CSCI 452: Network and Web Programming

TCP Socket Project

Due February 12th, 11:59:59 PM

- Submit your assignment at http://www.networks.howard.edu/lij/courses/submit_hw.html
- Double check your submission. (You may submit multiple times with the latest one considered for grading.) No correction will be allowed after deadline even if wrong files are submitted.
- Create a project on Github and commit your code periodically. Make your project private initially and make it public at the deadline. (To create a private repository for free, sign up with your .edu email address.)
- Materials to be submitted (Zip all the materials and submit the zip file)
  a. C code of the client.
  b. C code of the server.
  c. A plain text file named readme.txt that contains the following items:
     i. The URL of your project on Github.
     ii. The date and time, brief description in one or two sentences, and the URL of the most significant commits. At least five are needed for each of client and server.
- The commit history is expected to show the progress of your development over time. There must be at least 5 commits for the program(s) of each side (client and server). While developing, a commit must be done before any test run of the program or when any non-trivial changes have been made. At least one commit must be done every hour of development.
- Grading

No credits will be given and the submission will not be evaluated further if any of the following is true.

  - Any of the required files has a wrong format.
  - Any of the required files is missing.
  - The commit history does not satisfy the requirements.
  - The commit history shows suspicious practice.

After that, programs will be judged according to the following criteria.

  a. Indentation: 5%
  b. Comments in code: 10%
  c. Correction of programs: 85% (judged by the outcome of test cases)

In this project, you will develop two network programs in the C language by modifying your first project, one for client and one for server.

The client program keeps reading input from user and performs actions as the following:

- If the user enters 's', it will ask user to enter a string. It then sends "CAP\nxxx\n" to the server via UDP (where "xxx" is the string entered by the user), and receives a message from the server also via UDP and output the message on the screen.
- If the user enters 't', it will ask user to enter a string. It then sends "FILE\nxxx\nkkk\n" via UDP to the server (where "xxx" is the string entered by the user), and receives a message from the server also via UDP. "kkk" is a numeric string of the TCP port number that the client listens on for file transfer.
  - If the message is "OK\n###\n", it will accept a TCP connection request from the server and receives the file through the TCP connection. "###" is the number of bytes of the file. The file should be saved in the name "xxx" in the directory where the client program is in.
  - If the message is "NOT FOUND\n", it will display a message of "xxx not found." on the screen, where "xxx" is the file name.
- If the user enters 'q', it will exit.
- If the user enters anything else, it will ignore.

The server program performs the following actions:

- If "CAP\nxxx\n" is received via UDP from the client, it capitalizes the received string and sends "CCC\n" back to the client also via UDP, where "CCC" is the capitalized string.
- If "FILE\nxxx\nkkk\n" is received via UDP from the client,
  - If the file named "xxx" exists in the directory where the server program is in, it sends "OK\n###\n" to the client via UDP, where "###" is the number of bytes of the file. After that, it connects to the client at the port "kkk" via TCP and sends the latter the content of the file.
  - If the file named "xxx" does not exist in the directory where the server program is in, it sends "NOT FOUND\n" to the client via UDP.

The client should be invoked by the following command:

```
<client> <TCP port> <server IP> <server UDP port>
```

Where `<client>` is the name of the client executable file name, `<TCP port>` is the port that the client listens on for the server's TCP connection request, `<server IP>` is the IP address of the server, and `<server UDP port>` is the UDP port of the server.

The server should be invoked by the following command:

```
<server> <UDP port>
```

Where `<server>` is the name of the server executable file name, and `<UDP port>` is the port the server listens on.