JAVA

NETWORK

PROGRAMMING

TCP/IP

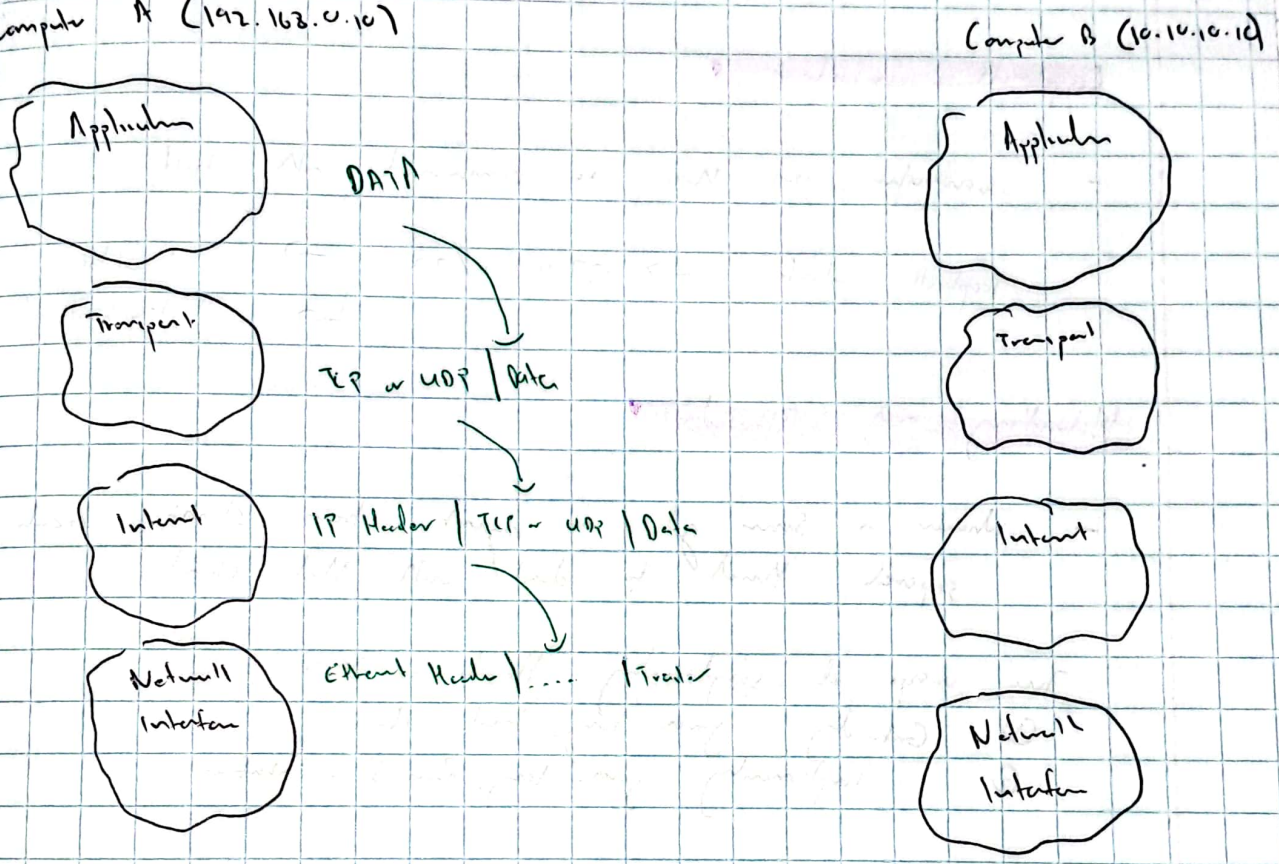
SOCKET

PROGRAMMING

Section 1 : Introduction to Networking

Networking Fundamentals

TCP/IP Model



TCP (Transmission Control Protocol)
- tracking transmitted data segments
- acknowledge received data
- retransmitting any unacknowledged data

UDP (User Datagram Protocol)
- packets may be lost or delivered out of order
- message split into datagrams, user sends datagram or packet on network layer (internet layer)
- unreliable but fast
- application must deal with lost packets

