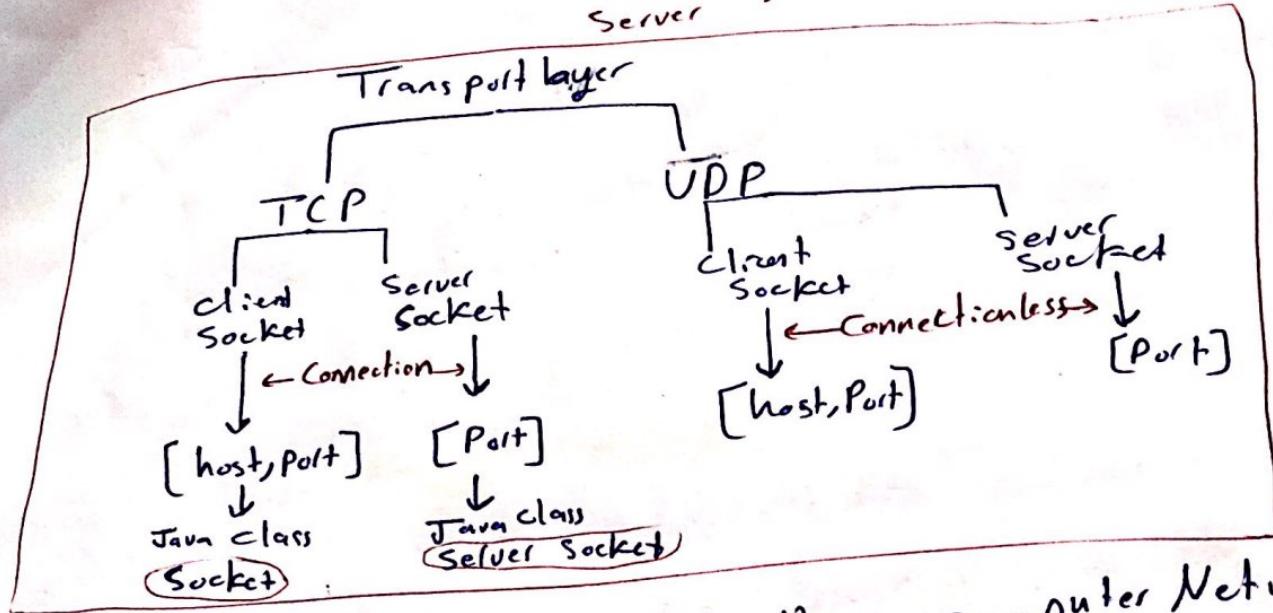
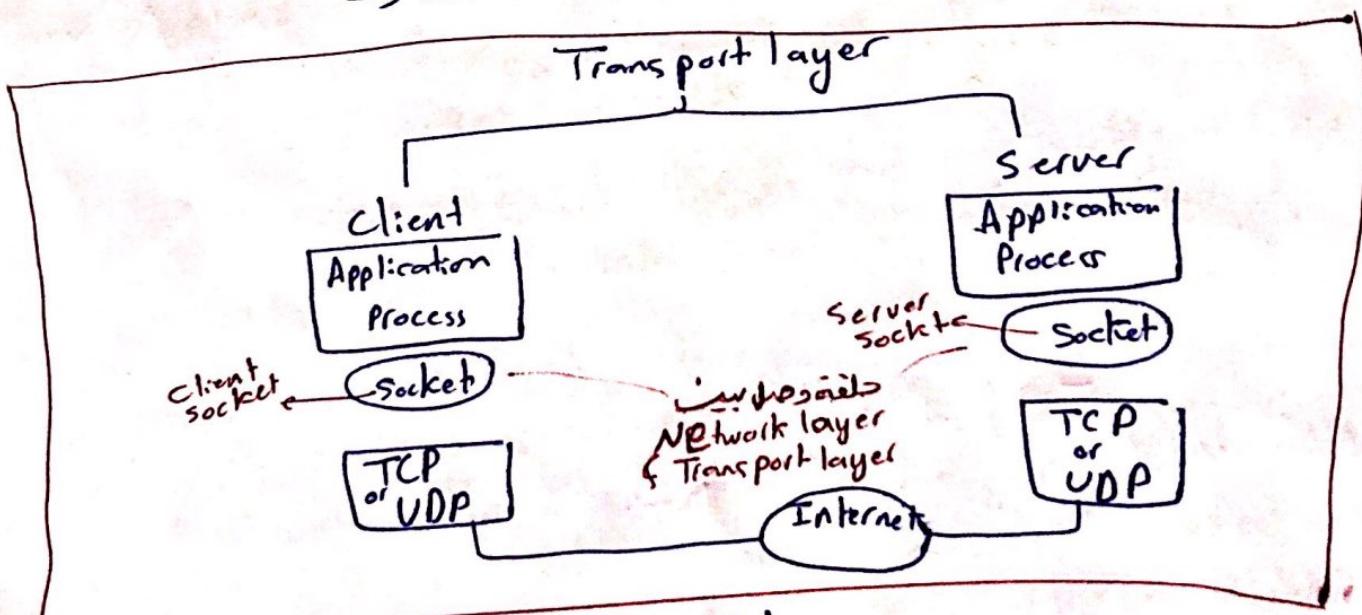


Transport layer: TCP/UDP
 كلما استخدم الاتجاه الرابع هنا أو من خلال من خلاله فـ client > Socket > Server
 وهناك نوعين من الـ socket



Network socket: as software structure within a computer Network,
 that server as an endpoint to send and receive data.
 فوهر كنها بجهل او مستقبل راتا
 ⇒ Its An API (Application Programming Interface)



Note: Ports
 محدود (0 - 1023)
 (1024 - 65535) مفتوح (cont-value)

مَرَادَه صفحته مُوقَع Yahoo <--
بِنَانَقْل بِرْنَاج لَد Client [بنانقر] <--
بِنَانَقْل Connection من خَلَال الـ Socket <--

```
import java.net.Socket;
import java.io.IOException;
import java.io.InputStream & InputStreamReader;
import java.io.BufferedReader;
import java.io.OutputStream;
import java.io.PrintWriter;
public class SocketRead {
    public static void main() throws IOException {
```

```
String host = "www.yahoo.com";  
Socket socket = new Socket(host, 80);
```

```
String host = "www.google.com";
Socket sc = new Socket(host, 80);
System.out.println("HTTP/1.1 POST /index.html");
System.out.println("User-Agent: Java/1.8.0_101");
```

