ifDescr.2 = STRING: "eth1"
RFC1213-MIB::ifDescr.3 = STRING: "eth0"
```

## Exercise 4

### What are the fields in an SNMP header?

Version number, community, data.
Data has a kind of tree for suboptions.

### How is the snmpwalk command implemented?

It’s like a script issuing the snmpgetnext command for iterating purposes.

### What is the maximum payload size?

In our case it was 165b for “Object Name: 1.3.6.1.2.1.1.9.1.3.2 (iso.3.6.1.2.1.1.9.1.3.2)”

### How is plain text represented? How are IP and MAC addresses encoded?

Plain text is represented as ASCII characters and IP/MAC addresses seemed not to be encoded at all. The OIDs are always encoded as the number-representation.

## Exercise 6

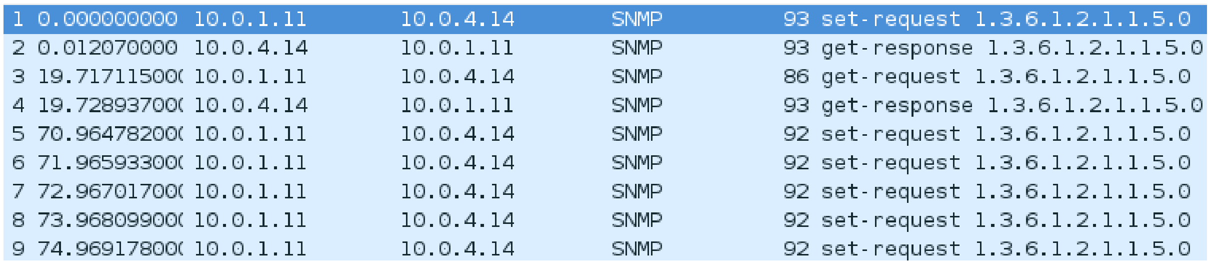
### What is the output of this command?

After we set the name to “NewName” the output was (as expected) NewName.

### How does the SNMP agent signal that the community string is not correct?

We didn’t get an answer for false questions. So the answer was just no answer at all.

As the captures are to much to be inserted at all, here is an excerpt. The capture itself can be found in the corresponding folder.



```
1 0.000000000 10.0.1.11    10.0.4.14    SNMP    93 set-request 1.3.6.1.2.1.1.5.0
2 0.012070000 10.0.4.14    10.0.1.11    SNMP    93 get-response 1.3.6.1.2.1.1.5.0
3 19.71711500( 10.0.1.11   10.0.4.14    SNMP    86 get-request 1.3.6.1.2.1.1.5.0
4 19.72893700( 10.0.4.14   10.0.1.11    SNMP    93 get-response 1.3.6.1.2.1.1.5.0
5 70.96478200( 10.0.1.11   10.0.4.14    SNMP    92 set-request 1.3.6.1.2.1.1.5.0
6 71.96593300( 10.0.1.11   10.0.4.14    SNMP    92 set-request 1.3.6.1.2.1.1.5.0
7 72.96701700( 10.0.1.11   10.0.4.14    SNMP    92 set-request 1.3.6.1.2.1.1.5.0
8 73.96809900( 10.0.1.11   10.0.4.14    SNMP    92 set-request 1.3.6.1.2.1.1.5.0
9 74.96917800( 10.0.1.11   10.0.4.14    SNMP    92 set-request 1.3.6.1.2.1.1.5.0
```

## Exercise 7

### What is the meaning of the *12*? Include all steps you performed to solve this question.

As we already hat to sum up the TCP states in the beginning the solution was simple. The 12 is for “deleteTCB” which resulted in a reset of the connection.

### What may be a problem with the community-based security?

There is not much knowledge needed to compromise the system by copying the packages (including the community name).

## Exercise 8

### What is the difference between the v1 and v2c messages?

I didn’t get any different output for both messages.

### Which fields in the SNMP GET are affected by this change?

snmp.msgAuthenticationParameters

### Which fields in the SNMP GET and the response are affected by this change?

The complete part within the SNMP (below UDP) gets encrypted.

## Exercise 9

root@PC1:~# snmptrapd -f -Le

```
Warning: no access control information configured.
  (Config search path: /usr/local/etc/snmp:/usr/local/share/snmp:/usr/local/lib/snmp:/root/.snmp)
This receiver will *NOT* accept any incoming notifications.
NET-SNMP version 5.7.2.1
No access configuration - dropping trap.
No access configuration - dropping trap.
No access configuration - dropping trap.
[...]
No access configuration - dropping trap.
No access configuration - dropping trap.
No access configuration - dropping trap.
No access configuration - dropping trap.
```

root@PC1:~# snmptrapd -f -Le

```
Warning: no access control information configured.
  (Config search path: /usr/local/etc/snmp:/usr/local/share/snmp:/usr/local/lib/snmp:/root/.snmp)
This receiver will *NOT* accept any incoming notifications.
NET-SNMP version 5.7.2.1
No access configuration - dropping trap.
No access configuration - dropping trap.
^C2015-08-22 14:51:25 NET-SNMP version 5.7.2.1 Stopped.
Stopping snmptrapd
```

As we can see here we couldn't get trap to work.

## Exercise 10