IP (Internet Protocol)

Basic characteristics of IP are :
- connectionless
- Best Effort (unreliable)
- Media independant

Loopback Interface

- computer uses this to communicate with itself

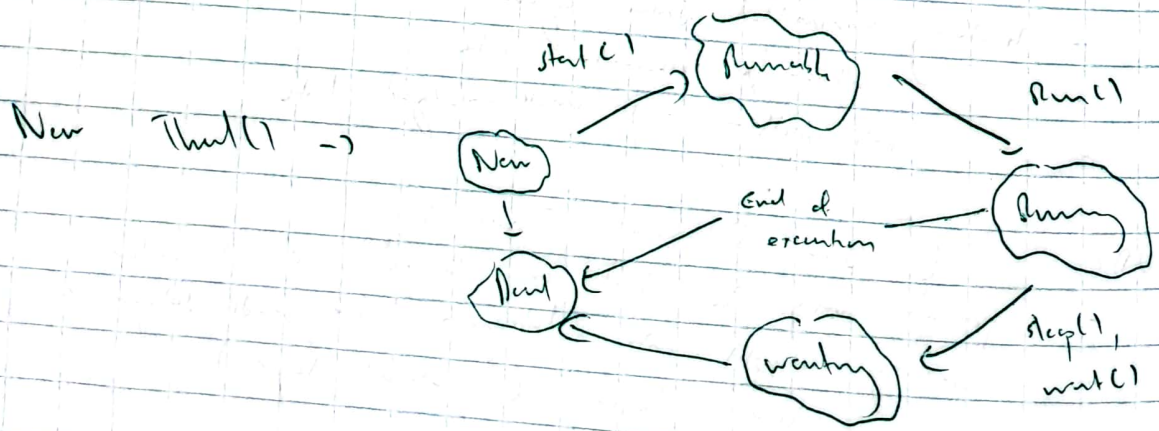
loopback interface → 127.0.0.1/32 → "localhost"
↳ "loopback"

Networking and Threads

- whenever a Server get a request from client, create a seperate thread to deal with that client

Two ways of implementing thread in Java
① Extending java.lang.Thread class
② Implementing java.lang.Runnable interface

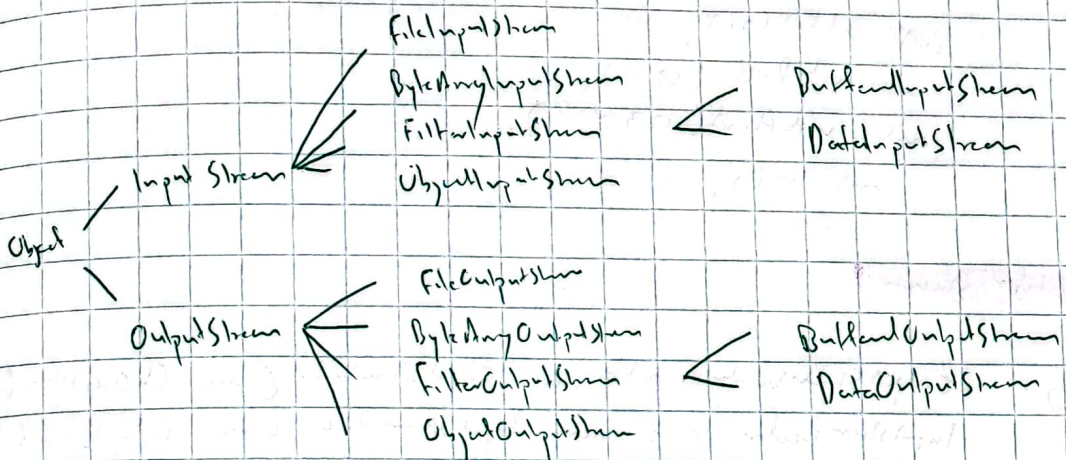
Life cycle of a thread



- start() → new Thread created and run
- run() → no new Thread

Section 3: I/O Streams

Java I/O



FileInputStream

Eg:
```
FileInputStream inputStream = new FileInputStream("example.txt");
int data = inputStream.read();

while (data != -1) {
    System.out.print(data); → char(data)
    data = inputStream.read();
}

inputStream.close();
```

