Project Netwerken

Semih Can Karakoç (695258)

June 8, 2023

Contents

1 Network Topology

The main IP-address that needs to be divided into four subnets is: 58.89.30.111/24. The subnet consists of the following IP-addresses:

- **Subnet A**: 58.89.30.64/26
  - A.1: 58.89.30.80/28
  - A.2: 58.89.30.96/28
- **Subnet B**: 58.89.30.128/26
  - B.1: 58.89.30.144/28
  - B.2: 58.89.30.160/28

This is done by determining the subnet mask, calculating the subnet increment, calculating the new subnet mask and determining subnet ranges. In Figure 1 is the whole network shown, which is made with *Cisco Packet Tracer*.

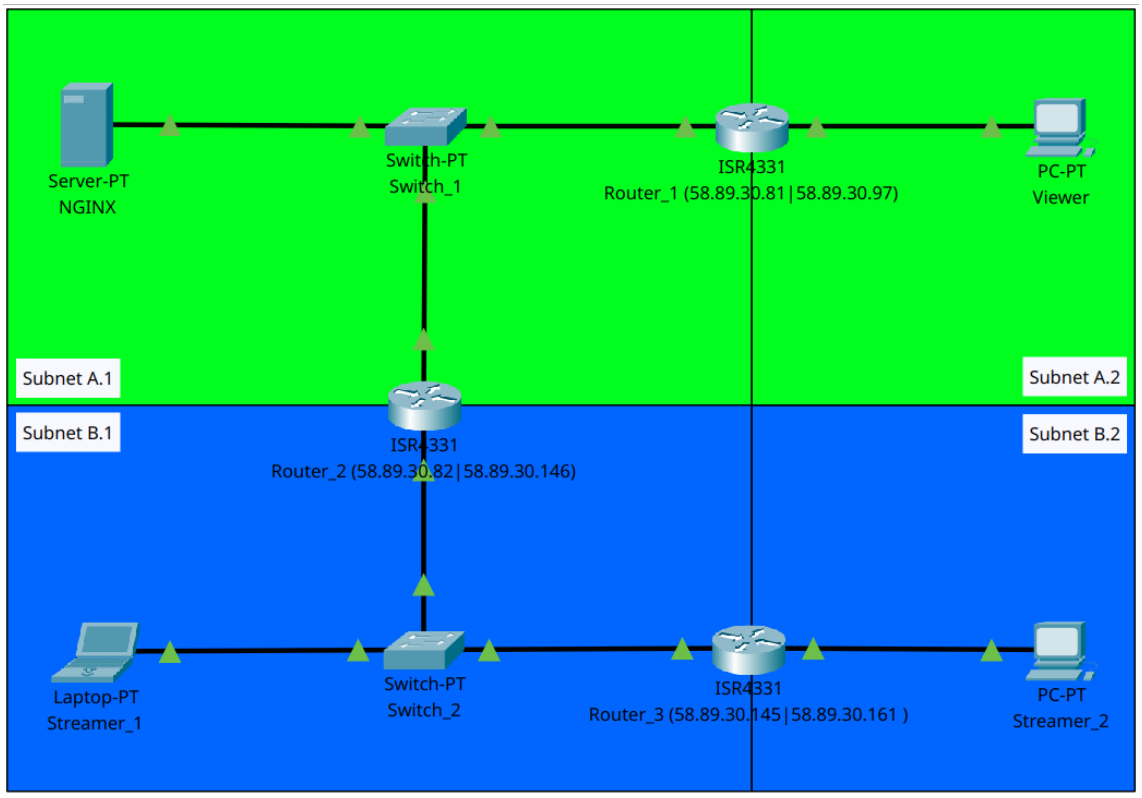

Figure 1: Our network made within *Cisco Packet Tracer*.

2 Roles

Our group for project network consists of four students and we all had a role:

- **Semih**:
  - Router 2. Main role is the connectivity between subnet *A* and *B*;
  - Streamer 1.
- **Ferhat**:
  - Router 3. Main role is the connectivity between subnet *B.1* and *B.2*. Router 3 is also a DHCP-server;
  - Streamer 2.
- **Marvin**:
  - Nginx. This server is going to receive the media stream from the Raspberry Pi, decrypt it, authenticate and serve it as both HLS and Dash.
- **Thom**:
  - Router 1. Main role is the connectivity between subnet *A.1* and *A.2*. Router 3 is also a DNS-server and DHCP-server;
  - Viewer.

3 Streaming

My role is to be both *Router_2* and *Streamer_1* in subnet B.1 as seen in Figure 1. *Router_2* has IPv4-forwarding enabled to act as a router.

3.1 Configuring IPv4-static routes

Adding an IP-route is important for connectivity between all the devices. This is done with the following Linux command: "*sudo ip route add ¡IP-1¿ via ¡IP-2¿*". *IP-1* and *IP-2* are both variable IPv4-addresses. There was also a static route added for *Router_1* to redirect any packages going from:

- 58.89.30.128/26 *(subnet A)* via 58.89.30.82 *(router_2)* with subnet mask 255.255.255.192.

There were also two static routes added for *Router_2* to redirect any packages going from:

- 58.89.30.96/28 *(subnet A.2)* via 58.89.30.81 *(subnet A.1)* with subnet mask 255.255.255.240;
- 58.89.30.160/28 *(subnet B.2)* via 58.89.30.145 *(subnet B.1)* with subnet mask 255.255.255.240.