ـ يُعرف أنه يُحيط بالـ (HTTP) - وتحت رقم الـ Port للـ HTTP * NOT
ـ يستخدم أو تُستخدم (Socket Sc) = new Socket (IP, Port)
ـ يُطلق وين الخطأ بازبط .
ـ يُحْلَل بـ catch & try افضل من الـ Exception
ـ لا نستخدم الـ Socket (Sc) ① لا نستخدم الـ Socket (Sc) ② نأخذ الـ data من خارج getOutputStream ()

getOutputStream();
outputStream OS = SC.get OutputStream();
socket (Yahoo) دفعطه لـ outputAll

boolean autoflush = true ;
بنقلف المسار أو توصيات

لـ (لتـنظـيف المسـار) اطـبع المـاخـوذ مـن الـ خـلال الـ Stream
ليـشـا مـتـخـصـصـا بـ autofl~ush يـكـيـمـيـلاـ مـسـاحـةـاـ لـ حـفـزـاـ نـاكـهـ بـ intـاـ

لـ [لـ تنظيف المسار] اطبع الماخوذ من الـ **Socket** من خلال الـ **output stream** **Print writer** **out** new **Print writer** (OS) لـ [لـ تنظيف المسار] اطبع الماخوذ من الـ **Socket** من خلال الـ **output stream** **Print writer** **out** new **Print writer** (OS)

مبايعة حالات
نحو طباقتهم على
ارسال طلب $\left[\begin{array}{l} S.O.P(\text{Get}/\text{Http } 1.1) \\ S.O.P(= \text{Host} + \text{host} + 80) \end{array} \right]$

S.O.P. \Leftarrow time Print writer

الفراء

للحملة

Input Stream is = (sc). get InputStream(); // user للـ socket

Input Stream Reader isr = new InputStreamReader(is);

BufferedReader br = new BufferedReader(isr);

String Builder sb = new String Builder(8192);

فليس بهذا تترد [السعة] بخزن فتح 8KB بعدين ينزل مره ومه [تسريجي كلبيه - الفراء]

لا زالت أحرف لحالها بتصير 1-6 خطوه

while (i != -1) {

- ضل لف لصت توصل نهاية للف -

هون حيلت لـ ASCII

← i = br.read();

صون رجعناها حرف // ن (char)i

sb.append((char)i);

}

loop (sb.toString());

sc.close(); // Socket

os.close(); // OutputStream

أغلقت البرنامج

بسطر يستغل

ابا خلاق ا اختياري هنا

اجاري هنا

Ports type:

- TCP, UDP ports [0 - 1023] "client socket"
- free Ports [1024 - 65535] "server socket"
- int value [0 - 65535] مخواه بين 50هـ

[0 - 1023] إلى عندي الموجوين local Host go TCP ports //

import java.net.UnknownHostException; socketException

import java.net.Socket;

import java.io.IOException;

Public class ViewTCPPorts

```

    Public static void main(String[] args) {
        String host = localHost // 127.0.0.1
        Socket s; // client socket
        for (int port = 1; port < 1024; port++) {
            try {
                s = new Socket(host, port); // low port with socket
            } catch (IOException e) {
                System.out.println(ex);
            }
        }
    }

```

ابا خلاق اختياري هنا

- 2 types of exception
- UnknownHostException:
 - IOException,

2) Multi-task

2) Multi-task

كود بعي اخر في المنهجية //

```
import java.net.ServerSocket;
import java.io.IOException;
public class ViewFreePorts {
    public static void main() {
        ServerSocket s0;
        for (int port = 1024, port <= 65535; port++) {
            try {
                s0 = new ServerSocket(port); // high Port with socket
                System.out.println("There is free port " + port);
                s0.close();
            } catch (IOException ex) {
                System.out.println(ex);
            }
        }
    }
}
```

نوع واحد لـ IOException ①

داخلي لـ IOException

في الـ TCP Ports موجودات الـ LocalHost [0 - 1023] ملخص البروتوكول
 البروتوكول يفتح منفذين على الماكنة
 1. LocalHost [0 - 1023] ملخص البروتوكول

2. TCP Ports

3. try

4. if (Port + " " == "host") {

5. S = new Sock ("host", Port)

6. for i <= 1024 {

7. if (S.bind ("", i) == 0) {

8. System.out.println ("Port " + i + " is free");

9. break;

10. }

11. }

12. }

FTP : file Transfer Protocol

"FTP + TCP"

⇒ its a Network Protocol for translating files between computer over (TCP/IP) connections consider as application layer Protocol (TCP/IP على الحواسيب بناء على نقل الملفات بين)

: خوارقها / ميزاتها

✓ 1. Backup: Copy data from one location to secure Server (نسخة احتياطية في التركيز والبنوك يكون يومياً بمعنى يومياً (نحو)) نقل البيانات للسيرفر

✓ 2. Replication: similar to Backup but it more resilience and higher availability (النسخة الاحتياطية من مزدوجة بين باباokus لكن بدرجة أعلى)

✓ 3. data badning: download from Server to Client

from Client to Server.

✓ 4. Data uploading: send data from Client to Server.

TCP : Transport layer protocol

(Transmission control Protocol)

FTP : Application layer Protocol (file Transfer Protocol)

UDP : Transport layer Protocol (User Datagram Protocol)

- O (load)

هذا السيرفر
Client N

+ download FTP Server

```

import java.io.File; // جريدة
import java.io.FileInputStream; => FTP بخضوره
import java.io.OutputStream;
import java.io.InputStream;
import java.net.ServerSocket; //server
import java.net.Socket; //client

