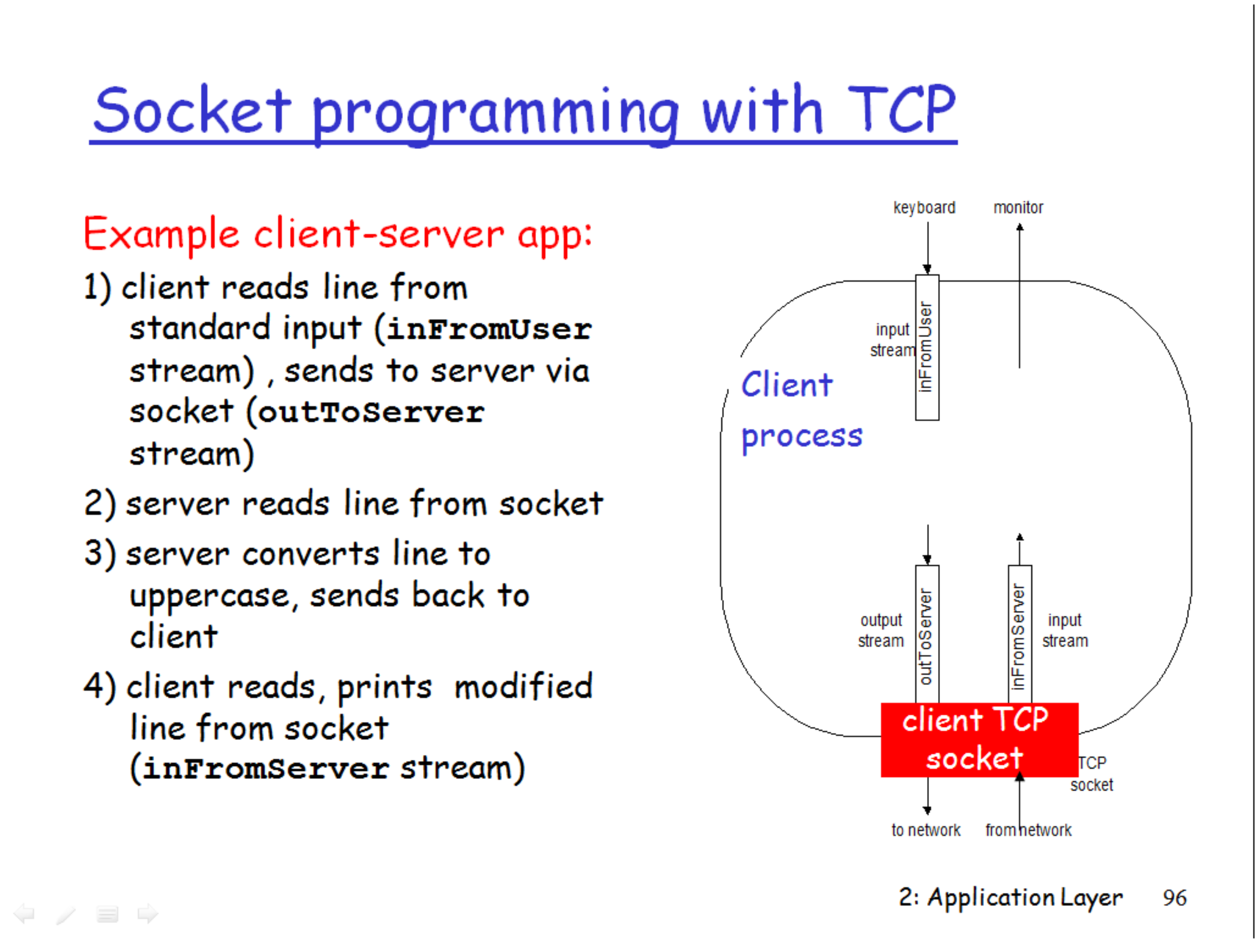
AMRITA VISHWA VIDYAPEETHAM
Department of Computer Science and Engineering
CSE394 Computer Networks Lab

Lab Exercise: 3 – Getting started with Socket Programming

Objective:

Familiarization with basic network programming in Java using TCP and UDP
For the programs listed below, try running the client and server on the same machine. So keep your hostname as "localhost".