FileOutputStream

Eg:
```
FileOutputStream fileOutput = new FileOutputStream("exampl.txt");
String string = "Hello";
fileOutput.write(string.getByte());
fileOutput.close();
```

DataStream

Eg:

```
File fil = new File("exmpl.txt");
file.createNewFile();

DataOutputStream out = new DataOutputStream(new FileOutputStream(fil.getName()));

out.writeInt(72);
out.writeDouble(68.44);
out.writeFloat(123.456f);
out.close();
```

Character Stream

Eg:

```
OutputStreamWriter out = new OutputStreamWriter(new FileOutputStream("exmpl.txt"
InputStreamReader in = new InputStreamReader(new FileInputStream("exmpl.txt")

out.write("Hello");

out.flush();

// out.close();        call flush method

int data = in.read();

while (data != -1)
        {
            System.out.print((char)data);
            data = in.read()
        }
```

BufferedStream

Eg: BufferedReader reader = new BufferedReader(new FileReader("example.txt"));
BufferedWriter writer = new BufferedWriter(new FileWriter("example.txt"));

String line = null;

while ((line = reader.readLine()) != null)
{
writer.write(line);
writer.newLine();
}
writer.close();
reader.close();

PrintStream

Eg: PrintStream out = new PrintStream("example.txt");
out.println(var);
out.close();

System.in

Eg: System.out.println("Enter part:");
InputStreamReader in = new InputStreamReader(System.in);

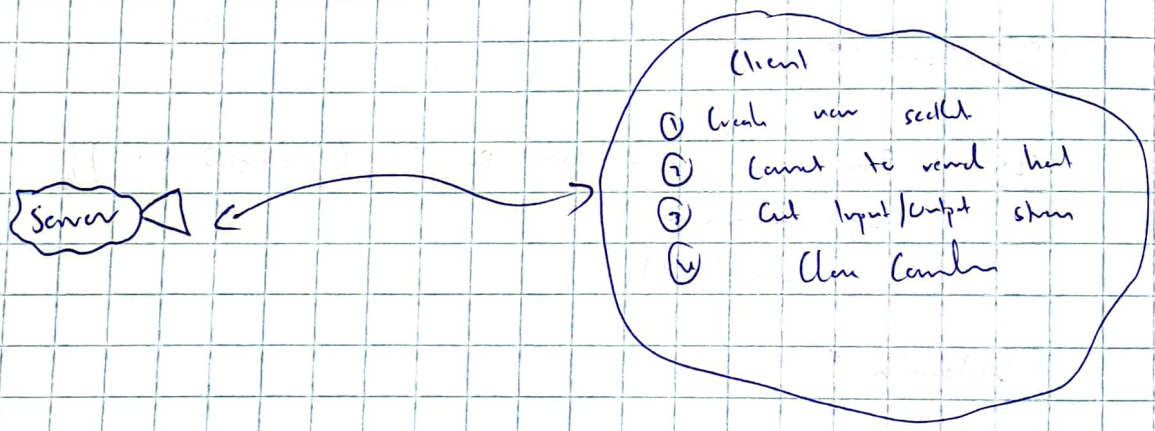
BufferedReader reader = new BufferedReader(in);