```

لأنه ينقل مثبات FTP

-(load) download from server
upload to server

public class FTP Server {

الملاصق مصدره إلى السيرفر بعنوان client

client

Public static void main() throws IOException {

Server

Server socket ssc = new ServerSocket(4000);
TCP port

Client

socket socket = ssc.accept();
على رجلين أو Client, Server

File file = new File("mah.txt"); // عز البرقر

InputStream in = new FileInputStream(file); // انتاد قبر

Output Stream out = socket.getOutputStream(); // يعنى الملف من الماكنة

byte[] b = new byte[20 * 1024]; // حجم معين

int i; // طالعني داتا فيها و الكتبها

while ((i = in.read(b)) > 0) {

out.write(b, 0, i); // read with output write

outputStream.close(); // سكرن اول

- بدى ادخل الفايل من او input stream
- و اخرج مع outputstream او output socket.getOutputStream()
- معتبرات او Client socket

inputStream.close(); // سكرن اول

clientSocket.close(); // سكرن اول

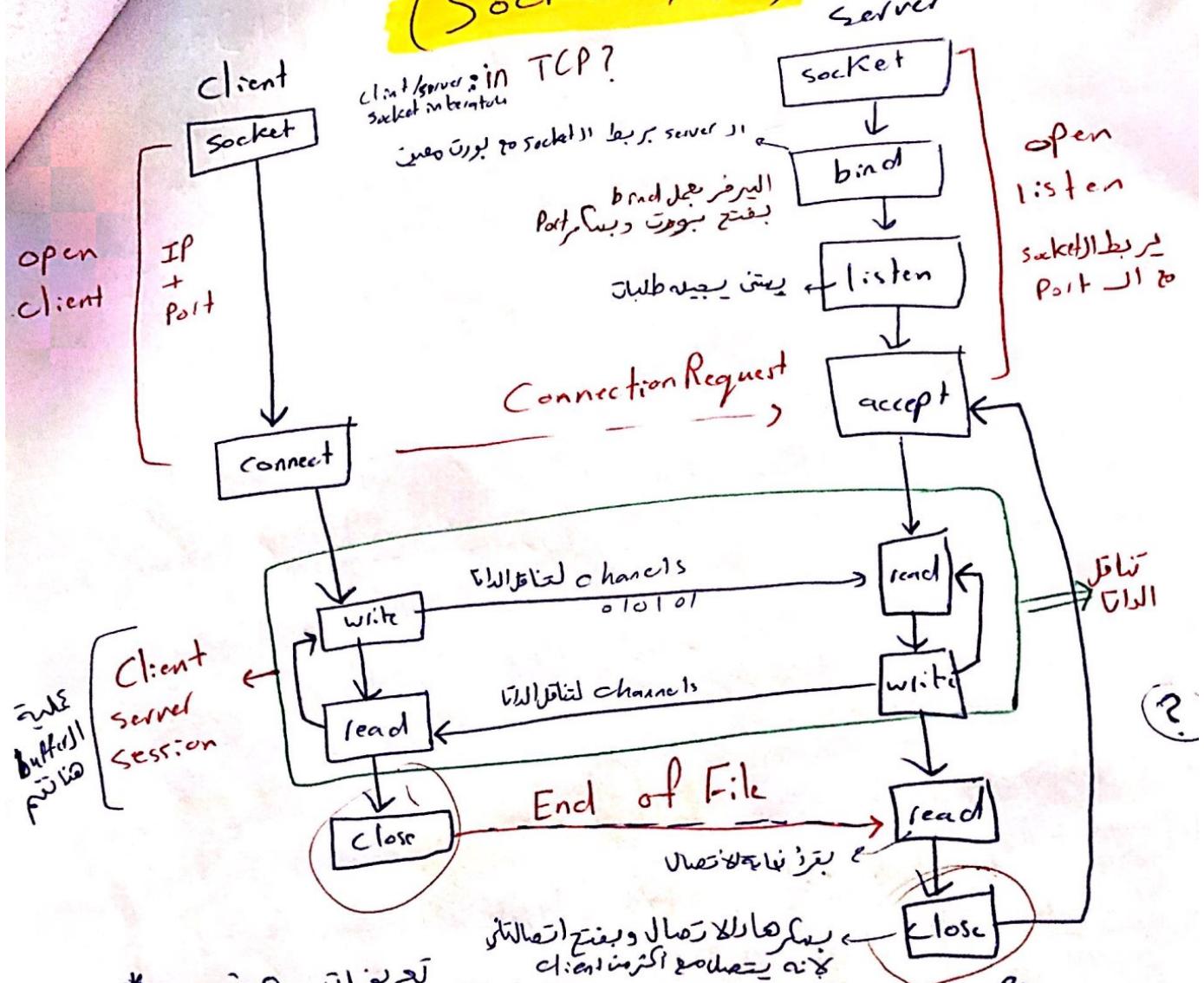
serverSocket.close(); // سكرن اول

هذا هو اختيارات الإختلاف
لكنه يفضل أن نلتقط

all output files to one output

^

Multithreading (Socket API) (السكتاتيون API) (API لبرمجة السكتاتيون)



* Stream تدفق

1. A Stream is a sequence of character that flow into or out of a process.

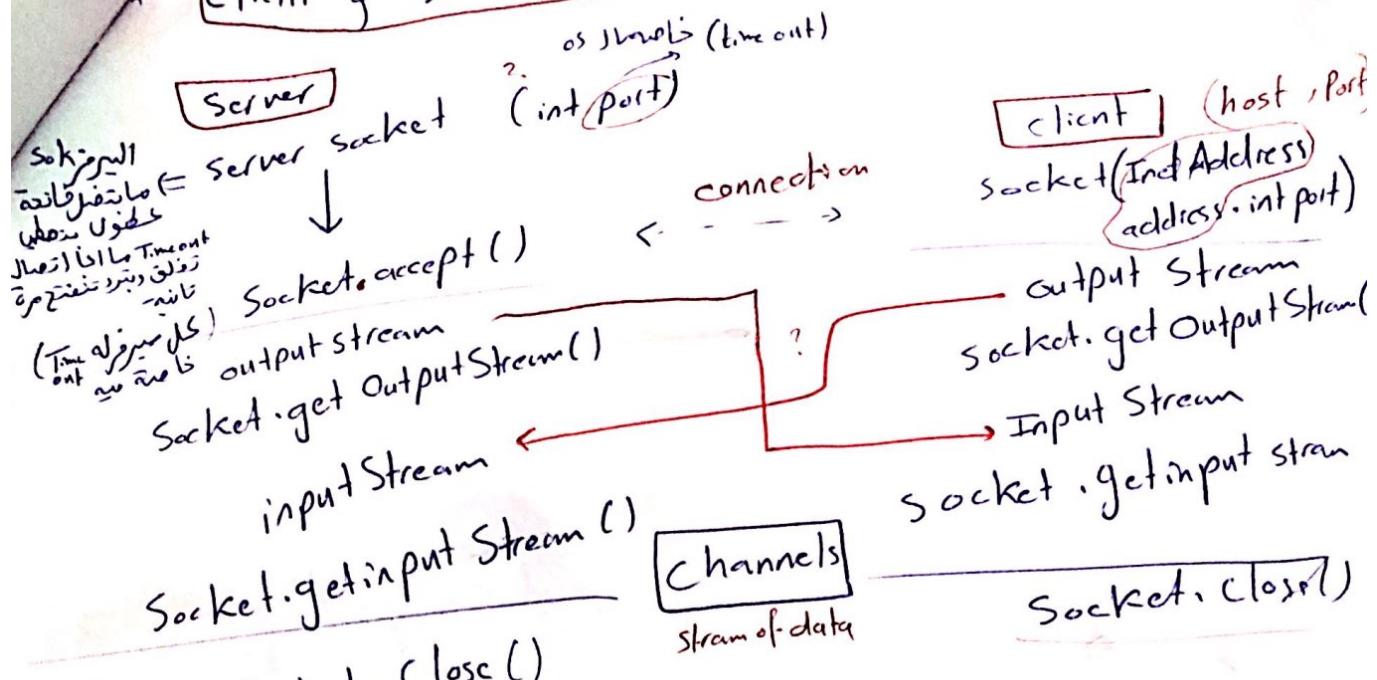
2. An input Stream: attached to some input source for the process ex: keyboard or socket.

