Daniel hor dhor051 516000089
SE364 a01

SOFTENG 364 Assignment 1

Due 5 pm May 5th 2020

Instructions for Assignment:

- The assignment contains 1 Socket programming exercise, 2 Wireshark exercises, and 1 traceroute exercise
- For the Socket programming, Python 3 is the coding platform. For full credit, please ensure that your code adheres to the usual guidelines of consistency and readability, and use pycodestyle to check conformance with PEP 8 (https://pypi.org/project/pycodestyle/).
- For the Wireshark question, you need to include the screenshots of Wireshark to support your answer. Submit your write up of this assignment as a pdf file. Your actually submission will be a zip file of your write up, plus the relevant files you have generated.

Exercise 1

I. File transfer using TCP.

In this exercise you will set up a server and client and transfer an audio file, using python 3.

You have been provided an audio file, audioTRM.wav. Set up a local server that hosts this file. Please note that the aim is to do a basic file transfer (wav file) from the server to the client using socket.recv( ). The details of the server are:

Item	**Requirement**
Server name	localhost
Server port	12000
File transfer protocol	TCP
Algorithm requirement	Use *socket.recv(*bufsize*[,* flags*]), bufsize=1024*
Message to be displayed when server is running	"Server started listening localhost: 12000"
Filename	Implement the above using python and save the code as TCPserver_Exercise1.py

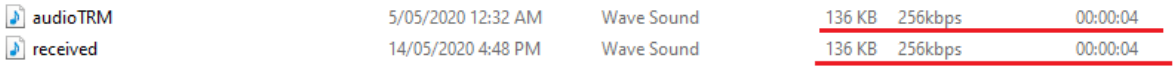
Now we need to setup a client that connects to this server and reads the audio file audioTRM.wav[1]. Set up a local client that connects to the local server you created. The client

[1] This audio file has been produced by a synthetic speech engine we have been working on for te reo Māori, it is saying the phrase *"Nau mai, ki te ao, o te hanga oro kōrero."*

should then read the audio file hosted by the server and save it to a file called received.wav. The details of the client are:

Item	**Requirement**
File transfer protocol	TCP
Algorithm requirement	Use *socket.recv(*bufsize*[,*flags*]), bufsize=1024*
Message to be displayed when connection is established to server	"Connection established to server localhost: 12000"
Message to be displayed when file transfer is complete	"File transfer complete – file name: audio.wav, saved as received.wav"
Filename	Implement the above using python and save the code as TCPclient_Exercise1.py

In addition to producing the above python code, comment on the original audio.wav file, and the received.wav file. Are they the same? How do you know?



Yes, the contents of the original audio.wav file and the received.wav file are the same. This is known because both files have the same size(136KB), Bit rate(256kbps) and Length(4 seconds).

Furthermore, by listening to both audio files, it is confirmed that the two audio files have the same audio output.

2. Timing of the file transfer

Calculate the time taken for the file transfer, using the time package in Python.

(i) Include the header: *import time*

(ii) Add *start_time = time.time( )* at the start of the file transfer (if you have any type of looping mechanism, as this step before the loop starts)

(iii) Add *end_time = time.time( )* at the end of the file transfer (Again put this instruction at the end of the loop – after the entire file transfer)

What is the time taken for the file transfer?

Daniel hor dhor051 516000089
SE364 a01

```
C:\Users\dh\SE364a01>python TCPserver_Exercise1.py
Server started listening localhost: 12000
Sending complete
Time taken: 0.003991842269897461 seconds

C:\Users\dh\SE364a01>
```

```
C:\Users\dh\SE364a01>python TCPclient_Exercise1.py
Connection establish to server localhost: 12000
File transfer complete - file name:  audio.wav, saved as received.wav
Time taken: 0.003999948501586914 seconds

C:\Users\dh\SE364a01>
```

The server took 0.003991842269897461 seconds to send the data.
The client took 0.003999948501586914 seconds to receive and save the data to a new file.

3. Wireshark analysis of your File transfer

Use wireshark to investigate the file transfer in your client server set up.

1. Start the server
2. Start Wireshark capture, using the loopback traffic mode.
3. Start the client
4. Stop the capture once the file transfer is complete, and save the results in a file called Exercise1.3.pcapng.

Identify the following, use appropriate wireshark screen shots to support your answers.:

5. which packets were the three-way handshake?

7	2.630015	127.0.0.1	127.0.0.1	TCP	56	61146 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
8	2.630061	127.0.0.1	127.0.0.1	TCP	56	12000 → 61146 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
9	2.630105	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0

The three-way handshake packet numbers = 7, 8, 9

6. which packets were the shut-down?

No.	Time	Source	Destination	Protocol	Length	Info
282	2.635298	127.0.0.1	127.0.0.1	TCP	44	12000 → 61146 [FIN, ACK] Seq=139245 Ack=1 Win=2619648 Len=0
283	2.635316	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=139246 Win=2611456 Len=0
284	2.635595	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [FIN, ACK] Seq=1 Ack=139246 Win=2611456 Len=0
285	2.635630	127.0.0.1	127.0.0.1	TCP	44	12000 → 61146 [ACK] Seq=139246 Ack=2 Win=2619648 Len=0

The shut-down packet numbers = 282, 283, 284, 285

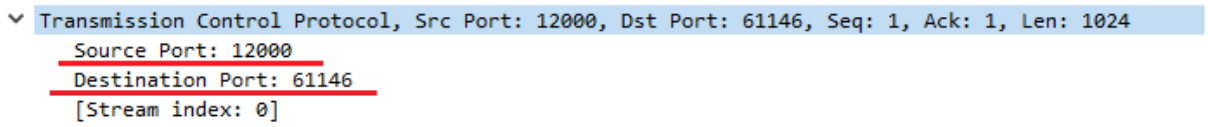
7. what the source and destination IP address in the file transfer,

No.	Time	Source	Destination	Protocol	Length	Info
10	2.630466	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=1024
11	2.630487	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=1025 Win=2618624 Len=0
12	2.630507	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=1025 Ack=1 Win=2619648 Len=1024
13	2.630522	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=2049 Win=2617600 Len=0
14	2.630540	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=2049 Ack=1 Win=2619648 Len=1024
15	2.630552	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=3073 Win=2616576 Len=0

Source IP address = 127.0.0.1

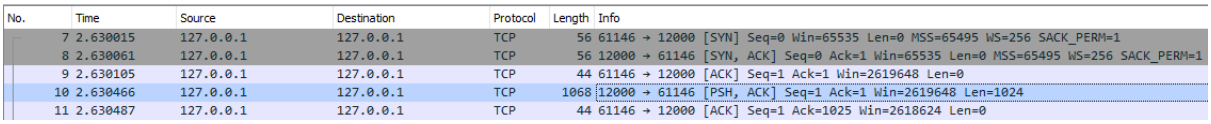
Destination IP address = 127.0.0.1

8. what port numbers were involved in the file transfer.