```
root@PC1:~# snmpwalk -v1 -c public 10.0.4.14 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.0.0.0.0 = IpAddress: 10.0.4.3
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 0.0.0.0
root@PC1:~# snmpwalk -v1 -c public 10.0.4.3 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.10.0.1.0 = IpAddress: 10.0.6.1
RFC1213-MIB::ipRouteNextHop.10.0.2.0 = IpAddress: 10.0.3.2
RFC1213-MIB::ipRouteNextHop.10.0.3.0 = IpAddress: 10.0.3.3
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 10.0.4.3
RFC1213-MIB::ipRouteNextHop.10.0.6.0 = IpAddress: 10.0.6.3
root@PC1:~# snmpwalk -v1 -c public 10.0.6.1 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.10.0.1.0 = IpAddress: 10.0.1.1
RFC1213-MIB::ipRouteNextHop.10.0.2.0 = IpAddress: 10.0.2.1
RFC1213-MIB::ipRouteNextHop.10.0.3.0 = IpAddress: 10.0.2.2
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 10.0.2.2
RFC1213-MIB::ipRouteNextHop.10.0.6.0 = IpAddress: 10.0.6.1
root@PC1:~# snmpwalk -v1 -c public 10.0.1.1 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.10.0.1.0 = IpAddress: 10.0.1.1
RFC1213-MIB::ipRouteNextHop.10.0.2.0 = IpAddress: 10.0.2.1
RFC1213-MIB::ipRouteNextHop.10.0.3.0 = IpAddress: 10.0.2.2
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 10.0.2.2
RFC1213-MIB::ipRouteNextHop.10.0.6.0 = IpAddress: 10.0.6.1
```

```
root@PC1:~# snmpwalk -v1 -c public 10.0.4.14 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.0.0.0.0 = IpAddress: 10.0.4.3
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 0.0.0.0
root@PC1:~# snmpwalk -v1 -c public 10.0.4.3 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.10.0.1.0 = IpAddress: 10.0.3.2
RFC1213-MIB::ipRouteNextHop.10.0.2.0 = IpAddress: 10.0.3.2
RFC1213-MIB::ipRouteNextHop.10.0.3.0 = IpAddress: 10.0.3.3
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 10.0.4.3
root@PC1:~# snmpwalk -v1 -c public 10.0.3.2 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.10.0.1.0 = IpAddress: 10.0.2.1
RFC1213-MIB::ipRouteNextHop.10.0.2.0 = IpAddress: 10.0.2.2
RFC1213-MIB::ipRouteNextHop.10.0.3.0 = IpAddress: 10.0.3.2
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 10.0.3.3
root@PC1:~# snmpwalk -v1 -c public 10.0.2.1 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.10.0.1.0 = IpAddress: 10.0.1.1
RFC1213-MIB::ipRouteNextHop.10.0.2.0 = IpAddress: 10.0.2.1
RFC1213-MIB::ipRouteNextHop.10.0.3.0 = IpAddress: 10.0.2.2
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 10.0.2.2
root@PC1:~# snmpwalk -v1 -c public 10.0.1.1 ip.ipRouteTable.ipRouteEntry.ipRouteNextHop
RFC1213-MIB::ipRouteNextHop.10.0.1.0 = IpAddress: 10.0.1.1
RFC1213-MIB::ipRouteNextHop.10.0.2.0 = IpAddress: 10.0.2.1
RFC1213-MIB::ipRouteNextHop.10.0.3.0 = IpAddress: 10.0.2.2
RFC1213-MIB::ipRouteNextHop.10.0.4.0 = IpAddress: 10.0.2.2
```

Explain how the route is resolved by the process described above.

Each host has it’s own table saved, where it just stores the next hop for a disired destination. This way the packages can arrive the destination without every host knowing the whole path.

Give the routes you determined before and after you disconnected the serial cable.

**Before: 10.0.4.14 → 10.0.4.3 → 10.0.6.1 → 10.0.1.1 → 10.0.1.12**
**After: 10.0.4.14 → 10.0.4.3 → 10.0.3.2 → 10.0.2.1 → 10.0.1.1**

# Lab 10

## Pre-Module Questions

### What IP address range is used in IP multicast?

224.0.0.0 to 239.255.255.255

### What is special about the addresses 224.0.0.0 to 224.0.0.255? How do routers treat packets with a destination in this address range?

These are multicast addresses which are used to send datagrams beyond the direct local area network.

### What are the purposes of the following addresses:

**224.0.0.5:** All OSPF routers address is used to send Hello packets to all OSPF routers on a network segment.

**224.0.0.6:** The OSPF All Designated Routers address is used to send OSPF routing information to designated routers on a network segment.

### Give the socat commands for the following actions:

#### Receiving IP multicast packets for group 224.0.10.1 containing a UDP datagram for port 1111 and writing the UDP payload to STDOUT.

```
$ socat -u UDP4-RECV:1111,ip-add-membership=224.0.10.1:10.0.1.12 STDIO
```

#### Sending IP multicast packets to group 224.0.10.1 with TTL 64 containing a UDP datagram to port 1111 that contains the payload "lab".

```
$ while true; do echo lab; sleep 1; done | socat -d -d STDIO UDP4-
                                  SENDTO:224.0.10.1:1111,ip-multicast-ttl=64
```

Explain the major differences between IGMP versions 1 to 3.

| Feature | IGMPv1 | IGMPv2 | IGMPv3 |
|---|---|---|---|
| **Default Query Interval** | 60s | 125s | 125s |
| **First octet value for the report** | 0x12 | 0x16 | 0x22 |
| **Group address for the report** | Joining multicast group address | Joining multicast group address | Joining multicast group address and source address |
| **Destination address for the report** | Joining multicast group address | Joining multicast group address | 224.0.0.22 |
| **Can maximum response time be configured** | No, fixed at 10s | Yes, 0 to 255 s | Yes, 0 to 53min |

Give an mtrace command, that traces the path from 10.0.1.11 to 10.0.2.12 using multicast group *224.0.10.1*.