There was also a static route added for *Router_3* to redirect any packages going from:

- 58.89.30.64/26 *(subnet B)* via 58.89.30.146 *(router_2)* with subnet mask 255.255.255.192.

After configuring the routers properly, Figure 2 and Figure 3 confirm that I (as *Router_2*) was able to ping every device in both subnet A and B.

3.2 FFmpeg streaming

In order to stream something, *streamer_1* and *streamer_2* had to install a program called *FFmpeg*. FFmpeg is a great tool to compress, convert, edit videos and images. It can also be used to stream a webcam or desktop environment. In order to stream my webcam, I entered the following command (for both *streamer_1* and *streamer_2*):

```
ffmpeg -f v4l2 -i /dev/video0 \
  -c:v libx264 -pix_fmt yuv420p -framerate 15 -g 30 -b:v 500k \
  -preset ultrafast -tune zerolatency \
  -f flv "rtmp://58.89.30.83:4000/live/cam1?streamkey=123"
```

In this command above, we captured the video source from the webcam using Video4Linux, compresses it and streams it to the RTMP-server, which in this case is: *Nginx* (58.89.30.83) in subnet A.1 seen in Figure 1. It is also possible to stream my desktop environment by using *x11grab* as shown in the following command below:

```
ffmpeg -video_size 1920x1080 -framerate 30 -f x11grab -i :0.0 \
  -c:v libx264 -pix_fmt yuv420p -framerate 15 -g 30 -b:v 500k \
  -preset ultrafast -tune zerolatency \
  -f flv "rtmp://58.89.30.83:4000/live/cam1?streamkey=123"
```

Both commands worked on *streamer_1* as well for *streamer_2*. The viewer was able to connect to the RTMP-server and fetch the stream successfully! It does work as shown in Figure 5, Figure 6, Figure 7 and Figure 8, unfortunately there is a lot of latency. In the real world however, this won't matter as much because the streamer and viewer won't be sitting in the same room anyway. So latency would not be noticed that rapidly.

```
semih@broodjesemih:~ $ ping 58.89.30.145
PING 58.89.30.145 (58.89.30.145) 56(84) bytes of data.
64 bytes from 58.89.30.145: icmp_seq=1 ttl=64 time=0.565 ms
64 bytes from 58.89.30.145: icmp_seq=2 ttl=64 time=0.483 ms
^C
--- 58.89.30.145 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1018ms
rtt min/avg/max/mdev = 0.483/0.524/0.565/0.041 ms
semih@broodjesemih:~ $ ping 58.89.30.161
PING 58.89.30.161 (58.89.30.161) 56(84) bytes of data.
64 bytes from 58.89.30.161: icmp_seq=1 ttl=64 time=0.562 ms
64 bytes from 58.89.30.161: icmp_seq=2 ttl=64 time=0.411 ms
^C
--- 58.89.30.161 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 0.411/0.486/0.562/0.075 ms
semih@broodjesemih:~ $ ping 58.89.30.162
PING 58.89.30.162 (58.89.30.162) 56(84) bytes of data.
64 bytes from 58.89.30.162: icmp_seq=1 ttl=63 time=1.87 ms
64 bytes from 58.89.30.162: icmp_seq=2 ttl=63 time=1.90 ms
^C
--- 58.89.30.162 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.865/1.880/1.896/0.015 ms
semih@broodjesemih:~ $ ping 58.89.30.147
PING 58.89.30.147 (58.89.30.147) 56(84) bytes of data.
64 bytes from 58.89.30.147: icmp_seq=1 ttl=64 time=0.654 ms
64 bytes from 58.89.30.147: icmp_seq=2 ttl=64 time=0.751 ms
^C
--- 58.89.30.147 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.654/0.702/0.751/0.048 ms
semih@broodjesemih:~ $
```

Figure 2: It's possible to ping every device on the network (pt.1).

```
semih@broodjesemih:~ $ ping 58.89.30.97
\PING 58.89.30.97 (58.89.30.97) 56(84) bytes of data.
64 bytes from 58.89.30.97: icmp_seq=1 ttl=64 time=1.58 ms
64 bytes from 58.89.30.97: icmp_seq=2 ttl=64 time=1.51 ms
^C
--- 58.89.30.97 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.513/1.545/1.578/0.032 ms
semih@broodjesemih:~ $ ping 58.89.30.81
PING 58.89.30.81 (58.89.30.81) 56(84) bytes of data.
64 bytes from 58.89.30.81: icmp_seq=1 ttl=64 time=1.57 ms
64 bytes from 58.89.30.81: icmp_seq=2 ttl=64 time=2.33 ms
^C
--- 58.89.30.81 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.565/1.947/2.330/0.382 ms
semih@broodjesemih:~ $ ping 58.89.30.82
PING 58.89.30.82 (58.89.30.82) 56(84) bytes of data.
64 bytes from 58.89.30.82: icmp_seq=1 ttl=64 time=0.148 ms
64 bytes from 58.89.30.82: icmp_seq=2 ttl=64 time=0.109 ms
^C
--- 58.89.30.82 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.109/0.128/0.148/0.019 ms
semih@broodjesemih:~ $ ping 58.89.30.83
PING 58.89.30.83 (58.89.30.83) 56(84) bytes of data.
64 bytes from 58.89.30.83: icmp_seq=1 ttl=64 time=0.427 ms
64 bytes from 58.89.30.83: icmp_seq=2 ttl=64 time=0.229 ms
^C
--- 58.89.30.83 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.229/0.328/0.427/0.099 ms
semih@broodjesemih:~ $
```

Figure 3: It's possible to ping every device on the network (pt.2).

