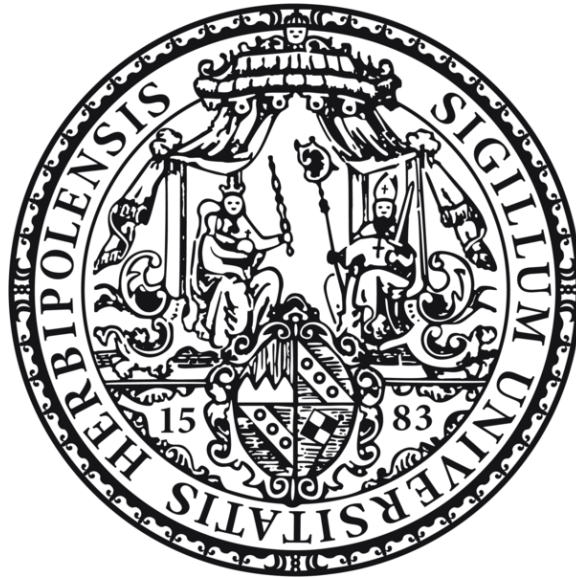


Hardwarepraktikum Internet-Technologien

Task 7: Packet-loss and Latency



Julius-Maximilians-Universität Würzburg

Chair of Computer Science III

A project report submitted by **Group 11**

Enrique Mesonero Ronco

Manuel Calvo Martín

Pablo E. Ortega Ureña

Summer term 2022

July, 2022

Contents

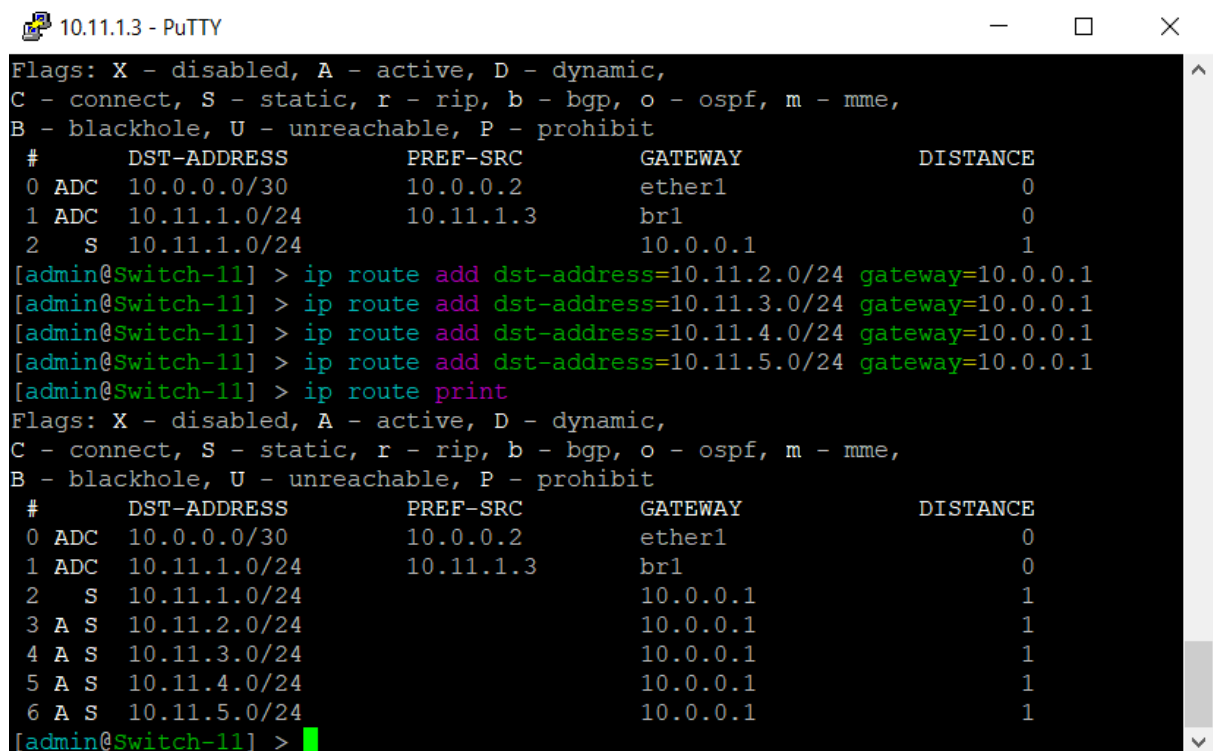
7. Packet-loss and Latency	3
7.2. Network configuration	3
7.3. Subjective Influence of Latency and Packet Loss	3
7.3.1. Latency minimisation and buffers	7
7.3.2. Packet loss and latency	7