3. An output Stream: attached to an output source for the process ex: monitor, socket

4. channel: represent a stream of data between applications
ex: 0110011001

5. buffer: work with channel to process data: Client

Client & server data Streaming



Socket.close() stop
 مفهوم وظيفة close() في سوكيل
 سوكيل

FTP Client : import java.net;
 download

Public Class **FTPClient** {
 public static void main (String [] args) throws IOException

socket [Socket] = new Socket ("127.0.0.1", 4000);
Port

InputStream [in] = socket.getInputStream();
 OutputStream [out] = new FileOutputStream ("math.txt");
انتهاء الملف في المكتبة

byte [] b = new byte [20 * 1024];

int i ;
 while (i = in.read (b)) > 0) {

out.write (b, 0, i);
offset بعد بسيط يمني

}

out.close ();
 in.close ();
 socket.close ();
نقطة انتهاء

socket
 لازم servell
 او " " ويعمل
 listen بعد
 client يستقبل من

TCP / UDP:

* TCP:

(Transmission control Protocol) is a communication protocol that enables application programs and computing devices to exchange messages over a network. It's designed to send packets across the internet & ensure the successful delivery of data & message over networks.

تبارد الرسائل / على مبنية على
Connection oriented

بروتوكول يعتمد على التطبيقات والاجهزه من تبارد الرسائل عبر انترنت ، لفتن وصولها باتجاه

* UDP: (User Datagram Protocol) is a communication Protocol that transmits independent packets over the network with no guarantee of arrival & no guarantee of the order of delivery.

رسائل عبر الانترنت دون حفظ وصولها او ترتيب دخولها

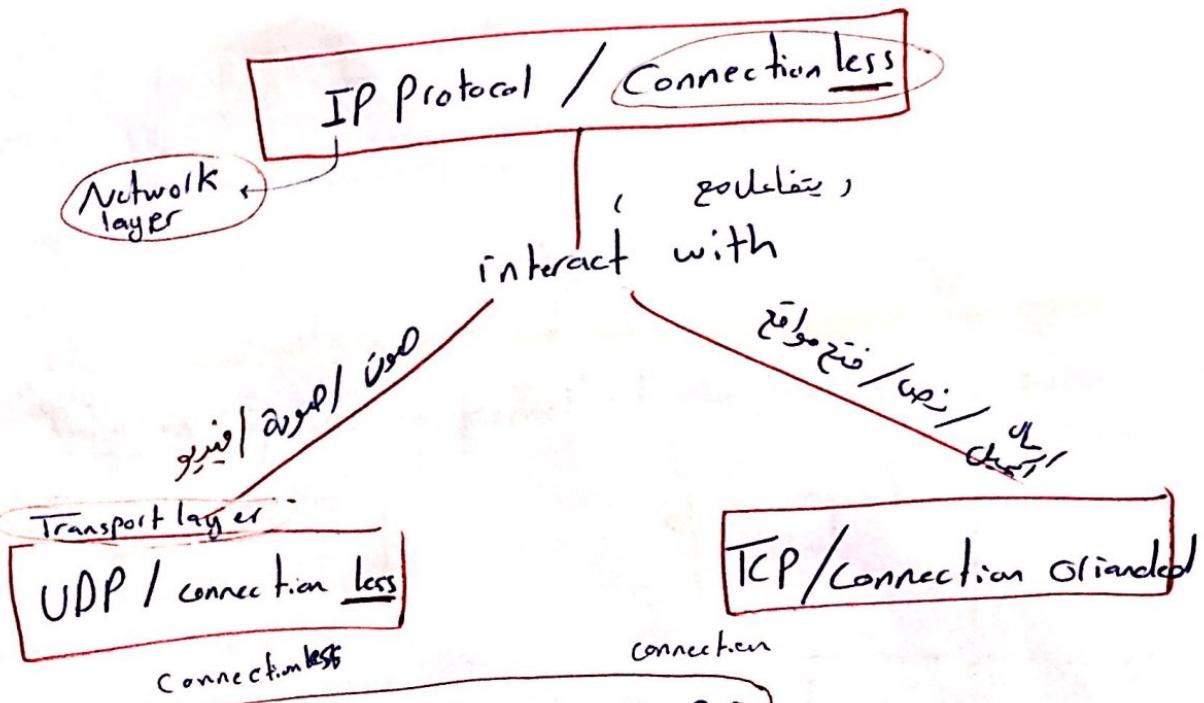
TCP	UDP
Connection oriented	connection less
Reliable	Unreliable
flow control	no flow control
بنك الموارد Congestion Control (inorder)	No Congestion control out of order
Used with HTTP , FTP Telnet , SMTP , POP3	media streaming , DNS ↓ Domain name server
slow transfer Protocol بطء في نقل البيانات و تأمينها	Rapid datagram Protocol اخذ بقدر انتقال البيانات ولستالم ولا تأخذه وصول المسبح
Slower	faster

Why use UDP: ??

TCP vs UDP

it's important to understand that the lack of overhead and latency can make it significantly faster than TCP.
رسالة ارسل الفيديوهات عالى توصيل بشكل أسرع

في بعض الحالات يُجبر تقطيع بعض الـ (صور، الفيديو) لفترة زرقاء بسيطة أو الصور تتأخر ببعضها



How IP work with TCP?

