T3 – UDP Java Sockets

Socket programming *with UDP*

UDP: no "connection" between client and server

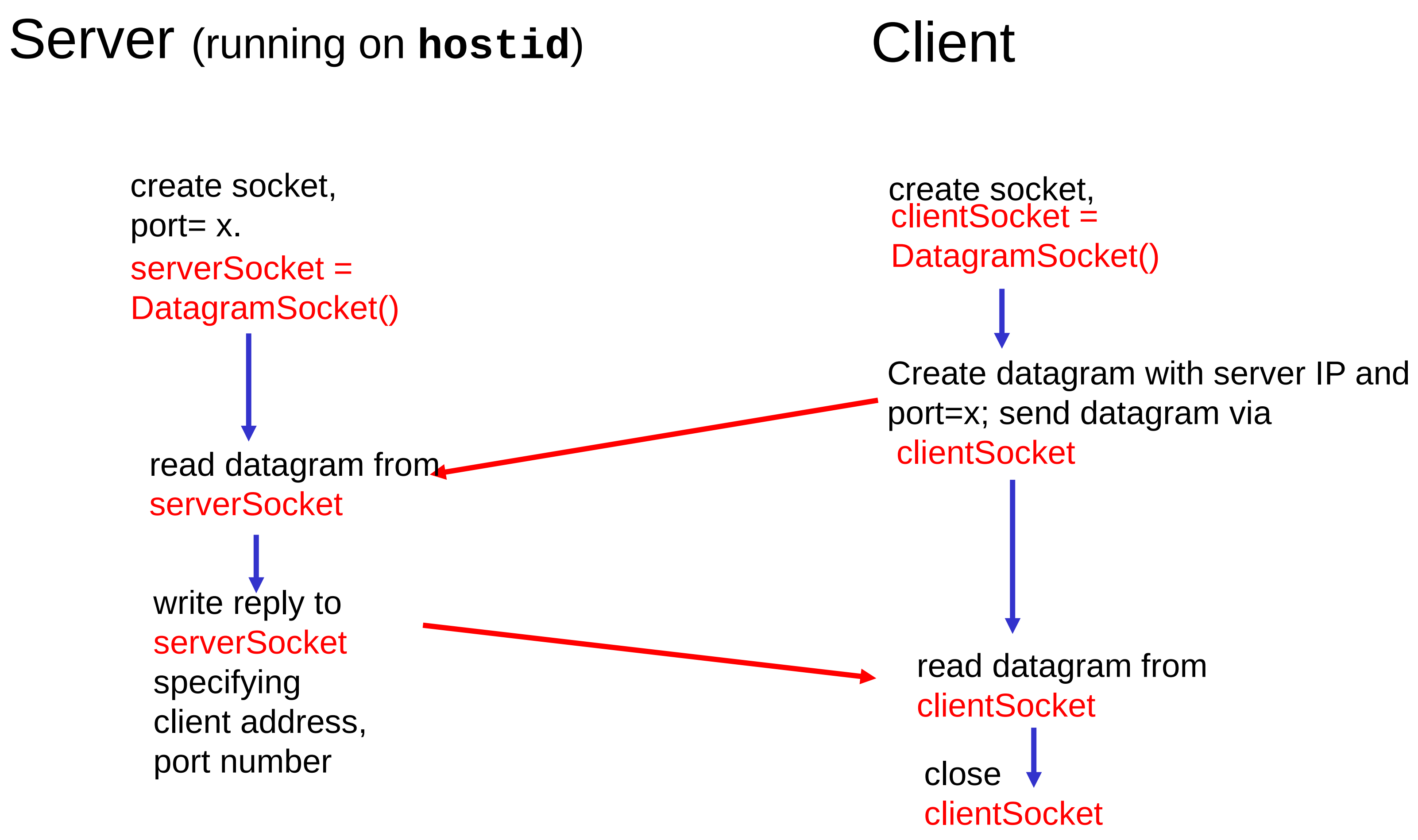
- no handshaking
- sender explicitly attaches IP address and port of destination to each packet
- server must extract IP address, port of sender from received packet

UDP: transmitted data may be received out of order, or lost

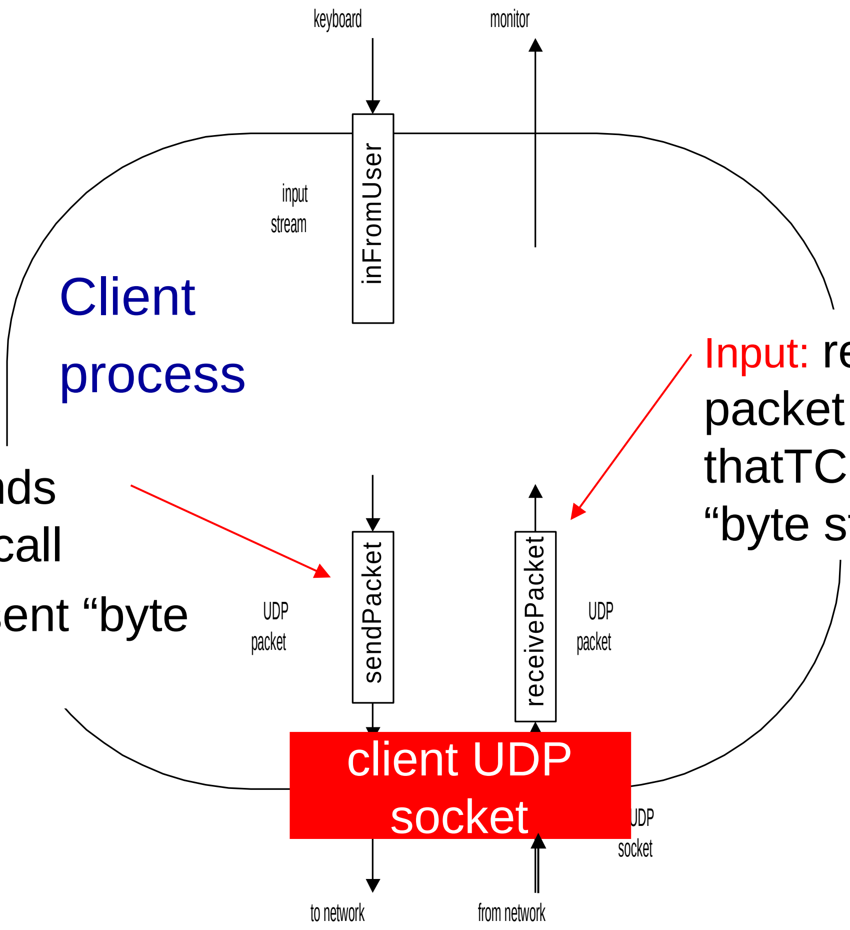
application viewpoint:

UDP provides unreliable transfer of groups of bytes ("datagrams") between client and server

Client/server socket interaction: UDP


Server (running on **hostid**)
Client
create socket, port= x. serverSocket = DatagramSocket()
read datagram from serverSocket
write reply to serverSocket specifying client address, port number
create socket, clientSocket = DatagramSocket()
Create datagram with server IP and port=x; send datagram via clientSocket
read datagram from clientSocket
close clientSocket

Example: Java client (UDP)



Output: sends packet (recall that TCP sent "byte stream")

Input: receives packet (recall thatTCP received "byte stream")

Example: Java client (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception
    {
        Scanner inFromKeyboard =  new Scanner(System.in);
      System.out.println(“ Introduce the data to send to the server.“);

    DatagramSocket clientSocket = new DatagramSocket();

  int p = clientSocket.getLocalPort();

System.out.println(“ Client uses port:  “ + p);

 InetAddress IPAddress = InetAddress.getByName("localhost");


   String sentence = inFromKeyboard.nextLine();
```

create input stream

create client socket

translate hostname to IP address using DNS

Example: Java client (UDP), cont.

create datagram with data-to-send, length, IP addr, port

```
DatagramPacket sendPacket =
   new DatagramPacket(sentence.getBytes(), sentence.getBytes().length, IPAddress, 7777);
```

```
clientSocket.send(sendPacket);
```

send datagram to server

Create Datagrampacket where to receive the incoming datagram

```
byte[] receiveDataBuffer = new byte[512];
DatagramPacket receivePacket =  new DatagramPacket(receiveDataBuffer, receiveData.length);
```

```
clientSocket.receive(receivePacket);
```

read datagram from server

```
 String modifiedSentence = new String(receivePacket.getData(), 0,receivePacket.getLength()) ;

 System.out.println("FROM SERVER:" + modifiedSentence);
 clientSocket.close();
 }
}
```

To read only the bytes sent by the server

Example: Java server (UDP)

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {

      DatagramSocket serverSocket = new DatagramSocket(7777);

      byte[] receiveDataBuffer = new byte[1024];
      byte[] sendDataBuffer  = new byte[1024];

      while(true)
        {

          DatagramPacket receivePacket =
             new DatagramPacket(receiveDataBuffer, receiveDataBuffer.length);
           serverSocket.receive(receivePacket);
```

create datagram socket at port 9876

create space for received datagram

receive datagram

Example: Java server (UDP), cont

```
            String sentence = new String( receivePacket.getData());

            InetAddress IPAddress = receivePacket.getAddress();

            int port = receivePacket.getPort();

            String capitalizedSentence = sentence.toUpperCase();

            sendDataBuffer = capitalizedSentence.getBytes();

            DatagramPacket sendPacket =
               new DatagramPacket(sendDataBuffer, sendDataBuffer.length, IPAddress,port);

            serverSocket.send(sendPacket);
         }
      }
   }
```

get IP addr port #, of sender

create datagram to send to client

write out datagram to socket

end of while loop, loop back and wait for another datagram

Example: Java server (UDP), cont

```
        String sentence = new String( receivePacket.getData());



        String capitalizedSentence = sentence.toUpperCase();

        sendDataBuffer = capitalizedSentence.getBytes();

       byte[] buffer = new byte[512];
       DatagramPacket sendPacket = new DatagramPacket(buffer, 512);

     sendPacket.setAddress(receivePacket.getAddress());
    sendPacket.setPort(receivePacket.getPort());
    sendPacket.setData(capitalizedSentence.getBytes());
    sendPacket.setLength(capitalizedSentence.getBytes().length);



      serverSocket.send(sendPacket);
      }
    }
  }
```

reate datagram
to send to client

write out
datagram
to socket

InetAddress class

getByName method

public static InetAddress getByName(String host) throws UnknownHostException

- Examples of use:

```
InetAddress ipServer = InetAddress.getByName(“zoltar.redes.upv.es”);
InetAddress ipServer = InetAddress.getByName(args[0]);
InetAddress ipServer = InetAddress.getByName(“127.0.0.1”);
```

getAllByName method

public static InetAddress[] getAllByName (String host) throws UnknownHostException

Examples of use:

```
InetAddress[] listaIps = InetAddress.getAllByName(“www.hotmail.es”);
```

To print it, we can use the method: Arrays.toString(Object[] a) (we will have to import java.util.Arrays).

Example:

```
System.out.println(Arrays.toString(InetAddress.getAllByName("www.hotmail.es")));
```