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?

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?

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?
6. which packets were the shut-down?
7. what the source and destination IP address in the file transfer,
8. what port numbers were involved in the file transfer.

9. What packet did the file transfer start?
10. What packet did the file transfer end?
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?
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).

(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).

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

[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.

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 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

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.

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.

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.

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.