Section 4: Java Socket

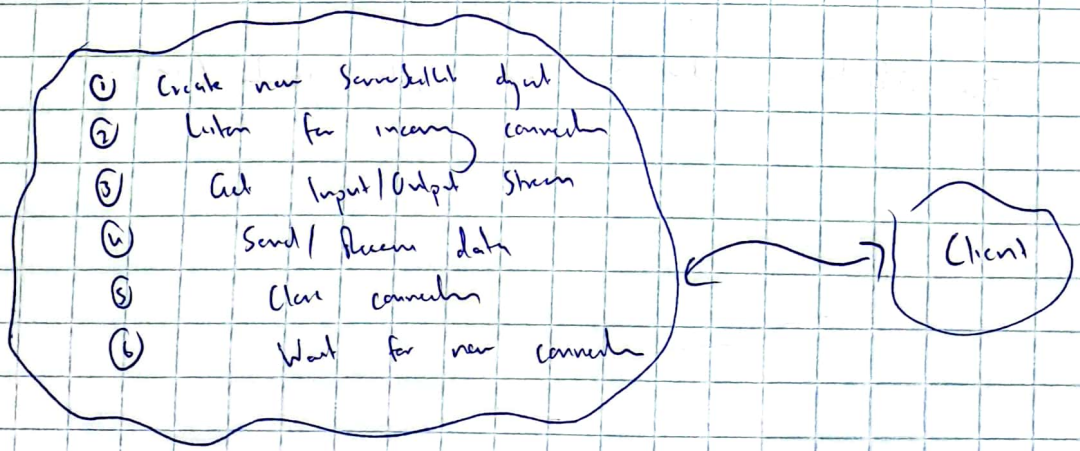
Introduction

Socket can perform 7 basic operations:
- connect to a remote machine
- send data
- receive data
- close a connection
- bind to a port
- listen for incoming data
- accept connections from remote machines on its bound port

Socket Class



Server Socket Class



Local Port Scanner

- ServerSocket constructor fails in port number
- to check port in cmd - netstat -a

TCP Server

Eg:

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

        ServerSocket serverSocket = new ServerSocket (6666);
        System.out.println (" waiting for clients ... ");
        Socket socket = serverSocket.accept ();
        PrintWriter out = new PrintWriter ( socket.getOutputStream (), true );
        out.println ( " Hello Client! " );
        BufferedReader input = new BufferedReader (
                                new InputStreamReader ( socket.getInputStream()));
        String clientInput = input.readLine ();
        System.out.println ( clientInput );
            input.close ();
            out.close ();
            socket.close ();
```

InetAddress

Eg:

```
InetAddress addr = InetAddress.getLocalHost ();
System.out.println ( addr.getHostAddress ());
        "   "   "  (addr.getHostName ());

InetAddress addr2 = InetAddress.getByName ( "google.com" );

Socket soc = new Socket ( addr, 6666 );
```

TCP Client

Eg:

```
try {
InetAddress serverAddr = InetAddress.getByName("localhost");
System.out.println("server ip addr" + serverAddr.getHostAddress());
Socket socket = new Socket(serverAddr, 9999);
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
BufferedReader input = new BufferedReader(new InputStreamReader(socket.
                                          socket.getInputStream()));
System.out.println(input.readLine());
out.println("hello server!");
  input.close();
    out.close();
     socket.close();
```

Remote Port Scanner

- scans remote host for open tcp port
- Exampl in Download.

Ping

Eg:

```
try {
    String hostAddress = "192.168.0.123";
    InetAddress host = InetAddress.getByName(hostAddress);
    System.out.println(host.isReachable(1000));

Process p = Runtime.getRuntime().exec("ping" + hostAddress);
BufferedReader inputStream = new BufferedReader(new InputStreamReader(p.getInputStream
String commandOutput = "";
boolean isReached = true;
while ((commandOutput = inputStream.readLine()) != null) {
    if (commandOutput.contains("Destination host unreached")
    {
        isReached = false;
        break;
    }
}
```

```
if( "Reachable" ) { System.out.println("Host is reachable!"); }
else { System.out.println("Host is not reachable!"); }
```

Validate IP address

Eg:

```
public boolean validIPAddress(String ip)
{
    String[] numbers = ip.split("\\.");
    if( numbers.length != 4) return false;

    for( String str : numbers )
    {
        int i = Integer.parseInt(str);
        if( (i<0) || (i>255)) return false;

        return true;
    }
```

URL and Web scraping

Eg:

```
URL url = new URL("http://www.yahoo.com/q?s=cricket");
URLConnection urlConn = url.openConnection();
BufferedReader in = new BufferedReader(new InputStreamReader(
                                    urlConn.getInputStream()));

String inputLine;
String pattern = "<span id=\"yfs_l84_cricket\">(.+?)</span>";
Pattern r = Pattern.compile(pattern);

while ((inputLine = in.readLine()) != null)
  {
      if(inputLine.contains("yfs_l84_cricket"))
      {
          Matcher m = r.matcher(inputLine);
        if (m.find()) {
            System.out.println(m.group(1));
        }
      }

    in.close();
```

Section 5 : UDP programming

UDP client

Eg :

