Networks Sub-module Assignment

(2022-2023)

1. Assignment introduction

This assignment includes three parts. Part 1 is to code an Instant Messenger in Python 3.10. Part 2 is to analyse a wireless transmission scenario and design your protocol. Part 3 is to demonstrate your understanding of switch operations. For detailed descriptions and tasks, please refer to the individual parts.

You should hand in your completed assignments via Blackboard Ultra. The deadline for your submissions is **2pm Thursday February 9th, 2023**. Each of you should include the following five files in your submission: the client source code (client.py), the server source code (server.py), the log file (server.log), a readme.txt containing any notes about the usage of the program and a PDF file answering the tasks in Part 2 and Part 3.

Please note that all submissions will be subject to plagiarism and collusion checks.

2. Assignment descriptions and tasks

Part 1: Implement an instant messenger (45 marks)

Note that all code should be written in Python 3.10 and must be stored in files named server.py and client.py. All code must run in Windows. Code must be stored in files named server.py and client.py. You must use the "socket" library directly (i.e., not via any other Python module) for network communications.

You are required to implement a client-server system, which implements an instant messenger **using TCP** allowing users to chat with each other. This instant messenger will consist of a client and a server program. You should be able to invoke server and client as follows:

```
python server.py [port]
python client.py [username] [hostname] [port]
```

For example, `python client.py John 127.0.0.1 12000` would enable a client using the username "`John`" to connect to `127.0.0.1` (i.e., the local system) via port number `12000`.

Please complete the following functions required for the server or clients in your programs.

1) The server and clients should implement the following functionality (28 marks):

- When a client connects, display a simple welcome message from the server. Note that this message must be sent over a network socket and should not be hardcoded on the client side. On the server, print where the connection is coming from (IP address and port number).
- Allow multiple clients to connect.
- On the client, provide an input prompt allowing the client to send messages.
- Client should be able to send multiple messages. These messages should be displayed to all currently connected clients.
- Use the username passed to client.py to print "[username] has joined" or "[username] has left" on all connected clients whenever a client connects or disconnects from the server (even when leaving due to connection loss).
- Also display the username in front of each message, so users can tell who is saying what.
- One of the connected clients disconnecting should not cause the server to crash.

2) Logging (5 marks)

Produce a log file called server.log when the server is run. This log should contain information about all clients connecting, disconnecting, as well as any messages sent. Sample output showing at least two clients connecting, chatting and disconnecting should be included in the submission.

3) Clients upload a file to server (12 marks)

This task is to enable a client to upload a file to the server. The ability to upload a simple text file will be acceptable. In detail,

- The server should be able to create an uploads folder for each client. This folder will contain the files being uploaded to the server from a client. Please use the username of the respective client to name each uploads folder. For example, a client with username "client1" should have an uploads folder named "client1" on the server.

  The uploads folder can be created at any time as long as it is present by the time uploaded files are received by the server. Please ensure that your code can create distinct uploads folders for multiple clients.

  (6 marks)
- Each client should be able to upload files to its uploads folder on the server. The files and uploads folders can be hosted within the same folder as your source code.

  Please ensure that your code can enable multiple clients to upload their files to their uploading folders.

  (6 marks)

****

Hints for Part 1 A:

1. Be sure to build up the functionality of your application step by step, for example, do not attempt to write the entire server before starting the client.
2. If you are unsure of how to start, I suggest you to have a look at the lab assignments 2 & 3.
3. I also suggest using a dictionary to store all current connections. Ensure that when a client drops out, that connection is removed from the dictionary.
4. You may find it useful to look at non-blocking sockets and select(), to ensure that all messages can be received. E.g. (note the format of the if statement):

```
r,w,e=select.select([client.connection],[],[client.connection])
   if client.connection in r: [...]
```

5. It is ok to send all messages directly as fixed length byte arrays over the sockets.

****

Part 2: Analyse a simple wireless network (20 marks)

Refer to the following wireless network scenario.

Assume a wireless network consists of an access point (denoted as X) and two wireless nodes (denoted as A and B respectively). Assume that wireless nodes **A and B cannot hear each other's transmissions**, but **they can hear X** (i.e. X can hear A and B). Suppose

- At time *0 μs*, X is sending a packet to some other node and it completes sending this packet at time *100 μs*.
- At time *20 μs*, a packet becomes available for transmission at A. A needs *150 μs* to send this packet.
- At time *60 μs*, a packet becomes available for transmission at B. B needs *100 μs* to send its packet.
- Let the value of the backoff timer for A be *40 μs* and the value of the backoff timer for B be *60 μs*.

1) Sketch the above described topology to include wireless nodes X, A, B, and their coverage. You may use tools such as Word, Paint, Visio, etc. to complete this topology.

(5 marks)

2) Analyse the above transmission situation and describe the transmission procedure. In your description, clearly state which node starts to sense and transmit at what time and how long it would take. Start your description from *0 us* in the network. Suppose the network only uses CSMA/CA without the RTS/CTS scheme.

