Institute of Information & Communication Technology
University of Sindh, Jamshoro

Distributed Computing (Lab. 3)
Lab Objective # 3: Communication using the UDP Protocol

Background

UDP: User Datagram Protocol (UDP) is said to be a connection-less protocol. Each datagram packet is sent as an isolated transmission whenever necessary. This is so, because, communication governed by UDP has not dedicated point-to-point link between the communicating ends (the client and the server), unlike TCP/IP, where once a link has been created, it remains lives for the duration of the dialogue. Moreover, UDP lacks the mechanism of acknowledgment. This makes UDP less reliable on the one hand, but fast on the other, when compared with the acknowledgment-based TCP protocol.

UDP and Java: Java allows the writing of server and client applications that are based on the UDP protocol. In Java, an object of `DatagramSocket` represents a socket that is able to send and receive packets (in the form of `DatagramPacket` objects) using the UDP protocol.

A - Creating the Server Application

The process of writing a server application involves nine major steps:

1. Create a `DatagramSocket` object
2. Create a buffer (a `byte[]` array, for example) for the incoming datagrams
3. Create a `DatagramPacket` object for incoming datagrams
4. Accept an incoming datagram
5. Obtain the sender's address and port from the incoming datagram
6. Retrieve the data from the buffer
7. Create the response datagram as a `DatagramPacket` object
8. Send the response datagram
9. Close the `DatagramSocket` created in the first step

Code for an ECHO SERVER using the above steps:

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

public class UDPEchoServer {
      private static final int PORT = 1234;
      private static DatagramSocket datagramSocket;
      private static DatagramPacket inPacket, outPacket;
      private static byte[] buffer;
      public static void main(String[] args) {
            System.out.println("Opening port...\n");
            try {
                  datagramSocket =
                  new DatagramSocket(PORT);                         //Step 1.
```

Distributed Computing (Lab. 3)
Lab Objective # 3: Communication using the UDP Protocol

<u>*Code for Echo Server continued...*</u>

```
            }
            catch(SocketException sockEx) {
                  System.out.println("Unable to attach to port!");
                  System.exit(1);
            }
            handleClient();
      }

      private static void handleClient() {
            try {
                  String messageIn,messageOut;
                  int numMessages = 0;
                  do {
                        buffer = new byte[256];                   //Step 2.
                        inPacket =
                              new DatagramPacket(
                        buffer, buffer.length);                   //Step 3.
                        datagramSocket.receive(inPacket);         //Step 4.
                        InetAddress clientAddress =
                              inPacket.getAddress();              //Step 5.
                        int clientPort =
                        inPacket.getPort();                       //Step 5.
                        messageIn =
                              new String(inPacket.getData(),
                                    0,inPacket.getLength());      //Step 6.
                        System.out.println("Message received.");
                        numMessages++;
                        messageOut = "Message " + numMessages
                              + ": " + messageIn;
                        outPacket =
                              new DatagramPacket(messageOut.getBytes(),
                              messageOut.length(),clientAddress,
                              clientPort);                        //Step 7.
                        datagramSocket.send(outPacket);           //Step 8.
                  }while (true);
            }
            catch(IOException ioEx) {
                  ioEx.printStackTrace();
            }
            finally { //If exception thrown, close connection.
                  System.out.println(
                        "\n* Closing connection... *");
                  datagramSocket.close();                         //Step 9.
            }
      }
}
```

Institute of Information & Communication Technology
University of Sindh, Jamshoro

Distributed Computing (Lab. 3)
Lab Objective # 3: Communication using the UDP Protocol

B - Creating the Client Application

Writing the client application involves the following steps:

1. Create a `DatagramSocket` object
2. Create the outgoing datagram as an object of `DatagramPacket`
3. Send the outgoing datagram
4. Create a buffer (as a `byte[]` array) for incoming datagram
5. Create a `DatagramPacket` object for incoming datagrams
6. Accept a incoming datagram
7. Retrieve the data from the buffer
8. Close the `DatagramSocket`

Here is the CLIENT program that communication with our ECHO SERVER.

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

public class UDPEchoClient {
      private static InetAddress host;
      private static final int PORT = 1234;
      private static DatagramSocket datagramSocket;
      private static DatagramPacket inPacket, outPacket;
      private static byte[] buffer;

      public static void main(String[] args) {
            try {
                  host = InetAddress.getLocalHost();
            }
            catch(UnknownHostException uhEx)
            {
                  System.out.println("Host ID not found!");
                  System.exit(1);
            }

                  accessServer();
      }
```

Distributed Computing (Lab. 3)
Lab Objective # 3: Communication using the UDP Protocol

Code for the CLIENT continued…

```
      private static void accessServer() {
            try {
                  datagramSocket = new DatagramSocket();           //Step 1...
                  //Set up stream for keyboard entry...
                  Scanner userEntry = new Scanner(System.in);

                  BufferedReader userEntry = new BufferedReader(
                  New InputStreamReader (System.in));

                  String message="", response="";
                  do {
                        System.out.print("Enter message: ");
                        message = userEntry.nextLine();
                        outPacket = new DatagramPacket(
                              message.getBytes(), message.length(),
                              host,PORT);                           //Step 2.
                                                                    //Step 3...
                        datagramSocket.send(outPacket);
                        buffer = new byte[256];                     //Step 4.
                        inPacket = new DatagramPacket(
                                    buffer, buffer.length);         //Step 5.
                                                                    //Step 6...
                        datagramSocket.receive(inPacket);
                        response = new String(inPacket.getData(),
                                    0, inPacket.getLength());       //Step 7.
                                    System.out.println(
                                          "\nSERVER> "+response);
                  } while (!message.equals("***CLOSE***"));
            }
            catch (IOException ioEx) { ioEx.printStackTrace(); }
            finally {
                  System.out.println("\n* Closing connection... *");
                  datagramSocket.close();                           //Step 8.
            }
      }
}
```