```
try {
    DatagramSocket clientSocket = new DatagramSocket();
    byte[] sendData = new byte[1024];
    byte[] receiveData = new byte[1024];

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

    String stringSendData = "Hello Server!";
    sendData = stringSendData.getBytes();
    DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverAddress, 9
    clientSocket.send(sendPacket);
    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
    clientSocket.receive(receivePacket);
    receiveData = receivePacket.getData();
    String stringReceiveData = new String(receiveData);
    System.out.println("FROM SERVER: " + stringReceiveData);
    clientSocket.close();
}
```

UDP Server

Eg :

```
try {
    DatagramSocket socket = new DatagramSocket(9876);
    byte[] receiveData = new byte[1024];
    byte[] sendData = new byte[1024];

while(true) {
    DatagramPacket packetReceive = new DatagramPacket(receiveData, receiveData.length);
    socket.receive(packetReceive);
    String sentence = new String(packetReceive.getData());
    System.out.println("RECEIVED" + sentence);
    String stringData = "hello client!";
    sendData = stringData.getBytes();
```

```
InetAddress clientIPAddress = receivePacket.getAddress();
int clientPort = receivePacket.getPort();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
                                               clientIPAddress, clientPort);
serverSocket.send(sendPacket);
```

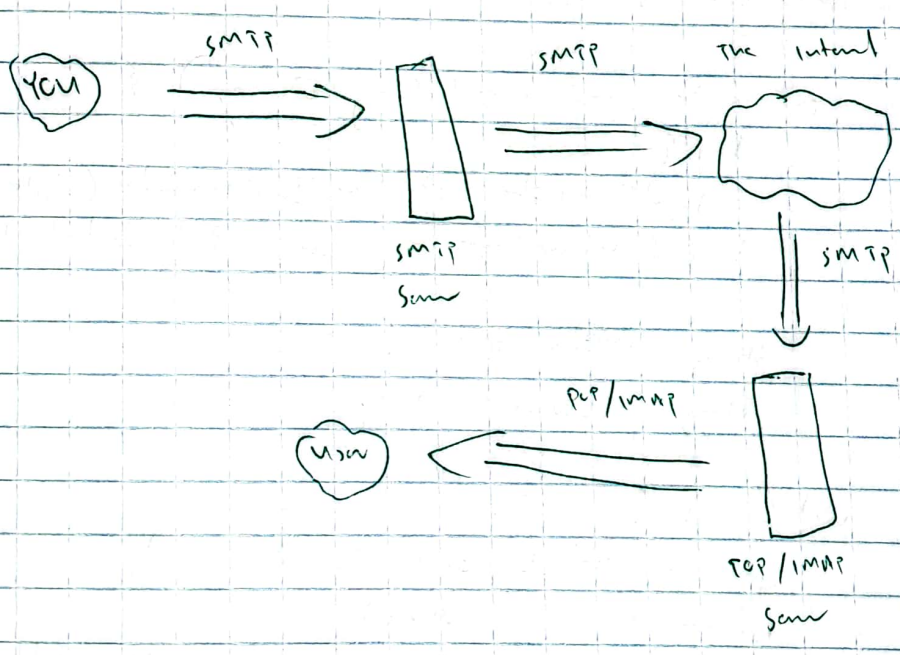
Section 6 : Send Email using SMTP protocol

SMTP protocol

- TCP/IP protocol used in sending and receiving email
- default port : 25

bob@gmail.com

Domain



Section 7 : File Transfer Application

Client Server

Cs:

```
try {
    ServerSocket serverSocket = new ServerSocket (7000);

while ( true ) {
    Socket clientSocket = serverSocket.accept();
    ClientThread clientThread = new ClientThread( clientSocket );
    clientThread.start();
    }
```

Client:

```
public class Client {

    public static void main ( String [] args ) {
        try {
            InputStreamReader in = new InputStreamReader ( System.in);
            BufferedReader reader = new BufferedReader(in);

