***EXPERIMENT NO: 6***

***AIM:*** To study UDP client server communication to reverse the string.

***THEORY:***

**User Datagram Protocol** (**UDP**) is one of the core members of the [Internet protocol suite](https://en.wikipedia.org/wiki/Internet_protocol_suite). UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network: there is no guarantee of delivery, ordering, or duplicate protection. If error-correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.

UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application; UDP avoids the overhead of such processing at the level of the network interface. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

***ATTRIBUTES OF UDP:***

* It is *transaction-oriented*, suitable for simple query-response protocols such as the [Domain Name System](https://en.wikipedia.org/wiki/Domain_Name_System) or the [Network Time Protocol](https://en.wikipedia.org/wiki/Network_Time_Protocol).
* It provides [*datagrams*](https://en.wikipedia.org/wiki/Datagram), suitable for modeling other protocols such as [IP tunneling](https://en.wikipedia.org/wiki/IP_tunneling) or [Remote Procedure Call](https://en.wikipedia.org/wiki/Remote_Procedure_Call) and the [Network File System](https://en.wikipedia.org/wiki/Network_File_System).
* It is *simple*, suitable for [bootstrapping](https://en.wikipedia.org/wiki/Bootstrapping) or other purposes without a full [protocol stack](https://en.wikipedia.org/wiki/Protocol_stack), such as the [DHCP](https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol) and [Trivial File Transfer Protocol](https://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol).
* It is *stateless*, suitable for very large numbers of clients, such as in [streaming media](https://en.wikipedia.org/wiki/Streaming_media) applications for example [IPTV](https://en.wikipedia.org/wiki/IPTV)
* The *lack of retransmission delays* makes it suitable for real-time applications such as [Voice over IP](https://en.wikipedia.org/wiki/Voice_over_IP), [online games](https://en.wikipedia.org/wiki/Online_games), and many protocols built on top of the [Real Time Streaming Protocol](https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol).
* Works well in *unidirectional* communication, suitable for broadcast information such as in many kinds of [service discovery](https://en.wikipedia.org/wiki/Service_discovery) and shared information such as broadcast time or Routing Information Protocol

***PACKET STRUCTURE OF UDP:***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **UDP Header** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ***Offsets*** | [**Octet**](https://en.wikipedia.org/wiki/Octet_(computing)) | **0** | | | | | | | | **1** | | | | | | | | **2** | | | | | | | | **3** | | | | | | | |
| **Octet** | [**Bit**](https://en.wikipedia.org/wiki/Bit) | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** |
| **0** | **0** | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| **4** | **32** | Length | | | | | | | | | | | | | | | | Checksum | | | | | | | | | | | | | | | |

The UDP header consists of 4 fields, each of which is 2 bytes (16 bits). The use of the fields "Checksum" and "Source port" is optional in IPv4 (pink background in table). In IPv6 only the source port is optional

***Source port number:***

This field identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.

***Destination port number:***

This field identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.

***Length:***

A field that specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes because that is the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. The practical limit for the data length which is imposed by the underlying [IPv4](https://en.wikipedia.org/wiki/IPv4) protocol is 65,507 bytes (65,535 − 8 byte UDP header − 20 byte [IP header](https://en.wikipedia.org/wiki/IPv4_header)).In IPv6 [jumbograms](https://en.wikipedia.org/wiki/Jumbogram) it is possible to have UDP packets of size greater than 65,535 bytes. [RFC 2675](https://tools.ietf.org/html/rfc2675) specifies that the length field is set to zero if the length of the UDP header plus UDP data is greater than 65,535.

***Checksum:***

The [checksum](https://en.wikipedia.org/wiki/Checksum) field may be used for error-checking of the header and data. This field is optional in IPv4, and mandatory in IPv6. The field carries all-zeros if unused.