How to program a simple TCP Client Server



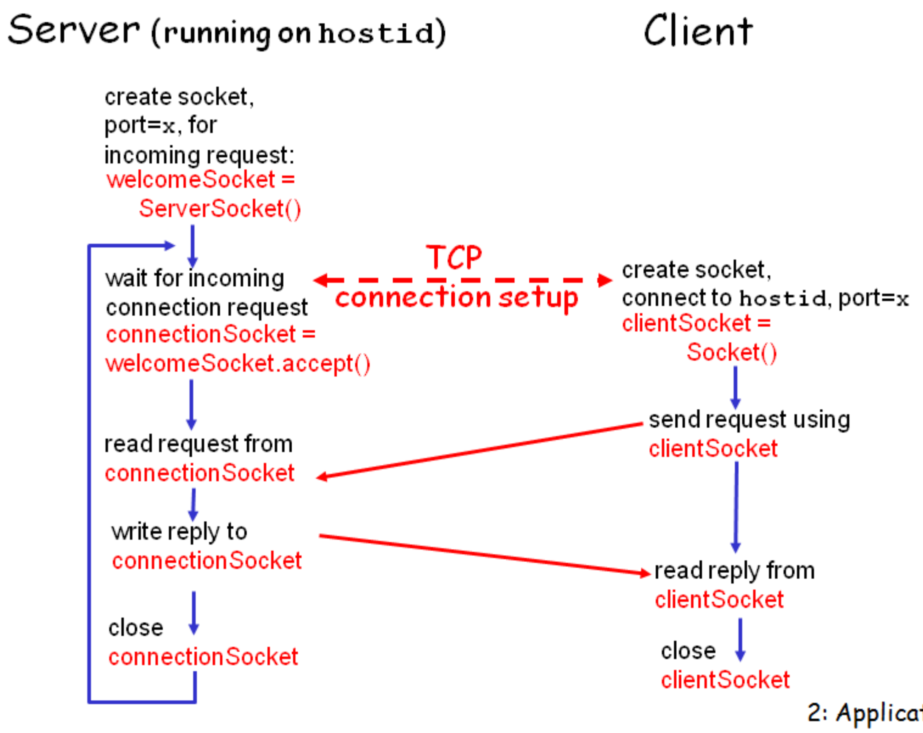
Socket programming with TCP

Example client-server app:

1) client reads line from standard input (**inFromUser** stream) , sends to server via socket (**outToServer** stream)
2) server reads line from socket
3) server converts line to uppercase, sends back to client
4) client reads, prints modified line from socket (**inFromServer** stream)

2: Application Layer 96

Client/server socket interaction: TCP


Server (running on hostid)
Client
create socket, port=x, for incoming request:
welcomeSocket = ServerSocket()
wait for incoming connection request
connectionSocket = welcomeSocket.accept()
TCP connection setup
create socket, connect to hostid, port=x
clientSocket = Socket()
send request using clientSocket
read request from connectionSocket
write reply to connectionSocket
read reply from clientSocket
close connectionSocket
close clientSocket

Example: Java client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;
```

Create input stream

```
        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));
```

Create client socket, connect to server

```
        Socket clientSocket = new Socket("hostname", 6789);
```

Create output stream attached to socket

```
        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Create input stream attached to socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();
```

Send line to server

```
outToServer.writeBytes(sentence + '\n');
```

Read line from server

```
modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

  }
}
```

Example: Java server (TCP)

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

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
    String clientSentence;
    String capitalizedSentence;
```