            String ipAddress = "";
            String fileName = "";

        while (true) {
            System.out.println(" Enter IP address ");
            ipAddress = reader.readLine();
            InetAddressValidator validator = new InetAddressValidator();
                isValid = validator.isValid(ipAddress);
                }

        System.out.println(" Enter a file name ");
        fileName = reader.readLine();
```

```
Socket socket = new Socket ( ipAddress, port );
InputStream inputByte = socket.getInputStream();
BufferedInputStream input = new BufferedInputStream (inputByte);
PrintWriter out = new PrintWriter ( socket.getOutputStream(), true );

out.println ( fileNum );
int code = input.read();

if ( code == 1 ) {
    BufferedOutputStream outputFile = new BufferedOutputStream (
                new FileOutputStream ( "D:\\download\\" + fileNum ));
    byte[] buffer = new byte [1024];
    int byteRead = 0;

    while ( ( byteRead = input.read(buffer) ) != -1 ) {
        System.out.println( "." ); // each . a download notice
        outputFile.write ( buffer, 0, byteRead );
        outputFile.flush();
    }
    System.out.println();
    System.out.println ("File " + fileNum + " successfully downloaded!");
}
    else
{
    System.out.println ( "File not on server" );
}
    catch ( Exception e ) {
        System.out.println ( e.toString() );
    }
}
}
```

ClientThread

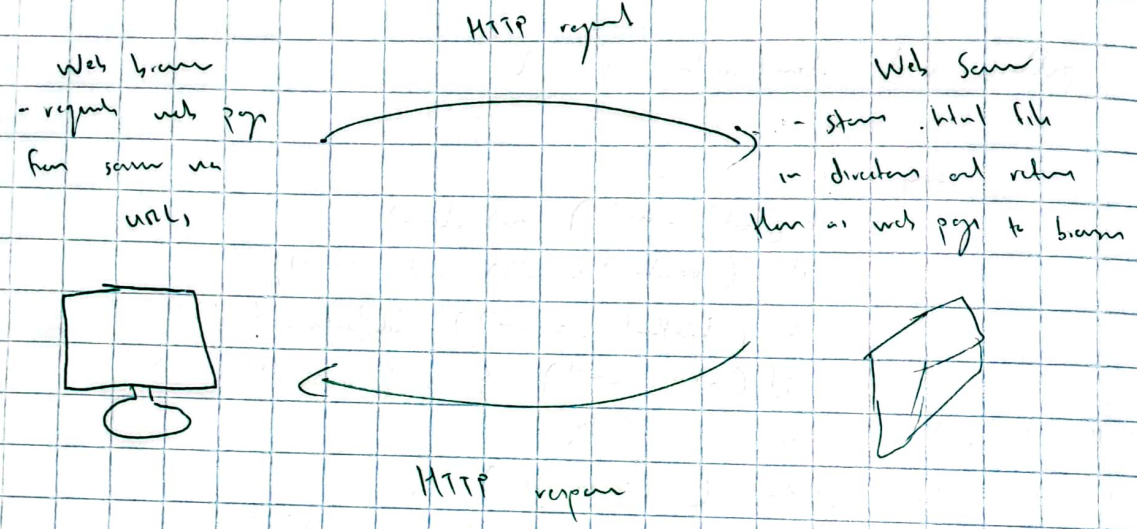
```
class ClientThread extends Thread {

    private Socket socket;
    private BufferedReader reader;
    private BufferedOutputStream out;
    private BufferedInputStream fileReader;

    public ClientThread(Socket socket) {
        this.socket = socket
    }

    public void run() {
        try {
            reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new BufferedOutputStream(socket.getOutputStream());

        String fileName = reader.readLine();
        System.out.println("File name: " + fileName + " requested by " + socket.getInetAddress().
                                                    getHostAddress());
        File file = new File(fileName);

        if (!file.exists()) {
                byte code = (byte) 0;
                out.write(code);
                    clean(out);
                    }

            else {
                out.write((byte)1);
                fileReader = new BufferedInputStream(new FileInputStream(file));
                byte[] buffer = new byte[1024];
                int bytesRead = 0;

        while((bytesRead = fileReader.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
                    out.flush();
                    }
```