Server (Source) port number = 12000
Client (Destination) port number = 61146

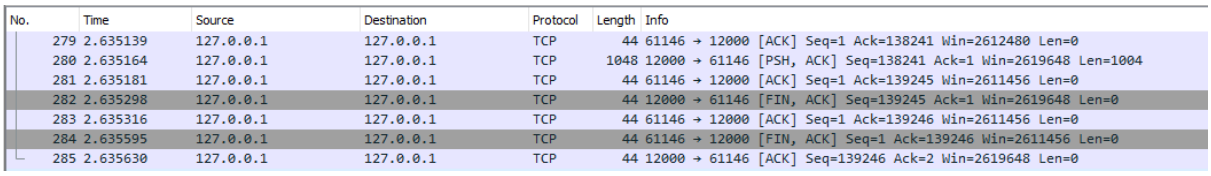
9. What packet did the file transfer start?



No.	Time	Source	Destination	Protocol	Length	Info
7	2.630015	127.0.0.1	127.0.0.1	TCP	56	61146 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
8	2.630061	127.0.0.1	127.0.0.1	TCP	56	12000 → 61146 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
9	2.630105	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
10	2.630466	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=1024
11	2.630487	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=1025 Win=2618624 Len=0

File transfer started on packet = 10
This is the first packet with the PSH flag.

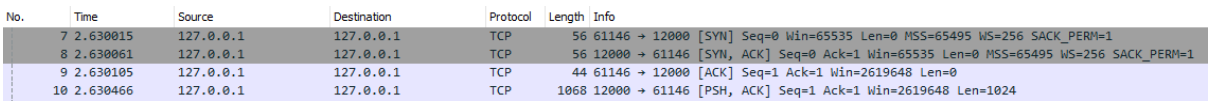
10. What packet did the file transfer end?



No.	Time	Source	Destination	Protocol	Length	Info
279	2.635139	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=138241 Win=2612480 Len=0
280	2.635164	127.0.0.1	127.0.0.1	TCP	1048	12000 → 61146 [PSH, ACK] Seq=138241 Ack=1 Win=2619648 Len=1004
281	2.635181	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=139245 Win=2611456 Len=0
282	2.635298	127.0.0.1	127.0.0.1	TCP	44	12000 → 61146 [FIN, ACK] Seq=139245 Ack=1 Win=2619648 Len=0
283	2.635316	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=139246 Win=2611456 Len=0
284	2.635595	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [FIN, ACK] Seq=1 Ack=139246 Win=2611456 Len=0
285	2.635630	127.0.0.1	127.0.0.1	TCP	44	12000 → 61146 [ACK] Seq=139246 Ack=2 Win=2619648 Len=0

File transfer ended on packet = 281
This is the packet of the acknowledgement sent by the client (and received by the server) for the final packet sent by the server.

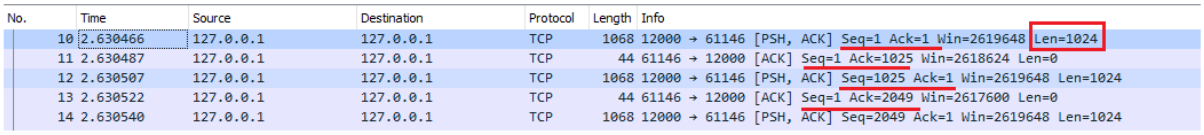
11. Discuss the file transfer process that you observed in wireshark, and how the client and server used sequence numbers and acknowledgements numbers to inform the other process of the file transfer process? Did any of the packets go missing?



No.	Time	Source	Destination	Protocol	Length	Info
7	2.630015	127.0.0.1	127.0.0.1	TCP	56	61146 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
8	2.630061	127.0.0.1	127.0.0.1	TCP	56	12000 → 61146 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
9	2.630105	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
10	2.630466	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=1024

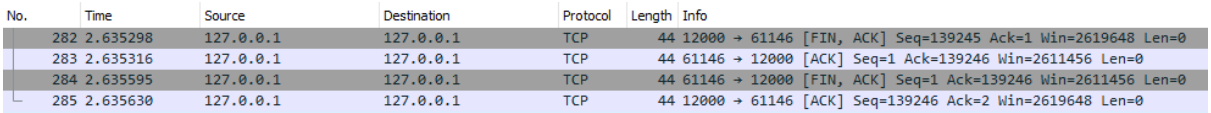
To begin the file transfer process, the connection is set up through the TCP 3-way handshake.
Packet 7 shows the client initiating the connection to the server by sending a TCP SYN message with a sequence number 0 indicating that it is the first segment of data of sent by the client.
Packet 8 shows the server sending a TCP SYNACK message that is acknowledging the SYN message of the client initiating the connection. The SYNACK message has

a sequence number 0 indicating that it is the first segment of data of sent by the server. The SYNACK message has acknowledgement number 1 indicating that the server is expecting that the sequence number of the next byte sent from the client is byte number 1.
Packet 9 shows the client sending an acknowledgement (ACK) message acknowledging the TCP SYNACK message(packet 8), with sequence number 1 showing the byte that the server is expecting (packet 8 Acknowledgment number 1) is sent. Packet 9 has Acknowledgment number 1 indicating that the client is expecting that the sequence number of the next byte sent from the server is byte number 1.
The TCP handshake is complete, and the connection is formed.



No.	Time	Source	Destination	Protocol	Length	Info
10	2.630466	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=1024
11	2.630487	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=1025 Win=2618624 Len=0
12	2.630507	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=1025 Ack=1 Win=2619648 Len=1024
13	2.630522	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=2049 Win=2617600 Len=0
14	2.630540	127.0.0.1	127.0.0.1	TCP	1068	12000 → 61146 [PSH, ACK] Seq=2049 Ack=1 Win=2619648 Len=1024

From packet 4 onwards, the server begins sending 1024 byte packets of data to the client.
When the server receives the ACK message that the handshake is complete (Packet 9), the server then sends Packet 10 with sequence number 1 and length 1024 bytes indicating that the first byte of the packet is byte stream number 1 and that packet 10 contains the 1st to 1024th byte.
When packet 10 is received by the client, the client sends packet 11 that is an ACK message with acknowledgement number 1025 (=1 + 1024). 1025 is the sequence number of the next byte that the client is expecting from the server.
When packet 11 is received by the server, the server sends Packet 12 with sequence number 1025 and length 1024 indicating that the first byte of the packet is byte stream number 1025 and that packet 12 contains the 1025th to 2049th byte. This sequence number 1025 of packet 12 correlates to the acknowledgement number 1025 of Packet 11.
The client uses acknowledgement numbers to indicate what bytes need to be sent and the server uses sequence numbers to indicate what bytes have been sent during the file transfer process. This process of the client sending acknowledgments with acknowledgement numbers and the server sending packets of data with sequence numbers continues until all the data is sent.