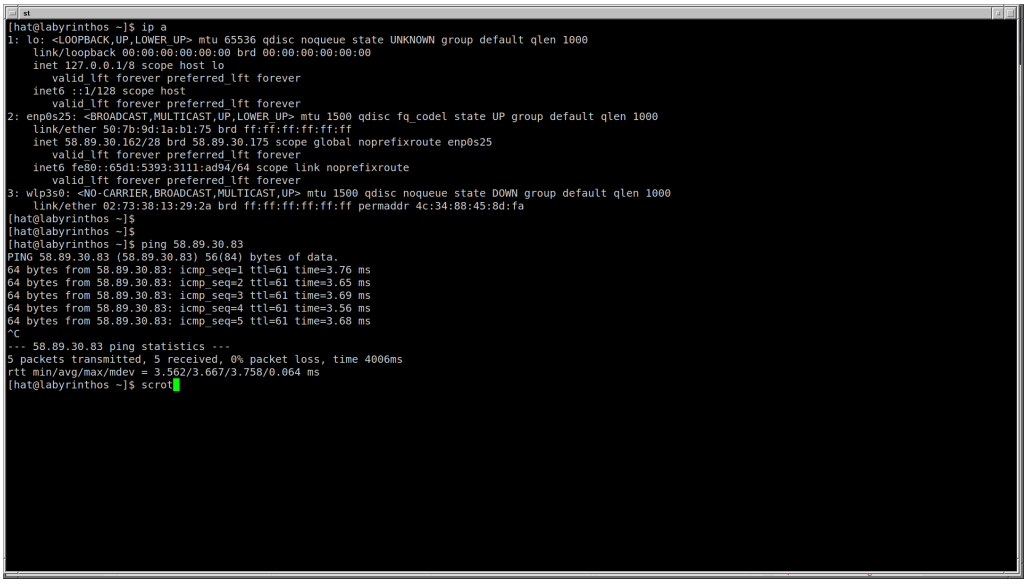

Figure 4: It's possible to ping from subnet B to A.

Figure 5: Viewer watching gameplay of Samurai Shodown II (*streamer_2*). Note that the compression artifacts does make the image quite smeary.

Figure 6: Viewer watching gameplay of Metal Slug 2 (*streamer_2*). The compression artifacts is even worse on this one.

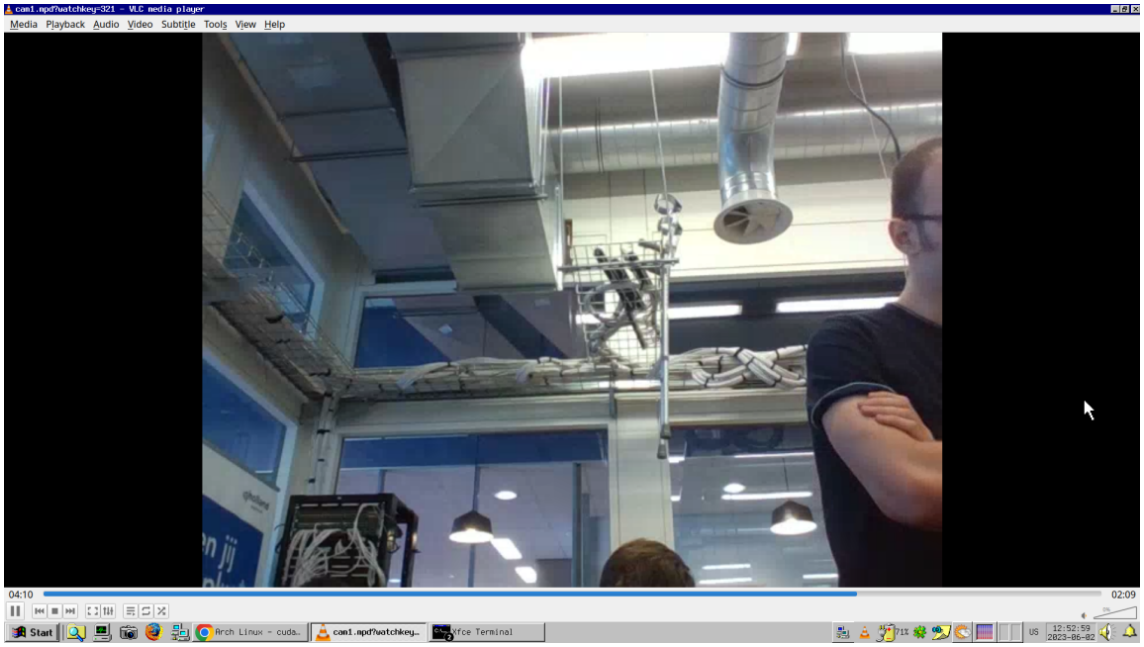

Figure 7: Viewer watching webcam of *streamer_2*.