7. Packet-loss and Latency

7.2. Network configuration

In order to emulate the different network conditions, we connect the NetEm to both the switch (on port 4) and the router (on the Internet port). We use the USB adapter to connect to the router and the native Ethernet interface to connect to the switch.

In addition, we configure the routing back to the static routes of section 5. For us to change the conditions, we change the way of reaching the Raspberry from the PC, changing the physical connections.



```
10.11.1.3 - PuTTY
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
#      DST-ADDRESS      PREF-SRC      GATEWAY      DISTANCE
0 ADC  10.0.0.0/30       10.0.0.2      ether1        0
1 ADC  10.11.1.0/24      10.11.1.3     br1           0
2  S   10.11.1.0/24              10.0.0.1      1
[admin@Switch-11] > ip route add dst-address=10.11.2.0/24 gateway=10.0.0.1
[admin@Switch-11] > ip route add dst-address=10.11.3.0/24 gateway=10.0.0.1
[admin@Switch-11] > ip route add dst-address=10.11.4.0/24 gateway=10.0.0.1
[admin@Switch-11] > ip route add dst-address=10.11.5.0/24 gateway=10.0.0.1
[admin@Switch-11] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
#      DST-ADDRESS      PREF-SRC      GATEWAY      DISTANCE
0 ADC  10.0.0.0/30       10.0.0.2      ether1        0
1 ADC  10.11.1.0/24      10.11.1.3     br1           0
2  S   10.11.1.0/24              10.0.0.1      1
3 A S  10.11.2.0/24              10.0.0.1      1
4 A S  10.11.3.0/24              10.0.0.1      1
5 A S  10.11.4.0/24              10.0.0.1      1
6 A S  10.11.5.0/24              10.0.0.1      1
[admin@Switch-11] >
```

Figure 7.1. Switch Configuration

7.3. Subjective Influence of Latency and Packet Loss

The first thing we do is try a ping test. With the direct connection, our pings happen at a rate of around 0.6ms. However, with the NetEm connected, this almost triples, to around 1.6ms. The effects of the NetEm are already noticeable by the SSH connection, which is much slower.

```

hwp@hwp-l: ~
64 Bytes von 10.11.1.1: icmp_seq=19 ttl=62 Zeit=0.573 ms
64 Bytes von 10.11.1.1: icmp_seq=20 ttl=62 Zeit=0.606 ms
64 Bytes von 10.11.1.1: icmp_seq=21 ttl=62 Zeit=0.709 ms
64 Bytes von 10.11.1.1: icmp_seq=22 ttl=62 Zeit=0.728 ms
64 Bytes von 10.11.1.1: icmp_seq=23 ttl=62 Zeit=0.674 ms
64 Bytes von 10.11.1.1: icmp_seq=24 ttl=62 Zeit=0.682 ms

--- 10.11.1.1 ping statistics ---
24 Pakete übertragen, 24 empfangen, 0% Paketverlust, Zeit 23555ms
rtt min/avg/max/mdev = 0.542/0.661/0.748/0.057 ms
hwp@hwp-l:~$ ping 10.11.1.1
PING 10.11.1.1 (10.11.1.1) 56(84) Bytes Daten.
64 Bytes von 10.11.1.1: icmp_seq=1 ttl=62 Zeit=2.52 ms
64 Bytes von 10.11.1.1: icmp_seq=2 ttl=62 Zeit=1.52 ms
64 Bytes von 10.11.1.1: icmp_seq=3 ttl=62 Zeit=1.46 ms
64 Bytes von 10.11.1.1: icmp_seq=4 ttl=62 Zeit=1.42 ms
64 Bytes von 10.11.1.1: icmp_seq=5 ttl=62 Zeit=1.43 ms
64 Bytes von 10.11.1.1: icmp_seq=6 ttl=62 Zeit=1.35 ms
^C
--- 10.11.1.1 ping statistics ---
6 Pakete übertragen, 6 empfangen, 0% Paketverlust, Zeit 5009ms
rtt min/avg/max/mdev = 1.353/1.615/2.518/0.406 ms