From sender to receiver IP datagram encapsulates TCP segment & then IP deals with addressing on the network
(TCP/IP System)

TCP يختلف عن IP في أن TCP يحتوي على البروتوكول والعنوان

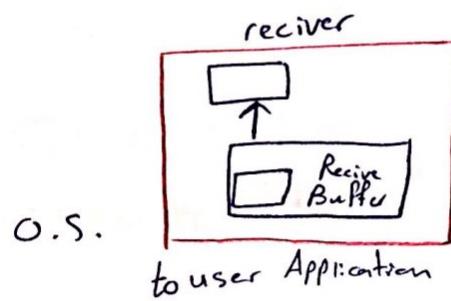
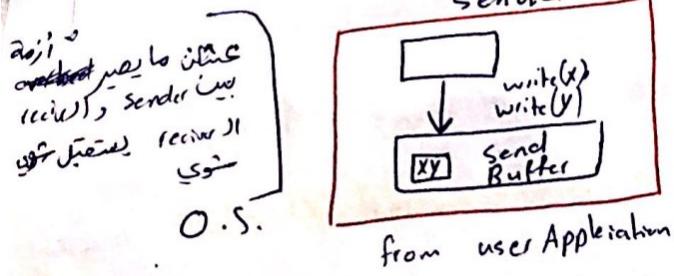
IP in Network layer \Rightarrow (Packet) (datagram)
UDP/TCP in Transport layer \Rightarrow (Segment)

Transport layer
↓
Network layer

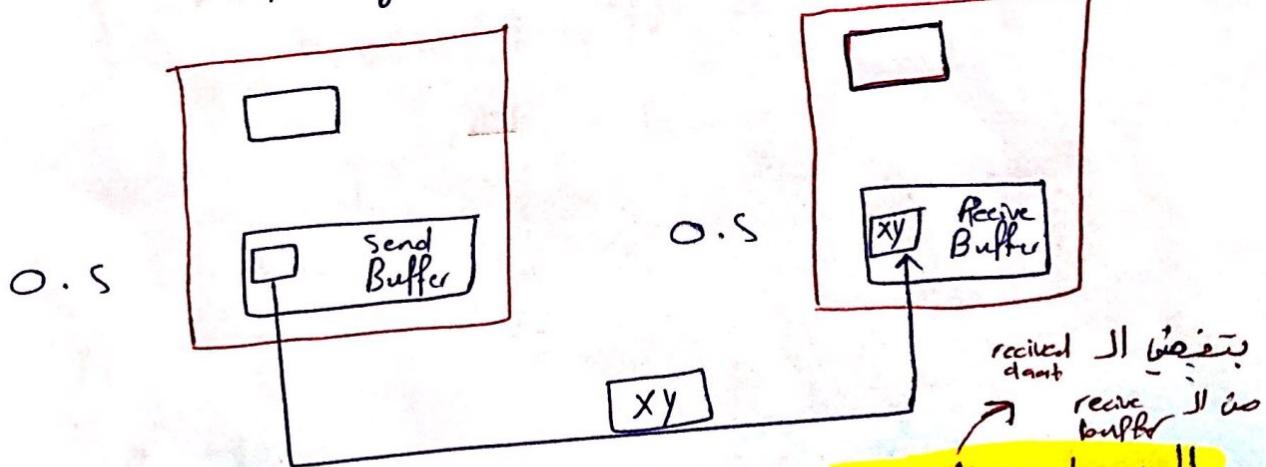
Flow Control in TCP:

Send & receive buffers:

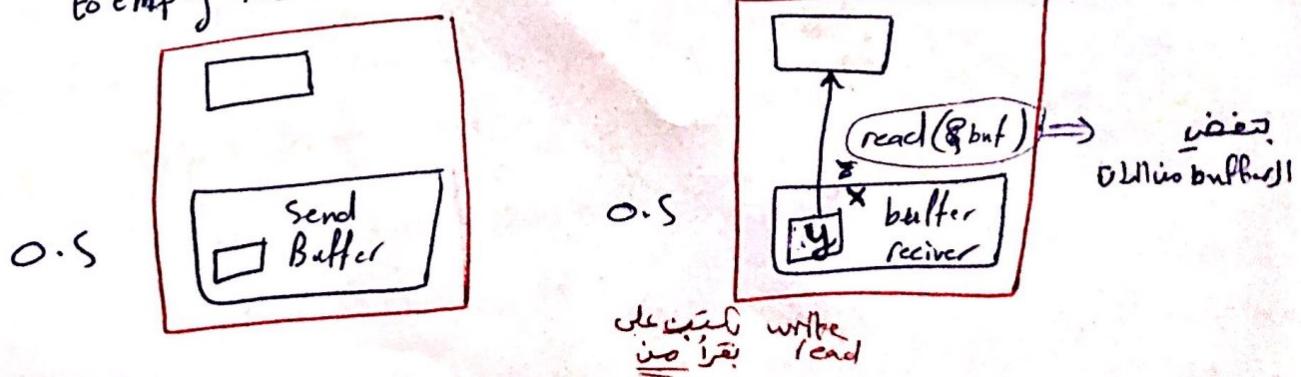
- when a User application sends data to another user application, The data is first stored in a send Buffer inside operating System kernel :



- The receive data is stored in a receive buffer inside The operating System kernel of the receiver Computer



- The Receiving application can use the `read()` system call to empty the Received data from the receiver buffer -



Upload:

server N client JI io ← send data

```

public class FTP Server {
    server socket ssc = new ServerSocket(4000);
    server socket ssc = new ServerSocket(4000);
    s.o.p ("listen to client connection");
    "السرفر يرسل اتصال الى العميل"
    Socket socket = ssc.accept();
    File file = new File("Mht.txt");
    Input Stream in = new FileInputStream(file);
    Output Stream out = new FileOutputStream("uploadfile.txt");
}

```

للتغیر هنا

byte [] b = new byte [20 + 1024]

byte [] b = new byte [20 + 1024]

while (i < n) {
 in.read(b);
 out.write(b, 0, i);
 i++;
}

out.close();
 in.close();
 socket.close();
 ssc.close();

8.