***SERVER PROGRAM:***

import java.io.IOException;

import java.net.\*;

public class SimpleUDPServer {

  public static void main(String[] args){

    DatagramSocket socket = null;

    DatagramPacket inPacket = null; //recieving packet

    DatagramPacket outPacket = null; //sending packet

    byte[] inBuf, outBuf;

    String msg;

    final int PORT = 8888;

    try{

      socket = new DatagramSocket(PORT);

      while(true){

        System.out.println("Waiting for client...");

        //Receiving datagram from client

        inBuf = new byte[256];

        inPacket = new DatagramPacket(inBuf, inBuf.length);

        socket.receive(inPacket);

        //Extract data, ip and port

        int source\_port = inPacket.getPort();

        InetAddress source\_address = inPacket.getAddress();

        msg = new String(inPacket.getData(), 0, inPacket.getLength());

        System.out.println("Client " + source\_address + ":" + msg);

        //Send back to client as an echo

        msg = reverseString(msg.trim());

        outBuf = msg.getBytes();

        outPacket = new DatagramPacket(outBuf, 0, outBuf.length,

                             source\_address, source\_port);

        socket.send(outPacket);

      }

    }catch(IOException ioe){

      ioe.printStackTrace();

    }

  }

  private static String reverseString(String input) {

    StringBuilder buf = new StringBuilder(input);

    return buf.reverse().toString();

  }

}

***CLIENT PROGRAM:***

import java.io.IOException;

import java.net.\*;

public class SimpleUDPClient {

public static void main(String[] args){

DatagramSocket socket = null;

DatagramPacket inPacket = null;

DatagramPacket outPacket = null;

byte[] inBuf, outBuf;

final int PORT = 8888;

String msg = null;

try {

InetAddress address = InetAddress.getByName("127.0.0.1");

socket = new DatagramSocket();

//Convert string to byte and send to server

msg = "Hello";

outBuf = msg.getBytes();

outPacket = new DatagramPacket(outBuf, 0, outBuf.length,

address, PORT);

socket.send(outPacket);

//Receive reversed message from server

inBuf = new byte[256];

inPacket = new DatagramPacket(inBuf, inBuf.length);

socket.receive(inPacket);

String data = new String(inPacket.getData(), 0, inPacket.getLength());

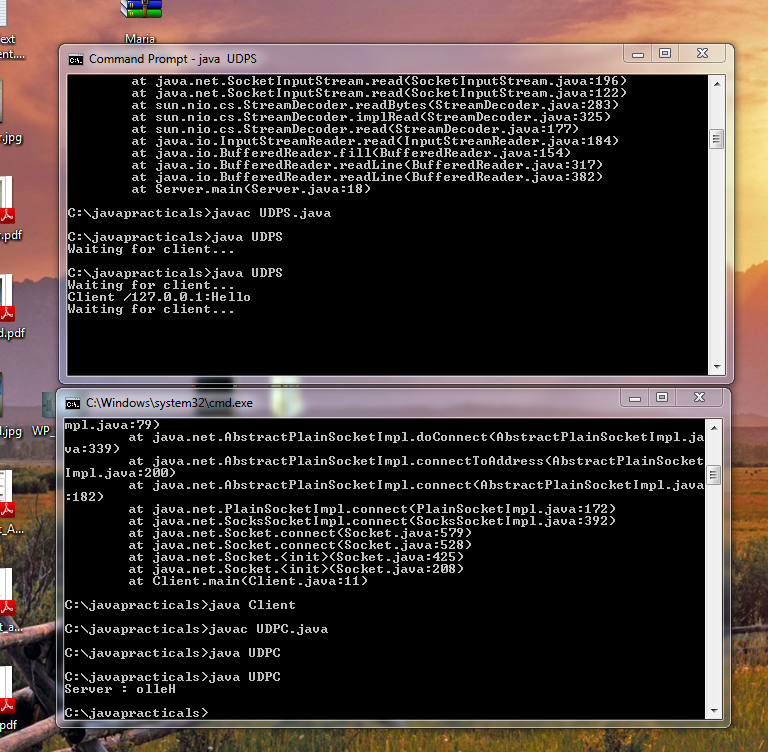
System.out.println("Server : " + data);

} catch (IOException ioe) {

System.out.println(ioe);

}}}

***OUTPUT:***

****

***CONCLUSION:***

Hence we have studied that UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss tolerating connections between applications on the Internet. Both UDP and TCP run on top of the Internet Protocol (IP) and are sometimes referred to as UDP/IP or TCP/IP. Both protocols send short packets of data, called datagrams.