CSC329 - Computer Networks
Assignment-2

King Saud University
Department of Computer Science
Fall 2021
Dr. Azzam Alsudais

Assignment Instructions

- **Teamwork**: Students need to perform this assignment in teams of only 4 students.
- **Academic integrity**: The work in this assignment should be the students' own original work. Any form of plagiarism will not be tolerated, and will result in a zero mark for the whole assignment.
- **Grading**: Grading is going to be offline. However, the instructor reserves the right to perform interview-based grading, in which students answer additional questions about their work on the assignment.
- **Submission**: See the end of this document for submission instructions.
- Please read the entire assignment prior to solving any question.

Software/Hardware Requirements

- **Host Machines**: You will need 3 host machines (team members' laptops) to test your work.
- **Operating System**: You may use whatever OS you find yourself comfortable with.
- **Programming Language**: You may use any programming language to implement the required functionality. However, you are highly encouraged to use Python.

Due Date is Dec 11, 2021 at 11:55PM

Student-1 Name: .....

Student-1 KSU-ID: .....

Student-2 Name: .....

Student-2 KSU-ID: .....

Student-3 Name: .....

Student-3 KSU-ID: .....

Student-4 Name: .....

Student-4 KSU-ID: .....

Introduction

The goal of this assignment is to program a chat application using TCP as an underlying transport-layer protocol.

Part-1 (5 Points)

In this part, you are asked to write a chat application. You will need to write a program that is able to connect to another instance of it and allow users (at both sides) to write and exchange messages.

Program Behavior

Here is a set of steps that describe the behavior of the program.

1. When the program is first launched, it needs to create a TCP socket that listens to (and accepts) new connections.
2. After it accepts a new connection, it needs to start taking input (text messages) from the client.
3. For simplicity, assume that users take turns writing messages. For instance, user-1 sends a message and waits for user-2 reply. During that time, user-1 cannot send a new message (until it receives a reply from user-2).
4. Whenever a user (at either side) types the word `exit`, the program should close down the connection and terminate.

Requirements

Here's a set of requirements that should be considered when writing the program.

1. The chat program must use TCP sockets.
2. Implement the chat program and name it `chat_v1.py`.
3. Each program instance must run on a separate host (i.e., programs run on different machines). These machines need to be connected to the same local area network.
4. The local port number as well as the IP address and port number of the other machine must be passed as command-line arguments. This is to help you use either machine to initiate the connection. For example, if the local port number is `11329` and the other machine's IP address is `192.168.1.5` and port number is `22329`, then it should look like the following (this order must be maintained):

```
python chat_v1.py 11329 192.168.1.5 22329
```

Part-2 (5 Points)

In this part, you need to modify the program so that clients do not necessarily need to know each other's IP and port number information. Instead, they consult another machine (e.g., a `tracker`) for the information of other clients.

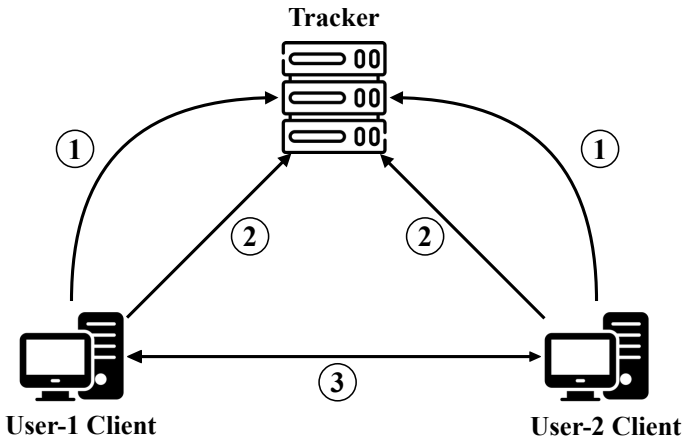

Figure 1: Program behavior of **Part-2**

Program Behavior

Figure 1 shows the steps that need to take place when a client wants to connect and chat with another client.

1. First, each client needs to register with the `tracker`. The `tracker` extracts each client's IP and port number information and keeps track of it (store it in memory).
2. Second, each client keeps checking with the `tracker` for new clients (i.e., are there any other registered clients?). Note that, you may utilize the reply from the `tracker` in step-1 to indicate that there are already other registered clients. When other clients have already registered, the `tracker` should respond with the appropriate information (IP address and port number of the other client).
3. Third, the TCP connection can now be established and clients can chat with one another (this is already done in **Part-1**).

Requirements

Here's a set of requirements that should be considered when writing the program.

1. The program must use TCP sockets for `client-client` and `tracker-client` communication.
2. Modify the chat client program so that it communicates with the `tracker` for the other client's information and name it `chat_v2.py`.
3. Implement the `tracker` program and name it `tracker.py`.
4. Each program instance must run on a separate host (i.e., programs run on different machines). This applies to the `clients` as well as the `tracker`. These machines need to be connected to the same local area network.
5. The IP address and port number of the `tracker` must be passed as command-line arguments to `chat_v2.py`. This is to help you use either machine to initiate the connection. For example, if the `tracker`'s IP address is `192.168.1.10` and port number is `55555`, then the following command should be used to run the application (this order must be maintained), and `tracker.py` must be already running:

   - `python chat_v2.py 192.168.1.10 55555` (**No local port is provided here**)

   - **Note**: the local port must not be provided as an argument. Instead, it should be selected at random each time the program gets executed (use the 21000-22000 range for the random number generator).

Part-3 (Extra credit: 2 Points)

In this part, you are asked to remove the `tracker`. In addition, modify the programs such that each client knows only the IP address of the other client. In other words, when user-1 wants to connect to user-2, assume that user-1 knows only the IP address of user-2 but not the port number. As such, user-1 will have to "scan" for open ports at user-2's host. You may use the same range of port numbers as in **Part-2**, and you need to name the program `chat_v3.py`.

Submission

Collect the following files in one folder, compress it, and name it `csc329_assignment2_gx.zip` where `x` is the group number in the google document (starting at 3). **Only one student needs to submit.** Here is what needs to be submitted.

- **Part-1**
  - Source code of `chat_v1.py`.
- **Part-2**
  - Source code of `chat_v2.py`.
  - Source code of `tracker.py`.
- **Part-3**
  - Source code of `chat_v3.py`.

Grading

Grading will follow the following criteria.

- 15% of the whole grade will be used to assess code quality. Easy-to-read and clean code is highly encouraged.
- 15% of the whole grade will be used to assess whether the programs run or not.
- 70% of the whole grade will be used to assess whether the programs work as intended. I will be running the programs on different machines to test whether they are capable of communicating with one another.

Late Submission Policy

The due date to submit your work is December 11, 2021 at 11:55PM. Late submission will be accepted (no later than final exam date), but they are subject to 30% deductions.