```
#mtrace 10.0.1.11 10.0.2.12 224.0.10.1
```

Translate the multicast IP address 224.85.170.42 to the corresponding MAC address.

01:00:5e:55:aa:2a

Is the mapping between multicast IP addresses and MAC addresses bijective? What does this imply?

Not a bijective, The multicast IP addresses all map to the same multicast MAC address.
**for example:** 5 bits of mapping information: 5†= 32. This means we will map 32 multicast IP addresses to 1 multicast MAC address.

## List all PIM message types and explain their function in one sentence

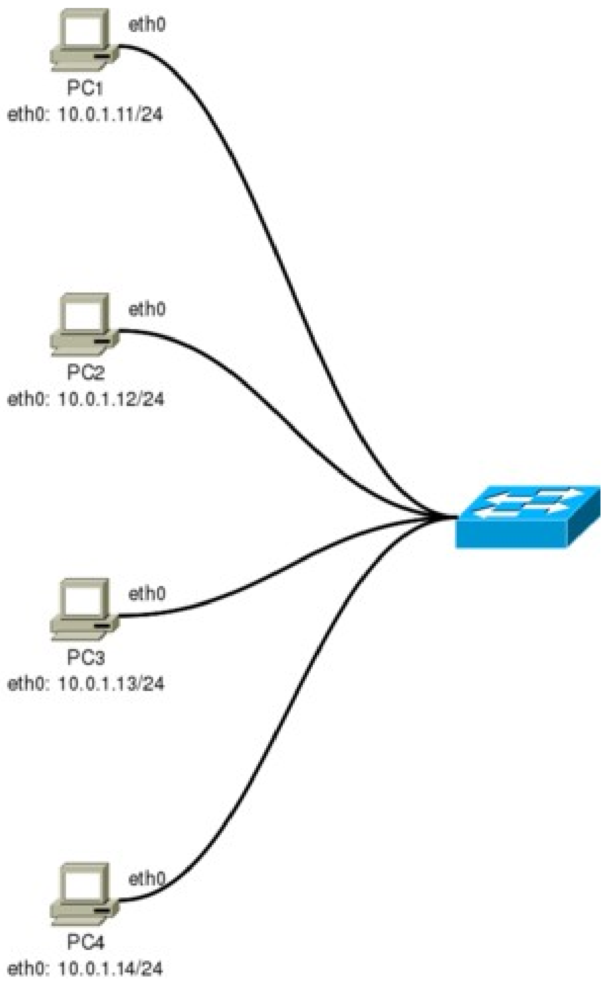
| Type | Name | Description |
|---|---|---|
| 0 | Hello | Are used to detect other PIM routers. |
| 1 | Register | |
| 2 | Register Stop | After this the DR starts a R-St. Timer to maintain it's state |
| 3 | Join/Prune | Join/prune to a tree. Prune messages are sent toward the upstream neighbor for S to indicate that traffic from S addressed to group G is not desired. |
| 4 | Bootstrap | BSR (Bootstrap Router) is one way for the receivers to discover the sources that send to a particular multicast group. |
| 5 | Assert | Used to select „Forwarder" in multi access environment. Contains information about metric preference, route metric and ip_address. Helps building the graph. |
| 6 | Graft | Is specificly for re-joining to a group which was pruned by the router earlier. |
| 7 | Graft-Ack | Acknowledges the Graft (in time window of 3s). |
| 8 | Candidate RP Advertisement | Even in the specified document () we couldn't find specific explanations. But it's a unicast to domain's BSR. |
| 9 | State Refresh | Skips unnecessary devices, but lets assert winner (i.e.) do a state refresh. |
| 10 | DF Election | The procedure selects one router as the DF for every RP of bidirectional groups. This router is responsible for forwarding multicast packets received on that network upstream to the RP. |

## How does a PIM join/prune handler decide if the message denotes a join, prune or both?

In (link[3]) we can see the Join/Prune message format. If the specified field has data it uses it. Especially the field with the number of joined/pruned sources is important as it denotes which value from which field is important.

3) http://www.faqs.org/rfcs/rfc3973.html – see 4.7.6

## Network Setup I



## Exercise 1

Here we set up the network and add routes to the multicast address 224.0.0.0/8 for each participant. We start wireshark for capturing the following command:

```
        $ while true;
                do echo ekelerregendesregenwetter; sleep 1;
        done | socat -d -d STDIO UDP4-SENDTO:224.0.10.1:6666
```

### Examine the destination MAC address:

What does the OUI look like? Is the multicast bit set? Is the globally unique bit set?

Multicast bit is set 1. Globally unique bit set to dst/src “0”.

Can you infer the multicast group from the NIC-specific part of the MAC address?

Yes. The prefix 01-00-5e identifies the frame as multicast.

### What is the IP TTL?

0

Do you observe IGMP messages?

No, we didn’t observe any IGMP messages.

## Exercise 2

**Have a look at the transmitted packages and answer the following questions.**

Examine the time frame before the first mark: What and how many IGMP messages do you observe?

Two IGMP messages by PC3 and PC2 (so 4 in all).

Examine the time frame between the first and the second mark: Do the receivers react on the new sender?

Yes.

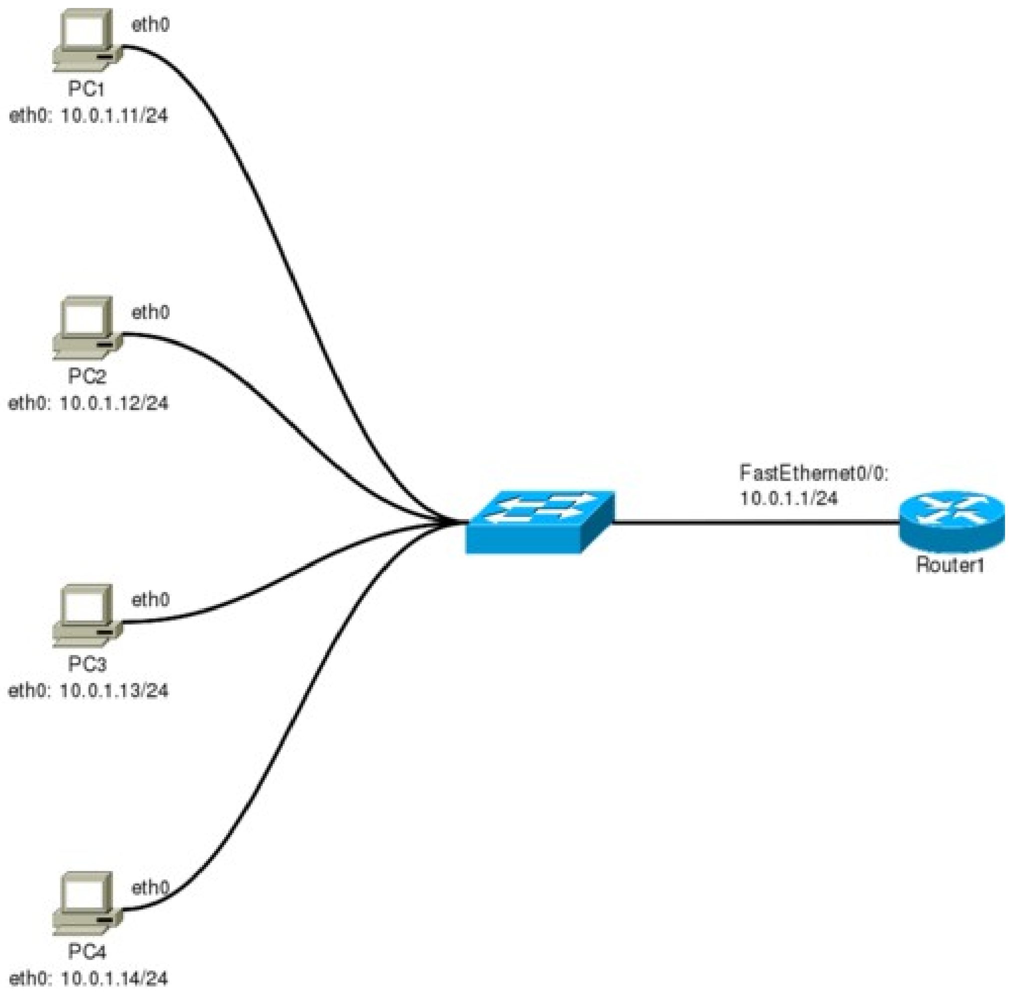
Examine the time frame after the second mark: What and how many IGMP messages do you observe?

Again two IGMP messages by PC2 and PC3.

Explain how adding and deleting senders and receivers to a multicast group works. Why is there no multicast support for TCP?

TCP would be rather senseless – how should acknowledge packages? And adding/deleting senders is done via specialized messages. Although deleting can be done by waiting for the TTL to kick in.

## Network Setup II



## Exercise 3

In this exercise we just enabled PIM dense mode for router 1.

## Exercise 4

We now test our setup with multiple bash-loops and socat and capture via wireshark again.

For all four receivers: Which PC is receiving data from which multicast group and which PC?

PC1 port 3333 : receives data from PC2 and PC3

PC4 port 1111 : receives data PC1

PC4 port 2222 : receives data PC1 and PC3

PC4 port 3333 : receives data PC3 and PC 2

Create a table that contains the IGMP message types you captured. Give typical TTL values and their destination IP addresses!

| Message Types | Info | TTL | Destination |
|---|---|---|---|
| PIMv2 | Hello | 1 | 224.0.0.13 |
| IGMPv2 | Membership query general | 1 | 224.0.0.1 |
| IGMPv3 | Membership report group 224.0.1.40 | 1 | 224.0.1.40 |

## Exercise 5

For this exercise we enabled responding to multicast pings for all PCs. Then we pinged to the multicast address 224.0.0.1 from PC4.

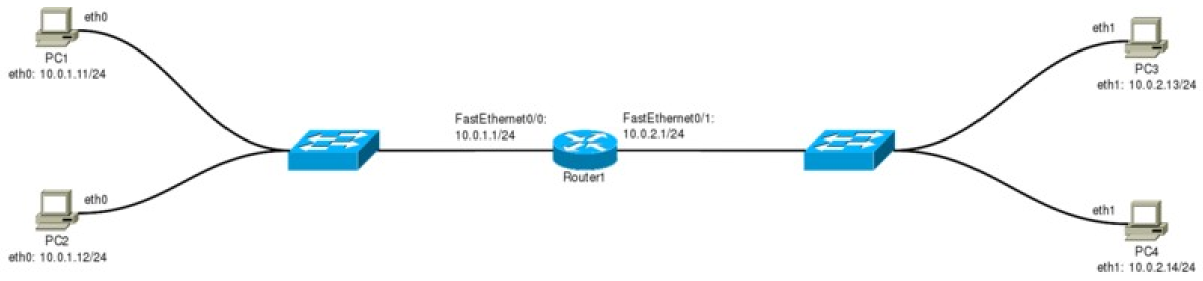
Give the numbers you wrote down.

3 nodes responded.

When multiple receivers were running on the same host, did you receive multiple replies from these hosts?

Yes.

## Network Setup III



## Exercise 6

We configured the network as shown in the network setup III. We disabled unused interfaces and removed unconnected routes from the routing tables. The default routes were set on router1. Router one got it’s PIM-DM (dense mode) enabled on both interfaces. After that we started the senders/receivers with soccat-bash-loops again.

Note for each PC from which sender multicast packets are received. How many reports are sent for each IGMP query?