No.	Time	Source	Destination	Protocol	Length	Info
282	2.635298	127.0.0.1	127.0.0.1	TCP	44	12000 → 61146 [FIN, ACK] Seq=139245 Ack=1 Win=2619648 Len=0
283	2.635316	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [ACK] Seq=1 Ack=139246 Win=2611456 Len=0
284	2.635595	127.0.0.1	127.0.0.1	TCP	44	61146 → 12000 [FIN, ACK] Seq=1 Ack=139246 Win=2611456 Len=0
285	2.635630	127.0.0.1	127.0.0.1	TCP	44	12000 → 61146 [ACK] Seq=139246 Ack=2 Win=2619648 Len=0

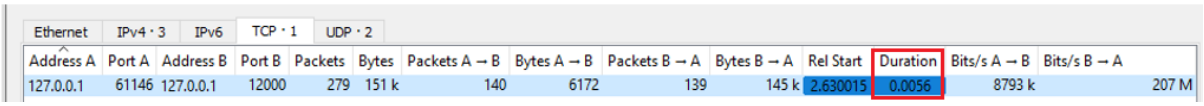
After all the data is sent the connection is closed. To close the connection, the server sends Packet 282 with the FINACK flag. The client then sends Packet 283 that is an acknowledgment message acknowledging the FINACK message and proceeds to send Packet 284 with the FINACK flag. Once the client's FINACK message is acknowledged by the server the connection is closed.

The sequence numbers and the acknowledgement numbers of these packets still indicate the byte stream number of the first byte in the packet and the sequence

number of the next byte expected. However, the numbers are not changing as the length of the packets sent is 0 bytes.

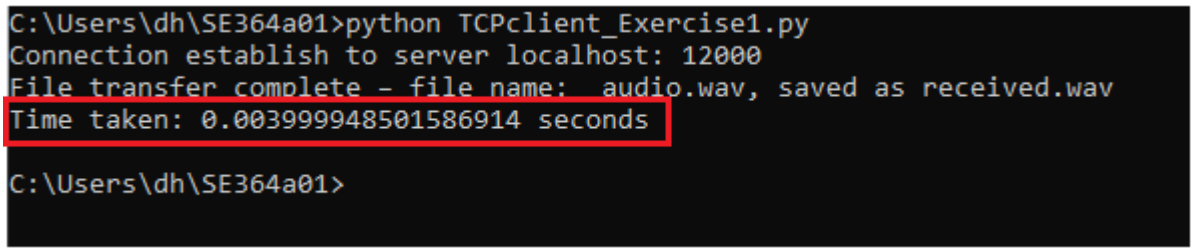
No packets went missing during the file transfer. This is observed as there are no duplicate packets with the same sequence number.

12. According to Wireshark how long did the file transfer take (including connection setup and connection tear down time) How did this compare to the time calculated in the client process (calculated in part 2).



Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
127.0.0.1	61146	127.0.0.1	12000	279	151 k	140	6172	139	145 k	2.630015	0.0056	8793 k	207 M

According to Wireshark the file transfer took = 0.0056 seconds



The file transfer time calculated in the client process in part 2 = 0.003999948501586914 seconds

The time indicated by Wireshark was longer than the time calculated in the client process by around 0.0016 seconds.

(Hint: you might find it useful going to conversations wireshark (Statistics->Conversations), selecting the TCP conversations, finding the stream which communicates to port 12000, clicking on this, and then click on follow on this stream).

Daniel hor dhor051 516000089
SE364 a01

Exercise 2

In this exercise you are going to compare the performance to two on-line speech synthesis demos, these can be found at http://mary.dfki.de:59125/ and http://engresdev09.its.auckland.ac.nz:59125/ Both on-line demos use the open source Mary TTS software, the http://mary.dfki.de:59125/ site is the site of the developers of MaryTTS, the http://engresdev09.its.auckland.ac.nz:59125/ is an installation of Mary TTS which runs on the University of Auckland network and has voices specially created for the New Zealand context. MaryTTS is a client/server system. On the client browser, the user can select the voice they wish, to use in the synthesis, input the phrase they wish to synthesis, and hear the results. The actual TTS engine is on a server, for the http://mary.dfki.de:59125/ site the server is in Germany, for the http://engresdev09.its.auckland.ac.nz:59125/ [2] site the server is based at the University of Auckland.

Figure 1: A screen shot of the MaryTTS demo page, see text for descriptions of annotations

Figure 1 is a screen shot of the on-line demo webpage, i.e, what is seen on the client's browser when MaryTTS is called. It can be seen 6 features have been highlighted in maroon. The first two features identify the input and output type, the default is text in, and audio out (which is usually what you want in a text to speech system). The third feature is the input text box. Some text, a place holder, is already placed in this box 'Welcome to the world of speech synthesis'. Feature 4 is a drop-down menu from where the voice can be selected. Often there are several voices that can be selected. Feature 5 is a drop-down menu from where the file type of the audio file, which will be produced when the SPEAK button (feature 6) is pressed. The audio out is set to be a wav format, but two other formats could be selected. Before starting this exercise, I suggest you visit http://mary.dfki.de:59125/, and have an investigation of how MaryTTS works. Place text in the input text box and listen to the results. Note this is a multi-lingual speech synthesis

[2] The MaryTTS based at the UoA can only be accessed on the UoA network, when off campus a VPN is needed. If you can do get on the UoA VPN please feel welcome to try out our system.

system, there are english voices to select from (look for GB and US), but try some of the other languages too.
This exercise is in three parts. In the first part you will be performing an analysis on a pcapng file, which was captured during an investigation of http://engresdev09.its.auckland.ac.nz:59125/, in the second part of the exercise you will be doing a wireshark capture of a visit to http://mary.dfki.de:59125/, where you will be doing some speech synthesis, and in the third part you will be comparing the two different MaryTTS systems, from a computer networking perspective.

1 Analysis of Interaction with the MaryTTS server in Auckland.

The MaryUoA_A1.pcapng file captured the follow actions with MaryTTS

1. Starting up MaryTTS on a client browser (IP 10.110.55.236) which is connecting to the MaryTTS engine based on the server at http://engresdev09.its.auckland.ac.nz:59125/ (IP 130.216.236.122).
2. Selecting the cmu-slt-hsmm en_US female hmm voice
3. Synthesizing the text "Welcome to the world of Speech Synthesis", which includes streaming an audio file
4. Selecting the sh_eu-hsmm en_NZ male Hmm voice
5. Synthesizing the text "Welcome to the world of Speech Synthesis" ", which includes streaming an audio file
6. Selecting the akl_mi_pk_voice1-hsmm mi male hmm voice
7. Synthesising the text "Haere Mai" ", which includes streaming an audio file

Open the pcapng file in Wireshark and create a table with every client HTTP GET to the MaryTTS server; for each GET include

i. the starting time and packet number of the packet with the GET,
ii. the time and packet number of the last packet of the server's response to the GET,
iii. the source and destination port numbers, and
iv. a brief description on what the GET command doing (note some GETs are really posts).

