UNIT- IV

UDP Datagrams and Socket

* It is a userdatagram protocol.
* It is an alternative protocol for sending data over IP that is very quick but not reliable
* When you send UDP data you have no way of knowing if the data is arrived
* And we are also not aware if the different pieces of data arrived in the order in which you sent them
* The pieces that arrive generally arrive quickly

The UDP Protocol:

* UDP is located between the Application layer and the IP layer
* It serves as an intermediary between the application programs and the network operations.

Position of UDP in the TCP/I protocol suite

Application Layer : SMTP, FTP, TFTP, DNS, SNMP, DHCP

Transport Layer: SCTP TCP UDP

Network Layer: IGMP, ICMP, IP, ARP

Data Link Layer: Underlying LAN or

Physical Layer: WAN Technology

User datagram

UDP packets called user datagrams have fixed size header of 8 bytes

Header : 8 bytes

Header and data: 8 to 65535

Header format: source port 0-16 destination port 16 – 31

Source port : gives the total length

Destination port: gives check sum

* Udp header add only 8 bytes to the IP header .
* Udp includes source and destination port number
* Port numbers are given as 2 byte unsigned integers 65,536-8bytes for the header this is redundant with the data gram length field of the IP header
* The exact number depends on the size of the header
* Check sum field is an optional and not used in or accessible from application laye
* If the checksum for data fails the native software silently discards the datagram
* Neither the sender or the receiver are notified
* UDP is an unreliable protocol

Structure of UDP datagram:

Version header length Types of service datagram length

Identification flags fragment

--------------------------------------------------------------------------------------------------------------------------------------Time to live (TTL) Protocol Header check sum

--------------------------------------------------------------------------------------------------------------------------------------

Source Address

Destination Address

Option

Souce port (0-65,535) destination port (0-65,535)

Combined length of data and UDP header(8-65535) destination port (0-65,535)

In java UDP datagram is represented as public final class DatagramPacket extends object

Datagram Socket Class:

* They are bound to local port
* They listen for incoming data and which they place in the header of outgoing datagram.
* If you are writing a server clients you should know which port the server is listening for incoming datagrams
* When server constructs a datagram socket it specifies the local port on which it will listen

The constructorsL

* The datagram socket constructors are used in different situation

1. 1st constructor opens a datagram on an anonymous local port
2. Opens a datagram on all local network interface
3. Opens a datagram socket on a well known local port on specific network interface

Program for look for local ports:

Import java.et.\*;

Public class UDPPort Scanner{

Public static void main(String [] args)

{

For(int port = 1024;port<=65535;port++)

Try

{

//the next line will fail and fall into the catch block if there is already a server running on port i

DatagramSocket server=new DatageramSocket(port);

Server.Close();

}

Catch(SocketException ex) {

System.out.println(“there is a server on port” +port +”.”);

}

}

}

Write a program to construct a Datagram packet to receive data

Import java.net.\*;

Public class datagram {

Public static void main(String[] args)

{

String s=”this is a test”;

Byte[] data=s.getBytes();

Try

{

InetAddress ia = InetAddress.getByName(“[www.google.com](http://www.google.com)”);

Int port=7;

DatagramPacket dp=new DatagramPacket(data,data.Length,ia,port);

System.out.println(“this packet is addressed to “ +dp.getAddress() +”on port” + dp.getPort());

System.out.println(“There are” +dp.getLength() + “bytes of data in the packet”);

System.out.println(new String(dp.getData(), dp.getOffset(), dp.getLength()));

}

Catch(UnknownHostException e)

{

System.err.println€;

}

}

}

Some useful Applications:

* In this section several internet servers and clients use the DatagramPacket and DatagramSocket.
* Many internet protocols have both TCP and UDP implementation
* When the IP packet is received by a host , the host determines whether the packet is a TCP packet or a UDP Datagram by inspecting the header
* TCP and UDP servers can share the same port number without problems

Simple UDP Client:

* Several internet services need to know only the clients address and port
* They ignore any data the client sends in the datagram
* Daytime, quote of the day,time and char gen are four such protocol
* Each of these responds the same way, regardless of the data contained in the datagram
* Clients for these protocol simply send a UDP datagrams to the server and read the response that come back

Sample program:

Import java.net.\*;

Import java.io.\*;

Import java.lang.\*;

Public class EC

{

Public static void main(String args[]) throws IOException

{

Byte[] buff = new byte[1024];

DatagramSocket soc = new DatagramSocket(9999);

String S = “ From Client hello server”;

Buff=s.getBytes();

InetAddress a= InetAddress.getByName(“Gfi-335”);

DatagramPacket pac=new DatagramPacket(buff,buff.length,a,8888);

Soc.send(pac);

System.Out.println(“end of sending”);

Byte[] buff1=new byte[1024];

Buff1=s.getBytes();

Pac=new DatagramPacket(buff1,buff.Length);

Soc.receive(pac);

String msg=new String(pac,getDate());

System.out.println(msg);

} System.out.println(“eof”)l

}

}

Import java.net.\*;

Import java.io.\*;

Import java.lang.\*;

Public class ES

{

Public static void main(String args[]) throws IOException

{

Byte[] buff=new byte[512];

DatagramSocket soc = new DatagramSocket(8888);

DatagramPacket pac=new DatagramPacket(buff, buff.length);

System.out.printl n(“serverver started”);

Soc.receive(pac);

String msg=new String(pac.getData());

System.out.println(msg);

System.out.println(“end of reception”);

String s=”From Server hello client”;

Byte[] buff=new byte[512];

Buff1=s.getBytes();

InetAddress a = pac.getAddress();

Int port=pac.getPort();

Pac=new DatagramPacket(buff, buff1.length,a,Port);

Soc.send(pac);

System.out.println(“end of sending”);

}

}

Datagram Channel:

\*datagram channel is a near complete alternate abstraction for UDP I/O

\*Datagram channel is a subclass of selectable that can be registered with a selector

\*datagram channel is used to bind a channel to a port.

Opening Socket:

\*The java.nio.channels.DatagramChannel class does not have any public constructor

\* You create a new Datagram Channel object using static open() method

Public static DatagramChannel Open() throws IOException

Datagram Channel=DatagramChannel.Open();

* Datagram channel is initially not bound to any port.
* To bind it you need to access the channels peer DatagramSocket object using the Socket method.
* Public abstract DatagramSocket Socket()

For e.g. this binds channel to port 3141.

SocketAddress address = new InetSocketAddress(3141);

DatagramSocket socket=channel.Socket();

Socket.bind(address);

Connecting:

Datagram Channel can be connected and it can be configured to only receive datagrams from and send datagrams to one host.

\*this is accomplished with connect method.

Public abstract DatagramChannel Connect(SocketAddress remote) throws IOException

TCP\_NODELAY() : it will makemultiple buffers as Individual packet

SO\_TIMEOUT(): java.net.socket.time out blocks from reader enable disable

SO\_LINGER(): socket is shut down

SO\_RCVBUF(): it is the size of the buffer kernel allocates to hold the data arriving into the given socket and then read

SO-SNDBUF: only in TCP communicates to kernel and stops sending data and accumulates in local buffer