USER DATAGRAM PROTOCOL (UDP)

- connectionless transport protocol

A UDP DATAGRAM FRAME FORMAT

-- 4 bytes--

Version	Header Length	TOS	Total Length
Identification		Flag	Frag Offset
TTL	Protocol	Header Checksum	
Source Address			
Destination Address			
Variable Size IP Options			
Source Port		Destination Port	
Length		Checksum	
Data Payload (up to 65508 bytes)			

TOS – Type of Service (minimum delay, maximum throughput, maximum reliabilty, etc.)

Total Length = Header length + payload length

TTL = Time to Live

Identification = Unique ID for each datagram.

Flag = (3-bit flag) First bit is 0, 2nd bit is 0 if fragmentation is allowed; 1 for not, 3rd bit is 0 is this is the last fragment; 1 if not.

Frag Offset = Fragment offset.

Common UDP Applications

- Domain Name Service (DNS)
- Simple Network Management Protocol (SNMP)
- Trivial File Transfer Protocol (TFTP)

DatagramPacket / DatagramSocket

The ***DatagramPacket*** class

Constructors:

```
// for receiving
DatagramPacket(byte [ ] buffer, int length)

//for transmitting
DatagramPacket(byte [ ] buffer, int length,
                InetAddress dest_addr, int dest_port)
```

Methods:

InetAddress getAddress( )
returns the IP address of either the sender or the recipient.

byte[ ] getData( )
returns the contents of the Datagram packet

int getLength( )
returns the length of data packet.

int getPort( )
returns the port number from which the DatagramPacket was sent or the destination port number.

void setAddress(InetAddress addr)
sets the new destination of the DatagramPacket

void setData(byte [ ] buffer)
sets the data in the packet with contents of buffer

void setLength(int len)
sets the new length for the DatagramPacket

void setPort(int port)
sets the destination port for the DatagramPacket

The ***DatagramSocket*** class

Constructors:

//for the client
DatagramSocket( ) throws java.net.SocketException

//for the server
DatagramSocket(int port) throws java.net.SocketException

Methods:

void close( )
closes a socket.
void connect(InetAddress remote_addr, int remote_port)
restricts access to the specified address and port.
void disconnect( )
disconnects the DatagramSocket and removes all restrictions.
InetAddress getInetAddress( )
returns the remote IP address to which socket is connected.
int getPort( )
returns the remote port number to which socket is connected or a -1 if no connection exists.
InetAddress getLocalAddress( )
returns the local address to which socket is bound.
byte[ ] getLocalPort( )
returns the local port to which socket is bound.
int getReceiveBufferSize( ) throws java.net.SocketException
int setReceiveBufferSize(int len) throws java.net.SocketException
returns/sets the maximum buffer size used for incoming UDP packets.
void getSendBufferSize( ) throws java.net.SocketException
void getSendBufferSize(int len) throws java.net.SocketException
returns/sets the maximum buffer size for outgoing UDP packets.
void getSoTimeout( ) throws java.net.SocketException
void setSoTimeout(int duration) throws java.net.SocketException
returns/sets the value of the timeout socket option.
By default, value is 0 indicating blocking I/O is used.
void receive(DatagramPacket packet) throws java.io.IOException
reads a UDP packet and saves its contents. The address and port number will be overwritten with the sender address and port fields.
void send(DatagramPacket packet)
sends the UDP DatagramPacket.

Java UDP Programming

1. Java UDP server creates a DatagramSocket instance bound to a UDP port.

2. Server creates a DatagramPacket instance that will be used to store the datagram that will be received.

3. Server invokes the DatagramSocket's receive( ) method. Note that this is a blocking operation.

4. Java UDP client creates a DatagramSocket instance bound to a particular UDP port.

5. Client creates a DatagramPacket destined to the Server's address and port with the data payload.

6. Datagram is then transmitted using the DatagramSocket's send( ) method.

7. The header is created before transmission.

8. If the UDP packet manages to get to the destination (server), the packet is copied into the server's DatagramPacket instance and the receive( ) method is unblocked. If the receive( ) method has not been invoked, the DatagramPacket instance is buffered until it is properly received.

```
//This is the UDP Receiver
import java.net.*;
import java.io.*;
import java.util.*;

public class UDPReceiver  {
        public static void main (String args[ ]) {
          try{
                // Create a datagram socket, bound to the specific port 2000
                DatagramSocket socket = new DatagramSocket(2000);

                System.out.println ("Bound to local port " + socket.getLocalPort());

                // Create a datagram packet, containing a maximum buffer of 256 bytes
                DatagramPacket packet = new DatagramPacket( new byte[256], 256 );

                // Receive a packet - remember by default this is a blocking operation
                socket.receive(packet);

                System.out.println ("Packet received at " + new Date( ));
                // Display packet information
                InetAddress remote_addr = packet.getAddress();
                System.out.println ("Sender: " + remote_addr.getHostAddress( ) );
                System.out.println ("from Port: " + packet.getPort());

                // Display packet contents, by reading from byte array
                ByteArrayInputStream bin = new ByteArrayInputStream
                                                (packet.getData());

                // Display only up to the length of the original UDP packet
                for (int i=0; i < packet.getLength(); i++)  {
                        int data = bin.read();
                        if (data == -1)
                                break;
                        else
                                System.out.print ( (char) data) ;
                }

                socket.close( );
        }
        catch (IOException e)           {
                System.out.println ("Error - " + e);
        }

   } //end of main
} //end of class definition
```

```
//The UDP Sender
import java.net.*;
import java.io.*;
import java.util.*;

public class UDPSender {
        public static void main (String args[ ]) {
           //use localhost to experiment on a standalone computer
           String hostname="localhost";    String message = "HELLO USING UDP!";
                try {
                  // Create a datagram socket, look for the first available port
                  DatagramSocket socket = new DatagramSocket();

                  System.out.println ("Using local port: " + socket.getLocalPort());
                  ByteArrayOutputStream bOut = new ByteArrayOutputStream();
                  PrintStream pOut = new PrintStream(bOut);
                  pOut.print(message);
                  //convert printstream to byte array
                  byte [ ] bArray = bOut.toByteArray();
                  // Create a datagram packet, containing a maximum buffer of 256 bytes
                  DatagramPacket packet=new DatagramPacket( bArray, bArray.length );

                System.out.println("Looking for hostname " + hostname);
                //get the InetAddress object
                InetAddress remote_addr = InetAddress.getByName(hostname);
                //check its IP number
                System.out.println("Hostname has IP address = " +
                                        remote_addr.getHostAddress());
                //configure the DataGramPacket
                packet.setAddress(remote_addr);
                packet.setPort(2000);
                //send the packet
                socket.send(packet);
                System.out.println ("Packet sent at!" + new Date());

                // Display packet information
                System.out.println ("Sent by  : " + remote_addr.getHostAddress() );
                System.out.println ("Send from: " + packet.getPort());

                }
           catch (UnknownHostException ue){
                System.out.println("Unknown host "+hostname);
           }
                catch (IOException e) {
                        System.out.println ("Error - " + e);
                }

        }//end of main
}//end of class definition
```