Distributed Systems
SOFE 4790U

Lab 1
September 19, 2024
CRN 43525

Jeremy Mark Tubongbanua	100849092

Table of Contents

Task 1

Task #1: Test your Internet speed (Bonus 5%)

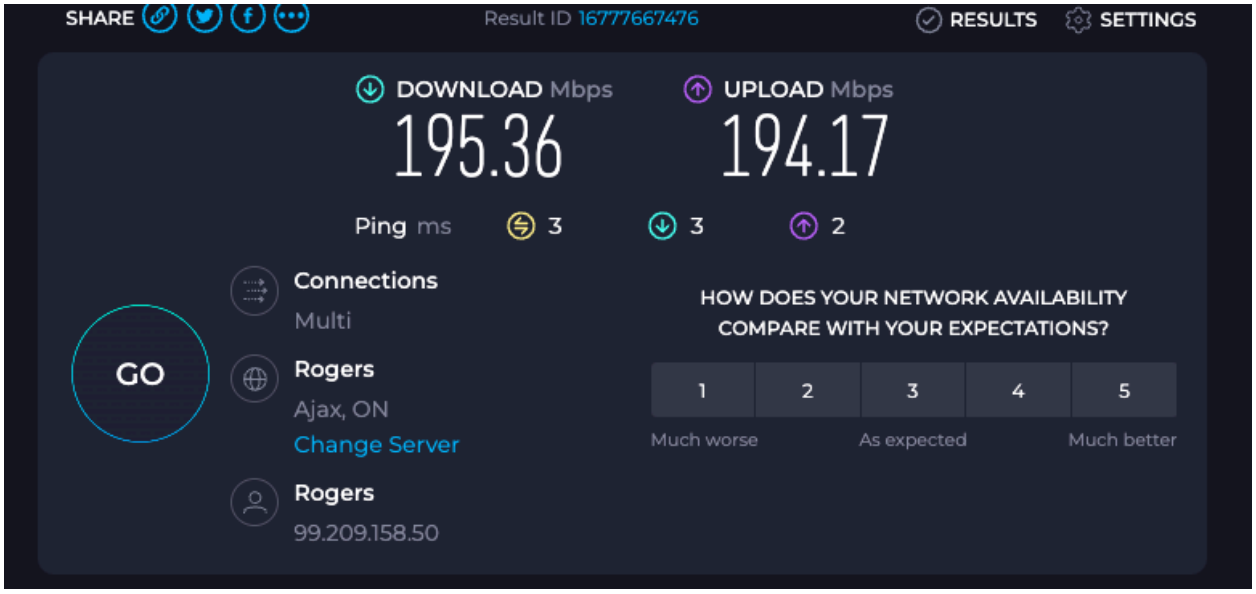
Check if your Internet service provider has a web service for testing your Internet speed (most service providers like Rogers, Bell, Telus, … do). If you cannot find it, try this: https://www.speedtest.net/.

Your task is to test your Internet speeds (2.4GHz, 5Ghz, and Ethernet) from your laptop, and document them in the report. Note: to test your Ethernet speed, attach your laptop to the ISP router. Answer the following questions in the report (use screenshots when needed):