Make sure the table entries are in chronological order. This table will have around 18 entries.
To create the table you may find the following functions in Wireshark useful

- Use the filter ip.addr=130.216.236.122
- Statistics->Conversations, go to TCP tab
- Statistics-> flow graph, (with Limit to display filter checked)
- TCP streams

Get object at URI: http://engresdev09.its.auckland.ac.nz:59125

GET packet	**Last RESPONSE Packet to GET**	**Ports**	**Description**
Packet: 73 Time: 9.020712	Packet: 81 Time: 9.030923	Source: 58626 Destination: 59125	Download request for main page of MaryTTS at http://engresdev09.its.auckland.ac.nz:59125
Packet: 88 Time: 9.056283	Packet: 106 Time: 9.063918	Source: 58627 Destination: 59125	Request for the object: /mary.js
Packet: 108 Time: 9.064732	Packet: 131 Time: 9.073571	Source: 58628 Destination: 59125	Request for the object: /favicon.ico
Packet: 147 Time: 9.153521	Packet: 161 Time: 9.159747	Source: 58630 Destination: 59125	Request for the object: /voices
Packet: 154 Time: 9.155872	Packet: 167 Time: 9.161303	Source: 58631 Destination: 59125	Request for the object: /datatypes

Packet: 155 Time: 9.155895	Packet: 171 Time: 9.161765	Source: 58632 Destination: 59125	Request for the object: /audioformats
Packet: 156 Time: 9.156218	Packet: 174 Time: 9.163647	Source: 58633 Destination: 59125	Request for the object: /audioeffects
Packet: 163 Time: 9.163647	Packet: 184 Time: 9.169961	Source: 58634 Destination: 59125	Request for the object: /favicon.ico
Packet: 187 Time: 9.170574	Packet: 191 Time: 9.191741	Source: 58631 Destination: 59125	Request for the object: /exampletext?voice=sh_neu-hsmm
Packet: 190 Time: 9.171599	Packet:194 Time:9.191871	Source: 58633 Destination: 59125	Request for the object: /exampletext?datatype=TEXT&locale=en_NZ
Packet:237 Time:16.498194	Packet:241 Time:16.524402	Source: 58633 Destination: 59125	Request for the object: /exampletext?datatype=TEXT&locale=en_US
Packet:238 Time:16.499392	Packet:239 Time:16.524310	Source: 58631 Destination: 59125	Request for the object: /exampletxt?voice=cmu-slt-hsmm
Packet:254 Time:18.961086	Packet:606 Time:19.626276	Source: 58633 Destination: 59125	Requests audio speech synthesis using “cmu-slt-hsmm en_US female hmm” voice
Packet:653 Time:27.033301	Packet:659 Time:27.100822	Source: 58633 Destination: 59125	Request for the object /exampletext?voice=sh_neu-hsmm
Packet:654 Time:27.033314	Packet:657 Time:27.100773	Source: 58631 Destination: 59125	Request for the object /exampletext?datatype=TEXTlocale=en_NZ
Packet:666 Time:28.878285	Packet:858 Time:29.399980	Source: 58631 Destination: 59125	Requests audio speech synthesis using: “sh_eu-hsmm en_NZ male Hmm” voice
Packet:900 Time:36.386060	Packet:905 Time:36.393649	Source: 58631 Destination: 59125	Request for the object: /exampletext?datatype=TEXT&locale=mi
Packet:901 Time:36.386308	Packet:903 Time:36.393561	Source: 58633 Destination: 59125	Request for the object: /exampletext?voice=akl_mi_pk_voice1-hsmm
Packet:949 Time:40.185855	Packet:1023 Time:40.321462	Source: 58631 Destination: 59125	Requests audio speech synthesis using: “akl_mi_pk_voice1-hsmm mi male hmm” voice

2 Analysis of the MaryTTS server based in Germany

Use Wireshark to capture the following

1. Starting up MaryTTS on your browser by clicking on http://mary.dfki.de:59125/
2. Select the dfki-spike-hsmm en_GB male hmm
3. Synthesize the text “Welcome to the world of Speech Synthesis” (remember to press the SPEAK button, and you should hear the voice saying the phrase)
4. Select the cmu-slt-hsmm en_US female hmm voice
5. Synthesize the text “Welcome to the world of Speech Synthesis”
6. Select the bits4 de female unitselection general
7. Synthesis the text “Willkommen in der Welt der Sprachsynthese!”
8. Stop the capture, and save the file, called Exercise2.2.pcapng for later analysis.

Using the same columns as in Part 1 create a second table for every client’s HTTP GET to the MaryTTS server.

GET packet	**Last RESPONSE Packet to GET**	**Ports**	**Description**
Packet: 8 Time: 2.053254	Packet: 19 Time: 2.394124	Source: 52025 Destination: 59125	Download request for main page of MaryTTS at http://engresdev09.its.auckland.ac.nz:59125
Packet: 23 Time: 2.418824	Packet: 69 Time: 3.102318	Source: 52026 Destination: 59125	Request for the object: /mary.js
Packet:28 Time: 2.748556	Packet: 54 Time: 3.084746	Source: 52027 Destination: 59125	Request for the object: /favicon.ico
Packet: 82 Time: 3.447884	Packet: 98 Time: 3.777302	Source: 52030 Destination: 59125	Request for the object: /voices
Packet: 85 Time: 3.448764	Packet: 102 Time: 3.781519	Source: 52028 Destination: 59125	Request for the object: /datatypes
Packet: 88 Time: 3.452474	Packet: 109 Time: 3.789736	Source: 52031 Destination: 59125	Request for the object: /audioformats
Packet: 91	Packet: 114	Source: 52029	Request for the object: /audioeffects

Time: 3.453568	Time: 3.792904	Destination: 59125	
Packet: 106 Time: 3.785573	Packet: 119 Time: 4.116684	Source: 52028 Destination: 59125	Request for the object: /exampletext?datatype=TEXT&locale=fr
Packet: 107 Time: 3.785652	Packet: 117 Time: 4.115398	Source: 52030 Destination: 59125	Request for the object: /exampletext?voice=upmc-pierre-hsmm
Packet: 153 Time: 13.187564	Packet: 157 Time: 13.517300	Source: 52028 Destination: 59125	Request for the object: /exampletext?datatype=TEXT&locale=en_GB
Packet: 154 Time: 13.187643	Packet: 155 Time: 13.515919	Source: 52030 Destination: 59125	Request for the object: /exampletext?voice=dfki-spike-hsmm
Packet: 164 Time: 16.707955	Packet: 264 Time: 17.936071	Source: 52028 Destination: 59125	Requests audio speech synthesis using “dfki-spike-hsmm en_GB male hmm”
Packet: 266 Time: 17.955342	Packet: 275 Time: 18.289653	Source: 52028 Destination: 59125	Request for the object: /favicon.ico
Packet: 346 Time: 45.915136	Packet: 350 Time: 46.246085	Source: 52030 Destination: 59125	request for the object /exampletext?datatype=TEXT&locale=en_US
Packet: 347 Time: 45.915210	Packet: 352 Time: 46.247837	Source: 52029 Destination: 59125	Request for the object: /exampletext?voice=cmu-slt-hsmm
Packet: 360 Time: 49.636735	Packet: 807 Time: 53.240792	Source: 52029 Destination: 59125	Requests audio speech synthesis using: “cmu-slt-hsmm en_US female hmm” voice
Packet: 632 Time: 52.475285	Packet: 676 Time: 52.828758	Source: 52030 Destination: 59125	Request for the object: /favicon.ico
Packet: 829 Time: 61.794877	Packet: 836 Time: 62.129097	Source: 52029 Destination: 59125	Request for the object: /exampletext?datatype=TEXT&locale=de
Packet: 830 Time: 61.794954	Packet: 834 Time: 62.124788	Source: 52031 Destination: 59125	Request for the object: /exampletext?voice=bits4
Packet: 843 Time: 65.593038	Packet: 940 Time: 66.185934	Source: 52029 Destination: 59125	Requests audio speech synthesis using: “bits4 de female unitselection general”
Packet: 942 Time: 66.829421	Packet: 951 Time: 67.16969	Source: 52029 Destination: 59125	Request for the object: /favicon.ico

3 Comparison of the two MaryTTS systems from a networks perspective.

The leader of the New Zealand group who have installed MaryTTS on http://engresdev09.its.auckland.ac.nz (henceforth refered to as MARYNZ), wishes you to do a comparison between their system, and the original german system (http://mary.dfki.de:59125/) (henceforth refered to as MARYDE). In both captures three different voices were selected, and a phrase was synthesized. The voice cmu-slt-hsmm en_US female was the first voice selected for MARYNZ and the second for MARYDE. The remaining 4 voices were all different for the two systems. In both captures the first two phrases that were synthesized then streamed as an audio file were english, and were the phrase “Welcome to the World of speech synthesis”, the final phrase in the two captures were Māori and German respectively.