PC1 wireshark first time for single IGMP query it send 4 reports and for second query it sends only two reports. and after some time it started sending 1 report for 1 query.

PC3 wireshark first two IGMP query it sends 8 reports later it sends 3 reports and then sends no reports.

What is the interval between the IGMP messages? Are all PCs sending those messages?

Default query interval 125 sec, default timeout interval 260. only PC1 and PC4 are sending IGMP messages.

## Exercise 7

We enabled multicast forwarding for router 1 and did the same testing setup like the last exercises.

What messages were received in what receiver?

PC1 gets PC2

PC4 gets PC3

Given the Wireshark capture, determine what IP address the IGMP queries and IGMP reports are sent to.

IGMP queries: 224.0.0.1, 224.0.10.3,224.0.10.2,224.0.10.1,224.0.1.40

IGMP reports: 224.0.10.3,224.0.10.2,224.0.10.1

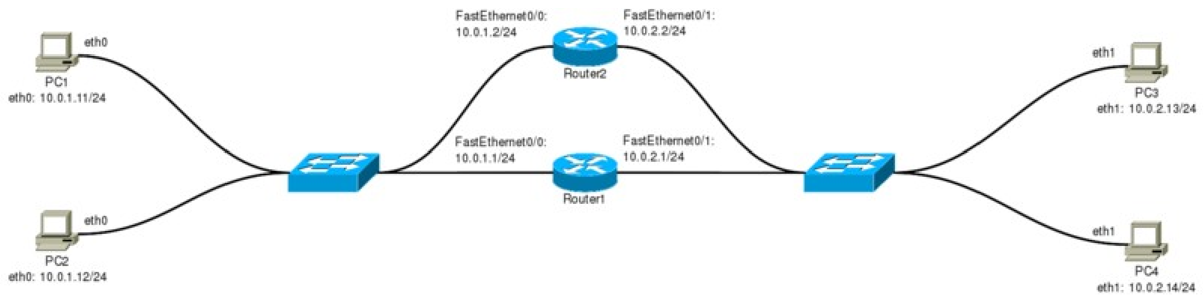
What can you conclude from this on the behavior of Router1 when it wants to receive IGMP reports?

PC4 received from PC1 and 3 - even after changing the ip address of the R1. Even after ip address change R1 is querying.

How many reports are sent for each IGMP query?

Tts sends 4 reports and later two reports.

## Exercise 8



We added router 2 to the network setup and configured it.

Before you changed the IP address of Router1:

Which router is the querier?

Router 1

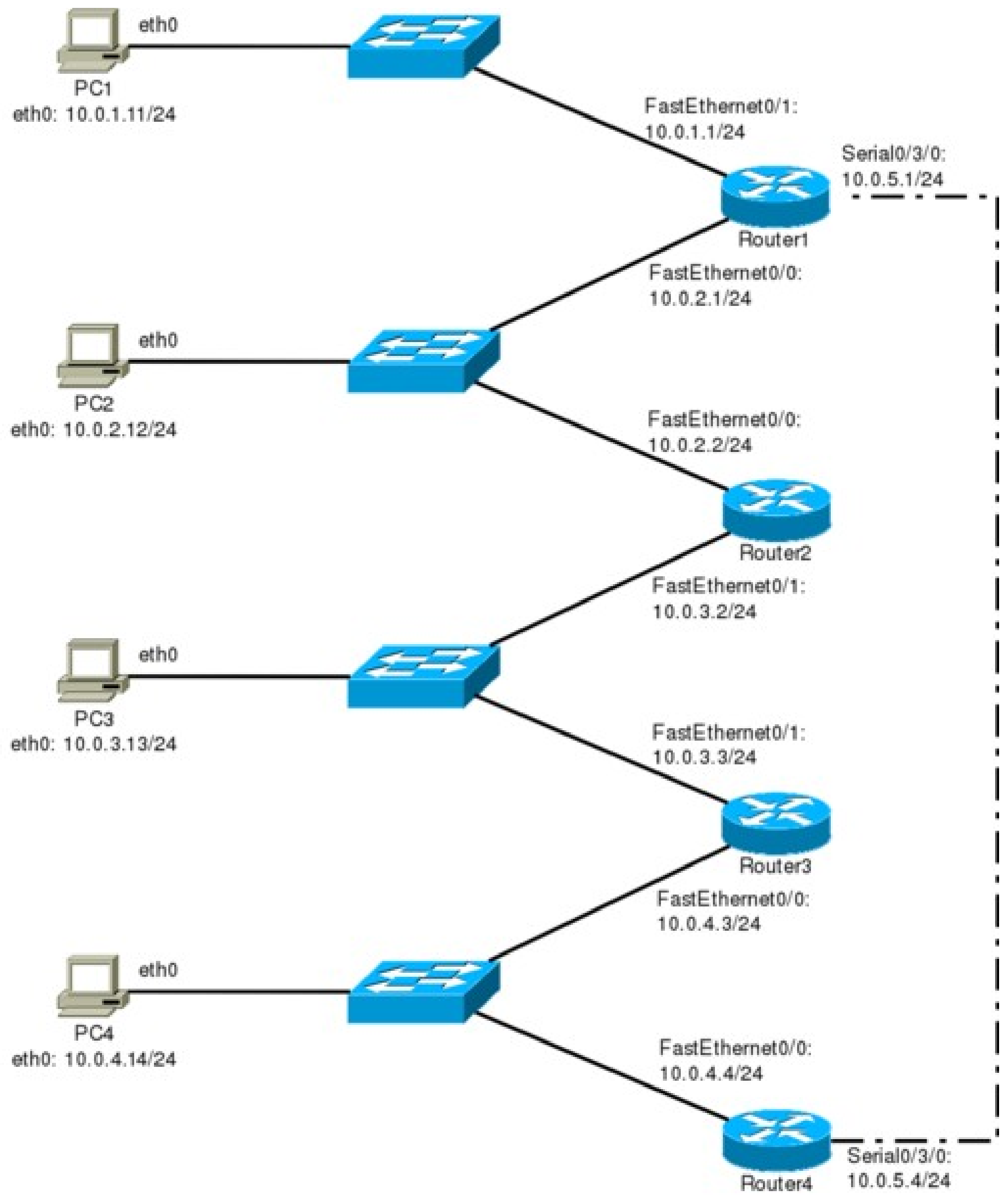
Which router is the forwarder?

Router 1

After you changed the IP address of Router1: Had this modification any impact on the role assignments you determined above?

Not at all.

## Exercise 9



Additionally we set the default routes for PC1-PC4 via 10.0.#.#, where # is the PC-nr. Each router was configured for multicast routing (RIP). The serial Link was configured for 64000b.

Determine the designated and querying routers for each network.

| | |
|---|---|
| Querying routers: | 10.0.1.1(R1), 10.0.2.1(R1), 10.0.3.2(R2), 10.0.4.3(R3) |
| Designated routers: | 10.0.1.1(R1), 10.0.2.2(R2), 10.0.3.3(R3), 10.0.4.4(R4) |

What multicast groups are joined by the routers?

Router 3: Multicast groups joined by this system (number of users): 224.0.1.40(1)

Router 2: Multicast groups joined by this system (number of users): 224.0.1.40(1)

## Exercise 10

In this exercise we explore the PIM dense mode, which uses prune and graft message to construct a shortes path tree for multicast messages.

What PIM message types do you observe?

> PC1 just observes PIM-hello messages. PC3 additionally observes PIM Join/Prune messages

Do you observe any IGMP messages that are related to this sender? List source nodes and message types.

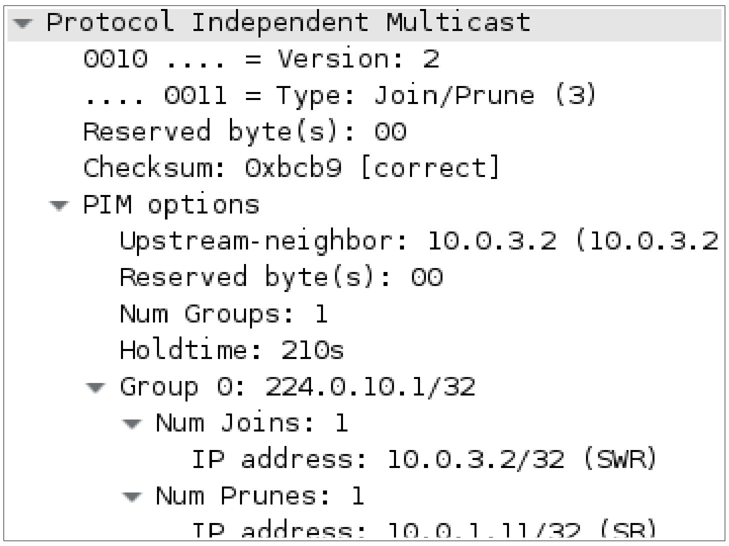
> Yes, but only Membership queries to 224.0.0.1 from PC1 (10.0.1.1)

Do you observe PIM messages that weren't sent before starting the socat sender? List the new message types.

> Just RIPv2 Responses.

Examine the destination address of the *PIM Prune* messages. Can you find any forwarded *PIM Prune* messages? Explain why or why not.

> The destination is always *224.0.0.13* and the MAC *01:00:5e:00:00:0d*. (→)



