EECE461/ECEG761 Network Security Systems

Homework 1

Socket Programming

(100 points)

Feb 24th, 2016

Objective

- Understand basic client/server network communications.
- Understand network communications using TCP protocol.
- Practice socket programming skills and asynchronous I/O.

Description

In a simple client/server paradigm network, usually a very large file is chunked into n small files, i.e. chunk 1 to chunk n, that are shared among the clients. A client is a user who wants to download some chunks from other clients, and eventually download all chunks and get the entire file. In turn, when asked, a client can also upload some chunks that he has to other clients. In order to let other clients know that he has these chunks, a centralized server is utilized to keep track of the chunks status. In this project, there are 64 chunks. Each client maintains a bit vector, called "chunk vector" to indicate that whether he has this chunk or not. If he has this chunk, the bit is set 1. Otherwise, the bit is set to 0.

Client Functionalities

1. **Initialization.** The client will be invoked with a command line which contains two arguments. The first argument could be the server's IP address in dotted decimal notation or the server's domain name. If it is given the IP address, gethostbyaddr() function is called to get the domain name and print it out. If it is given the domain name, gethostbyname() is called to find the server IP address and prints it out. The second argument is the name of a configuration file that he can read information from. Configuration file includes client id number, server's listening port, his own listening port, his initial chunk vector.

2. **TCP connection to server**. Client tries to connect to server by creating a socket and then calls the connect function. The client prints out a success message if the connection is successful. Otherwise, an error message is printed out and the client quits the connection.

3. **Register with server**. Client sends server a message with information of his id, listening port, and chunk vector. A send function will be used for sending information.

4. **Command processing from user**. Client enters an infinite loop waiting to accept the two commands, "f" and "q". If the user types "f", the program will ask the user which chunk that he needs, then the user will input the chunk index. If the user types "q", the client quits and sends the server a message "quit" to notify them. Then the client waits until the server closes the connection

with him, and quits the connection. Select () function will be used for asynchronous I/O that can process user command and read message coming from server. For example, if the user types "f" and indicates that he needs chunk 8, he will first check his chunk vector if he has chunk 8. If he has it, a message will print out like "I already have chunk 8". Otherwise, he will send a query to the central server and the server will reply with a list of clients with chunk 8. This list will be printed out. After that, he waits for the next command.

5. **Command processing from server**. Client will read commands from the server. Only one command: "exit" will be handled by the client. If the client receives "exit" command from the server, client will call close () function and close the connection with server.

Server Functionalities

1. **Initialization.** Server creates a socket, and bind this socket to port 5000. This port is the listening port for this project. Listen () function is called to wait for incoming connection requests.

2. **Request acceptation.** Server will call accept () function to accept the connection request from clients. The return value of accept () function is a new socket that is used for communication with clients. Server reads the message from clients that include client id, listening port, and chunk vector.

3. **Query processing from client.** If a client sends the server a query for a file, the server first prints out the client id, IP address and chunk index that client is asking. Server will print out a list of clients who have the chunk, and send the list back to the query client.

4. **Command processing from user.** Server receives only one command from the user, "q". If user types in this command, the server sends to all connected clients an "exit" message. Server waits all until all clients have closed their connections, then exit finally.

5. **Command processing from client.** If a client sends a "quit" message to the server, server prints out a message like "(client id) at (IP address) has quit," then close the connection with the client.

Grading Policy

1) A proper README file (5pts)
2) Client functionalities (40pts)
3) Server functionalities (40pts)
4) Code comments (10pts)
5) Code readability (5pts)

Submission

You should submit a single file via Moodle. Tar or zip your readme file, makefile, all source code files into one file. You should demonstrate your work before submission.