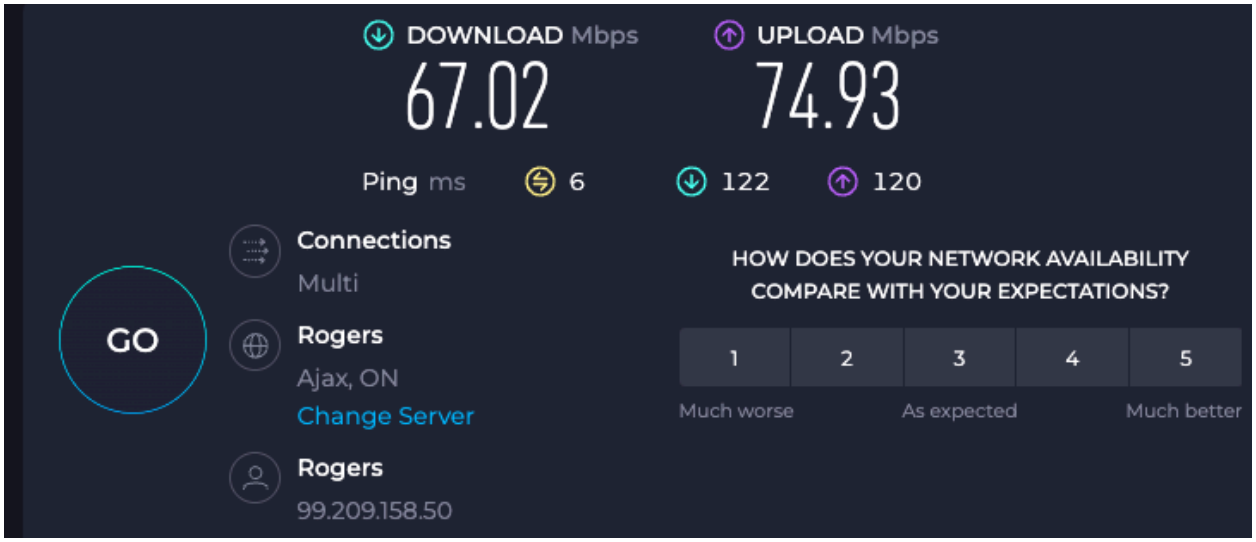
a) Do your Internet speeds match your ISP subscription?
b) How does the wi-fi speed compare to the Ethernet speed? If you're unable to test your Ethernet speed for any reason, then comment on the wi-fi speeds.
c) How does the speed test work? Please do not copy the detailed answers from online resources, just provide one simple paragraph in your own words.

You need to submit a report answering the previous questions before the deadline mentioned below in order the receive the bonus marks.

Ethernet speed



WiFi Speed



a) Do your internet speeds match with your ISP?

No they do not, it is advertised as 300 Mbps but I only get 195 Mbps via Ethernet.

b) How does your WiFi-speed compare with ethernet speed

My Wifi speed is 67 Mbps download and 75 Mbps upload while my Ethernet speed is 195 Mbps download and 194 Mbps upload. So they are very different and I would prefer ethernet due to higher speeds.

c) How does the speed test work?

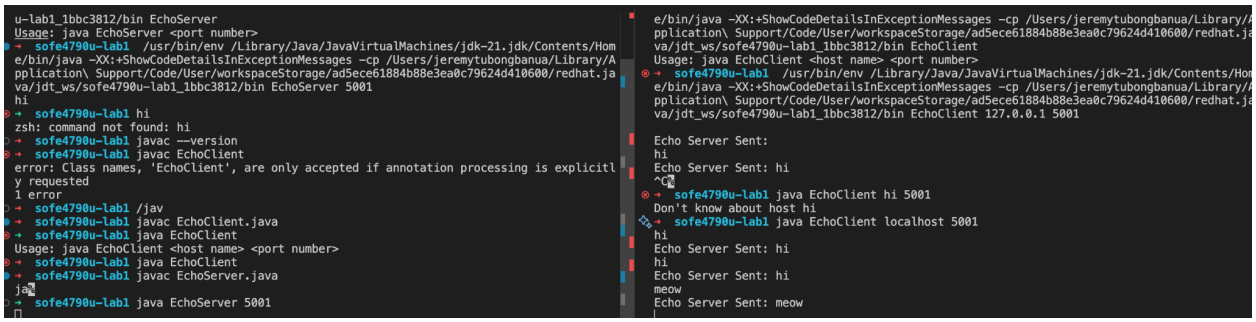
We contact speedtest.net's servers and test upload and download speeds through their servers. We send chunks of data and they measure transfer over durations.

Task 2

> **Task #2: Client IP address and port number (30%)**
> Download, compile and run the EchoServer/Client (code available on Canvas) (single-thread), and then modify the server code to print on the console the client's IP address and port number (Hint: check the java.net.Socket class for a method to return the client's IP address and port number).

