UNIT 10
UDP

LH - 5HRS

PRESENTED BY: **ER. SHARAT MAHARJAN**

NETWORK PRGORAMMING

PRIME COLLEGE, NAYABAZAAR

CONTENTS (LH - 5HRS)

10.1 UDP Protocol

- In computer networking, the UDP stands for User Datagram Protocol and is part of TCP/IP protocol.
- The UDP protocol allows the computer applications to send the messages in the form of datagram from one machine to another machine over the Internet Protocol network.
- The UDP is an alternative communication protocol to the TCP protocol.
- The UDP works by encapsulating the data into the packet and providing its own header information to the packet. Then, this UDP packet is encapsulated to the IP packet and sent off to its destination.
- UDP is a connectionless protocol.

10.2 UDP Clients

```
import java.net.*;
public class Client {
 public static void main(String[] args) {
  try {
   DatagramSocket socket = new DatagramSocket();
   //Sending Part
   byte[] buffer = new byte[1500];
   String message = "hello server";
   buffer = message.getBytes();
   DatagramPacket packet = new DatagramPacket(buffer, buffer.length,   InetAddress.getByName("127.0.0.1"),4567);
   socket.send(packet);
   System.out.println("Message sent: "+message);
   //Receiving Part
   packet = new DatagramPacket(buffer,buffer.length);
   socket.receive(packet);
   message = new String(buffer).trim();
   System.out.println("Message received: "+message);
   } catch (Exception e) {
    e.printStackTrace();}}}
```

10.3 UDP Servers

```
import java.net.*;
public class Server {
 public static void main(String[] args) {
  try {
   DatagramSocket socket = new DatagramSocket(4567);
   System.out.println("Server is running");
   //Receiving part
   byte[] buffer = new byte[1500];
   DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
   socket.receive(packet);//retrieving client message
   String message = new String(buffer).trim();
   System.out.println("Message received: "+message);
   //Sending part
   InetAddress client_address = packet.getAddress();
   int client_port = packet.getPort();
   message = "hello client";
   buffer = message.getBytes();
   packet = new DatagramPacket(buffer, buffer.length,client_address,client_port);
   socket.send(packet);
   System.out.println("Message sent: "+message);
   } catch (Exception e) {
   e.printStackTrace();}}}
```

10.4 The Datagram Packet Class

- It represents a data packet intended for transmission using the User Datagram Protocol.
- Packets are containers for a small sequence of bytes and include addressing information such as an IP address and a port.
- These two constructors create new DatagramPacket objects for **receiving** data from the network:

✓**public DatagramPacket(byte[] buffer, int length)**

✓**public DatagramPacket(byte[] buffer, int offset, int length)**

- When a socket receives a datagram, it stores the datagram's data part in buffer beginning at buffer[0] and continuing until the packet is completely stored or until length bytes have been written into the buffer .
- If the second constructor is used, storage begins at buffer[offset] instead. Otherwise , these two constructors are identical.

- These four constructors create new DatagramPacket objects for **sending** data across the network:

✓**public DatagramPacket(byte[] data, int length, InetAddress destination, int port)**

✓**public DatagramPacket(byte[] data, int offset, int length, InetAddress destination, int port)**

✓**public DatagramPacket(byte[] data, int length, SocketAddress destination, int port)**

✓**public DatagramPacket(byte[] data, int offset, int length, SocketAddress destination, int port)**

Methods:

- ✓**InetAddress getAddress():** returns IP address from which a DatagramPacket was sent.
- ✓**byte[] getData():** returns the contents of packet as an array of bytes.
- ✓**int getLength():** returns length of data stored in packet.
- ✓**int getPort():** returns the port number from which a packet was sent.

- 4 corresponding setter methods for above four getter methods.

10.5 The Datagram Socket Class

- To send or receive a DatagramPacket , you must open a datagram **socket**. In Java, a datagram socket is created and accessed through the **DatagramSocket** class:

public class DatagramSocket extends Object

- Creating DatagramSocket objects:

✓**public DatagramSocket( ) throws SocketException**

- This constructor creates a socket that is bound to an anonymous port. For example:

```
 try {DatagramSocket client = new DatagramSocket( );
        //send packets... } catch (SocketException ex) {
           System.err.println(ex); }
```

✓**public DatagramSocket(int port) throws SocketException**

This constructor creates a socket that listens for incoming datagrams on a particular port, specified by the port argument.

