University of Dhaka

Department of Computer Science and Engineering

CSE 3111 – Computer Networking Laboratory Credits: 1.5 Batch: 27/3rd Year 1st Sem 2023

Instructors: Prof. Dr. Md. Abdur Razzaque (AR), Prof. Dr. Md. Mamun-Or-Rashid (MOR),
Dr. Muhammad Ibrahim (MIb) and Mr. Md. Redwan Ahmed Rizvee (RAR)

Lab Experiment # 3

Title of the Experiment: Implementing File transfer using Socket Programming and HTTP GET/POST requests

Objective:

The objective of this lab is to give hands-on experience with socket programming and HTTP file transfer. You will

- implement multithreaded chat from many clients to one server

- set up an HTTP server process with a few objects

- use GET and POST methods to upload and download objects in between HTTP clients and a server

Task 1: File Transfer via Socket Programming

1. Implement a simple file server that listens for incoming connections on a specified port. The server should be able to handle multiple clients simultaneously.

2. When a client connects to the server, the server should prompt the client for the name of the file they want to download.

3. The server should then locate the file on disk and send it to the client over the socket.

4. Implement a simple file client that can connect to the server and request a file. The client should save the file to disk once it has been received.

Task 2: File Transfer via HTTP

1. Implement a simple HTTP file server that listens for incoming connections on a specified port. The server should be able to handle multiple clients simultaneously.

2. When a client sends a GET request to the server with the path of the file they want to download, the server should locate the file on disk and send it to the client with the appropriate HTTP response headers. The client should save the file to disk once it has been received.

3. When a client sends a POST request to the server with a file, the server saves it.

Deliverables:

1. Source code zip for the file server and client implemented in Task 1.

2. Source code zip for the HTTP file server and client implemented in Task 2.

3. A report that explains the design and implementation of the file server and client in both Task 1 and Task 2. The report should also include screenshots of the file server and client in action for both tasks.

Grading Criteria:

1. Correctness and functionality of the file server and client in Task 1.

2. Correctness and functionality of the HTTP file server and client in Task 2.

3. Quality and completeness of the report.

Note: Students are encouraged to test their file server and client with different file types and sizes to ensure that they work correctly. Additionally, students should handle errors and exceptions gracefully in their code.

References :

1. Chat with multiple clients : https://www.geeksforgeeks.org/introducing-threads-socket-programming-java
2. File Transfer via socket: https://www.geeksforgeeks.org/transfer-the-file-client-socket-to-server-socket-in-java/
3. Sending GET/POST requests : https://www.digitalocean.com/community/tutorials/java-httpurlconnection-example-java-http-request-get-post
4. Additional Details on GET/POST : https://www.javatpoint.com/java-get-post

Background on HTTP Requests

HTTP

HTTP/2 is a new version of the HTTP protocol. The protocol had three versions prior to 2. They were 0.9, 1.0 and 1.1. The first one was only an experiment starting in 1991. The first real version was 1.0 released in 1996. This version was soon followed by the version 1.1 next year, 1997. The major difference between 1.0 and 1.1 was the `Host` header field that made it possible to operate several websites on one machine, one server, one IP address, and one port.

Both versions 1.0 and 1.1 are extremely simple. The client opens a TCP channel to the server and writes the request into it as a plain text. The request starts with a request line, it is followed by header lines, an empty line and the body of the request. The body may be binary. The response has the same structure except the first line is not a request specific line, but rather a response obviously.

HTTP/1.1 limitations

With HTTP/1.1 the browser downloads the resources (CSS, JS, Image, Video) in a web page one after the other in separate TCP channels. The number of the TCP channels the server or for that matter the client can handle is finite,

therefore the browsers limit themselves not to open more than four TCP channels. It means that while four content elements for the page is downloading the other elements wait in a queue to be downloaded. If we have some very slow downloading content it may choke the download of the whole web page.

HTTP/2

HTTP/2 uses a single TCP channel between the server and the client. When the content is downloaded from different servers the client will eventually open separate TCP channels to each server, but for the content pieces that come from the same server, HTTP/2 uses only one TCP channel. Using multiple TCP channels between a certain client and a single server does not speed up the communication, it was only a workaround in HTTP/1.1 to partially mitigate the choking effect of slow resources.

A request and a corresponding response do not use exclusively the TCP channel in HTTP/2. There are frames and the request and the response travels in these frames. If a content piece is created slowly and does not use the channel for some time then other resources can get frames and can travel in the same TCP channel. This is a significant boost of download speed in many cases. (Detail can be found here)

Usage of HTTP on the web :

Every HTTP server website has several web pages inside it. Each web page can be separately accessed by the clients getting connected to the web server. These accesses can be of many various forms and ways. Separate HTTP methods exist for each separate applications from the clients' side for the HTTP server. Two such type of operations can be requested by the client to the server using POST and GET methods.

GET : GET is used to request data from a specified resource.

POST : POST is used to send data to a server to create/update a resource. The data sent to the server with POST is stored in the request body of the HTTP request. It is also used when uploading a file or when submitting a completed web form.

As first, the connection is to be made with the server, where client needs to POST the contents. Then the materials to be posted are collected. After that, client sends information to the web page, that client wants the POST method for that connection application.

Finally, a response is sent from the server. With this response message, the server sends a certain response code and a response message. For POST method, this code is 201 and for GET method, this code is generally 200 for success. This information can be nicely utilized by the client for cross-checking. If the returned code matches the success code, then client can decide that the sent contents are correctly posted in the web page. Later, the client can retrieve that content with the help of GET HTTP method.