Task 2.1 - Compile and Run (Single-threaded)

Here is a screenshot of myself getting it working without any code modifications yet.



Task 2.2 - Modify Server Code to Print Client IP Address And Port

To get the Client IP's address, we can simply add this line of code

```
System.out.println("Client IP: " + clientSocket.getInetAddress().getHostAddress());
System.out.println("Client Port: " + clientSocket.getPort());
```

This is what the output looks like

```
➜  sofe4790u-lab1 javac EchoServer.java && java EchoServer 5001
Client IP: 127.0.0.1
Client Port: 50831
```

This is because EchoClient is connect via localhost (I decided to run both the server and the client on the same machine). Then, the client connected via an ephemeral port, and it just so happened to be 50831 in this one instance.

Here is the full code of EchoClient.java as of this task.

```
import java.io.*;
import java.net.*;
```

```java
public class EchoServer {
    public static void main(String[] args) {

        if (args.length != 1) {
            System.err.println("Usage: java EchoServer <port number>");
            System.exit(1);
        }

        int portNumber = Integer.parseInt(args[0]);

        try {
            ServerSocket serverSocket = new
ServerSocket(Integer.parseInt(args[0]));
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client IP: " +
clientSocket.getInetAddress().getHostAddress());
            System.out.println("Client Port: " + clientSocket.getPort());
            PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(clientSocket.getInputStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                out.println(inputLine);
            }
        } catch (IOException e) {
            System.out.println("Exception caught when trying to listen on
port "
                    + portNumber + " or listening for a connection");
            System.out.println(e.getMessage());
        }
    }
}
```

Task 3

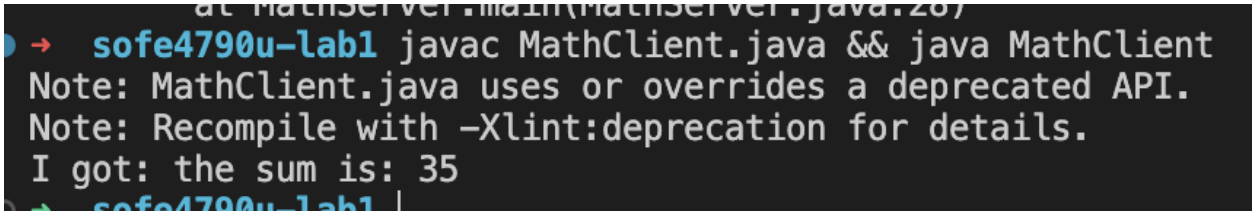
Task #3: Binomial Coefficient (30%)

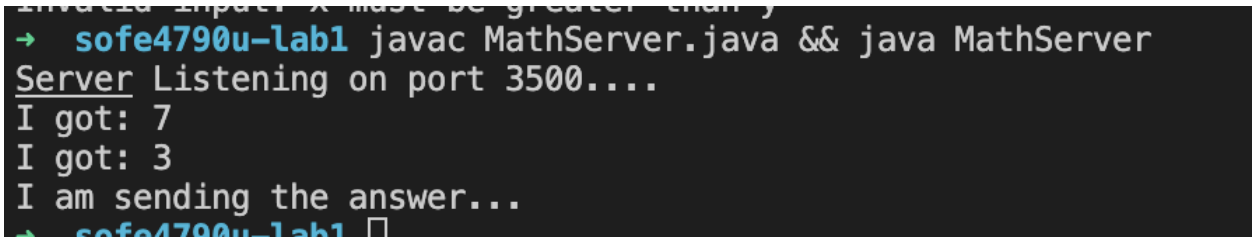
Modify the MathServer/Client application (code available on Canvas) so that when the server receives the two numbers (n, k), it finds the binomial coefficient (for simplicity always start with the small number). The server displays the coefficient value, and the client displays result received from the client

$_n C_k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$ where n > k

Example: if the client sends the numbers 7 and 3 then the server will calculate binomial coefficient $_7C_3$ = 35 (the server prints this number) and sends value 35 to the client to display. If the n <= k, then the server should send **-1** to the client.

