UDP & TCP

TUTORIAL 2

UDP Programming

- Connection less protocol which means that one program can send a load of packets to another and that would be the end of the relationship
- There is no guarantee of delivery, ordering, or duplicate protection of data
- UDP is faster because error recovery is not attempted.
- UDP is suitable for applications that need fast, efficient transmission, such as games.

UDP Programming

From class notes:

sender program

create a datagram socket and
bind it to any local port;
place data in a byte array;
create a datagram packet, specifying
the data array and the receiver's
address;
invoke the send method of the
socket with a reference to the
datagram packet;

receiver program

create a datagram socket and
bind it to a specific local port;
create a byte array for receiving the data;
create a datagram packet, specifying
the data array;
invoke the receive method of the
socket with a reference to the
datagram packet;

UDP Programming

So we need to create 2 classes:

- A client class (sender)
- A server class (receiver)

UDP Programming

- Create a new Project named UDPProgramming
- In your src default package, create 2 classes:
  - Client
  - Server
- Check the main method for both

UDP Programming - Client

1. Create a Datagram socket and bind it to any local port
2. Place data in byte array
3. Create a datagram packet and specify data array and receiver address
4. Invoke the send method with a reference to the packet

UDPClient.java

```java
import java.net.*;
import java.io.*;
public class UDPClient {

    public static void main(String args[]){
        //args give message contents and destination hostname
        DatagramSocket aSocket = null;
        try{
            aSocket = new DatagramSocket();
            byte [] m = "hello".getBytes();
            InetAddress aHost = InetAddress.getByName("localhost");
            int serverPort = 6789;

            DatagramPacket request = new DatagramPacket(m, "hello".length(), aHost, serverPort);
            aSocket.send(request);

            byte[] buffer = new byte[1000];
            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
            aSocket.receive(reply);
            System.out.println("Reply: " + new String (reply.getData()));

        }
        catch(SocketException e){
            System.out.println("Socket: " + e.getMessage());
        }
        catch(IOException e){
            System.out.println("IO: " + e.getMessage());
        }
        finally {
            if(aSocket != null) aSocket.close();
        }
    }
}
```

UDP Programming - Server

1. Create a datagram socket and bind it to a port
2. Create a byte array to receive the data
3. Create a datagram packet and specify the data array
4. Invoke the receive method of the socket with a reference the datagram packet

UDPServer.java

```java
import java.net.*;
import java.io.*;
public class UDPServer {

    public static void main(String[] args) {
        DatagramSocket aSocket = null;
        try{
            //create a socket at agreed port
            aSocket = new DatagramSocket(6789);
            byte[] buffer = new byte[1000];

            while(true){
                DatagramPacket request = new DatagramPacket(buffer, buffer.length);
                aSocket.receive(request);
                DatagramPacket reply = new DatagramPacket(request.getData(), request.getLength(),
                        request.getAddress(),request.getPort());
                aSocket.send(reply);
            }
        }
        catch(SocketException e){
            System.out.println("Socket: " + e.getMessage());
        }
        catch(IOException e){
            System.out.println("IO: " + e.getMessage());
        }
        finally {
            if(aSocket != null) aSocket.close();
        }

    }

}
```

UDP Programming - Run

How to make it work?

1. Run the UDPServer
2. Run the UDPClient

UDP Programming - Output

Problems @ Javadoc Declaration Console

```
<terminated> UDPClient [Java Application] /Library/Java/JavaVirt
Reply: hello
```

TCP Programming

- Connection oriented protocol
- It provides reliable, ordered, and error-checked delivery of data
- TCP is suited for applications that require high reliability, and transmission time is relatively less critical.
- The speed for TCP is slower than UDP.
- WWW, email, remote administration and file transfer rely on TCP

TCP Programming

Write a TCP client and TCP server.

The client sends a message to the server, the server reverses that messages and sends it back.

Create a class TCPServer that accepts input string from client and reverses it.

Create a class TCPClient that gets values from the user.

It also sends requests to server to reverse the string.

TCPServer.java

```java
import java.net.*;
import java.io.*;
public class TCPServer {

    public static void main(String[] args) throws IOException{
        //Initialization
        String server_inputMsg = "";
        String server_reverseMsg = "";

        //Create a socket at agreed port(5000)
        ServerSocket serverSocket = new ServerSocket(5000);

        while(true){
            //Establish the connection between the client and the server
            Socket connectionSocket = serverSocket.accept();

            //Get InputStream at server to get values from client
            BufferedReader inFromClient  =
                    new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));

            //Get OutputStream at server to send values to client
            DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());

            //Get the input message from client and then print
            server_inputMsg = inFromClient.readLine();
            System.out.println("Received: " + server_inputMsg);

            //Reverse the string
            server_reverseMsg = new StringBuffer(server_inputMsg).reverse().toString() + "\n";

            //Send the result to the client
            outToClient.writeBytes(server_reverseMsg);
        }
    }
}
```

TCPClient.java

```java
import java.net.*;
public class TCPClient {

    public static void main(String[] args) throws Exception{
        //Initialization
        String client_inputMsg = "";
        String client_reverseMsg = "";

        //Create a socket at agreed port
        Socket clientSocket = new Socket("localhost", 5000);

        //Get OutputStream at client to send values to server
        DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());

        //Get InputStream at client to get values from server
        BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

        //Get the input from the user
        BufferedReader inFromUser = new BufferedReader( new InputStreamReader(System.in));
        client_inputMsg = inFromUser.readLine();

        //Send the message received from user to server to be reversed
        outToServer.writeBytes(client_inputMsg + "\n");

        client_reverseMsg = inFromServer.readLine();
        System.out.println("FROM SERVER: " + client_reverseMsg);

        //close the socket for good measure
        clientSocket.close();
    }
}
```

TCP Programming

- To get Output:
  - Run the TCPServer first
  - Runt the TCPClient next
  - Enter a string at the console of TCPClient

At TCPClient

```
Problems  @ Javadoc  Declaration  Console
<terminated> TCPClient [Java Application] /Library/Java/JavaVi
Hello World!
FROM SERVER: !dlroW olleH
```

At TCPServer

```
Problems  @ Javadoc  Declaration  Console
TCPServer [Java Application] /Library/Java/JavaVirtualMachines/
Received: Hello World!
```

References

1. https://www.tutorialspoint.com/java/java_networking.htm

2. https://docs.oracle.com/javase/tutorial/networking/overview/networking.html

Lecture Note: Network and Process Communication, Dr. Jayakumar
http://www.cs.rutgers.edu/~pxk/417/notes/sockets/udp.html
Distributed Systems Concepts and Design, Fifth Edition, ISBN: 0-13-214301-1