Create welcoming socket at port 6789

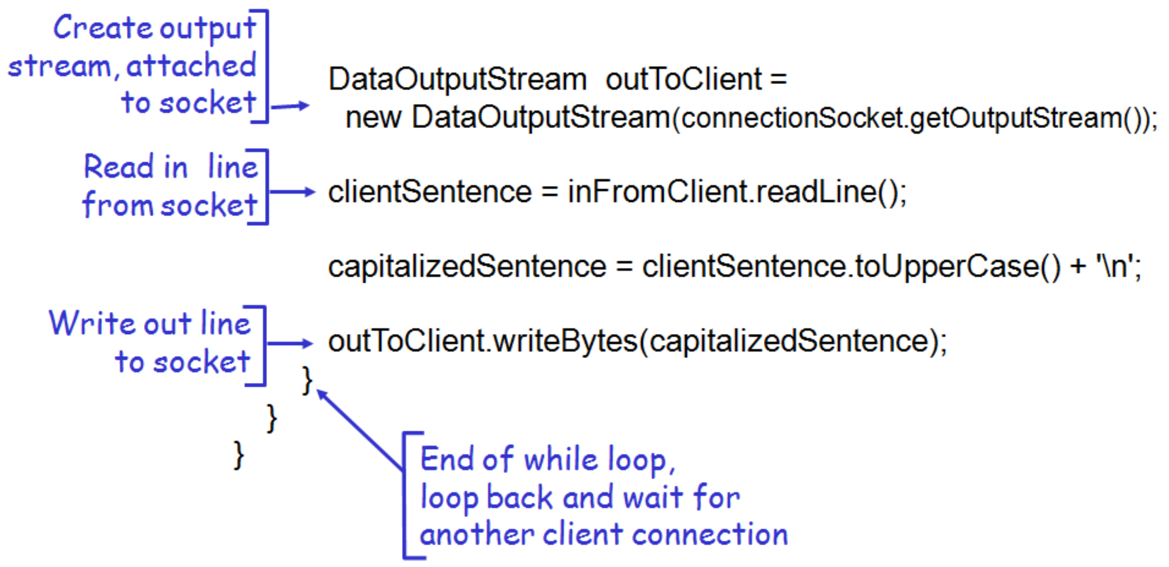
```
    ServerSocket welcomeSocket = new ServerSocket(6789);

    while(true) {
```

Wait, on welcoming socket for contact by client

```
        Socket connectionSocket = welcomeSocket.accept();
```

Create input stream, attached to socket

```
       BufferedReader inFromClient =
         new BufferedReader(new
         InputStreamReader(connectionSocket.getInputStream()));
```



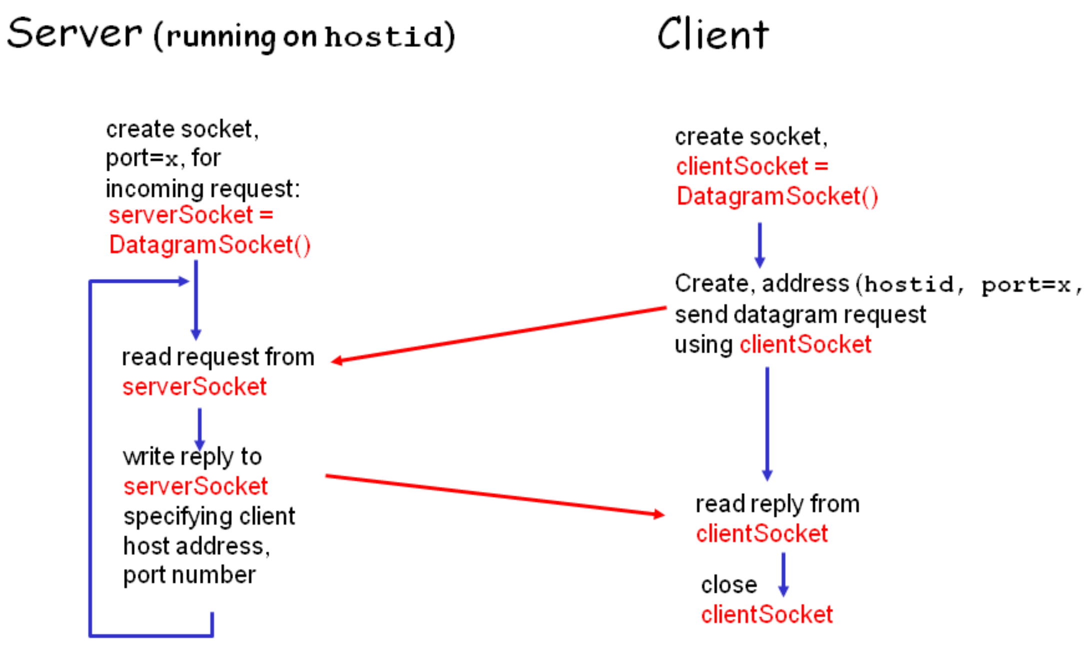
```
        DataOutputStream  outToClient =
          new DataOutputStream(connectionSocket.getOutputStream());

        clientSentence = inFromClient.readLine();

        capitalizedSentence = clientSentence.toUpperCase() + '\n';

        outToClient.writeBytes(capitalizedSentence);
       }
     }
 }
```