When n > k





When n <= k

MathServer.java

```
/**
 * @author Qusay H. Mahmoud
 */

import java.io.*;
import java.net.*;
```

```
public class MathServer {

  ServerSocket hi;
  Socket client;
  DataInputStream br;
  DataOutputStream dos;

  public static int add(int a, int b) {
    return a + b;
  }

  public static int binomialCoefficient(int n, int k) {
    if (k == 0 || k == n) {
      return 1;
    }
    return binomialCoefficient(n - 1, k - 1) + binomialCoefficient(n
- 1, k);
  }

  public static void main(String argv[]) throws Exception {
    new MathServer();
  }

  public MathServer() throws Exception {
    hi = new ServerSocket(3500);
    System.out.println("Server Listening on port 3500....");
    client = hi.accept();
    br = new DataInputStream(client.getInputStream());
    dos = new DataOutputStream(client.getOutputStream());

    int x = br.readInt();
    System.out.println("I got: " + x);
    int y = br.readInt();
    System.out.println("I got: " + y);
    // int ans = add(x,y);
    if (x <= y) {
      System.out.println("Invalid input: x must be greater than y");
      dos.writeBytes("the sum is: " + -1 + "\n");
```

```
    } else {
      int ans = binomialCoefficient(x, y);
      System.out.println("I am sending the answer...");
      dos.writeBytes("the sum is: " + ans + "\n");
    }
  }
}
```

MathClient.java

```
/**
 * @author Qusay H. Mahmoud
 */

 import java.io.*;
 import java.net.*;

 public class MathClient {
    public static void main(String argv[]) throws Exception {
      Socket echo;
      DataInputStream br;
      DataOutputStream dos;

      echo = new Socket("localhost", 3500);
      br = new DataInputStream(echo.getInputStream());
      dos = new DataOutputStream(echo.getOutputStream());
      int x = 7, y = 3;
      dos.writeInt(x);
      dos.flush();
      dos.writeInt(y);
      dos.flush();
      String str = br.readLine();
      System.out.println("I got: "+str);
    }
 }
```

Task 4

Task #4: Multi-threaded Server (40%)

Revise the MathServer/Client application from Task#3 to accomplish the following:

1. The server should be able to process requests from multiple clients simultaneously (multi-threaded server). Test it with multiple clients as I have done in class.

   In your report, you would show this with screenshots of the server window displaying the clients' IP addresses and port numbers.

2. Next, revise the server to handle a fixed number of clients. If the number of connected clients reaches the maximum (you define it) and a new client tries to connect with the server, the server should send a "busy" message to the client and close the connection to that client.

Here is an example of me reaching the max threads reached (2)

Server

```
→  sofe4790u-lab1 javac MathClient4.java && java MathClient4
→  sofe4790u-lab1 javac MathClient4.java && java MathClient4
→  sofe4790u-lab1 javac MathServer4.java && java MathServer4
Server Listening on port 3500....
Staring thread... Now, there are 1 threads running
IP: 127.0.0.1
Port: 55396
Staring thread... Now, there are 2 threads running
IP: 127.0.0.1
Port: 55397
Max threads reached. Closing connection...
I got: 7
I got: 3
I am sending the answer...
Thread is closing... Now, there are 1 threads running
Staring thread... Now, there are 2 threads running
IP: 127.0.0.1
Port: 55400
I got: 7
I got: 3
I am sending the answer...
Thread is closing... Now, there are 1 threads running
I got: 7
I got: 3
I am sending the answer...
Thread is closing... Now, there are 0 threads running
```

The server outputs the IP and port of every socket client created.

