# COMP713 Distributed and Mobile Systems

**Laboratory 1: Client-Server Communication**

## Introduction

There are two main aims of this laboratory. The first is to learn about programming client-server communications using Transport Control Protocol (TCP) and User Datagram Protocol (UDP), protocols at the network layer. The second is to learn about programming communication at the application layer. More specifically, web based communication using the Hypertext Transfer Protocol (HTTP) is explored.

## Resources

The software used will be Java SE, Java EE, and Netbeans. These programs should already be available on the computers in the lab. If you want to use your own computer, you will need to have these programs installed yourself. Program codes for each part of the laboratory can be downloaded from the course Blackboard site on the same page as this document.

Students should refer to Sections 1.1 to 1.3 of the course manual.

## Pre-Lab Questions

1. Which are the two most commonly used transport layer protocols?
2. What are the main differences between these two protocols?
3. List two application layer protocols that make use of each of these two transport layer protocols.

## Part 1: Animation Sender and Receiver

This example can be found on pages 14-19 of the course manual. The Java code for the animation sender and receiver can be found in the files `AnimationSender.java` and `AnimationReceiver.java` respectively in `Animation.zip`. This program makes use of UDP for communication between server and client. There are also two associated GIF files.

To run an example code, download and unzip `Animation.zip`. Open the Netbeans IDE and start a New Project. Select "Java Project with Existing Sources". Then give the project a suitable name and add the folder with the respective source code to "Source Package Folders" in the next window. Open the source file and run it.

You must start the Animation Receiver program before starting the Animation Sender. The GUI window of the Receiver will show the animation.

**Exercise 1:** UDP does not guarantee that the received datagram are in order. It is up to the application program to deal with out of order datagrams. Enhance the classes `AnimationSender` and `AnimationReceiver` so that outdated datagrams are discarded.

## Part 2: Echo Server and Client

This example can be found on pages 8-12 of the course manual. The Java code for the echo server and client can be found in the files `EchoServer.java` and `EchoClient.java` respectively in `Echo.zip`. This program makes use of TCP for communication between server and clients.

You must start the Echo Server program before starting the Echo Client.

To run an example code, download and unzip `Echo.zip`. Open the Netbeans IDE and start a New Project. Select "Java Project with Existing Sources". Then give the project a suitable name and add the folder with the respective source code to "Source Package Folders" in the next window. Open the source file and run it.

Enter some text into the Echo Client and the same message should be returned.

**Exercise 2:** Try to run more than one client – (i) on the same machine; and (ii) on different machines.

**Question 1:** What happens when there are more than one client connected to this server? How does the server know which client to echo the message back to?

## Part 3: Echo Web Server and Web Client

The echo service is now implemented in a web server. The `EchoWebServer` program code can be found on pages 22-24 of the course manual and in the file `EchoWebServer.java` in `WebEcho.zip`. Start running the server. This server can be tested with a Web browser such as Firefox using the URL http://localhost:8080/. Check that you get a similar response to that shown on page 21 of the course manual.

In this case, the clients are HTML pages. Three HTML forms are included: `GetForm.html`, `PostForm.html`, and `PostMultipleForm.html`.

**Exercise 3:** Open GetForm.html using a Web browser. Enter a name and press the Submit button. Examine the response from the server. Compare the responses to the other two Web pages.

**Question 2:** How many connections are made? You can tell from the `EchoWebServer` output. Explain why there are this number of connections made.

## Part 4: Echo Web Servlet with Session Tracking

The `EchoWebServer` above runs as a user application instead of a web application. We shall now examine a Java Servlet that performs session tracking. A Java Servlet is a Java program that extends the capabilities of a Web server to implement various responses to client requests. We shall deploy this Web application on the GlassFish server that has been installed on Netbeans.

To start the GlassFish server in Netbeans:

- Select Window->Services from the Netbeans menu bar at the top.
- In the "Services" panel on the top left, expand the "Servers" tab. You should see GlassFIsh Server as one of the items under it. Right-click on it and select "Start".
- You should see some notification messages on the GlassFish tab of the "Output" panel.

Create a Web application by:

- Start a new project on Netbeans. Choose the "Java Web" category and a "Web Application". Provide a project name as usual (e.g. HelloWeb).
- Then select GlassFIsh as the Server and Select Java EE 7 Web as the Java EE version.
- Note that the path where we can access this web application is "localhost:8080/HelloWeb" (replace HelloWeb by the project name you have chosen above).
- You don't need to select any web framework for this project.

After successfully created the project, you should see your project in the "Projects" tab in the top left panel. Under this tab there are four sections:

1. Web Pages – these contain the web pages through which a user interacts with the web application.
2. Source Packages – these contain the functional classes of the web application.
3. Libraries – the JDK and GlassFish (in our case).
4. Configuration Files – configure the web application on startup.

Create a new Servlet source file – expand "Source Packages", right-click on "<default package>" and select "New->Servlet". Give your servlet class a name (we shall use "MyServlet" here).

In "Configure Servlet Deployment":

- Check the "Add information to deployment descriptor (web.xml)" box.
- Change the Servlet name to "SessionTest" and the URL pattern to "/SessionTest". This means that the servlet's functionality can be accessed at "localhost:8080/HelloWeb/SessionTest".

There is now a Java servlet program named `MyServlet.java`. Use the code found in the file `RequestCounter.java` (found in `WebCounter.zip`) to change the code in `Myservlet.java`.

**Question 3:** How many functions are there in the `Myservlet` class?

Replace the code in the "index.html" file under the "Web Pages" of HelloWeb by the one provided for this lab (found in `WebCounter.zip`).

Now we are ready to test the servlet. Start the GlassFish server by:

- Go to the "Services" tab in the top left panel.
- Expand "Servers".
- Right-click on "GlassFish Server" and select "Start".

You should now get notifications on the GlassFish server tab of the "Output" panel.

Now Run the `HelloWeb` project. A web browser will open http://localhost:8080/HelloWeb. Enter a name and click "Submit".

**Question 4:** What is the response displayed on the web browser?

**Exercise 4:** Set the URL on the web browser to http://localhost:8080/HelloWeb again. Enter another name and click "Submit". Notice the session number displayed. Repeat the above several more times to confirm that the session number is correct. Then open another browser and go to that URL. Check that the servlet detects a separate session and provides a different session number for this instance.

## Part 5: Number Guessing Game

The files in `guessNum.zip` implements a number guessing game. It contains a guess server in `GuessServer.java` and a guess client in `GuessClient.java`. The user guesses a number between 1 and 99 that has been randomly selected by the server at the start of the game. The game ends when the client guesses correctly. Try it by running the guess server first before starting a guess client.

**Exercise 5:** Convert the server into a Web Application using servlet with session tracking. The game can then be played in a web browser.

**NOTE:** Laboratory exercises do not need to be submitted for marking. They do not carry any marks. However, they are an important part of learning for this paper and doing them could be useful in completing the assignments.

**********