Your comparison should:

(i) First look at the setting up phase of the MaryTSS in the two MaryTTS systems. This will involve looking at the first series of GET statements (about 7) and identifying what the are doing (in a general sense). So, this starts from the initial down loading of the Mary TTS page, to the selection of the first voice in the capture (these are the set up commands, before any TTS session can start). Are all these steps similar in the two MaryTTS systems or are they different?

(ii) Your comparison should also include:

a) the total number of processes used in each capture used to synthesize the three phrases
b) the port numbers used
c) the total number of GET statements

d) the time taken to download the synthesised text spoken by the cmu-slt-hsmm en_US fem
e) the number of packets in the GET statements that precedes the streaming of the audio wav
f) the number of packets in the streamed audio files

Where differences are found spectulate on the causes of the differences.

(iii) The leader of the New Zealand group also wants you to investigate the time it takes to down load the synthesised text spoken by the sh_eu-hsmm en_NZ male Hmm voice and the akl_mi_pk_voice1-hsmm mi male hmm voice and compare these times to the the time it takes to taken to download the synthesised text spoken by the cmu-slt-hsmm en_US fem on MARYNZ.
Once again where there are differences speculate on the cause of the differences.

The two tables you produced earlier in parts 1 and 2 of this exercise will be very useful to you here.
This comparison should be written in formal english, i.e. with sentences, and is expected to be around 750 words.

The setting up phase of the MaryTSS in both the MaryTTS systems are identical for the first 7 GET statements. Afterwards, MARYNZ will receive a duplicate GET statement requesting for the /favicon.ico object. Both MARYNZ and MARYDE will then receive a GET statement corresponding to the selection of the first voice at packets 9 and 8 respectively.
The first GET statement for both MaryTTS systems is a request to download the main page of MaryTTS.
The second GET statement for both MaryTTS systems is a request for the object mary.js. This could possibly be code related to the functionality of the MaryTTS website.
The third GET statement is a request for the object favicon.ico. Browsers will automatically request the favicon when browsing to different sites.
The fourth GET statement is a request for the object /voices. This could be the list of the available voice options for the text-to-speech synthesis output.
The fifth GET statement is a request for the object /datatypes. This could be the list of available datatype options for the input text and output text fields used in the text-to-speech synthesis.
The sixth GET statement request for the object /audioformats. This could be the list of supported format types for the audio file output of the system.
The seventh GET statement request for the object /audioeffects. This could be the list of available audio effects and their default values that MaryTTS can use to modify the text-to-speech synthesis output.

To synthesize the three phrases, MARYNZ used 2 processes(“sh_eu-hsmm_en_NZ_male_Hmm” and “akl_mi_pk_voice1-hsmm_mi_male_hmm” on same process) and MARYDE used 2 processes(“cmu-slt-hsmm en_US_female_hmm” and “bits4_de_female_unitselection_general” on same process).

Daniel hor dhor051 516000089
SE364 a01

Both the MaryTTS systems have the same server(destination) port number of 59125. This is reasonable as a server socket will listen on a single port. The two MaryTTS systems both have a variety of client(source) ports. The MARYNZ capture shows the port numbers 58626, 58627, 58628, 58630, 58631, 58632, 58633, and 58634 are used as source ports. The MARYDE capture shows that the port numbers 52025, 52026, 52027, 52028, 52029, 52030, and 52031 are used as source ports. This difference is because source ports are randomly generated and assigned from the dynamic port range.

The MARYNZ capture contains 19 GET statements whereas the MARYDE capture contains 21 GET statements. This difference is due to the multiple duplicated GET statement requests for the /favicon.ico object. This GET statement request is duplicated 1 time in the MARYNZ capture and duplicated 3 times in the MARYDE capture. Without these duplicated GET statements, both MaryTTS captures would have the same 18 GET statements.

The captures show that the time taken to download the synthesised text spoken by the "cmu-slt-hsmm en_US fem" with MARYNZ was 0.076149seconds (=19.626276(last response)-19.550127(first response)) and with MARYDE was 3.604057seconds (=53.240792-50.611766). MARYNZ was faster by 2.938867 seconds(=3.604057-0.66519). This could be due to the difference in connection distance to the two servers. Long connection-distance could lead to latency-delay and the closer-distanced UoA-based server would have a faster download time compared to the further-distanced Germany-based server. The two servers could also have a differing bandwidth directly affecting data transfer rate.

The number of packets in the GET statements that precede the streaming of the audio wav is 12 for MARYNZ and 11 for MARYDE. This difference is due to the duplicate GET statement requesting for the /favicon.ico object. The remaining GET statements are identical for both captures.

The number of packets in the streamed audio files for MARYNZ was around 172 for "cmu-slt-hsmm en_US_female_hmm", 73 for "sh_eu-hsmm_en_NZ_male_Hmm", 27 for "akl_mi_pk_voice1-hsmm_mi_male_hmm"
The number of packets in the streamed audio files for MARYDE was around 62 for "dfki-spike-hsmm en_GB_male_hmm", 294 for "cmu-slt-hsmm en_US_female_hmm", 62 for "bits4_de_female_unitselection_general".
The difference in the number of packets for "cmu-slt-hsmm en_US_female_hmm" is possibly related to the difference in packet length of MARYDE(1466) MARYNZ(1420) used in the transfer.

On MARYNZ, the time taken to download the synthesised text was 0.093488seconds (=29.399980(last reponse)-29.306492(first reponse)) for the "sh_eu-hsmm_en_NZ_male_Hmm" voice, 0.013861secconds (=40.321462-40.307601) for the

"akl_mi_pk_voice1-hsmm_mi_male_hmm" voice and 0.076149seconds (=19.626276-19.550127) for the "cmu-slt-hsmm_en_US_fem" voice.
The lower time for the "akl_mi_pk_voice1-hsmm_mi_male_hmm" voice is possibly due to "Haere_Mai" text only having 3 syllable compared to the "Welcome_to_the_world_of_Speech_Synthesis", synthesised by the other two voices, which has 10 syllables. The lower number of syllables would result in a shorter video and a faster download.
Both "akl_mi_pk_voice1-hsmm_mi_male_hmm" voice and "sh_eu-hsmm_en_NZ_male_Hmm" voice synthesised the same message of "Welcome_to_the_world_of_Speech_Synthesis". However, the "sh_eu-hsmm_en_NZ_male_Hmm" voice had a faster download by 0.017339 seconds(=0.093488-0.076149). This difference in time could possibly be due to the "sh_eu-hsmm_en_NZ_male_Hmm" voice having a faster speaking-rate causing a smaller sized audio file to be generated.
Furthermore, there was a difference in the number of packets to transfer the three files from server to client. The server must wait for an acknowledgement before sending more packets resulting in longer download times for a larger number of packets.