```

Figure 7.2. ICMP without and with the NetEm

Our second test is establishing a connection using Netcat. For that, we put the Raspberry listening on port 3333, using the command `nc -l 3333`. By default, Netcat uses TCP for the connections. If we wanted to use UDP, we should have put the flag `-u`.

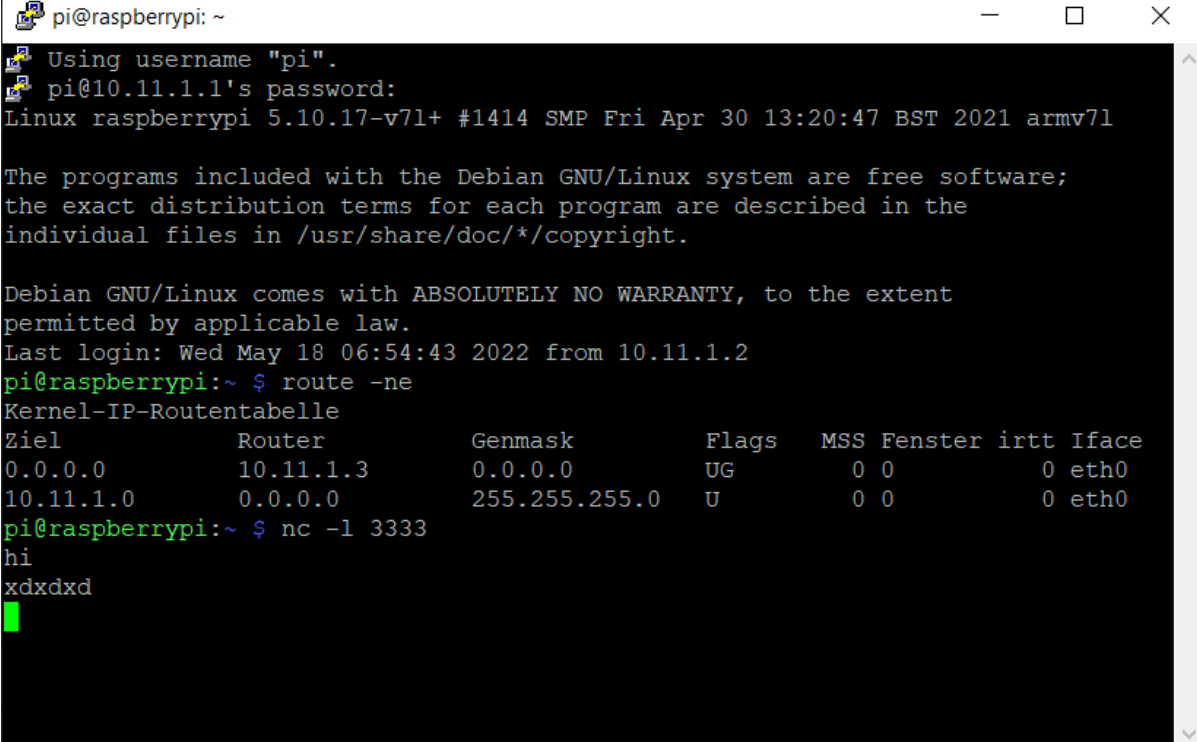
On our end device, we connect to the Raspberry using `nc 10.11.1.1 3333`, in other words, the Raspberry's IP address and listening port. We write some lines on both devices, without any issue.

