destination based on information contained within that packet. That means, each packet needs to have destination address and each packet might be routed differently, and might arrive in any order. Packet delivery is not guaranteed.

Java supports datagram communication through the following classes:

- DatagramPacket
- DatagramSocket

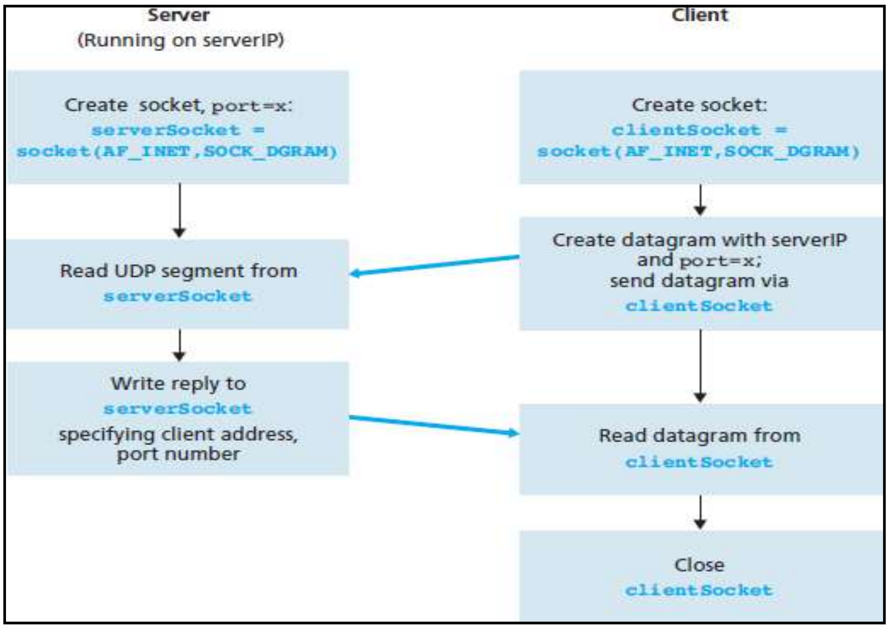

Fig -7-UDP client/server communication flow:

/* Datagram Socket Program */

<u>UDP Server</u>

```
import java.io.*;
import java.net.*;
public class UDPServer
{
 public static void main(String[] args)
{
DatagramSocket skt=null;
try
{
System.out.println("server is started");
skt=new DatagramSocket(6788);
byte[] buffer = new byte[1000];
while(true)
{
DatagramPacket request = new DatagramPacket(buffer,buffer.length);
skt.receive(request);
String[] message = (new String(request.getData())).split(" ");
byte[] sendMsg= (message[1].toUpperCase()+ " from server to client").getBytes();
DatagramPacket reply = new
DatagramPacket(sendMsg,sendMsg.length,request.getAddress(),request.getPort());
skt.send(reply);
}
}
catch(Exception ex)
{
System.out.println(ex.getMessage());
}
}
}
```

UDP Client

```
import java.io.*;
import java.net.*;
public class UDPClient
{
 public static void main(String[] args)
{
DatagramSocket skt;
try
{
skt=new DatagramSocket();
String msg= "atme college ";
byte[] b = msg.getBytes();
InetAddress host=InetAddress.getByName("127.0.0.1");
int serverSocket=6788;
DatagramPacket request =new DatagramPacket (b,b.length,host,serverSocket);
skt.send(request);
byte[] buffer =new byte[1000];
DatagramPacket reply= new DatagramPacket(buffer,buffer.length);
skt.receive(reply);
System.out.println("client received:" +new String(reply.getData()));
skt.close();
}
catch(Exception ex)
{
   System.out.println(ex.getMessage());
}
}
}
```