EE 419 Introduction to Computer-Communication Networks
Final Project

Due Wednesday, 06/08/22 at 11:59 pm

(Rev. 1)

Individual

1. Describe a day in the life of a web page request. This specific request is for a video on www.youtube.com, which uses DASH (Dynamic Adaptive Streaming over HTTP) to enable video streaming. Illustrate the **sequence**, **purpose**, **header format** of the different protocols at various levels of the stack. Measure the **time taken** by each protocol *wherever possible*. Support your answers by providing Wireshark packet capture screenshots for the protocols. In order to capture a packet either from or to YouTube's servers, start a capture session in Wireshark and open up www.youtube.com in your browser and stop the packet capture. Use filter "tcp contains youtube" to filter your results. Additionally, other protocol packets (DHCP, ARP etc.) may also be captured in Wireshark.

   Each protocol should be described in at least 150 – 200 words. It is expected that your investigation will result in a document 5 – 10 pages long (12 pt., 1 – 1.5 pt. spaced with screenshots).

2. Choose your favorite protocol among the ones you explored before and write a 3-page summary of its RFC.

(Rev. 1) – P1. Added packet capture instructions & length guidance

Group

3. Design a TCP-based chat application using Python. Use a server-client model with a `server.py` and a `client.py` script.

Requirements

1. `Server.py`
   - Define all functions within a `Server` class.
   - Use of `threading` library to allow multiple connections.
   - `server_init()` creates a server TCP socket object. The function should also take user input for the server IP and port number and bind it to the socket created. This socket should be always listening. Additionally, make a list to maintain clients. This function should also implement code to accept clients and have relevant print statements.
   - `broadcast(user_message)` Broadcasts a client's `user_message` to all other clients.
   - `client_handling(client_socket, client_address)` Receives messages from different clients. Optionally, when a client joins the server for the first time, broadcast its name using the `broadcast()` function.

2. `Client.py`
   - Define all functions within a `Client` class.
   - Use of `threading` library to allow multiple functions to run simultaneously.
   - `connection_init()` Creates a client socket with the user entering server IP and port. After the socket is created, a connection to the server should be established. Once a connection is established between the client and server, ask the user to enter a username and send it to the server.
   - `message_handling()` Prints messages received from other users.
   - `input_handling()` Sends user messages with their username concatenated to the server.

3. Comment your individual contributions at the start of `Client.py`.

Deliverables

- `Server.py, Client.py`
- Video demo (30 – 60 seconds) showcasing two different computers (on the same subnet or LAN) running `Client.py` and able to exchange messages successfully. As always, `Server.py` should be run first – you may choose to run it on either of the computers.

(Rev. 1) – P3. Added optional functionality in `client_handling()`
Added comment requirement
Specified video length