3109	2437.8089618...	10.11.5.2	10.11.1.1	ICMP	98 Echo (ping) request id=0x004b, seq=13/3328, ttl=64 (reply in
3110	2437.9960951...	10.11.1.1	10.11.5.2	TCP	78 [TCP Previous segment not captured] 22 → 35832 [ACK] Seq=6805
3111	2438.0980363...	10.11.1.1	10.11.5.2	TCP	66 22 → 35832 [ACK] Seq=6805 Ack=3730 Win=64128 Len=0 TSval=7603
3112	2438.1060235...	10.11.1.1	10.11.5.2	SSHv2	110 Server: Encrypted packet (len=44)
3113	2438.1060398...	10.11.5.2	10.11.1.1	TCP	78 [TCP Dup ACK 3104#1] 35832 → 22 [ACK] Seq=3730 Ack=6753 Win=6
3114	2438.4403633...	10.11.1.1	10.11.5.2	ICMP	98 Echo (ping) reply id=0x004b, seq=13/3328, ttl=62 (request
3115	2438.4521000...	10.11.5.2	10.11.1.1	SSHv2	122 Client: Encrypted packet (len=44)
3116	2438.6438342...	10.11.1.1	10.11.5.2	TCP	110 [TCP Retransmission] 22 → 35832 [PSH, ACK] Seq=6805 Ack=3730
3117	2438.6438743...	10.11.5.2	10.11.1.1	TCP	86 [TCP Dup ACK 3104#2] 35832 → 22 [ACK] Seq=3774 Ack=6753 Win=6
3118	2438.7766532...	10.11.1.1	10.11.5.2	TCP	118 [TCP Retransmission] 22 → 35832 [PSH, ACK] Seq=6753 Ack=3730
3119	2438.7767144...	10.11.5.2	10.11.1.1	TCP	66 35832 → 22 [ACK] Seq=3774 Ack=6849 Win=64128 Len=0 TSval=3014
3120	2438.8089265...	10.11.5.2	10.11.1.1	ICMP	98 Echo (ping) request id=0x004b, seq=14/3584, ttl=64 (reply in
3121	2439.0304590...	10.11.1.1	10.11.5.2	TCP	66 22 → 35832 [ACK] Seq=6849 Ack=3774 Win=64128 Len=0 TSval=7603
3122	2439.0972300...	10.11.1.1	10.11.5.2	SSHv2	126 Server: Encrypted packet (len=60)
3123	2439.0972472...	10.11.5.2	10.11.1.1	TCP	66 35832 → 22 [ACK] Seq=3774 Ack=6909 Win=64128 Len=0 TSval=3014
3124	2439.3384846...	10.11.1.1	10.11.5.2	ICMP	98 Echo (ping) reply id=0x004b, seq=14/3584, ttl=62 (request
3125	2439.7647002...	10.11.5.2	10.11.1.1	SSHv2	102 Client: Encrypted packet (len=36)
3126	2439.8113425...	10.11.5.2	10.11.1.1	ICMP	98 Echo (ping) request id=0x004b, seq=15/3840, ttl=64 (reply in
3127	2440.2445921...	10.11.1.1	10.11.5.2	TCP	66 22 → 35832 [ACK] Seq=6909 Ack=3810 Win=64128 Len=0 TSval=7603
3128	2440.2548264...	10.11.1.1	10.11.5.2	SSHv2	230 Server: [TCP Previous segment not captured], Encrypted packe
3129	2440.2548414...	10.11.5.2	10.11.1.1	TCP	78 [TCP Dup ACK 3123#1] 35832 → 22 [ACK] Seq=3810 Ack=6909 Win=6
3130	2440.2634117...	10.11.1.1	10.11.5.2	TCP	102 [TCP Retransmission] 22 → 35832 [PSH, ACK] Seq=6909 Ack=3810
3131	2440.2634394...	10.11.5.2	10.11.1.1	TCP	66 35832 → 22 [ACK] Seq=3810 Ack=7109 Win=64128 Len=0 TSval=3014
3132	2440.2667054...	10.11.1.1	10.11.5.2	SSHv2	158 Server: [TCP Previous segment not captured], Encrypted packe
3133	2440.2667209...	10.11.5.2	10.11.1.1	TCP	78 [TCP Dup ACK 3131#1] 35832 → 22 [ACK] Seq=3810 Ack=7109 Win=6
3134	2440.2767517...	10.11.1.1	10.11.5.2	TCP	214 [TCP Retransmission] 22 → 35832 [PSH, ACK] Seq=7217 Ack=3810
3135	2440.2767666...	10.11.5.2	10.11.1.1	TCP	78 [TCP Dup ACK 3131#2] 35832 → 22 [ACK] Seq=3810 Ack=7109 Win=6
3136	2440.2835729...	10.11.1.1	10.11.5.2	SSHv2	174 Server: [TCP Fast Retransmission], Encrypted packet (len=108
3137	2440.2835032...	10.11.5.2	10.11.1.1	TCP	66 35832 → 22 [ACK] Seq=3810 Ack=7457 Win=64128 Len=0 TSval=3014
3138	2440.3445036...	10.11.1.1	10.11.5.2	ICMP	98 Echo (ping) reply id=0x004b, seq=15/3840, ttl=62 (request