Exercise 3

In this exercise you are going to compare the round trip times and throughput between the TCP stream which does the audio file transfer in Exercise 1 (saved in part 3), and a TCP stream that does an audio file transfer in Exercise 2 from file MaryUOA_A1.pcapng . To capture the streams look at the TCP conversations (Statistics->Conversation then click on the TCP tab) and identify which conversation contains the audio file transfer (Hint the port numbers, and the size of the bytes in the conversation may be useful to identify the conversations you wish to look at). The "follow the stream" option will enable you to check you have got an audio file transfer in the stream, then click graph a TCP conversation. Note in MaryUOA_A1.pcapng there are several TCP conversations with audio file transfers, choose one. Look at the roundtrip time graph for the two conversations, and the also the throughput graphs. Write up a discussion (around 750 words) on what the two graphs show and compare the plots for the two conversations, be sure to indicate

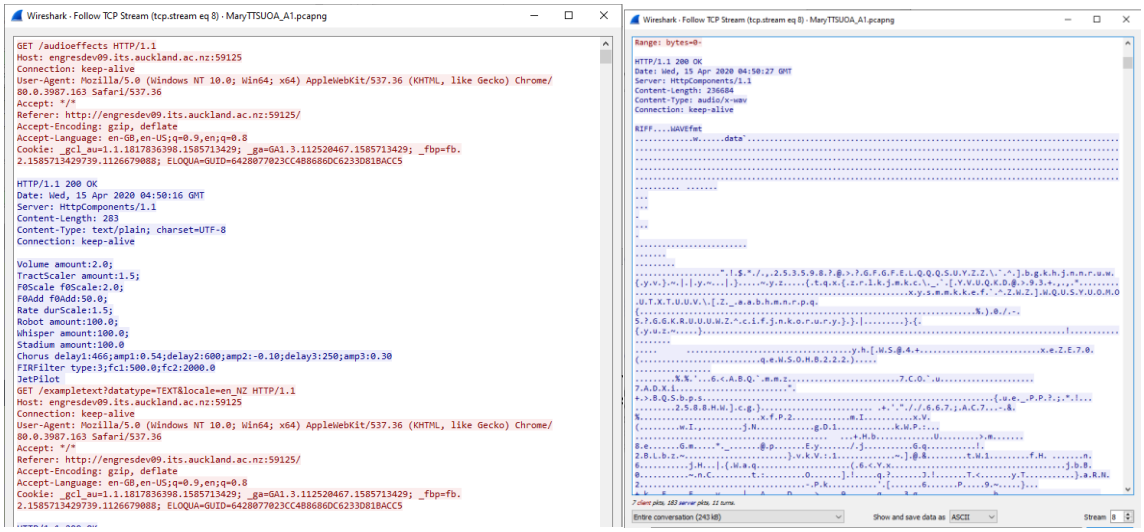
i. what data is being transferred in each TCP conversation,
ii. which parts of the data are the audio file, discuss differences in the round trip time ranges, and throughput
iii. what is goodput and discuss whether differences between goodput and throughput can be seen in the two conversations.

Be sure to include relevant wireshark screen shots to support your discussion.

Data

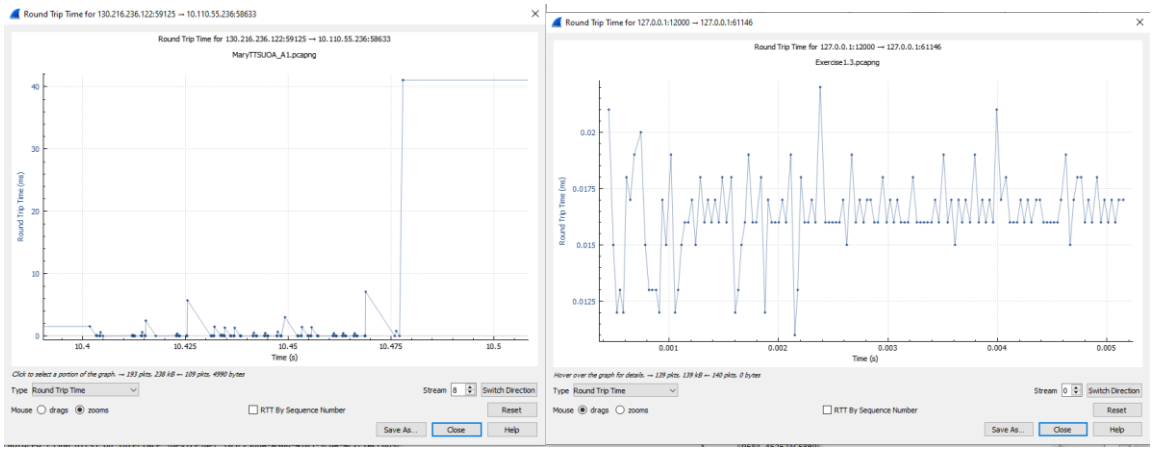
Daniel hor dhor051 516000089
SE364 a01

For the TCP conversation of Exercise 1 we can see that all the packets of data being transferred after the handshake and before the connection closing are audio files represented by ASCII characters only.



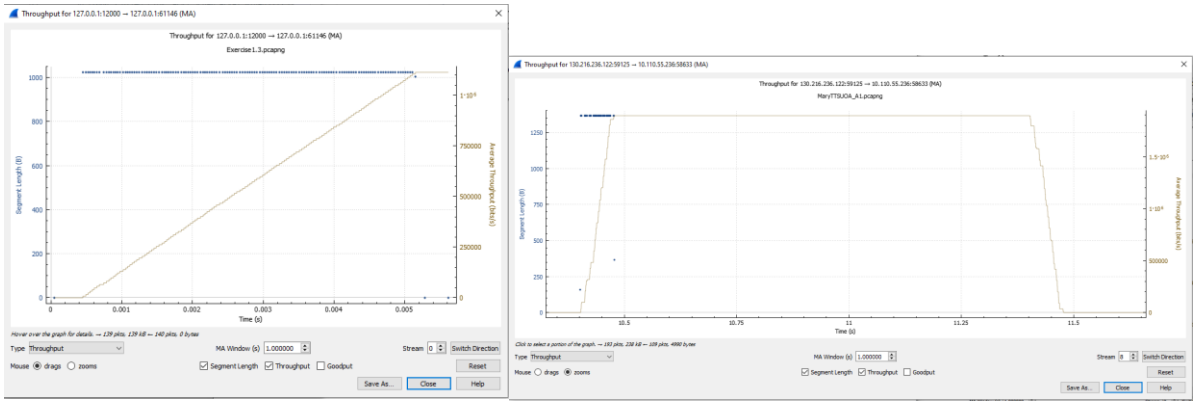
For the TCP conversation of MaryUOA_A1 there were data types other than audio transfered such as “text/plain” data types that relates to “/audioformats” and “/audioeffects”. The audio files transferred in this conversation were represented by ASCII characters. The connection has mixed data types observed due to the server using the same stream(in this case: stream 8) to respond to requests other than the audio file trasnfer.