Create output stream, attached to socket
Read in line from socket
Write out line to socket
End of while loop, loop back and wait for another client connection

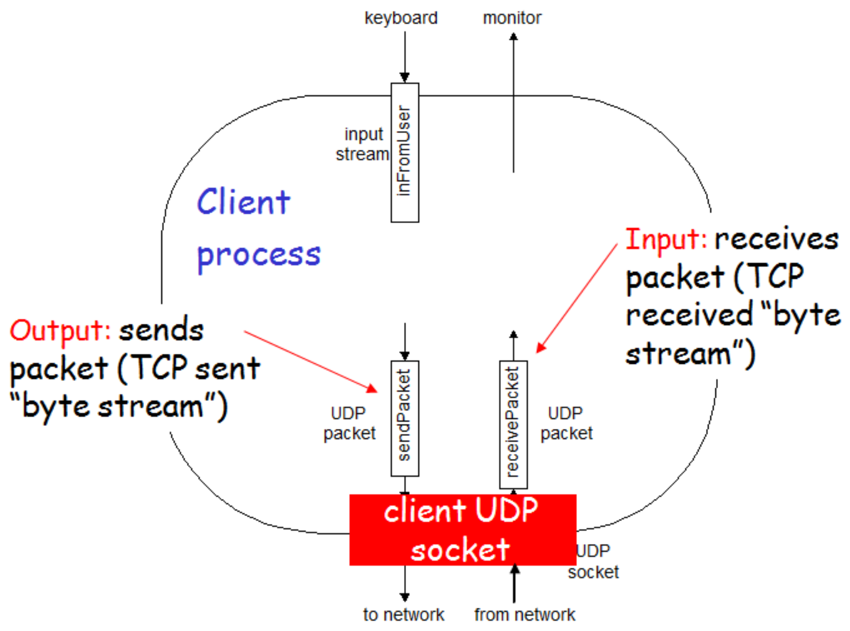
---

How to program a simple UDP Client Server

Client/server socket interaction: UDP


Server (running on hostid)
Client
create socket, port=x, for incoming request: serverSocket = DatagramSocket()
read request from serverSocket
write reply to serverSocket specifying client host address, port number
create socket, clientSocket = DatagramSocket()
Create, address (hostid, port=x, send datagram request using clientSocket
read reply from clientSocket
close clientSocket

Example: Java client (UDP)



Example: Java client (UDP)

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

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

     BufferedReader inFromUser =
       new BufferedReader(new InputStreamReader(System.in));

     DatagramSocket clientSocket = new DatagramSocket();

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

     byte[] sendData = new byte[1024];
     byte[] receiveData = new byte[1024];

     String sentence = inFromUser.readLine();
     sendData = sentence.getBytes();
```

Create input stream → `BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));`

Create client socket → `DatagramSocket clientSocket = new DatagramSocket();`

Translate hostname to IP address using DNS → `InetAddress IPAddress = InetAddress.getByName("hostname");`

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

```
DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
```

Send datagram to server

```
clientSocket.send(sendPacket);

DatagramPacket receivePacket =
  new DatagramPacket(receiveData, receiveData.length);
```

Read datagram from server

```
clientSocket.receive(receivePacket);

String modifiedSentence =
   new String(receivePacket.getData());

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

```
}
```

Example: Java server (UDP)

