Assignment 1
Introduction to Socket Programming in C/C++

1 Introduction

Berners-Lee and his team are credited for inventing the original HyperText Transfer Protocol (HTTP) along with Hyper Text Markup Language (HTML) and the associated technology for a web server and a text-based web browser. The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page. What you're about to do is to reinvent the wheel on the motivation of getting a deep understanding of how HTTP works!

In this assignment, you will use sockets to implement a simple web client that communicates with a web server using a restricted subset of HTTP. The main objective of this assignment is to give you hands-on experience with UNIX sockets.

2 Part 1: Multi-threaded Web Server

2.1 Introduction and Background

Please refer to the lectures' slides for the format and the use of HTTP.

2.2 Specifications

Your web server should accept incoming connection requests. It should then look for the GET request and pick out the name of the requested file. If the request is POST then it sends an OK message and waits for the uploaded file from the client. Note that a GET request from a real **WWW** client may have several lines of optional information following the GET. These optional lines, though, will be terminated by a blank line (i.e., a line containing zero or more spaces, terminated by a '\r\n' (carriage return then newline characters). Your server should first print out the received command as well as any optional lines following it (and preceding the empty line).

The server should then respond with the line, this is a very simple version of the real HTTP reply message:

HTTP/1.1 200 OK\r\n

then in case of GET command only:

{data, data, ..., data}

followed by a blank line (i.e., a string with only blanks, terminated by a '\r\n'). After finishing the transmission, the server should keep the connection open waiting for new requests from the same client. If the document is not found (in case of GET), the server should respond with (as would a real http server):

HTTP/1.1 404 Not Found\r\n

2.3 Server-Side Pseudo Code

while true:
do Listen for connections
Accept new connection from incoming client and delegate it to worker thread/process
Parse HTTP/1.1 request and determine the command (GET or POST)
Determine if target file exists (in case of GET) and return error otherwise
Transmit contents of file (reads from the file and writes on the socket) (in case of GET)
Wait for new requests (persistent connection)
Close if connection timed out
end while

2.4 Notes

- You should handle the both commands GET (to get file from the server) and POST (to send file to the server).
- In the case of POST command, file content is sent in the request message body.
- No validation (on requests) required. However, you should write the described format of the command.
- There are different types of HTTP status codes, you're only required to handle 404 and 200.
- You will want to become familiar with the interactions of the following system calls to build your system: socket(), select(), listen(), accept(), connect().
- Command to run the server:

  ./my_server port_number

- You are supposed to handle HTML, txt and images.
- You're free to choose whether to implement a multi-threaded or multi-process approach, justifying your choice.

3 Part 2: HTTP Web Client

3.1 Specifications

Your web client must read and parse a series of commands from the input file. For this assignment, only the GET and POST commands are required to be handled. The commands syntax should be as follows, where file path is the path of the file on the server (including the file itself):

client_get file-path host-name (port-number)

client_post file-path host-name (port-number)

Note that the port-number is optional. If it is not specified, use the default HTTP port number, 80. In response to the specified operation (GET or POST), the client must open a connection to an HTTP server on the specified host listening on the specified (or default) port number. The receiver must display the file and then store it in the local directory (i.e., the directory from which the client or server program was run). The client should shut down when reaching the end of the file.

3.2 Client-Side Pseudo Code

Create a TCP connection with the server **while**
more operations exist **do**
    Send next requests to the server
    Receives data from the server (in case of GET) or sends data (in case of POST)
end while
Close the connection

3.3 Notes

- Command to run the client: ./my_client server_ip port_number
- Your client program should use the reliable stream protocol (SOCK STREAM) and the Internet domain protocols (AF INET).

4 Part 3: HTTP 1.1

You are required to add simple HTTP/1.1 support to your web server, consisting of persistent connections and pipelining of client requests to your web browser. You will also need to add some heuristic to your web server to determine when it will close a" persistent" connection. That is, after the results of a single request are returned (e.g., index.html), the server should by default leave the connection open for some period of time, allowing the client to reuse that connection to make subsequent requests. This timeout needs to be configured in the server and ideally should be dynamic based on the number of other active connections the server is currently supporting. That is, if the server is idle, it can afford to leave the connection open for a relatively long period of time. If the server is busy, it may not

be able to afford to have an idle connection sitting around (consuming kernel/thread resources) for very long.

5 Bonus

5.1 Test your server with a real web browser

Test your server with a web browser of your choice.

5.2 Performance Evaluation

Explore the performance of your server when the number of clients increases or the requests to the server increases. You can observe the time delay or throughput with changing number of clients. Draw a chart to clarify the relation between time delay (for example) and number of requests to the server.

6 List of Useful Resources

- Online:" Beej's Guide to Network Programming: Using Internet Sockets"
- Use 'TCP\IP Sockets in C' book as a reference (check chapter 2 and the API reference).

7 Policy

1. You should develop your application in C/C++ programming language.
2. You are required to submit external documentation for your program and to comment your code thoroughly and clearly. Your documentation should describe the overall organization of your program, the major function and data structures.
3. You should work individually.