Figure 8: *streamer_1* streaming and watching his own webcam.

4 Miscellaneous screenshots

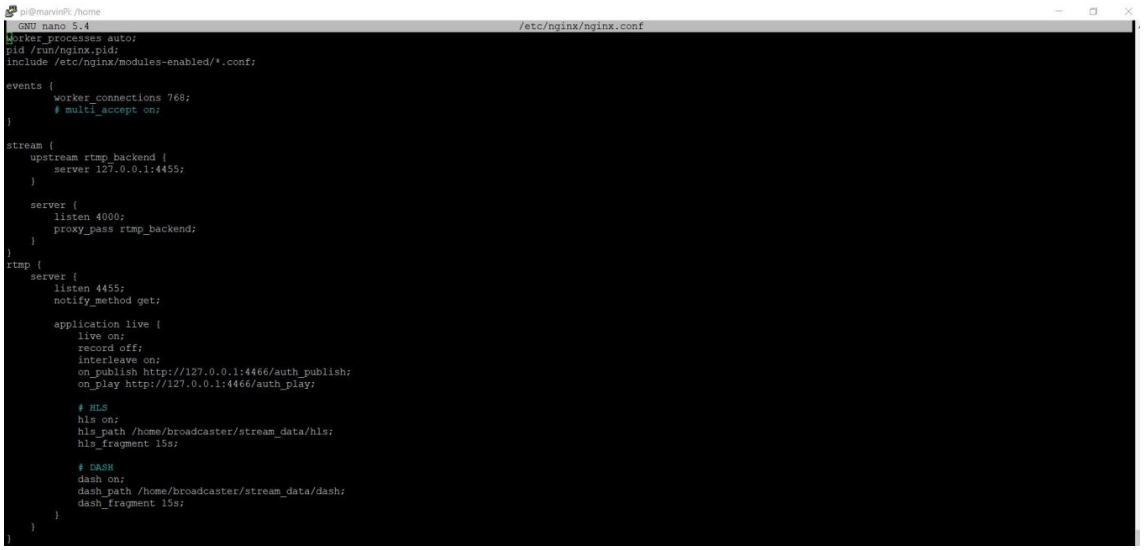

Figure 9: Nginx configuration for RMTP-server, which all has been done by Marvin (pt.1).

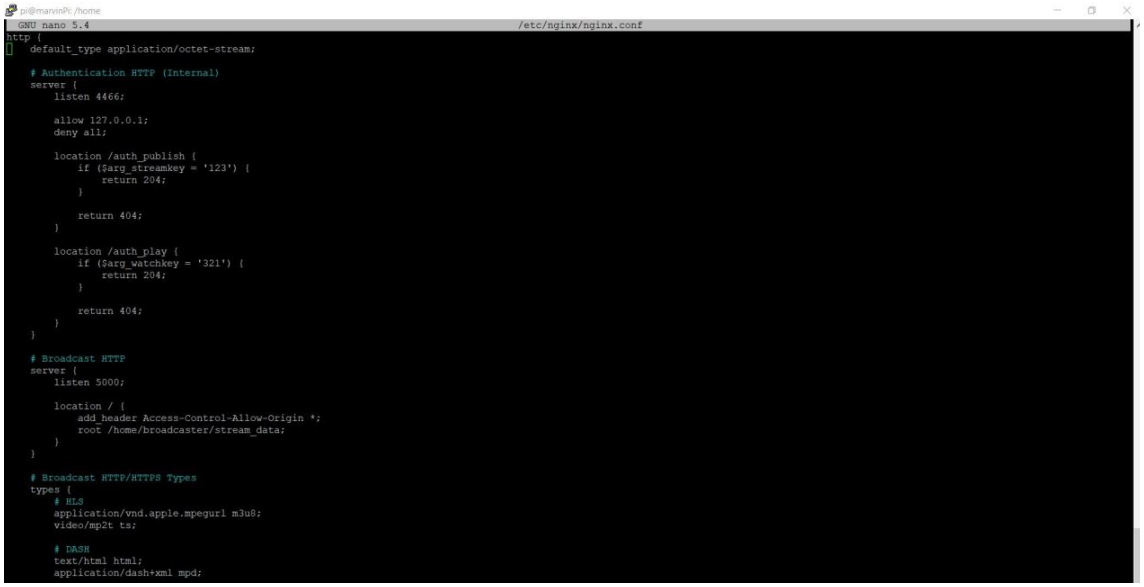

Figure 10: Nginx configuration for RMTP-server, which all has been done by Marvin (pt.2).

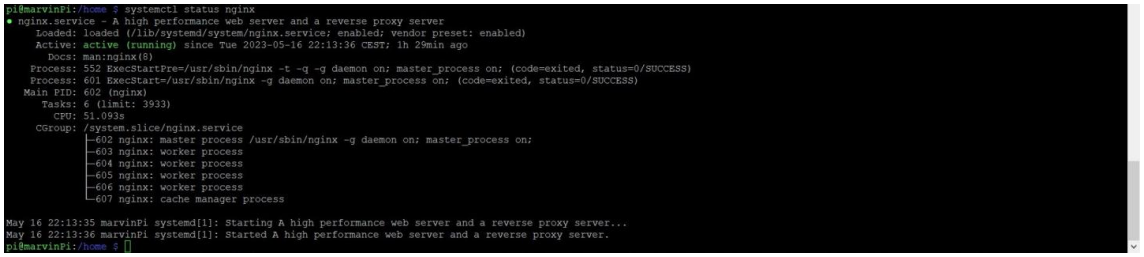

Figure 11: Proof of that the Nginx-server is running without any problems!