* FTP Client:

```

public class FTP Client {
    Socket socket = new Socket("127.0.0.1", 4000);
    File file = new File("uploadfile.txt");
    Input Stream in = new FileInputStream(file);
    Output Stream out = socket.getOutputStream();
    s.o.p ("File upload");
}

```

للتغیر هنا

TCP Client and Server:

connect to one server

```

import java.net.ServerSocket
import java.net.Socket
import java.io.InputStream
import java.io.BufferedReader
import java.io.InputStreamReader
import java.io.PrintWriter
import java.io.IOException

```

(echo Server) برنامج للتأكد أن المدخلات

برنامجه لارساله بين client و server بستخدمو IP

Public class **TCP Server** سيرفر يستخدم Socket

Private static ServerSocket SerSock

Private static final int Port = 1234 [مفروض كلام رقم غير معين]

كم البورت الذي يفتحه الميرور نفسه حال

Public static void main() { } ... الفايدة بهير يعرف دين بالربط صارى مسلكة وسوبي Try / catch a

always ON ---> دلائل على Server

S.o.P("Connecting to port") ;] جمل توصيفية

S.o.P("Server starts") ;] اتصال بين الاتصال والميرور

S.o.P("Waiting for client") ;] الميرور ينتظر الاتصال وبعد

الميرور فتح اتصال وبذل

أولاً سبلاج الـ Port ، فتح اتصال ، فاخذ متن (Step A)

```
try {  
    SerSock = new ServerSocket( Port );  
}
```

ثانياً ما عرف يثبت له البورة ;
عذران يكون الرقم ممحوز / غلط

```
} catch ( IOException e ) {  
    S.O.P ( "unable to connect Port" );  
    System . exit ( 1 );  
}
```

بعد ماclient يبعث رسالة يستقبل السيرفر رسالة ويعالجها
وبرمج يرسل رسالة انه مستقبلها

2 do {
 ClientHandle();

طوال ما السيرفر مشغّل

السيرفر يفعل بسأله client
يفتح وجوه TCP port

```
} while ( true );
```

الثاني main كذا
try {
 ClientHandle();
 Socket link = null;

عنده من متصل client وله فقط

step B
try {
 link = SerSock . accept ();
 S.O.P ("client Accepted");
 Input Stream input = link . getInputStream ();
 Input Stream Reader isr = new Input Stream Reader (input);
 BufferedReader br = new BufferedReader (isr);
 output // PrintWriter output = new PrintWriter (link . getOutputStream (),
 flush = true);
 input . readLine ();
 input . close ();
 output . close ();
}

```

String message = null;
int numMessage = 0;
try {
    do {
        message = br.readLine();
        numMessage++;
        System.out.println("Received message " + message);
    } while (!message.equals("close"));
    output.println("Message received");
    br.close();
    output.close();
} catch (IOException e) {
    System.out.println("unable to close connection");
    System.exit(1);
}
}

// ClientHandle Method();
// Class TCP Server

```

عدد الرسائل
 عناوين تتنفس Reader
 قراءة الرسالة على شكل input من client
 th.
 numMessage++
 System.out.println("Message " + numMessage + ":" + message);
 لـ هنا بطبع رقم الرسالة و ايضـ هـ الرسالة المبقـونـة مـنـاـ (client)
 مـنـاـ حـكـمـهـاـ اـلـيـخـرـهـاـ مـنـاـ "close"
 على اعتبار انه كل طـرـفـ ، مـنـاـ الرسـلةـ مـنـاـ clientـ ، السـرـفـرـ مـنـاـ clientـ ،
 مـنـاـ عـنـاـ رسـلـهـاـ مـنـاـ "close"
 تطـبـيـكـ مـاـ يـتـبـعـهـاـ مـنـاـ
 output

while (!message.equals("close")) {
 message = br.readLine();
 output.println("Received message " + message);
 }
 output.println("Message received");
 br.close();
 output.close();
}

catch (IOException e) {
 System.out.println("unable to close connection");
 System.exit(1);
}
}

// ClientHandle Method();
// Class TCP Server

TCP Echo Server System

```

import java.net.*;
import java.io.*;
import java.util.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;
import java.util.Scanner // لبيان المدخلات
import java.util.Scanner // لبيان المدخلات
    
```

Client طرف الارسال *
for input (رسالة ارسال)
for output (رسالة ارسال)
th
n

Public Class TCP Client

```

public static final int Port = 1234;
private static InetAddress host;
private static String hostIP; // لبيان المدخلات
    
```

```
public static void main( ) {
```

127.0.0.1

```
try { host = new InetAddress().getLocalHost(); }
```

accessServer();

```
catch (IOException e) {
    S.O.P("Host not found");
}
```

} // main()

ClientHandle()

* البيرفر بخارج ادسرس الـclient

* الـclient بده يوصل للـserver

```
public static void accessServer() {
```

S.O.P("connected to Server");

socket link = null;

try {

link = new Socket(host, PORT); // (request) طلب اتصال

// سمح للموزر بدخول رسالة

Scanner UserEntry = new Scanner(System.in);

step 2

InputStream input = link.getInputStream();

InputStreamReader isr = new InputStreamReader(input);

BufferedReader br = new BufferedReader(isr);

الـuser يستقبل من جهتيه ① الرسالة ② الـserver

// step 2

(R.)

2-12

```

    print writer output = new PrintWriter(link.getOutputStream());
    string message, response; // Step 3
    do {
        s.o.p("Enter message:");
        message = UserEntry.nextLine();
        output.println(message);
        output.flush();
        response = br.readLine();
        System.out.println("Received message from server: " + response);
    } while (!message.equalsIgnoreCase("close"));
    br.close();
    input.close();
    output.close();
}

```

مترجم عن input و output بعدهما خطوة flush للقناة بخلاف طرق **printWriter**

تحزين (تختزين) ترضيع للفهان

→ تنظيف للسار وما يكون في timeout

لخط الرسالة على البورز من السيرفر

بروز دخل رسالة طبعتها طبعتها output طبعتها على السيرفر رايت للسيرفر السيرفر يطبع الجملة بترجمة منه على input (br) بظهورها كمسخر

while (!message.equalsIgnoreCase("close")) {
 UserEntry.nextLine();
 if (client != null) {
 client.sendMessage("close");
 }
 br.close();
 input.close();
 output.close();
 }
}