```
        close();
    }
}
catch (Exception e) {
    System.out.println(e.toString());
}
}

public void close() {

try {
    if (out != null) out.close();
    if (reader != null) reader.close();
    if (fileReader != null) fileReader.close();
    if (out != null) { socket.close(); }
}
catch (Exception e) {
    System.out.println(e.toString());
}
}
```

Section 8: Basic web server application

HTTP server

- set of rules for browsing files on world wide web
- communication, media independent and stateless.

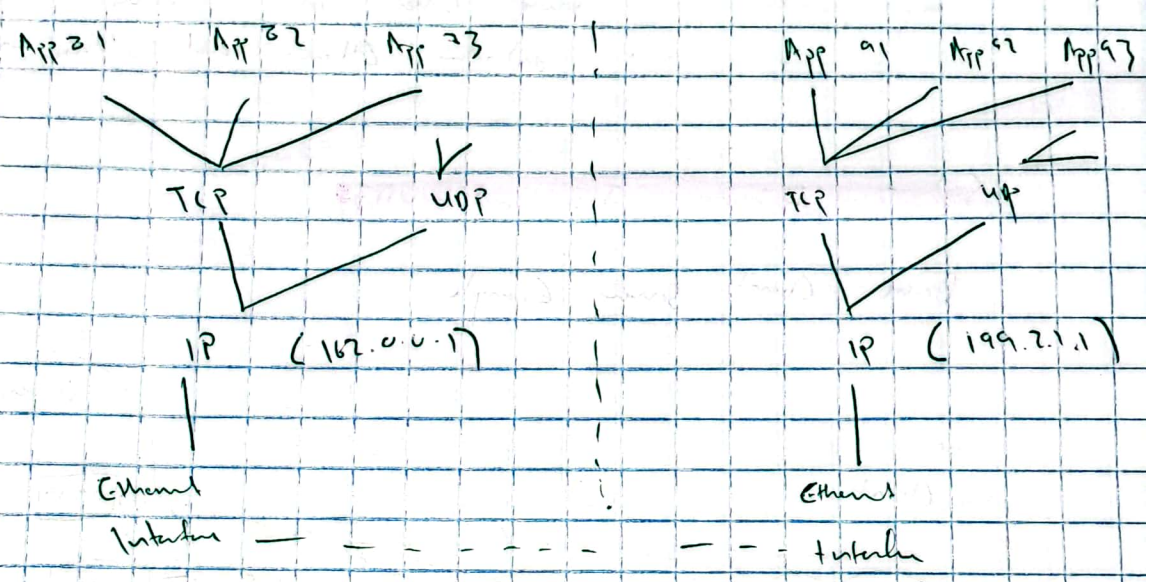


UDEMY : Java : Socket Programming Simplified

Section 1 : Introduction

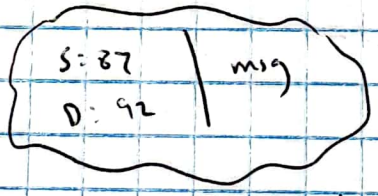
Understanding message transfer between applications

Example : Sending a msg from App 82 to App 92



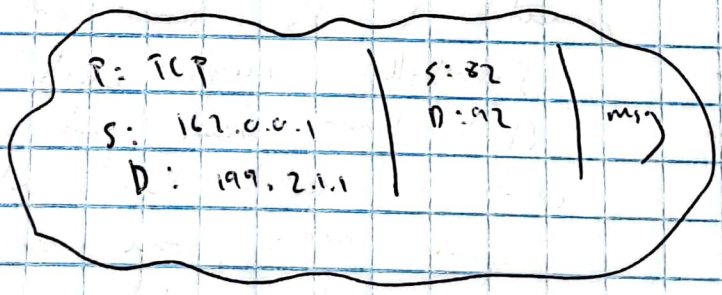
- each app identified by a 16-bit port number

TCP :



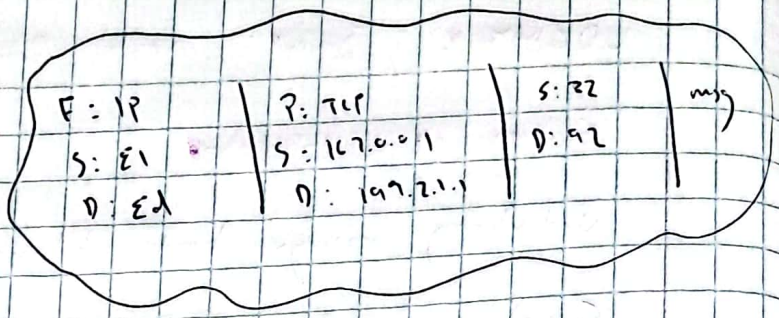
- adds source and destination port number

IP :



- adds type of protocol