```
▼ Protocol Independent Multicast
    0010 .... = Version: 2
    .... 0011 = Type: Join/Prune (3)
    Reserved byte(s): 00
    Checksum: 0xbcb9 [correct]
  ▼ PIM options
      Upstream-neighbor: 10.0.3.2 (10.0.3.2
      Reserved byte(s): 00
      Num Groups: 1
      Holdtime: 210s
    ▼ Group 0: 224.0.10.1/32
      ▼ Num Joins: 1
          IP address: 10.0.3.2/32 (SWR)
      ▼ Num Prunes: 1
          IP address: 10.0.1.11/32 (SR)
```

Are any of the multicast packets actually forwarded?

> We couldn't observe any forwarded multicast packages.

What new PIM message type do you observe?

> Supposedly the Join/Prune message, but this was already shown in the first capture. Additionally we captured 2 assert packages which is too less to see a flooding period (next question).

Determine the flooding period.

> We couldn' t observe static periods. In our case we had times from 20s to 40s for the join/prune messages.

(We then removed the link to between router 4 and 10.0.4.0/24)

How long does it take for all routes to recover?

We could see a recovering in about 5s.

(We reconnected the previously removed link)

Use the previous evaluations to explain how the PIM routesr build their distribution tree.

It implicitly builds shortest-path trees by flooding multicast traffic domain wide, and then pruning back branches of the tree where no receivers are present. PIM-DM is straightforward to implement but generally has poor scaling properties.

## Exercise 11

Same setup like before, but with two additional senders on PC2/4.

Sketch all multicast distribution trees.

It’s really simple like the setup itself. R1 → R2 → R3 → R4 . And each one to its respective network. As these routers are the querying routers, they can provide and take all the necessary information.

Include a multicast routing table in your lab report.