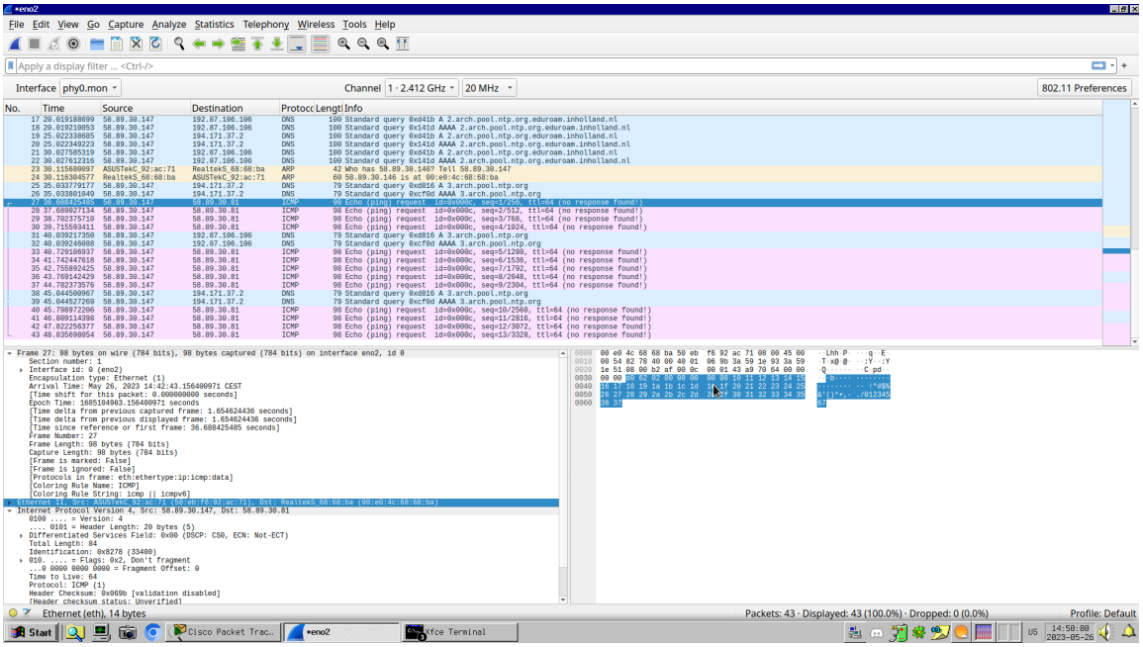

Figure 12: Here was I (Semih) troubleshooting the ICMP (ping) request via my laptop (58.89.30.147) to *router_1* (58.89.30.81).

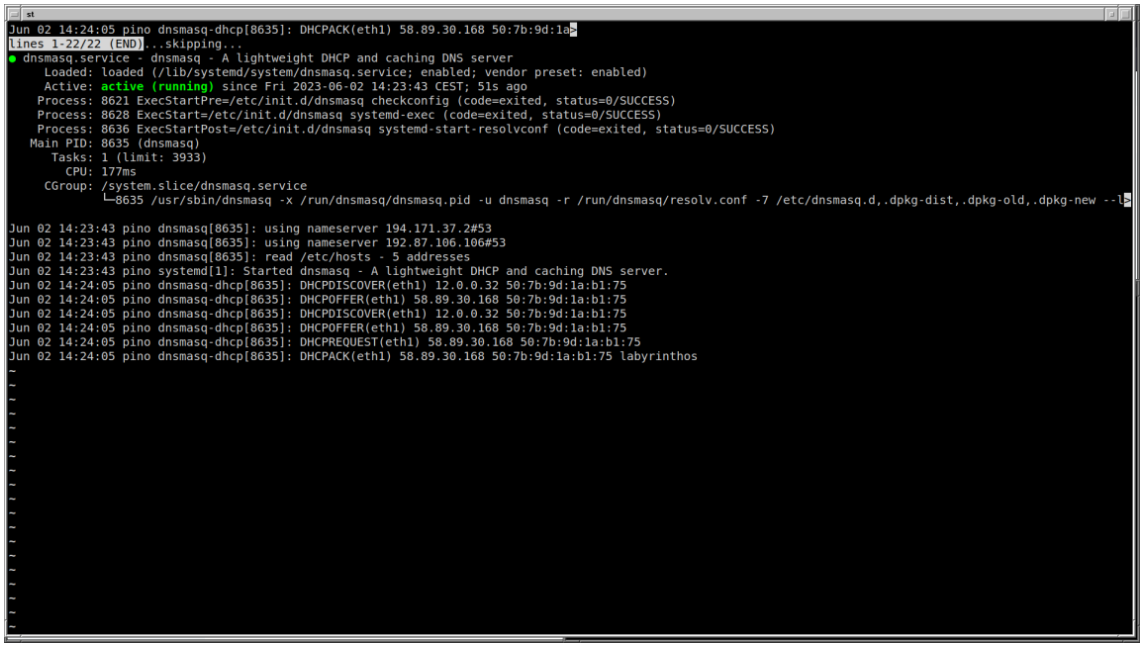

Figure 13: DHCP-server is working fine in subnet B.2 (*router_3*).

