CO324 Lab 01

Sockets and Error handling

In this lab we will look at the way of creating a simple UDP client and server in order to learn basics about network sockets, exception handling and timeouts of network programming.

Sockets

A **network socket** is an endpoint of an inter-process communication flow across a computer network. Every socket needs to have an address. A **socket address** consists of the IP address and the port number. Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

The "`java.net`" package provides classes for implementing network applications.

A comprehensive source on java.net package can be found at :

http://bioinfo2.ugr.es/OReillyReferenceLibrary/java/fclass/ch15_js.htm

Read the sample code given with the lab sheet. There are two programs, a server and a client. These two run on two computers (hosts) that are connected by a network and communicate with each other. Since we do not have two computers to run these we would run both on the same computer and use the internal connection to communicate. Therefore, to test the given code, compile them and run the server first. When running the server, we should give a port as a command line argument.

```
Java UDPServer 20000
```

Open another terminal and run the client while the server is still running. The client needs the ip address of the server and the port number of the server to connect to the server. Since we are running the server and the client in the same computer, the ip address of the server is 127.0.0.1

```
Java UDPClient 127.0.0.1 20000
```

If you want test this using two machines use following commands.

```
Server Machine: java UDPServer <PORT>
Client Machine: java UDPClient <Server IP> <Server PORT>
```

Exercise 1: Create a new socket object in the client code. (Hint: Check the socket object created in the server code. It is not a must to give the port number to the client socket)

Datagrams

With the **User Datagram Protocol** (**UDP**) computer applications can send messages, sometimes known as ***datagrams***, to other hosts on an Internet Protocol (IP) network. The packet must contain the host address and the port of the destination.

In the server code it has created an empty datagram packet using the DatagramPacket constructor in order to receive data from the client.

Exercise 2: Construct a datagram packet for the client. Note that the client needs to send data (a message) to a specific host through a specific port. Therefore you must use `DatagramPacket(byte[] data, int datalength, InetAddress addr, int port)` constructor.

Error handling

Using a combination of the try and catch blocks is a method to catch an exception. A `try/catch` block is placed around the code that might generate an exception. Code within a `try/catch` block is referred to as protected code.

A fatal error or a fatal exception is a condition under which the program cannot continue executing anymore. For an example, if the server cannot initialize a socket, the server cannot continue its task and therefore should be terminated. However, if a connected client stops responding or responses with an unrecognized message, the server could take an action, such as closing the connection or sending an error message and continue normally. This kind of error is not a fatal error and is considered non-fatal.

Exercise 3: Consider the server code. Which of these are recoverable? And which of them are non-recoverable?

Exercise 4: The server receives one packet and sends one packet at each iteration of the `'for'` loop. If the program throws an exception at receiving a packet, what will happen to the `'for'` loop?

If this error might be temporary and could affect only one or few packets received, do you think that this is the suitable behavior for the server?

If not, modify the server code such that the server deals with such problems.

The finally keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred. Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.

Exercise 5: What is the disadvantage of placing the `socket.close()` function at the end of the try block in the client code? Use the finally keyword to do necessary cleanups for the client code.