Daniel hor dhor051 516000089
SE364 a01



The round-trip time graph of MaryUOA_A1 conversation captures round trip times between 0.01ms to 41.2ms whereas the round trip time graph of Exercise 1 conversation captures round trip times between 0.011 ms to 0.022ms. The range of captured RTTs for TCP conversation of MaryUOA_A1 was much larger than the conversation of Exercise 1. This shows that the packets for the MaryUOA_A1 audio file transfer took much longer to send and recive compared to the packets for the Excercise audio file transfer. The difference in RTTs is possibly due to Exercise 1 creating a local server to host the audio file. This results in minimal latency delay when transferring data packets as the packet does not need to travel. On the other hand, the MaryUOA_A1 required for packets to be sent between the client and the UOA server. This possibly explains the large variablity in the MaryUOA_A1 conversation RTTs as different packet may take different route to reach the UOA server. The latency delay caused by the connection length to the UOA server could explain the overal increase in RTTs of the MaryUOA_A1 conversation.

Throughput

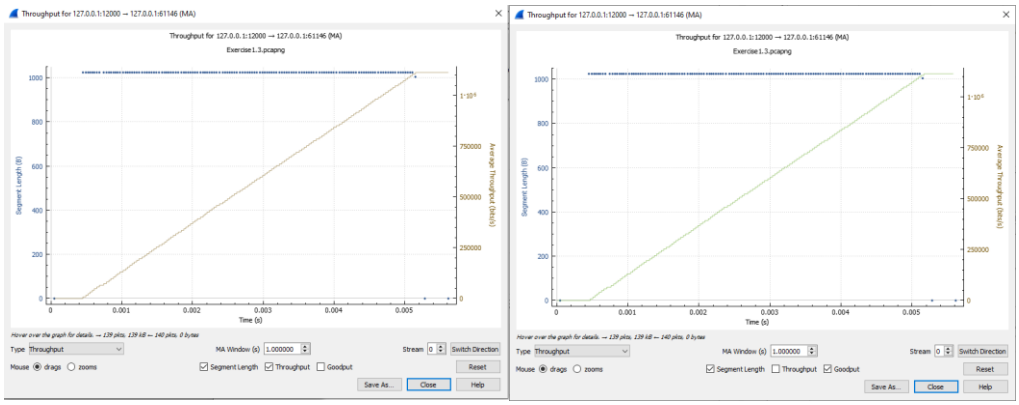


The throughput graph for the MaryUOA_A1(left) conversation shows a linear increase in throughput between 5 seconds and 10 seconds. This is due to the audio file transfer form the speech synthesis and the increase in throughput continues until the audio file transfer is complete and reached a peak throughput of approximately 1.9x10^6 bits/s.
The throughput graph for the Excercise1 conversation also shows a similar linear increase between 0.0005 to 0.005 seconds and reaches a peak throughput of approximately 1.1x10^6 bits/s.

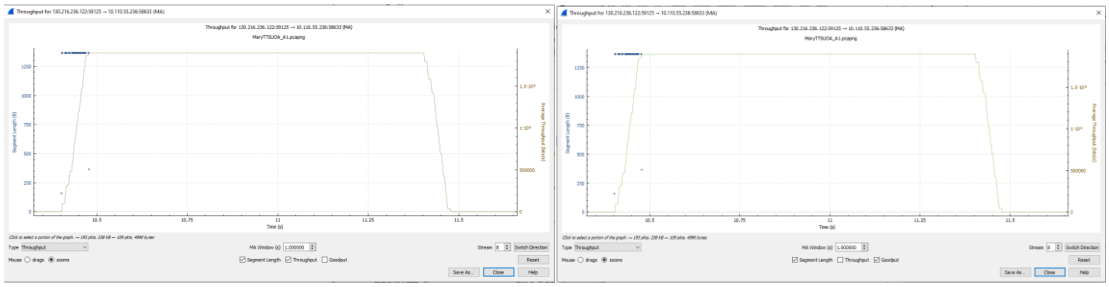
Daniel hor dhor051 516000089
SE364 a01

The throughput of the Excercise1(right) conversation increased at a significantly faster compared to the MaryUOA_A1 conversation. This indicates that the audio transfer of the Excercise1 conversation was faster compared to the Excercise1 conversation.
The MaryUOA_A1 conversation is observed to have more fluctation. This is possibly due to the differences in routing of different packets.

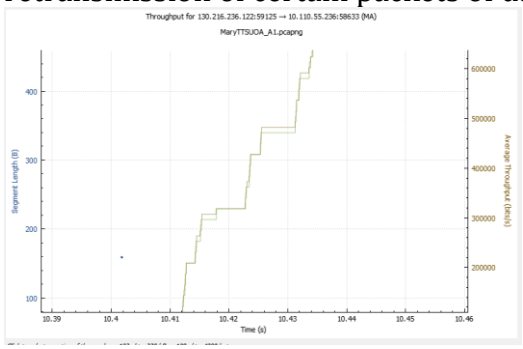
Goodput



Throughput is the measurement of all data of all dataypes flowing through stream whereas goodput is a measure of the rate of the useful data transferred.
From the the throughput(left) and goodput(right) graph for the Exercise 1 conversation, there is not much variance of the graphs indicating that the majority of the data trasnferred was useful data.



From the the throughput(left) and goodput(right) graph for the MaryUOA_A1 conversation, there is slight variance of the graphs. This could possibly be due to the retransmission of certain packets of data.



The MaryUOA_A1 graphs had little variance. This is possibly because the audio file data transferred was much larger in size than other data types that were transferred.

Daniel hor dhor051 516000089
SE364 a01

Exercise 4

Traceroute (called `tracert` on windows and `traceroute` on linux/Mac) can be used to identify the IP routers on the network path a packet travels on between the source and destination. It is invoked form the command line e.g. tracert mary.dfki.de). Traceroute, using the protocol ICMP, elicits responses from the router 1 hop away from the source, 2 hops away from the source, then 3 hops, 4 hops, and so forth until the destination is reached. Typically the output from traceroute is the measured round-trip time to each router on the path, the IP address and the DNS name of the router. The DNS name is useful for working out the organization to which the router belongs. Since traceroute takes advantage of common router implementations, there is no guarantee that it will work for all routers along the path, and it is usual to see "*" responses when it fails for some portions of the path. Use traceroute to look at the network path to both MaryTTS engines you have been investigating, i.e. tracert 130.216.236.1223, and tracert 134.96.190.2084 (if you are using mac/linux use traceroute, rather than tracert).

Using the traceroute output, sketch a drawing of the network path, (note you may not get all the routers, it is highly likely you will get many *, however complete the exercise with the routers you do know about). Show your computer (lefthand side) and the remote server (righthand side), both with IP addresses, as well as the routers along the path between them numbered by their distance on hops from the start of the path. The output of traceroute will tell you the hop number for each router.