(15 marks: 3 marks for the access point, 6 marks for each of the wireless nodes)

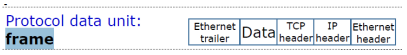
Part 3: Understand switch operations (20 marks)

We have 5 devices connected to an 8-port switch. Each device is numbered starting with 0, and each port of the switch is also numbered starting with 0.

- Device 0 has MAC address 40-4A-18-B2-63-DA and is connected to port 0 of the Switch.
- Device 1 has MAC address 00-0C-2B-AF-18-7B and is connected to port 1 of the Switch.
- Device 2 has MAC address 00-1D-D1-BC-DF-73 and is connected to port 2 of the Switch.
- Device 3 has MAC address AC-D9-D6-57-24-A3 and is connected to port 3 of the Switch.
- Device 4 has MAC address 04-5D-56-3E-A3-B4 and is connected to port 4 of the Switch.

The switch and all the devices are cold-started, and the following Ethernet frames are sent on the network in sequence:

- Frame 0 (source 40-4A-18-B2-63-DA destination 04-5D-56-3E-A3-B4)
- Frame 1 (source AC-D9-D6-57-24-A3 destination 00-0C-2B-AF-18-7B)
- Frame 2 (source 40-4A-18-B2-63-DA destination 00-1D-D1-BC-DF-73)
- Frame 3 (source 00-0C-2B-AF-18-7B destination 00-1D-D1-BC-DF-73)
- Frame 4 (source 40-4A-18-B2-63-DA destination 00-0C-2B-AF-18-7B)
- Frame 5 (source 04-5D-56-3E-A3-B4 destination 00-0C-2B-AF-18-7B)



1) The following questions ask you to find out which of these frames will be sent out from selected ports. (18 marks)

- Assume you are monitoring port 0 of the switch, which of the above frames would be sent out from this port?
- Assume you are monitoring port 1 of the switch, which of the above frames would be sent out from this port?
- Assume you are monitoring port 2 of the switch, which of the above frame would be sent out from this port?
- Assume you are monitoring port 3 of the switch, which of the above frame would be sent out from this port?

2) Give the switching table that the switch forms after forwarding these 5 frames.

(2 marks)

3. How to submit?

You will submit five files:

- server.py
- client.py
- server.log
- readme.txt
- a .pdf file. In this file, you present your answers for Part 2 and Part 3. A template for the .pdf file is in the "Assignment" folder of the Networks sub-module on Blackboard Ultra.

Please compress all files into a single .zip file, and name your .zip file as "YourName_YourID.pdf" (make sure to replace "YourName" with your own name and "YourID" with the ID given to you by the University. For example, `JohnDavidson_cgab36.pdf`). Please upload your compressed file to Blackboard Ultra for submission.

The submission deadline is **2pm Thursday February 9th 2023**.

4. Collaboration policy

You may discuss your work with anyone, but you must do your work yourself and comply with the University rules regarding plagiarism and collusion (https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/1/). Your submissions will be assessed for collusion and plagiarism.

5. Feedback Sheet

Criterion	Marks	Comments
Part 1 Instant messenger (45)		
Sever and client programs are implemented. Code realises all the functional requirements	/30	
Log file.	/5	
Error handling.	/10	
Part 2 Analyse a simple wireless network (20)		
Topology.	/5	
Transmission procedure description.	/15	
Part 3 Understand switch operations (20)		

Frames and their output ports.	/18	
Switching table.	/2	
Others (15)		
Code quality.	/8	
Presentation and readability (for the .PDF file).	/7	

Note: while your readme.txt file won't be marked, the information in this file will be used to support marking for the required coding functions. Therefore, please do remember to include key notes about the usage of your program so that the function requirements listed in Part A can be checked appropriately.

Code quality:

The code should be short, easy to read and understand. In particular, this means that:

- Program code is commented where necessary for understanding. Excessive comments should be avoided.
- Programs are appropriately structured (e.g., correct and consistent indentation style).
- An appropriate naming convention is used for variables, methods/functions and classes throughout the project. See https://www.python.org/dev/peps/pep-0008/.
- Appropriate use is made of external libraries and there is no evidence of redundant code, e.g. importing unused Python modules, functions that are never called, etc.
- Correct/appropriate use of conditional statement and iterative statements.
- The code is not over-engineered, i.e. it does not introduce unnecessary data structures or include overly-generic functions.

Presentation and readability:

Content should be concise yet clearly describing the full idea and protocol. In particular, this means that:

- Words should highly focus on answering the questions or describing the protocol steps. Too much background content or other irrelevant content will lose points.
- Avoid very long sentences. Consider using different ways of presenting your ideas (say tables, figures, etc.) as appropriate.
- Font(s) should be legible, particularly for those in a figure or a table if you use figures or tables.
- Format and fonts should be consistent throughout the file.