Figure 7.3. Packet loss on SSH with NetEm connected

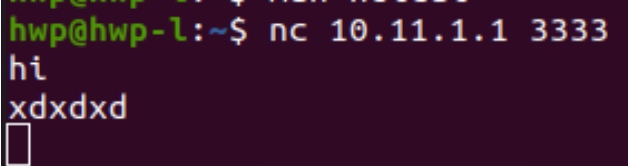
We also try the same thing using the `-u` flag to do it via UDP. Again, no issues on either end, and no variation detected. On Wireshark, we see that

TCP used three kinds of packet in order to keep the synchronization, the first one being SYN, sent by the end device to the Raspberry, the second one the Raspberry's response, SYN ACK to do the handshake. Then we have PSH and ACK packets to acknowledge the reception of packets in order and request more. None of those packets are seen on the UDP connection.



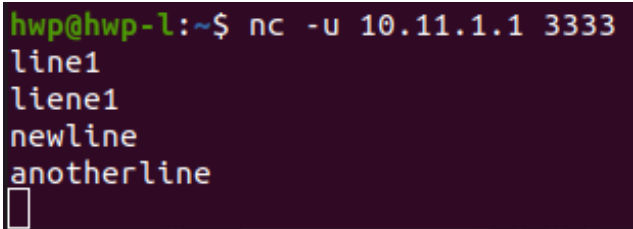
```
pi@raspberrypi: ~  
Using username "pi".  
pi@10.11.1.1's password:  
Linux raspberrypi 5.10.17-v7l+ #1414 SMP Fri Apr 30 13:20:47 BST 2021 armv7l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Wed May 18 06:54:43 2022 from 10.11.1.2  
pi@raspberrypi:~$ route -ne  
Kernel-IP-Routentabelle  
Ziel          Router        Genmask       Flags  MSS  Fenster  irtt  Iface  
0.0.0.0       10.11.1.3     0.0.0.0       UG     0 0      0     eth0  
10.11.1.0     0.0.0.0       255.255.255.0 U      0 0      0     eth0  
pi@raspberrypi:~$ nc -l 3333  
hi  
xdxdxd  
█
```

Figure 7.4. Raspberry receiving and sending messages



```
hwp@hwp-l:~$ nc 10.11.1.1 3333  
hi  
xdxdxd  
█
```

Figure 7.5. End device connection with the Raspberry



```
hwp@hwp-l:~$ nc -u 10.11.1.1 3333  
line1  
liene1  
newline  
anotherline  
█
```

Figure 7.6. UDP communication

Next up, we try forwarding the USB microphone recording from the Raspberry to the end device using the network. We forward it using Netcat, and using the arecord/aplay interface.

On the sending device, which is the Raspberry, we need the arecord -f S16_LE -r 44100 -t raw command piped along the nc 10.11.5.2 3333 command. This sends a raw audio signal on a 16 bit little endian with a 44100 Hz sampling rate, to the server located on the IP address and port designated (our end device).

Meanwhile, on our end device, we open our 3333 port first using Netcat, with nc -l 3333 and we pipe that with the aplay -f S16_LE -r 44100 command, exactly what we had on the sending device in order to process the signal correctly.