```
Router2#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
 L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group,
      V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode
(*, 224.0.10.1), 00:19:13/00:02:59, RP 10.0.3.2, flags: SJC
 Incoming interface: Null, RPF nbr 0.0.0.0
 Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:01:40/00:00:00
    FastEthernet0/1, Forward/Dense, 00:19:13/00:00:00
(10.0.1.11, 224.0.10.1), 00:19:13/00:02:29, flags: PJTX
 Incoming interface: FastEthernet0/0, RPF nbr 10.0.2.1
 Outgoing interface list:
    FastEthernet0/1, Prune/Dense, 00:00:25/00:02:34
(*, 224.0.1.40), 00:39:41/00:03:22, RP 10.0.3.2, flags: SJCL
 Incoming interface: Null, RPF nbr 0.0.0.0
 Outgoing interface list:
    FastEthernet0/1, Forward/Dense, 00:39:41/00:00:00
    FastEthernet0/0, Forward/Dense, 00:39:41/00:00:00
```

# Excise 12

Here we gather information for some of the group and infos for multicast.

```
Router1# mrinfo

10.0.2.1 [version  12.4] [flags: PMSA]:
10.0.2.1 -> 10.0.2.2 [1/0/pim]
10.0.1.1 -> 0.0.0.0 [1/0/pim/querier/leaf]
10.0.5.1 -> 10.0.5.4 [1/0/pim]
```

```
Router2# mrinfo

10.0.2.2 [version  12.4] [flags: PMA]:
10.0.2.2 -> 10.0.2.1 [1/0/pim/querier]
10.0.3.2 -> 10.0.3.3 [1/0/pim]
```

```
Router3# mrinfo

10.0.3.3 [version  12.4] [flags: PMA]:
10.0.3.3 -> 10.0.3.2 [1/0/pim/querier]
10.0.4.3 -> 10.0.4.4 [1/0/pim]
```

```
Router4# mrinfo

10.0.4.4 [version  12.4] [flags: PMA]:
10.0.4.4 -> 10.0.4.3 [1/0/pim/querier]
10.0.5.4 -> 10.0.5.1 [1/0/pim]
```

```
Router4#mtrace 10.0.1.11 10.0.3.13 224.0.10.1
Type escape sequence to abort.
Mtrace from 10.0.1.11 to 10.0.3.13 via group 224.0.10.1
From source (?) to destination (?)
Querying full reverse path...
 0  10.0.3.13
-1  10.0.4.3 PIM Prune sent upstream [10.0.1.0/24]
-2  10.0.4.4 PIM Prune sent upstream [10.0.1.0/24]
-3  10.0.5.1 PIM  [10.0.1.0/24]
```

Do any of the commands in step 3 display information that you cannot obtain using mrinfo or mtrace?

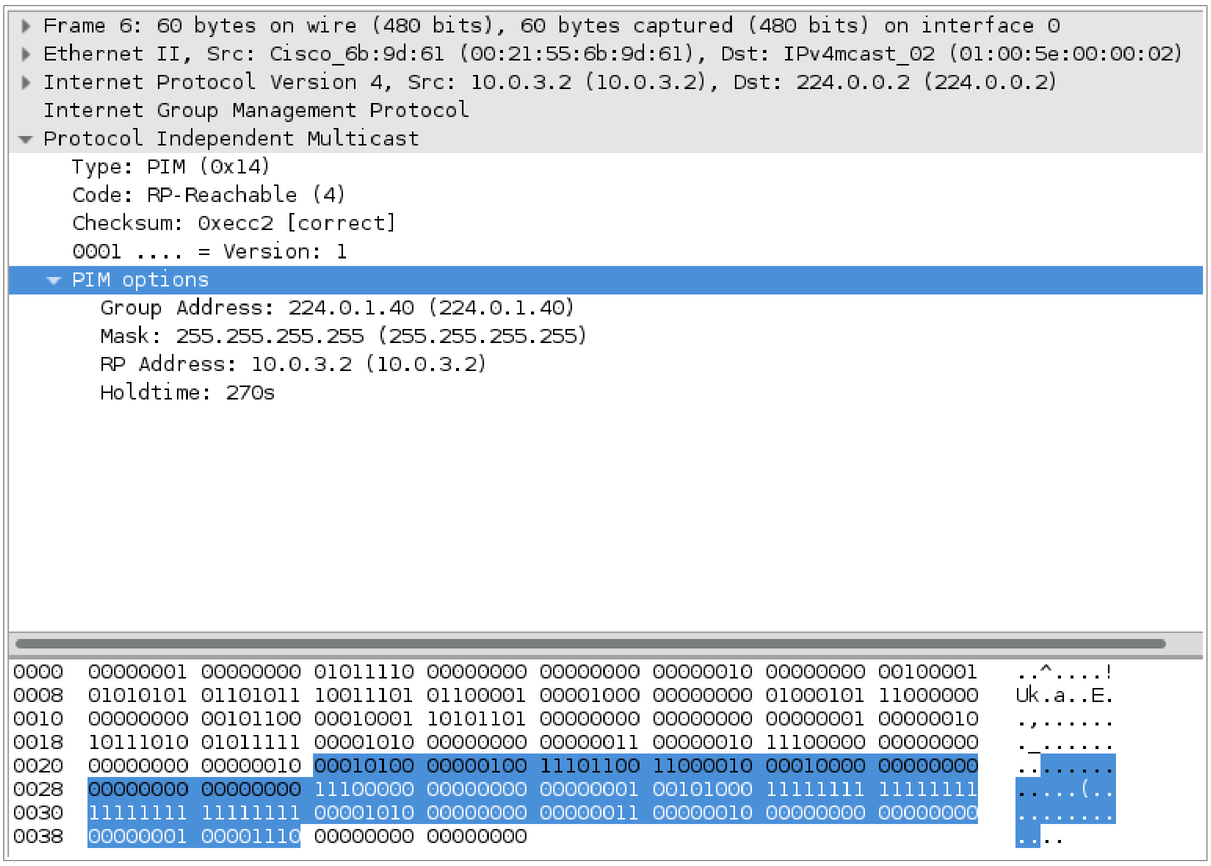
```
Router1#show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
     S - State Refresh Capable
Neighbor         Interface               Uptime/Expires    Ver   DR
Address                                                          Prio/Mode
10.0.2.2         FastEthernet0/0         00:51:29/00:01:26 v2    1 / DR S P
10.0.5.4         Serial0/3/0             00:58:10/00:01:35 v2    1 / S P
```

## Exercise 13

Here we switched to PIM sparse mode.

What new PIM message types do you observe? Attach a detailed printout of one specimen for each type.

We observed just **RP-Reachable**



