University of Calgary
Faculty of Science
Department of Computer Science

CPSC 441- Computer Networks
Assignment 1
Socket Programming - TCP
Winter 2023

1 Objectives

The goal of this assignment is to practice principles of socket programming in C/C++ languages as well as the concepts that you have learned about TCP protocol in the course material. In this assignment you will implement a simple server and client in either C or C++ that interact with each other in a certain application-layer protocol discussed below.

2 Details

2.1 Introduction

TCP is a connection-oriented protocol, whereas UDP is a connectionless protocol

A server and client can connect to the network and send/receive data using either of the two important transport-layer protocols: TCP and UDP. In this assignment, you will implement both the server and client using TCP protocol. Moreover, a server and a client can communicate with each other semantically using different application layer protocols. The most commonly used application layer protocols include HTTP, FTP, SMTP, etc. In this project however, for the purpose of simplicity, you are going to use a customized application-layer protocol that is explained in section 2.2. Protocol Behavior. As also mentioned above, the high-level (application-layer) protocol used in this assignment is a customized protocol that is described in pseudocodes below. You should follow these behaviors in your client and server program to communicate with each other.

2.2 Protocol Explanation

When the client connects to the server, the client application should prompt the user to ask their UCID. Afterwards, this UCID is sent to the server. In response, the server sends

the current datetime of the system (i.e., "Mon Oct 2 10:59:33 2017"). The datetime can be in any format but it is important that both the server and client agree on the same format. Then, the server generates a PASSCODE which works like an authorization code (but it's not safe at all!). The PASSCODE is generated by extracting the seconds part of the time (i.e., 33) plus the last four digits of UCID (i.e., 5359). So, the PASSCODE in this example is 5392. Please note that the PASSCODE is not sent to the client directly and the client is responsible to also create it in the same way. Client will then send this PASSCODE to the server as an authorization code and if both PASSCODEs (the PASSCODE server generates and the PASSCODE server receives from client) are the same, server will open and read a text file (from its own run directory) and send the content to the client. Client should receive the content and save it in a text file in its own directory. Below are the pseudocodes for both server and client behaviors.

Server Behavior

```
1: while TRUE do
2: Listen for incoming requests (from clients using TCP) over a specific port
3: Accept a new connection
4: Receive user's UCID from the client
5: Send current day and time to the connected client
6: Create a PASSCODE.
        //PASSCODE = seconds part of that date time + last four digits of UCID
7: Read an integer from the client
       //It should be the same passcode but constructed and sent by the client
8:     Open the file \data.txt" from the run directory and
       send its contents to client
9:     Close the client socket
```

Client Behavior

```
1: Create a TCP connection with the server
2: Get user's UCID as an input in the client and send it to the server
3: Read a string which is the current time sent by the server
4: Create the PASSCODE
    //PASSCODE = seconds part of the current time + last four digits of UCID
5: Send the PASSCODE to the server
6: Read the content of the text file sent by the server and
    save what it reads from the socket into a text file in the run directory
7: Close the socket and file stream
```

2.3 Implementation of the Sever and the Client

In this assignment, you are responsible for implementing both the server and client in either C or C++ using TCP sockets. Both server and client should be run on local host (client uses local host and server uses INADDR_ANY as their IP address). The port on which the server listens to for incoming requests must be given to both server and client as a command line argument. Server uses this port number to listen for any incoming request, and the client will use it to reach out to the server. In this assignment, for the purpose of simplicity, you can assume that the server is always serving at most 1 client and not more. In other words, you are not required to handle non-blocking (asynchronous) IOs. However, it's important to note that this is not the case in real web applications since the server should respond to several clients in parallel and not be blocked on one of their IOs. Important note: Your programs (both client and server) should have sufficient logging for every stage or event, i.e., upon connection or disconnection of a client in the server, message/file transfers, etc.

2.4 Exception Handling

There are many different exceptions that might happen when implementing socket programming. You should include handle of some of them that were discussed in the course material. For example, when the connection attempt has failed because the IP address or port was wrong. Another case would be related to error handling while reading from a file.