We could establish a real-time audio transmission, both using TCP and UDP. There was no noticeable difference in the quality and delay, this last one being about half a second.

```
pi@raspberrypi:~ $ arecord -f S16_LE -r 44100 -t raw | nc -u 10.11.5.2 3333
Aufnahme: Rohdaten 'stdin' : Signed 16 bit Little Endian, Rate: 44100 Hz, mono
^CAbbruch durch Signal Unterbrechung ...
pi@raspberrypi:~ $ arecord -f S16_LE -r 44100 -t raw | nc 10.11.5.2 3333
Aufnahme: Rohdaten 'stdin' : Signed 16 bit Little Endian, Rate: 44100 Hz, mono
^CAbbruch durch Signal Unterbrechung ...
pi@raspberrypi:~ $
```

Figure 7.7. Sending audio via UDP and TCP

71334	5121.8321770...	10.11.5.2	10.11.1.1	TCP	66	3333 → 44692	[ACK]	Seq=1 Ack=2160129 Win=64128 Len=0 TSval=36
71335	5121.8460309...	10.11.1.1	10.11.5.2	TCP	322	44692 → 3333	[PSH, ACK]	Seq=2160129 Ack=1 Win=64256 Len=256 T
71336	5121.8460533...	10.11.5.2	10.11.1.1	TCP	66	3333 → 44692	[ACK]	Seq=1 Ack=2160385 Win=64128 Len=0 TSval=36
71337	5121.8466856...	10.11.1.1	10.11.5.2	TCP	834	44692 → 3333	[PSH, ACK]	Seq=2160385 Ack=1 Win=64256 Len=768 T
71338	5121.8467064...	10.11.5.2	10.11.1.1	TCP	66	3333 → 44692	[ACK]	Seq=1 Ack=2161153 Win=64128 Len=0 TSval=36
71339	5121.8471855...	10.11.1.1	10.11.5.2	TCP	322	44692 → 3333	[PSH, ACK]	Seq=2161153 Ack=1 Win=64256 Len=256 T
71340	5121.8472026...	10.11.5.2	10.11.1.1	TCP	66	3333 → 44692	[ACK]	Seq=1 Ack=2161409 Win=64128 Len=0 TSval=36
71341	5121.8609976...	10.11.5.2	10.11.1.1	TCP	66	3333 → 44692	[FIN, ACK]	Seq=1 Ack=2161409 Win=64128 Len=0 TSv
71342	5121.8610378...	10.11.1.1	10.11.5.2	TCP	322	44692 → 3333	[PSH, ACK]	Seq=2161409 Ack=1 Win=64256 Len=256 T
71343	5121.8610537...	10.11.5.2	10.11.1.1	TCP	54	3333 → 44692	[RST]	Seq=1 Win=0 Len=0

Figure 7.8. Audio through TCP

5473	3861.4866949...	10.11.1.1	10.11.5.2	UDP	706	41748 → 3333	Len=11024	
5474	3861.6210438...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=0, ID=7c17)	[Reasse
5475	3861.6210440...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=1480, ID=7c17)	[Rea
5476	3861.6212883...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=2960, ID=7c17)	[Rea
5477	3861.6212885...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=4440, ID=7c17)	[Rea
5478	3861.6215360...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=5920, ID=7c17)	[Rea
5479	3861.6215362...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=7400, ID=7c17)	[Rea
5480	3861.6217855...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=8880, ID=7c17)	[Rea
5481	3861.6217857...	10.11.1.1	10.11.5.2	UDP	706	41748 → 3333	Len=11024	
5482	3861.7559980...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=0, ID=7c24)	[Reasse
5483	3861.7559984...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=1480, ID=7c24)	[Rea
5484	3861.7562190...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=2960, ID=7c24)	[Rea
5485	3861.7562194...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=4440, ID=7c24)	[Rea
5486	3861.7564887...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=5920, ID=7c24)	[Rea
5487	3861.7564891...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=7400, ID=7c24)	[Rea
5488	3861.7567473...	10.11.1.1	10.11.5.2	IPv4	1514	Fragmented IP protocol	(proto=UDP 17, off=8880, ID=7c24)	[Rea

Figure 7.9. Audio through UDP

7.3.1. Latency minimisation and buffers

The default buffer time arecord and aplay use is 500ms, which was around the time we experienced when listening. In order to tweak this time, the buffer-time (-B) flag can be used. It measures in ms.