المحرك القائم للبوز يدخل ارسال بـ close

الرسالة خرجت من عنوان client

بروز دخل رسالة طبعتها طبعتها output طبعتها على السيرفر رايت للسيرفر السيرفر يطبع الجملة بترجمة منه على input (br) بظهورها كمسخر

catch (IOException e) {
 s.o.p("No message Received");
 }
}

راس ما استقبل مسبح

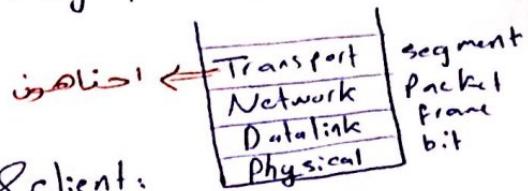
Try {
 s.o.p("Close Connection ---");
 link.close(); // socket
 client.close(); // socket
 } catch (IOException e) {
 s.o.p("Unable to close socket");
 system.exit(1);
 }
}

method access server

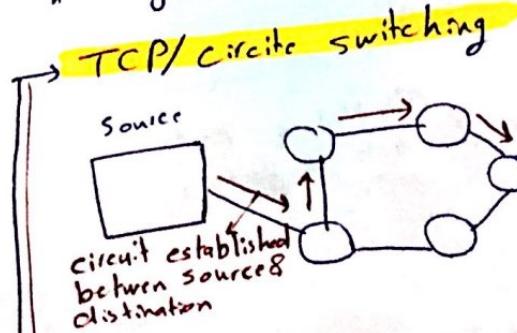
// class TCP Client

socket using UDP: (user datagram protocol)
بيانات المستخدم الموجة

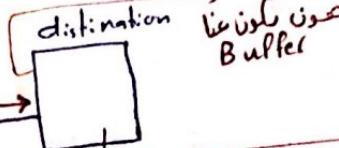
- transfer unreliable data (group of bytes / datagram)



عسان انقل البيانات
+ using Datagram socket



(in order) تهتم بالترتيب



كل البيانات خط وادر ← circuit n

أولاً، إذا متن بنفس الرتيب يفحص إذا وصلت البيانات بنفس الرتيب
SourceChecksum like

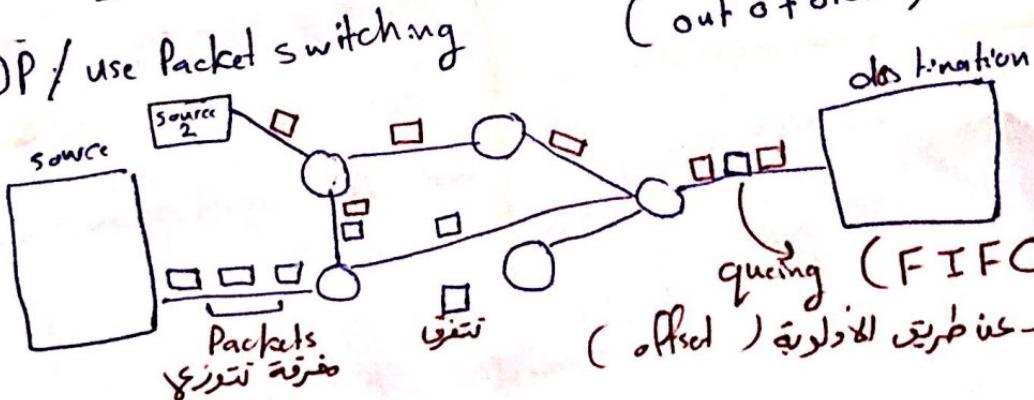
للسـخدم بنقل الـ

تعريف [Circuit switching] : a connection that requires a directed circuit to be established before data can be transmitted [اتصال يقتضي إنشاء دارة محددة لم انشئها لنقل البيانات]

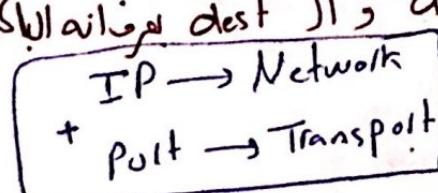
سلبيات : take all bandwidth / high overhead.

(out of order)

UDP / use Packet switching



لوكـان في 2 مـاري بـرسـالـ Packـetـ إلى نفسـ الـ Sourceـ 2 لـ destـ لاـ IPـ مـاري بـرسـالـ Packـetـ إلى نفسـ الـ Sourceـ 1 لـ destـ لاـ



تعريف [Packet switching] : data transmission model in which long messages are split into packets and sent to receive individually. الحالـاـ تـرقـلـ

$\text{MTU} \Rightarrow$ Network size
 $\text{MSS} \Rightarrow$ Transport size
max signal size

TCP

- Transport layer
- **TCP header** (Port number)
 $20 + 1460$ data
- slower
- $\text{mss} = 1500 \text{ Byte}$

IP (Network layer)
- $\text{MTU} = 1500 \text{ Byte}$

UDP

- transport layer
- UDP header
 $8 + 1472$ data
- faster

IP header
 $20 + 1480$

- Connection oriented

- MSS
- MTU
↓
Input Stream
Output Stream

repeat packet
or

in come Packt
out come Packt

= Multithreading

why we use Datagram Packet??

VDP

محاضرة الـ 12 : 12 - 12

1- each packet delivered only (كل بacket لوحده)

2- it delivered out of order (غير مرتبة)

3- un reliability (عدم موثوقية)

Private static DatagramPacket inComePacket, outcomePacket;

private static bytes[] buffer;

public static void main() {

try {

s.o.println("openingPort...");

Socket Sock = new DatagramSocket(PORT);

server side

SpecificPort

connectionless (UDP is connection less)

- client any free Port

- server for a specific port

mt ← " المبروت = "

ClientHandling();

}

catch (IOException e) {

s.o.ph("unable to attach Port");

System.exit(1);

} main " اخرج من البرنامج كله عدا المبروت "

في حالة ما عرفت برجلي
البيورت او حط بيورت
محجوز او حط رقم
الب

Server طرف الـ

private static void ClientHandling() {

String messageIn, messageOut;

int numMessages = 0;

try {

do {

buffer = new byte[1500];

inComePacket = new DatagramPacket

(buffer, buffer.length);

(3) ادخل البيانات

step 4) استقبل البيانات

step 5) عرض المبروت

step 6) اخراج المبروت

step 7) اخراج المبروت

step 8) اخراج المبروت

step 9) اخراج المبروت

step 10) اخراج المبروت

step 11) اخراج المبروت

step 12) اخراج المبروت

بروت بيعت و نيفيل
ختر لر سكريت والر
دلا و هغيره انه
صا ز تفيري و تساكيه)

// Step 3

1) فتح السوكت 2) عرض المبروت 4) استقبل البيانات 6) اخراج المبروت

= Multithreading
 message In = new String(incomppacket.GetData()); // خطوة 5
 incomppacket.length(); // حجم المESSAGE
 s.o.println("message received"); // حزم المESSAGE
 numMessages++; + (numMessages) + ":" + messageIn;
 ② MessageOut = "Message";
 s.o.println(MessageOut);
 step 6
 Inet Address host = incomppacket.getAddress();
 int clientPort = incomppacket.getPort(); my free port
 step 7 // outComppacket = new DatagramPacket
 (messageOut.getBytes(), messageOut.length(), host, Client Port)
 Byte clientPort // العنصر في المESSAGE
 sock.send(outComppacket); // step 8
 } while(true); // Server will be on.