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

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

Create datagram socket at port 9876

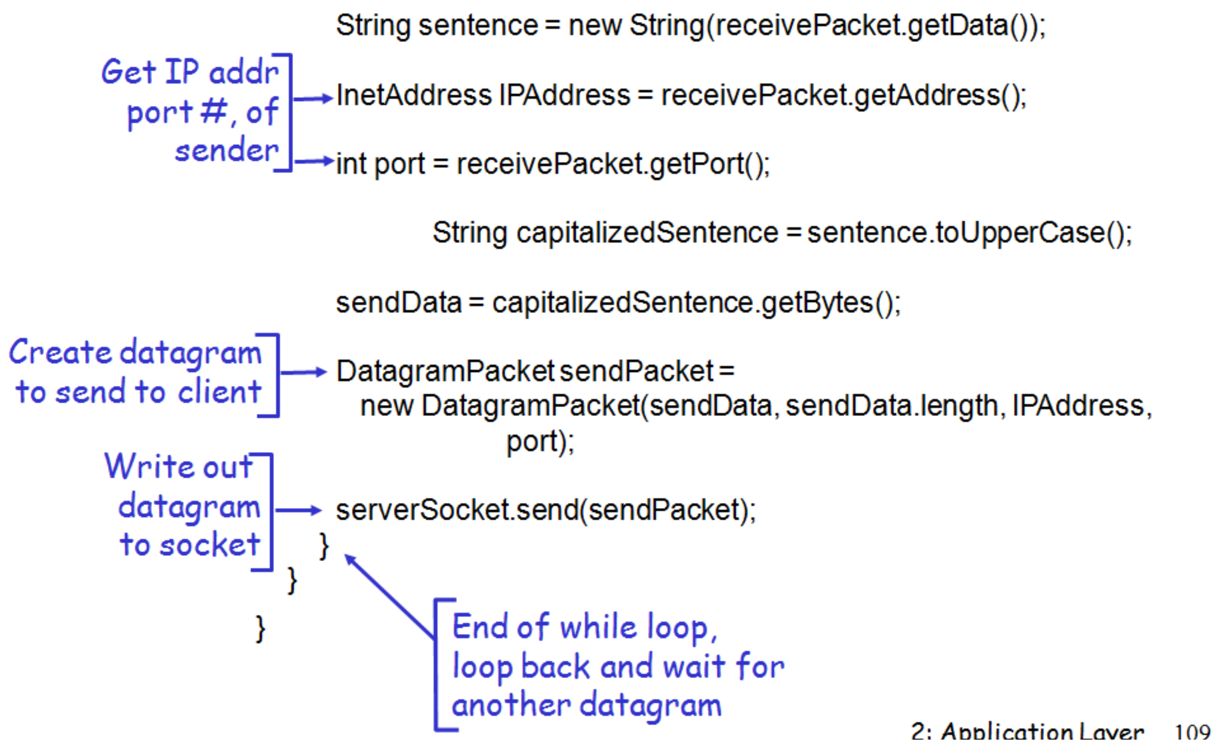
```
    DatagramSocket serverSocket = new DatagramSocket(9876);

    byte[] receiveData = new byte[1024];
    byte[] sendData  = new byte[1024];

    while(true)
      {
```

Create space for received datagram

```
        DatagramPacket receivePacket =
          new DatagramPacket(receiveData, receiveData.length);
```

Receive datagram

```
        serverSocket.receive(receivePacket);
```



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

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

        String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.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
2: Application Layer 109

Lab Questions

1. Implement the simple TCP server and UDP server mentioned above. (10 Marks)

2. Try running the TCP server twice. Note down in your observation record, the message that comes when you try to run the TCP server for the second time. Why do you think that this error is coming? (5 Marks)

3. With the server running on port 6789, try connecting the client to a different port. Write down the message that you get in your observation record. (5 Marks)

4. Modify the TCP Client Server example to create a chat program. For the sake of simplicity, let one user be on the client and let the other user be on the server. The user on client process and the user on the server process should be able to pass messages between each other. (20 Marks)