Client 1

```
→  sofe4790u-lab1 javac MathClient4.java && java MathClient4
Note: MathClient4.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Connected to server...
Enter n:
7
Enter k:
3
Server says: the sum is: 35
→  sofe4790u-lab1
```

Client 2

```
→  sofe4790u-lab1 javac MathClient4.java && java MathClient4
Note: MathClient4.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Connected to server...
Enter n:
7
Enter k:
3
Server says: the sum is: 35
→  sofe4790u-lab1
```

Client 3

```
→  sofe4790u-lab1 javac MathClient4.java && java MathClient4
Note: MathClient4.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Server is busy... Try again later.
→  sofe4790u-lab1 javac MathClient4.java && java MathClient4
Note: MathClient4.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Connected to server...
Enter n:
7
Enter k:
3
Server says: the sum is: 35
→  sofe4790u-lab1
```

Notice when it says “server is busy”

This is because it tried to connect to the server but there were already 2 threads running.

MathServer.java

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.*;

public class MathServer4 {

  private static int threadsRunning;
  private final static int MAX_THREADS = 2;

  public static class MathThread implements Runnable {
    Socket client;

    public MathThread(Socket client) {
      this.client = client;
    }

    @Override
    public void run() {
      try {
        System.out.println("IP: " +
client.getInetAddress().getHostAddress());
        System.out.println("Port: " + client.getPort());
        DataInputStream br;
        DataOutputStream dos;
        br = new DataInputStream(client.getInputStream());
        dos = new DataOutputStream(client.getOutputStream());
        int x = br.readInt();
        System.out.println("I got: " + x);
        int y = br.readInt();
        System.out.println("I got: " + y);
        // int ans = add(x,y);
        if (x <= y) {
```

```
          System.out.println("Invalid input: x must be greater than
y");
          dos.writeBytes("the sum is: " + -1 + "\n");
        } else {
          int ans = binomialCoefficient(x, y);
          System.out.println("I am sending the answer...");
          dos.writeBytes("the sum is: " + ans + "\n");
        }
      } catch (Exception e) {
        System.out.println("Error: " + e);
      } finally {
        try {
          client.close();
          threadsRunning--;
          System.out.println("Thread is closing... Now, there are " +
threadsRunning + " threads running");
        } catch (Exception e) {
          System.out.println("Error closing client socket");
        }
      }
    }
  }

  public static int binomialCoefficient(int n, int k) {
    if (k == 0 || k == n) {
      return 1;
    }
    return binomialCoefficient(n - 1, k - 1) + binomialCoefficient(n
- 1, k);
  }

  public static void main(String argv[]) throws Exception {
    ServerSocket hi = new ServerSocket(3500);
    System.out.println("Server Listening on port 3500....");

    while (threadsRunning <= MAX_THREADS) {
      Socket client = hi.accept();
      threadsRunning++;
      if(threadsRunning > MAX_THREADS) {
```

```
        System.out.println("Max threads reached. Closing
connection...");
        DataOutputStream dos = new
DataOutputStream(client.getOutputStream());
        dos.writeBytes("Server is busy\n");
        client.close();
        threadsRunning--;
        continue;
      } else {
        DataOutputStream dos = new
DataOutputStream(client.getOutputStream());
        dos.writeBytes("Server is ready\n");
      }
      System.out.println("Staring thread... Now, there are " +
threadsRunning + " threads running");
      new Thread(new MathThread(client)).start();
    }

    hi.close();
  }

}
```

MathClient.java

```
 import java.io.*;
 import java.net.*;

 public class MathClient4 {
    public static void main(String argv[]) throws Exception {
      DataInputStream br;
      DataOutputStream dos;

      Socket echo = new Socket("localhost", 3500);
      br = new DataInputStream(echo.getInputStream());
      // check if received "Server is busy"
      if (br.readLine().equalsIgnoreCase("Server is busy")) {
```

```
        System.out.println("Server is busy... Try again later.");
        return;
      } else {
        System.out.println("Connected to server...");
      }
      dos = new DataOutputStream(echo.getOutputStream());
      BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
      int n = -1;
      int k = -1;
      System.out.println("Enter n: ");
      n = Integer.parseInt(in.readLine());
      System.out.println("Enter k: ");
      k = Integer.parseInt(in.readLine());
      dos.writeInt(n);
      dos.writeInt(k);
      String response = br.readLine();
      System.out.println("Server says: " + response);
      echo.close();
    }
 }
```