We tried lowering it from 500ms, however, no difference was noticed, it did not lower the reception time, using both TCP and UDP. We only detected anomalies using 0 as a buffer, where we could hear cuts in the audio.

7.3.2. Packet loss and latency

Now, we connected the NetEm to the network and ran the same commands. At first, we tried TCP, and the difference was highly noticeable. We heard the complete signal, but with a lot of stops and cuts, which was not pleasant. When trying later with UDP, the result was similar in the amount of cuts, however, there was not as much delay. Instead, the audio was either lost or scrambled. However, with the default buffer, it was mostly unintelligible.

73869	5297.9027640...	10.11.5.2	10.11.1.1	TCP	94 [TCP Dup ACK 73855#6] 3333 → 44696 [ACK] Seq=1 Ack=609377 Win=64256 Len=1448 TSval=1514 TSecr=1514
73870	5297.9052533...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [ACK] Seq=625305 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73871	5297.9064790...	10.11.5.2	10.11.1.1	TCP	94 [TCP Dup ACK 73855#7] 3333 → 44696 [ACK] Seq=1 Ack=609377 Win=64256 Len=1448 TSval=1514 TSecr=1514
73872	5297.9431952...	10.11.1.1	10.11.5.2	TCP	1514 [TCP Out-Of-Order] 44696 → 3333 [ACK] Seq=622409 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73873	5297.9444206...	10.11.5.2	10.11.1.1	TCP	86 [TCP Dup ACK 73855#8] 3333 → 44696 [ACK] Seq=1 Ack=609377 Win=64256 Len=1448 TSval=1514 TSecr=1514
73874	5297.9929622...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [PSH, ACK] Seq=626753 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73875	5297.9941869...	10.11.5.2	10.11.1.1	TCP	86 [TCP Dup ACK 73855#9] 3333 → 44696 [ACK] Seq=1 Ack=609377 Win=64256 Len=1448 TSval=1514 TSecr=1514
73876	5298.2284628...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [ACK] Seq=628201 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73877	5298.2296868...	10.11.5.2	10.11.1.1	TCP	86 [TCP Dup ACK 73855#10] 3333 → 44696 [ACK] Seq=1 Ack=609377 Win=64256 Len=1448 TSval=1514 TSecr=1514
73878	5298.3025968...	10.11.1.1	10.11.5.2	TCP	1514 [TCP Out-Of-Order] 44696 → 3333 [ACK] Seq=609377 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73879	5298.3026564...	10.11.5.2	10.11.1.1	TCP	78 3333 → 44696 [ACK] Seq=1 Ack=619513 Win=331776 Len=0 TSval=36 TSecr=1514
73880	5298.3104656...	10.11.1.1	10.11.5.2	TCP	1514 [TCP Previous segment not captured] 44696 → 3333 [PSH, ACK] Seq=629649 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73881	5298.3105024...	10.11.5.2	10.11.1.1	TCP	78 [TCP Window Update] 3333 → 44696 [ACK] Seq=1 Ack=619513 Win=331776 Len=0 TSval=1514 TSecr=1514
73882	5298.4276344...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [ACK] Seq=632545 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73883	5298.4276707...	10.11.5.2	10.11.1.1	TCP	86 [TCP Dup ACK 73879#1] 3333 → 44696 [ACK] Seq=1 Ack=619513 Win=64256 Len=1448 TSval=1514 TSecr=1514
73884	5298.5275587...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [ACK] Seq=633993 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73885	5298.5275743...	10.11.5.2	10.11.1.1	TCP	86 [TCP Dup ACK 73879#2] 3333 → 44696 [ACK] Seq=1 Ack=619513 Win=64256 Len=1448 TSval=1514 TSecr=1514
73886	5298.6360243...	10.11.1.1	10.11.5.2	TCP	1514 [TCP Out-Of-Order] 44696 → 3333 [ACK] Seq=619513 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73887	5298.6360521...	10.11.5.2	10.11.1.1	TCP	78 3333 → 44696 [ACK] Seq=1 Ack=629649 Win=331648 Len=0 TSval=36 TSecr=1514
73888	5298.7018124...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [ACK] Seq=635441 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73889	5298.7018449...	10.11.5.2	10.11.1.1	TCP	78 [TCP Window Update] 3333 → 44696 [ACK] Seq=1 Ack=629649 Win=331648 Len=0 TSval=1514 TSecr=1514
73890	5298.7371944...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [ACK] Seq=636889 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73891	5298.7372275...	10.11.5.2	10.11.1.1	TCP	78 [TCP Dup ACK 73887#1] 3333 → 44696 [ACK] Seq=1 Ack=629649 Win=331648 Len=0 TSval=1514 TSecr=1514
73892	5298.8143359...	10.11.1.1	10.11.5.2	TCP	78 3333 → 44696 [FIN, ACK] Seq=1 Ack=629649 Win=339328 Len=0 TSval=1514 TSecr=1514
73893	5298.8905910...	10.11.1.1	10.11.5.2	TCP	1514 [TCP Out-Of-Order] 44696 → 3333 [PSH, ACK] Seq=631097 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73894	5298.8906443...	10.11.5.2	10.11.1.1	TCP	54 3333 → 44696 [RST] Seq=1 Win=0 Len=0
73895	5298.9165525...	10.11.1.1	10.11.5.2	TCP	1514 44696 → 3333 [ACK] Seq=638337 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73896	5298.9165916...	10.11.5.2	10.11.1.1	TCP	54 3333 → 44696 [RST] Seq=1 Win=0 Len=0
73897	5298.9911314...	10.11.1.1	10.11.5.2	TCP	1514 [TCP Out-Of-Order] 44696 → 3333 [ACK] Seq=629649 Ack=1 Win=64256 Len=1448 TSval=1514 TSecr=1514
73898	5298.9911699...	10.11.5.2	10.11.1.1	TCP	54 3333 → 44696 [RST] Seq=1 Win=0 Len=0