To finish your drawing, label the routers along the path with the name of the real-world organization to which they belong. To do this, you will need to interpret the domain names of the routers given by traceroute. If you are unsure, label the routers with the domain name of what you take to be the organization. Ignore or leave blank any routers for which there is no domain name (or no IP address). This is not an exact science, so we will give some examples. Suppose that traceroute identifies a router along the path with a domain name like arouter.syd.aarnet.net.au. Ignore at least the "arouter" part as indicating a computer within a specific organization. For country-code top-level domains like ".au" (for Australia) the last three domains in the name will normally give the organization. In this case the organization's domain name is aarnet.net.au. Using a web search, we find this domain represents AARNET, Australia's research and education network. The "syd" portion is internal structure, and a good guess is that it means the router is located in the Sydney part of AARNET. Therefore for all routers with domain names of the form *.aarnet.net.au, you would write "AARNET" on your figure. While there are no guarantees, you should be able to reason similarly and at least give the domain name of the organizations near the ends of the path.

Remember to include a screen shot of your two traceroutes along with the above analysis.

[3] If you do have access to the University of Auckland VPN I encourage you to start this up before doing this traceroute, otherwize its likely not to complete, although you will still be able to do the exercise.

[4] This traceroute is likely not complete, but you will still be able to do the exercise.

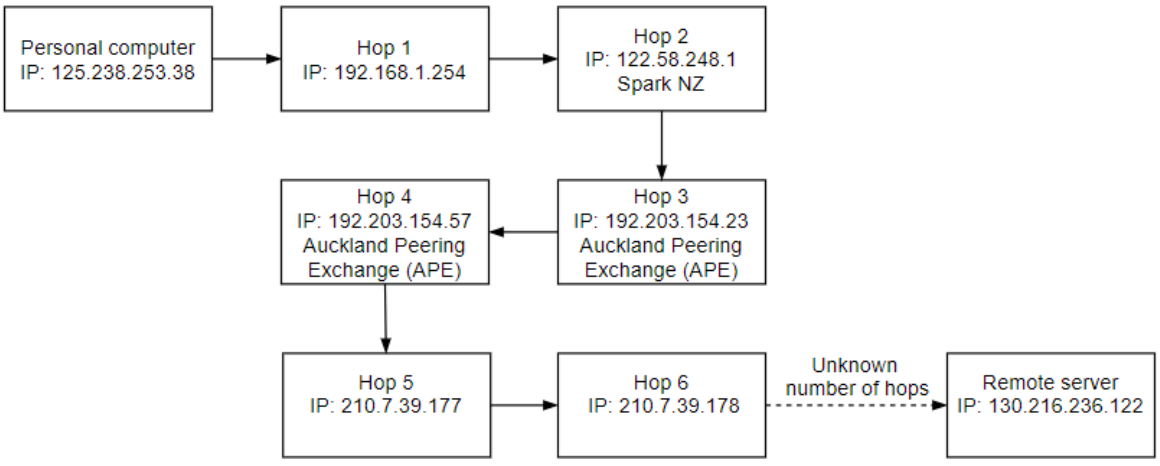
Daniel hor dhor051 516000089
SE364 a01

Traceroute for 130.216.236.122

Command Prompt

```
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\dh>tracert 130.216.236.122

Tracing route to engresdev09.its.auckland.ac.nz [130.216.236.122]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  192.168.1.254
  2     9 ms     5 ms     8 ms  122-58-248-1-vdsl.sparkbb.co.nz [122.58.248.1]
  3    36 ms     7 ms     7 ms  natcom.ape.nzix.net [192.203.154.23]
  4    11 ms     8 ms    11 ms  reannz.ape.nzix.net [192.203.154.57]
  5     8 ms    11 ms     5 ms  210.7.39.177
  6    10 ms     7 ms     7 ms  210.7.39.178
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9     *        *        *     Request timed out.
 10     *        *        *     Request timed out.
 11     *        *        *     Request timed out.
 12     *        *        *     Request timed out.
 13     *        *        *     Request timed out.
 14     *        *        *     Request timed out.
 15     *        *        *     Request timed out.
 16     *        *        *     Request timed out.
 17     *        *        *     Request timed out.
 18     *        *        *     Request timed out.
 19     *        *        *     Request timed out.
 20     *        *        *     Request timed out.
 21     *        *        *     Request timed out.
 22     *        *        *     Request timed out.
 23     *        *        *     Request timed out.
 24     *        *        *     Request timed out.
 25     *        *        *     Request timed out.
 26     *        *        *     Request timed out.
 27     *        *        *     Request timed out.
 28     *        *        *     Request timed out.
 29     *        *        *     Request timed out.
 30     *        *        *     Request timed out.

Trace complete.
```

Drawing of the network path for 130.216.236.122



Daniel hor dhor051 516000089
SE364 a01

Traceroute for 134.96.190.208

Command Prompt

```
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\dh>tracert 134.96.190.208

Tracing route to lnv-90208.sb.dfki.de [134.96.190.208]
over a maximum of 30 hops:

  1     5 ms     2 ms     2 ms  192.168.1.254
  2     9 ms     7 ms     6 ms  122-58-248-1-vdsl.sparkbb.co.nz [122.58.248.1]
  3    32 ms     7 ms     6 ms  222.152.41.139
  4     6 ms     8 ms    13 ms  122.56.113.4
  5    12 ms    13 ms     7 ms  ae7-2.akbr7.global-gateway.net.nz [122.56.119.53]
  6    36 ms    31 ms    30 ms  et10-0-5.sgbr3.global-gateway.net.nz [202.50.232.110]
  7    32 ms    33 ms    33 ms  ae2-10.sgbr4.global-gateway.net.nz [202.50.232.246]
  8    32 ms    31 ms    34 ms  103.13.80.125
  9   146 ms   148 ms   147 ms  ae-11.r31.tokyjp05.jp.bb.gin.ntt.net [129.250.5.34]
 10   145 ms   147 ms   146 ms  ae-3.r03.tokyjp05.jp.bb.gin.ntt.net [129.250.3.56]
 11   149 ms   158 ms   144 ms  ae-0.level3.tokyjp05.jp.bb.gin.ntt.net [129.250.8.202]
 12   324 ms   323 ms   330 ms  ae-1-5.bar1.Hamburg1.Level3.net [4.69.142.209]
 13   328 ms   329 ms   327 ms  195.122.181.62
 14   323 ms   323 ms   322 ms  cr-han2-be3.x-win.dfn.de [188.1.144.38]
 15   323 ms   322 ms   325 ms  cr-fra2-be12.x-win.dfn.de [188.1.144.133]
 16   336 ms   341 ms   340 ms  kr-saa17-0.x-win.dfn.de [188.1.245.14]
 17     *        *        *     Request timed out.
 18     *        *        *     Request timed out.
 19     *        *        *     Request timed out.
 20     *        *        *     Request timed out.
 21     *        *        *     Request timed out.
 22     *        *        *     Request timed out.
 23     *        *        *     Request timed out.
 24     *        *        *     Request timed out.
 25     *        *        *     Request timed out.
 26     *        *        *     Request timed out.
 27     *        *        *     Request timed out.
 28     *        *        *     Request timed out.
 29     *        *        *     Request timed out.
 30     *        *        *     Request timed out.

Trace complete.
```

Drawing of the network path for 134.96.190.208