

CS2105 Computer Networking I
Laboratory Experiment 3

Socket Programming II

Objectives:

- (1) Understand how a simple Web server in Java is built to support HTTP.
- (2) Develop a multi-threaded (or concurrent) Web server.

This laboratory experiment is an extension to Laboratory Experiment 2 (Socket Programming I), and is based on **Section 2.9 – Building a Simple Web Server** of the textbook (p. 164 – 169).

(Total 100 marks)

Part A: Demonstrate socket programming for a connection-oriented/connectionless iterative/concurrent server (10 marks)

This part provides two sample programs, i.e., the client/server program pair, for a TCP implementation of the application developed for Part B of Laboratory Experiment 2. Do the following:

- (1) Start up the Ethereal software on PC1–“BenchNo”.
- (2) Run *sudo lab3* on Router–“BenchNo” to retrieve the sample server program called **TCPServer.java**.
- (3) Run *sudo lab3* on PC1–“BenchNo” to retrieve the sample client program called **TCPClient.java**.
- (4) Run the TCP server on Router–“BenchNo” and two TCP clients on PC1–“BenchNo”.
- (5) Observe the operations and check to see if the server supports connection-oriented iterative, or connectionless iterative, or connection-oriented concurrent, or connection-oriented iterative client-server communication. Explain your observations to your Lab Tutor.

Part B: Using threads for concurrency (10 marks)

In concurrent programming, a thread of execution is an abstraction of independent computation and a single process can contain one or more threads. Follow the steps given below to convert the iterative to concurrent client-server communication that the server’s program supports:

- Create a new thread class called **ClientHandler** that has the following statements:

```

class ClientHandler extends Thread
{
    private Socket client=null;
    public ClientHandler(Socket c)
    {
        this.client=c;
    }

    public void run()
    {
    }
}

```

- Move the statements used to handle the client connection after accepting it to the run() method of the new class.
- Call the start() method of the new class to start the thread in the main class:
new ClientHandler(client).start();

Repeat steps (1) to (5) and demonstrate your results.

Part C: Develop a multi-threaded Web server

(80 marks)

By the end of this part, you will have developed, in Java, a multi-threaded Web server that is capable of serving multiple requests in parallel. You are going to implement some functions of HTTP version 1.0, as defined in RFC 1945.

From Section 2.9, you should have understood how a simple Web server in java is built to support HTTP. To test this Web server, run **WebServer.java** on Router-“BenchNo”. Then use a browser running on PC1-“BenchNo” to request a file from the server:

`http://server.lab:6789/index.html` .

Demonstrate the results to your Lab Tutor.

Now, refer to Part B and develop a multi-threaded Web server to support HTTP. Recall that HTTP/1.0 creates a separate TCP connection for each request/response pair. A separate thread handles each of these connections. There will also be a main thread, in which the server listens for clients that want to establish connections. To simplify the programming task, we can develop the code in two stages. In the first stage, you can write a multi-threaded server that simply displays the contents of the HTTP request message that it receives. After this program is running properly, you can add the code required to generate an appropriate response. Also, when the server encounters an error, it should send a response message with an appropriate HTML source so that the error information is displayed in the browser window. Use a browser running on PC1-“BenchNo” to request files from the server:

`http://server.lab:6789/index.html` , and,

`http://server.lab:6789/index.jpg` .

Demonstrate the results to your Lab Tutor.

– END –