Figure 7.10. Audio through TCP using NetEm

```

hwp@hwp-l:~$ nc -u -l 3333 | aplay -f S16_LE -r 44100 -B 45000
Wiedergabe: Rohdaten 'stdin' : Signed 16 bit Little Endian, Rate: 44100 Hz, mono
Unterlauf!!! (mindestens 53,394 ms)
Unterlauf!!! (mindestens 214,517 ms)
Unterlauf!!! (mindestens 914,209 ms)
Unterlauf!!! (mindestens 378,894 ms)
Unterlauf!!! (mindestens 142,103 ms)
Unterlauf!!! (mindestens 118,247 ms)
Unterlauf!!! (mindestens 392,838 ms)
Unterlauf!!! (mindestens 883,585 ms)
Unterlauf!!! (mindestens 166,746 ms)
Unterlauf!!! (mindestens 762,234 ms)
Unterlauf!!! (mindestens 242,852 ms)
Unterlauf!!! (mindestens 873,072 ms)
Unterlauf!!! (mindestens 480,980 ms)
Unterlauf!!! (mindestens 315,377 ms)
^CAbbruch durch Signal Unterbrechung ...
aplay: pcm_write:2061: Schreibfehler: Unterbrechung während des Betriebssystemsaufrufs

```

Figure 7.11. Warnings received on audio through UDP using NetEm

We experimented with various buffer sizes, obtaining different results for TCP and UDP.

TCP:

Buffer Size	Experience
0	Same as using 500
1	Lots of cuts, nothing can be heard
500	Bits are able to understand, but really delayed
3000	Slight improvement, but still really delayed
20000	Perfect point, audio is a bit delayed but is more constant and intelligible
100000	Cuts start to be experienced again

UDP:

Buffer Size	Experience
0	Same as using 500
1	Nothing intelligible
500	Barely anything intelligible
3000	Slight improvement, but still scrambled
20000	Biggest improvement from previous sizes, but still unpleasant
40000	Plateau point, from this point onwards, there is no improvement

We need to highlight that while with TCP we could reach a buffer time in which the packet loss and latency did not matter as much, as the audio was perfectly intelligible, with UDP that could not be reached. We avoided the audio scrambling with high buffer sizes. However, there were still some lost packets that made the experience worse.