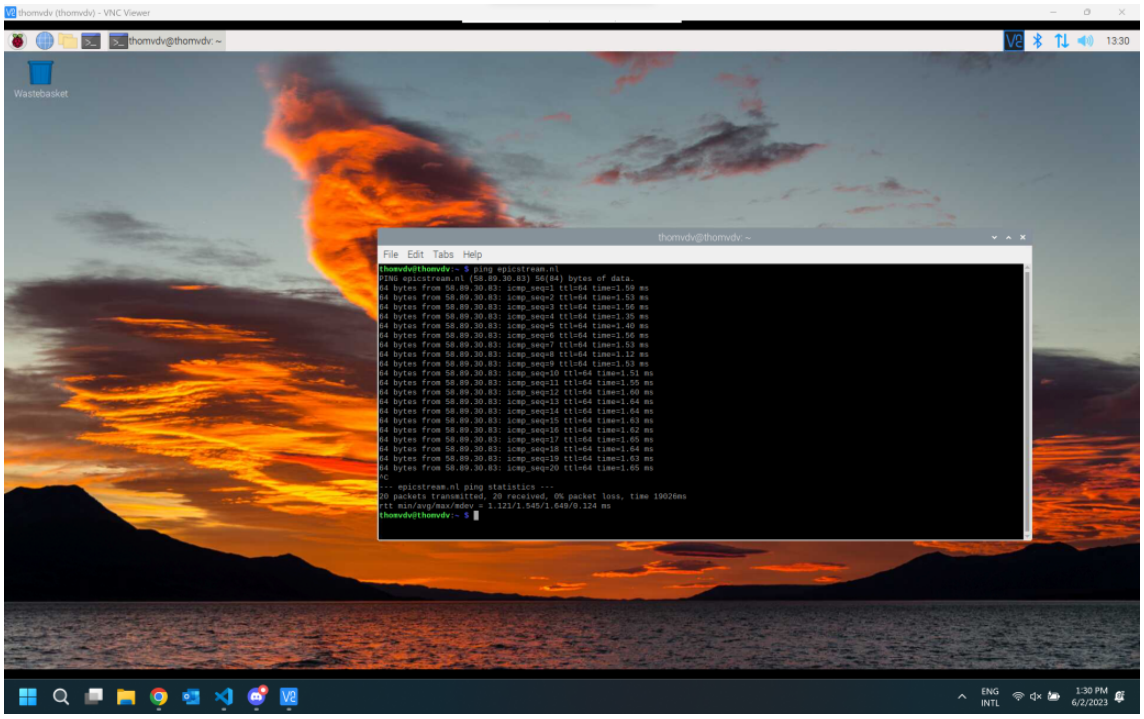

Figure 14: DNS-server is working fine in subnet A.1 *router_1*.

```
# Example static IP configuration:
interface eth1
static ip_address=58.89.30.146/28
static routers=58.89.30.145
#static domain_name_servers=58.89.30.81
#tatic routers=58.89.30.82

interface eth0
static ip_address=58.89.30.82/28
static routers=58.89.30.81
#static domain_name_servers=58.89.30.81
#static routers=58.89.30.146
```

Figure 15: dhcpcd.conf file of *router_2*.

```
pi@marvinPi:/home $ ls
broadcaster  broadcastercache  pi
pi@marvinPi:/home $ cd broadcaster
pi@marvinPi:/home/broadcaster $ ls
conf  log  run  stream_data
pi@marvinPi:/home/broadcaster $
```

Figure 16: ls command in MarvinPi (*Nginx*).