Use this constructor to write a server that listens on a well-known port; if servers listened on anonymous ports, clients would not be able to contact them.

A SocketException is thrown if the socket can't be created.

Sending and Receiving Datagrams

- The primary task of the DatagramSocket class is to send and receive UDP datagrams.
- One socket can both send and receive. Indeed, it can send and receive to and from multiple hosts at the same time.

public void send(DatagramPacket dp) throws IOException

- Once a DatagramPacket is created and a DatagramSocket is constructed , send the packet by passing it to the socket's send() method. For example, if socket is a DatagramSocket object and packet is a DatagramPacket object, send packet using socket like this:

```
socket.send(packet);
```

10.6 Socket Options

1. **SO_TIMEOUT**

- It is the amount of time, in **milliseconds** , that **receive**( ) waits for an incoming datagram before throwing an IOException.
- If SO_TIMEOUT is 0, receive( ) never times out.
- This value can be **changed** with the **setSoTimeout**( ) method and **inspected** with the **getSoTimeout**( ) method.

2. **SO_RCVBUF**

- It determines the **size of the buffer** used for network I/O.
- Sufficiently **large receive buffers** are even more important for UDP than for TCP, since a UDP datagram that arrives when the buffer is full will be lost.
- The **setReceiveBufferSize**( ) method suggests a number of bytes to use for buffering input from this socket.
- You may wish to check the actual size of the receive buffer with **getReceiveBufferSize**( ) after setting it.

3. <u>SO_SNDBUF</u>

- DatagramSocket has methods to get and set the suggested send buffer size used for network output.
- The **setSendBufferSize**( ) method suggests a number of bytes to use for buffering output on this socket.
- You'll want to check the result of **setSendBufferSize**( ) by immediately following it with a call to **getSendBufferSize**( ) to find out the real the buffer size.

4. <u>SO_REUSEADDR Option:</u>

- It enables reuse address for a socket.
- It has two methods setReuseAddress() and getReuseAddress().

5. **SO_BROADCAST:**

- This socket option enables or disables the ability of the process to send broadcast messages.
- It has two methods: setBroadcast() and getBroadcast().

6. **IP_TOS:**

- Sets the type-of-service or traffic class(maximum throughput, minimum delay, etc) field in the IP header for UDP socket.
- Two methods: setTrafficClass() and getTrafficClass().

10.7 UDP Applications

WAP to create an UDP Client:

```
import java.net.*;
import java.util.Scanner;
public class Client {
 public static void main(String[] args) {
  try {
   DatagramSocket socket = new DatagramSocket();
   Scanner scanner = new Scanner(System.in);
   String message;
   while(!(message=scanner.nextLine()).equalsIgnoreCase("exit")) {
    //Sending Part
    byte[] buffer = new byte[1500];
    buffer = message.getBytes();
    DatagramPacket packet = new DatagramPacket(buffer, buffer.length, InetAddress.getByName("127.0.0.1"),4567);
    socket.send(packet);
    System.out.println("Message sent: "+message);
    //Receiving Part
    buffer  = new byte[1500];
    packet = new DatagramPacket(buffer,buffer.length);
    socket.receive(packet);
    message = new String(buffer).trim();
    System.out.println("Message received: "+message);
    }} catch (Exception e) {
       e.printStackTrace();}}}
```

WAP to create UDP Server.

```
import java.net.*;
public class Server {
 public static void main(String[] args) {
  try {
   DatagramSocket socket = new DatagramSocket(4567);
   System.out.println("Server is running");
   while(true) {
   //Receiving part-hello server
    byte[] buffer = new byte[1500];
    DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
    socket.receive(packet);//retrieving client message
    String message = new String(buffer).trim();
    System.out.println("Client: "+message);
    //Sending part
    InetAddress client_address = packet.getAddress();
    int client_port = packet.getPort();
    buffer = message.getBytes();
    packet = new DatagramPacket(buffer, buffer.length,client_address,client_port);
    socket.send(packet);
    System.out.println("Server: "+message);}
    } catch (Exception e) {
        e.printStackTrace();}}}
```

THANK YOU FOR YOUR ATTENTION

PREPARED BY: ER. SHARAT MAHARJAN