- source IP and destination IP

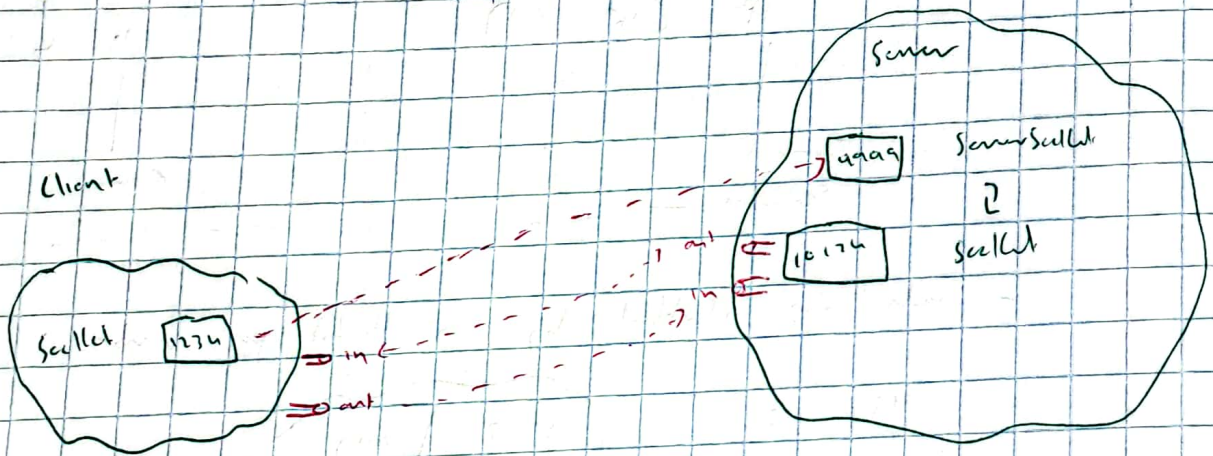
Ethernet Interface :



- adds : • from type is IP
  • source Ethernet card address
  • destination Ethernet card address → a variable

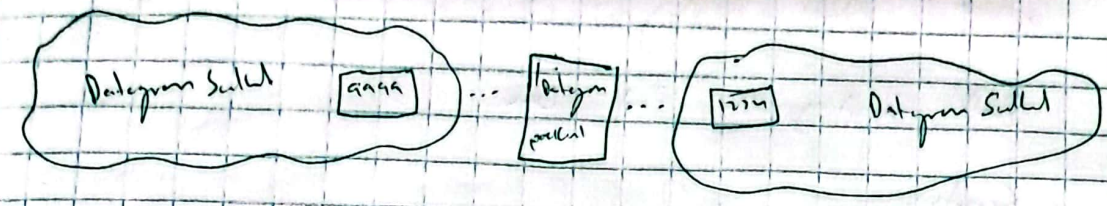
Section 2 : TCP and UDP

Basic Client Server Example



- Clients can use Server port number (9999) to get connected to M Server and communicate
- ServerSocket accept client connection
- Client knows about port number 9999, Client will create a socket that will send a connect request to port 9999, once connection is established, a separate socket is created (10134), that port number is communicated with M client,
- Client has 2 stream : input and output
- Server socket also has 2 stream : input and output

User Datagram Protocol



- So for Socket connection with a TCP protocol, you need a connection
- UDP is connectionless, but you need endpoint
- Example in Intellij

Section 3 : HTTP

HTTP interaction explained

Browser