```
▸ Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: Cisco_6b:9d:61 (00:21:55:6b:9d:61), Dst: IPv4mcast_02 (01:00:5e:00:00:02)
▸ Internet Protocol Version 4, Src: 10.0.3.2 (10.0.3.2), Dst: 224.0.0.2 (224.0.0.2)
  Internet Group Management Protocol
▾ Protocol Independent Multicast
    Type: PIM (0x14)
    Code: RP-Reachable (4)
    Checksum: 0xecc2 [correct]
    0001 .... = Version: 1
  ▾ PIM options
      Group Address: 224.0.1.40 (224.0.1.40)
      Mask: 255.255.255.255 (255.255.255.255)
      RP Address: 10.0.3.2 (10.0.3.2)
      Holdtime: 270s

0000  00000001 00000000 01011110 00000000 00000000 00000010 00000000 00100001   ..^....!
0008  01010101 01101011 10011101 01100001 00001000 00000000 01000101 11000000   Uk.a..E.
0010  00000000 00101100 00010001 10101101 00000000 00000000 00000001 00000010   .,......
0018  10111010 01011111 00001010 00000000 00000011 00000010 11100000 00000000   ._......
0020  00000000 00000010 00010100 00000100 11101100 11000010 00010000 00000000   ........
0028  00000000 00000000 11100000 00000000 00000001 00101000 11111111 11111111   .....(..
0030  11111111 11111111 00001010 00000000 00000011 00000010 00000000 00000000   ........
0038  00000001 00001110 00000000 00000000                                       ....
```

For one router include the contents of the multicast routing table in your lab report

```
Router2#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode
(*, 224.0.10.1), 00:09:34/00:03:15, RP 10.0.3.2, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/1, Forward/Sparse, 00:06:11/00:03:15
    FastEthernet0/0, Forward/Sparse, 00:08:30/00:02:24
(*, 224.0.1.40), 00:10:06/00:03:28, RP 10.0.3.2, flags: SJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:09:32/00:02:49
    FastEthernet0/1, Forward/Sparse, 00:09:52/00:03:27
```

Draw a schematic overview of the RPT

## Exercise 14

Same setup as before. Restarted wireshark sessions and new pim threshold of 0 instead of infinity. Starting senders and receivers again.

Use the saved data to explain the events that occured while mode switching.

The network gets reconfigured for the ospf protocol with an spt-threshold of 0.

```
P Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode
```

```
Router1#show ip mroute
(*, 224.0.10.1), 00:03:18/00:02:34, RP
10.0.3.2, flags: SP
  Incoming interface: FastEthernet0/0, RPF
nbr 10.0.2.2
  Outgoing interface list: Null
(*, 224.0.1.40), 00:27:20/stopped, RP
10.0.3.2, flags: SJCL
  Incoming interface: FastEthernet0/0, RPF
```

```
Router2#show ip mroute
(*, 224.0.10.1), 00:26:54/00:02:17, RP
10.0.3.2, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse,
00:02:49/00:02:03
    FastEthernet0/1, Forward/Sparse,
00:23:31/00:02:38
```

| | |
|---|---|
| nbr 10.0.2.2<br>  Outgoing interface list:<br>    Serial0/3/0, Forward/Sparse,<br>00:27:20/00:00:00 | (*, 224.0.1.40), 00:27:26/00:03:15, RP<br>10.0.3.2, flags: SJCL<br>  Incoming interface: Null, RPF nbr 0.0.0.0<br>  Outgoing interface list:<br>    FastEthernet0/0, Forward/Sparse,<br>00:26:52/00:03:14<br>    FastEthernet0/1, Forward/Sparse,<br>00:27:12/00:02:51 |
| Router3#show ip mroute<br>(*, 224.0.10.1), 00:23:03/00:03:24, RP<br>10.0.3.2, flags: SJC<br>  Incoming interface: FastEthernet0/1, RPF<br>nbr 10.0.3.2<br>  Outgoing interface list:<br>    FastEthernet0/0, Forward/Sparse,<br>00:02:03/00:03:24<br><br>(*, 224.0.1.40), 01:21:49/00:02:45, RP<br>10.0.3.2, flags: SJCL<br>  Incoming interface: FastEthernet0/1, RPF<br>nbr 10.0.3.2<br>  Outgoing interface list:<br>    FastEthernet0/0, Forward/Sparse,<br>01:21:50/00:02:46 | Router4#show ip mroute<br>(*, 224.0.10.1), 00:01:44/00:02:11, RP<br>10.0.3.2, flags: SJPC<br>  Incoming interface: FastEthernet0/0, RPF<br>nbr 10.0.4.3<br>  Outgoing interface list: Null<br><br>(*, 224.0.1.40), 00:26:59/00:02:19, RP<br>10.0.3.2, flags: SJCL<br>  Incoming interface: FastEthernet0/0, RPF<br>nbr 10.0.4.3<br>  Outgoing interface list:<br>    Serial0/3/0, Forward/Sparse,<br>00:26:59/00:00:00 |

As we can see here, all the devices are using ospf. As the name states each router/host is using the **s**hortest **p**ath **f**irst.