2.5 Running the Code

In your implementation, you should have the following folders:

1. Server directory: this folder contains the server code file (server.c or server.cpp) and a text file named data.txt. This text file can contain any texts.

2. client directory: this folder contains the server code file (client.c or client.cpp)

To compile and run your code, follow the steps below:

1. Firstly, go to server directory and open a terminal. Use one of these commands to compile your code and generate an output file:

   ```
   gcc server.c -o server (for C)
   g++ server.cpp -o server (for C++)
   ```

2. Then, your server should run using this command (port_x is the port number that server will listen to):

   ```
   ./server port_x
   ```

3. Now go to the client directory and open a terminal. Use one of these commands to compile your code and generate an output file:

   ```
   gcc client.c -o client (for C)
   g++ client.cpp -o client (for C++)
   ```

4. Then, your client should be run and connected to the server using this command:

   ```
   ./client port_x
   ```

5. Now that both server and client are running, you can start inputting commands in the client terminal.

2.6 Correct Output

As mentioned earlier, the output messages (loggings) should be informative and clear.

1. First, the received UCID should be printed on the server terminal.
2. Second, the received time from the server should be printed on the client terminal.
3. Third, the client should print the PASSCODE that it sends to the server.
4. Forth, each time the client receives the text content from the server, it should Print: readBytes + " Received from the Server!" where "readBytes" is the number of bytes read from the server in each iteration in the loop.
5. Finally, it should save the text file in its own directory (Client folder). So, you should be able to open the text file from the client's directory and see the exact content from the original text file from the server's directory.

2.7 Submission and Demonstration

All submissions must be uploaded through the D2L dropbox. Please follow the instructions of your TA. The following should be submitted: A zipped file containing two folders (client and server). The client directory contains client code and the server directory contains server code and a text file with some content. The file should be named as follows: your first name, last name, zipped extension. For example if you name is John Doe, your submitted document would be called 'JohnDoe.zip'.

You will be asked to demonstrate your program to your TA on your laptop in a time slot during one of the tutorials (more information will be uploaded on d2l in due time). During your demo time, you should download the submitted file from D2L before the deadline and present that version to your TA.

You are URGED to check your submission after uploading it on D2L. Empty submission after the deadline will be subject to the indicated penalties.

2.8 Grading Rubric

This assignment will graded out of 25% but it will carry 12.5% of the total grade. Work should be done individually. Any source of help MUST be indicated and cited.

The following are considered in grading your assignment:

1. 2 points for a suitable demonstration of your code to your TA. A successful demo will include points for clear answers to questions asked during your code walk-through
2. 7 points for sending and receiving UCID
3. 7 points for sending and receiving time
4. 8 points for comparing PASSCODEs and sending the text file
5. 1 points for exception handling
6. You will receive less than 20% if your program compiles, but it doesn't work at all.
7. You will receive 0 points if your program will not compile.
8. For identical code, in part or full, the UofC rules and regulations for plagiarism will apply.

   It's better to have something working, even with little functionality, than a big program that crashes (or doesn't compile). We expect to see your own program, otherwise the above will apply.

2.9 Tips

1. If you have never done socket programming in C/C++ before, please make sure to attend your CPSC 441 tutorials on this topic and also use online resources.
2. Make sure that you have understood a basic introduction to client-server socket programming.
3. Start with very simple version of socket programming. Once you have these working (simple messages are being sent and received between client and server), then you can try to add more details to your code.

3 Deadline

Friday February 3, 2022 before 11:59 P.M. After the due time, a %20 deduction will apply on every late day or part of the day. The first 15 minutes after the deadline is not considered as part of the delay.

3.1 Plagiarism

Any resources which have significantly helped you in making your code should be cited, and you should not copy paste code from online websites but rather write all code by hand. A tool used to check plagiarism will be ran on student submissions. This is an individual assignment and students MUST not collaborate on the assignment in terms of sharing code under any circumstances. Even after the deadline, you must not share your code or make it available for others.