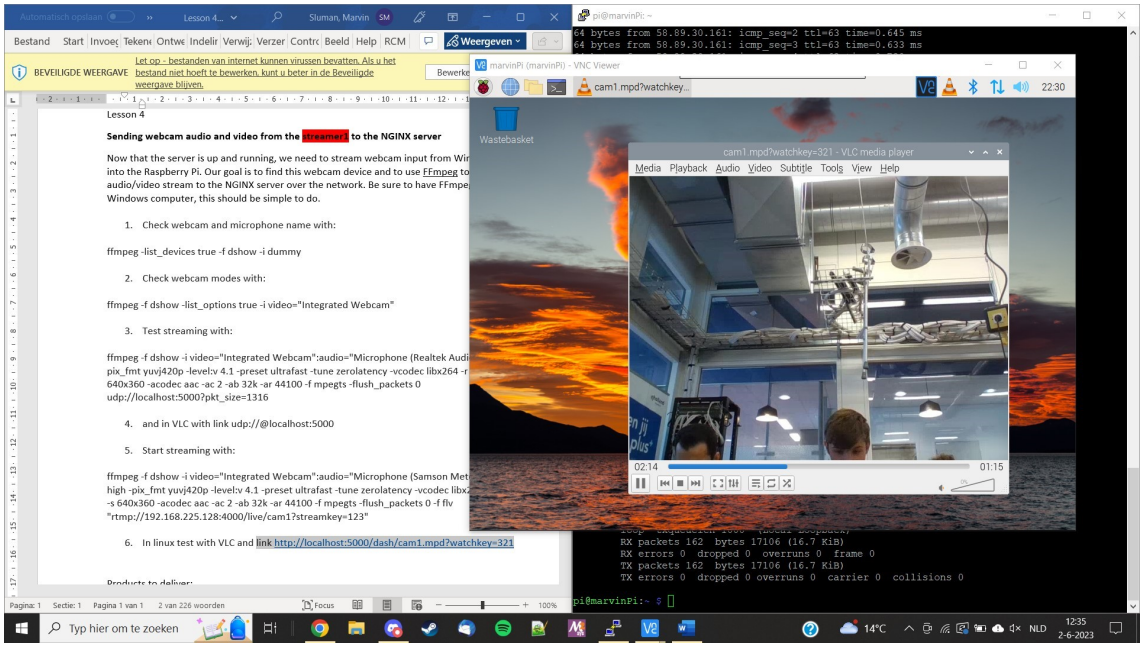

Figure 17: Marvin watching *streamer_2* (via *Nginx*).

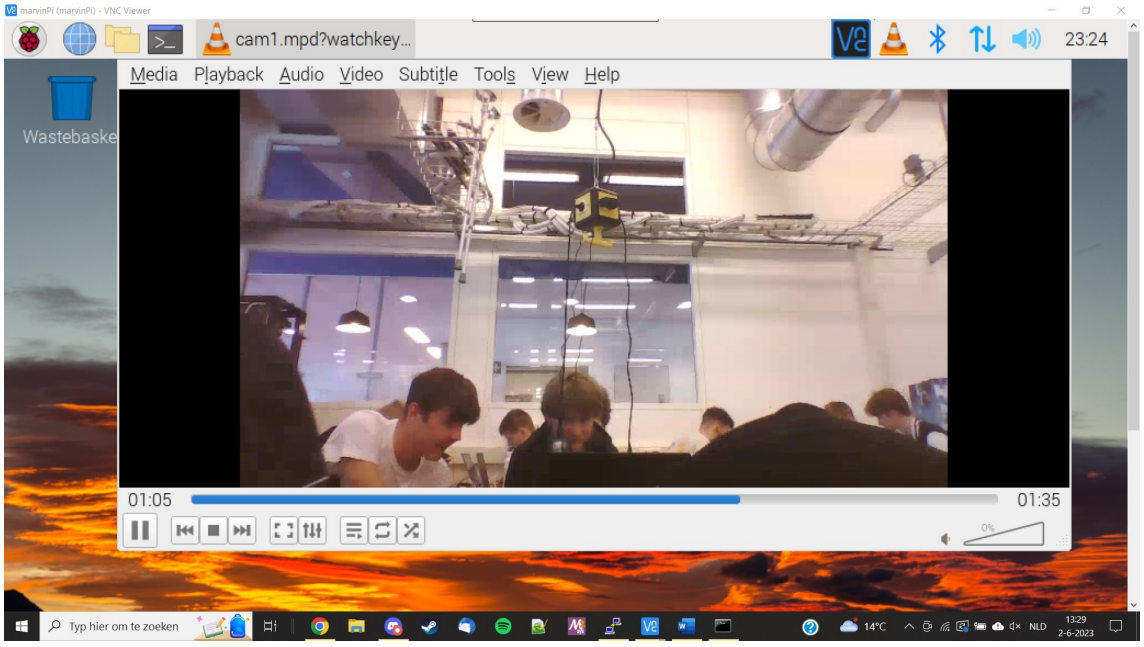

Figure 18: Marvin watching *streamer_1* (via *Nginx*).

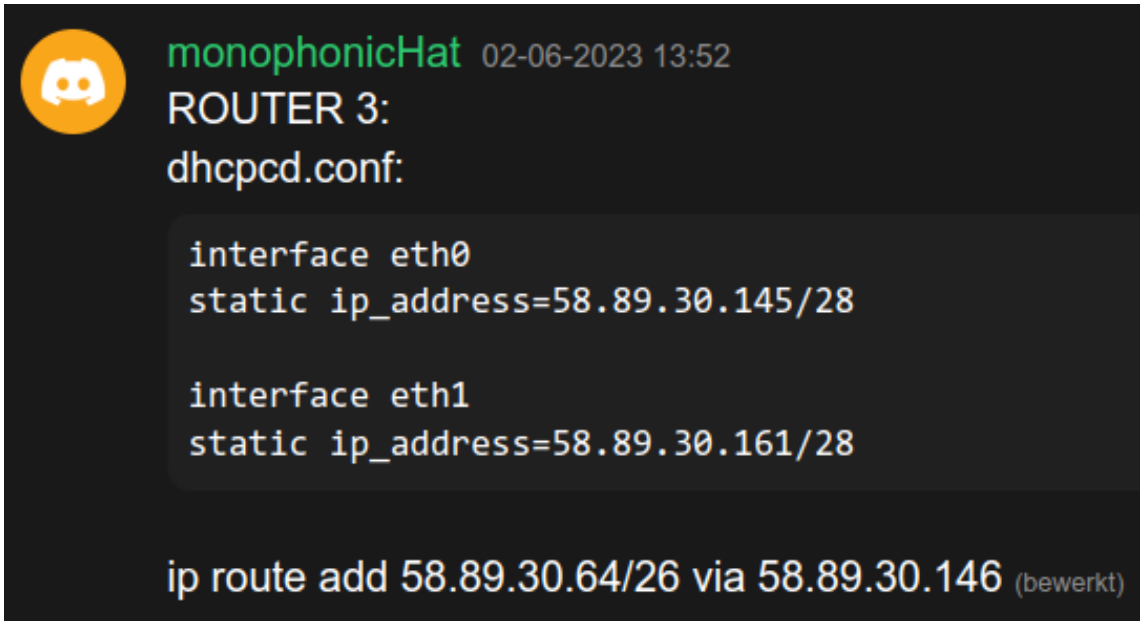

Figure 19: *Router_3* dhcpcd.conf file.

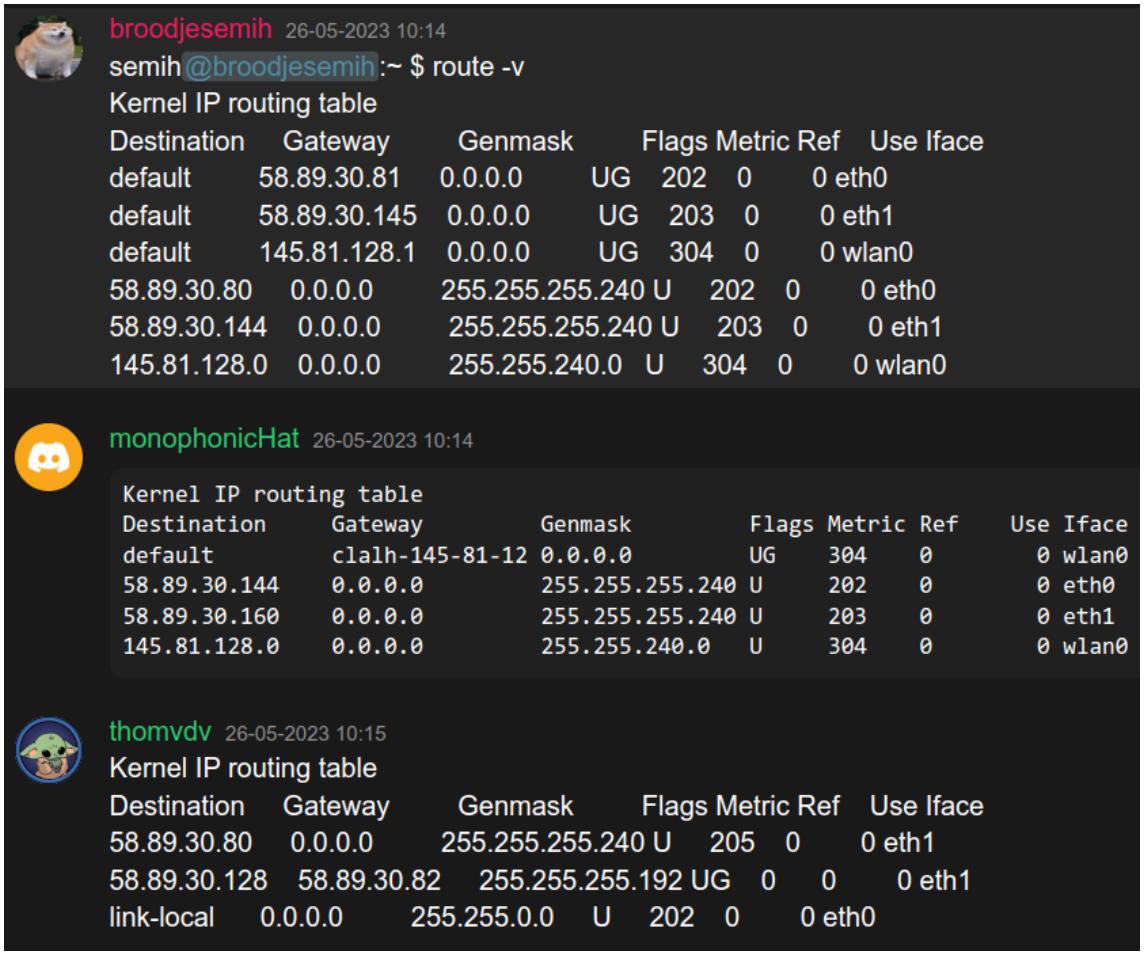

Figure 20: All “route -v” outputs.

```
Router 1 settings:
/etc/dhcpcd.conf:
interface eth1
static ip_address=58.89.30.81/28
interface eth0
inform 58.89.30.97/28

/etc/dnsmasq.conf
interface=eth0
dhcp-range=58.89.30.100,58.89.30.110,24h

conf-dir=/etc/dnsmasq.d

/etc/dnsmasq.d/epicstream.conf
address=/epicstream.nl/58.89.30.83

/etc/sysctl.conf
net.ipv4.ip_forward=1

Na opstarten: sudo ip route add 58.89.30.128/26 via 58.89.30.82
```

Figure 21: *Router_1* config files.