الشكل =
 import java.io.IOException;
 import java.net.DatagramPacket;
 import java.net.DatagramSocket;
 import java.net.InetAddress;
 } // try
 catch (IOException ex)
 System.err.println("message not received");
 System.exit(1);
 } // catch
 } // End handling Method

= Multithreaded
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Scanner;

client side

Public Class **UDPClient**

private static final int Ports 1234;
private static InetAddress host;
private static DatagramPacket incompaket, outkomepacket;
private static DatagramSocket clientSock;
private static byte[] buffer;

public static void main()

try{
host = InetAddress.getLocalHost(); // step 1
}
catch(UnknownHostException e){
System.out.println("unable to find the host");
System.exit(1);
}

server Accessing(); // method

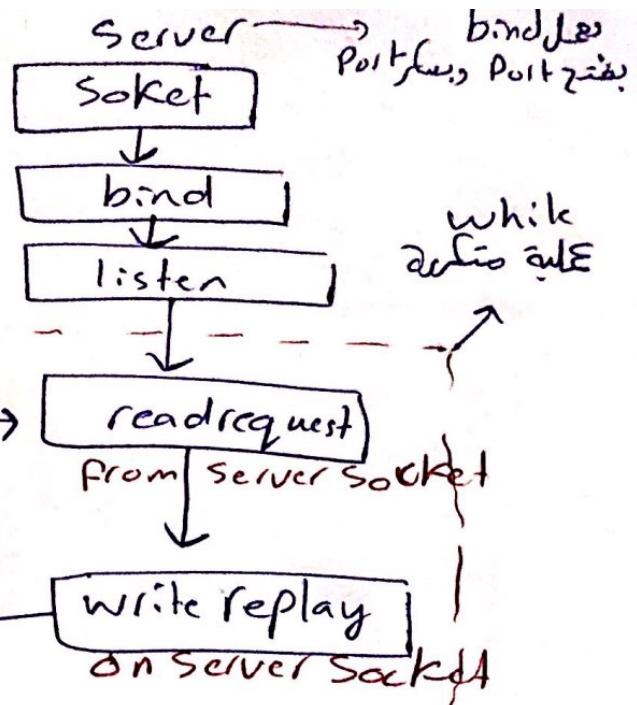
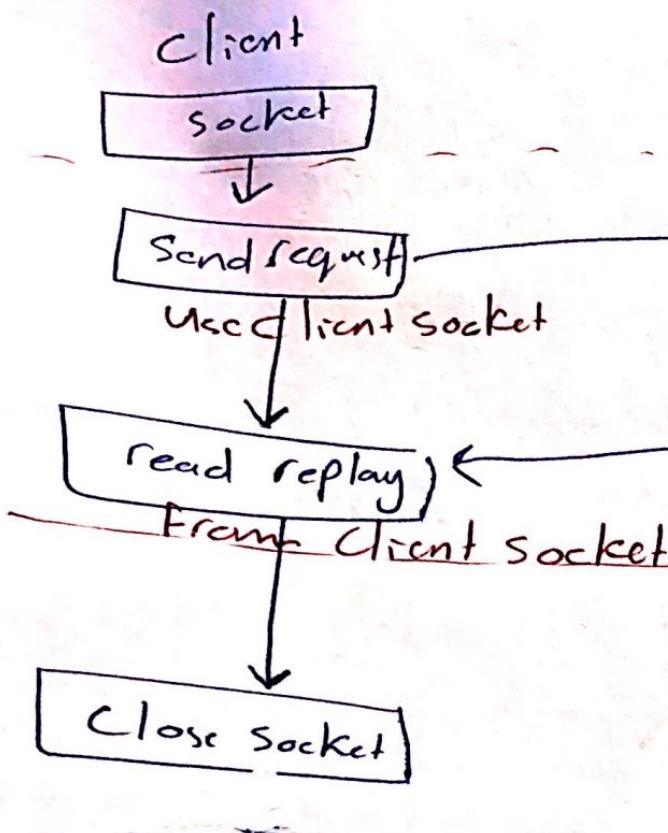
} // main

Multi-threading

```
private static void serverAccessing() {  
    try {  
        clientSock = new DatagramSocket(); // step 2  
        (no connection between client & server) اتصال بين client & server لا يوجد // 1 Port number  
        client socket binds to available port // 2 يربط client بPort رقم المكتوب  
        Client socket binds to available port // 3 يربط client بPort رقم المكتوب  
        Scanner userEntry = new Scanner(System.in); // step 3  
        String message, response;  
        do {  
            System.out.println("Enter message:");  
            message = userEntry.nextLine();  
            System.out.println("Message length: " + message.length());  
            System.out.println("Host IP address: " + hostIP);  
            System.out.println("Port number: " + port);  
            System.out.println("Message: " + message);  
            outcomepacket = new DatagramPacket(message.getBytes(),  
                message.length(), host, port); // step 4  
            clientSock.send(outcomepacket); // step 5  
            buffer = new byte[1500]; // step 6  
            incomepacket = new DatagramPacket(buffer, buffer.length()); // step 7  
            clientSock.receive(incomepacket); // step 8  
            response = new String(incomepacket.getData(), incomepacket.getLength());  
            System.out.println("Response from server: " + response);  
            if (!response.equalsIgnoreCase("close")) {  
                UserEntry.close();  
            }  
        } catch (IOException e) {  
            System.out.println("unable to open socket");  
        } catch (IOException e) {  
            System.out.println("No Response from server");  
        }  
        System.out.println("close socket");  
    } catch (Exception e) {  
        System.out.println("close socket");  
    }  
}
```

Client using free Port
Port num
packets = Port + IP num

UDP Steps:



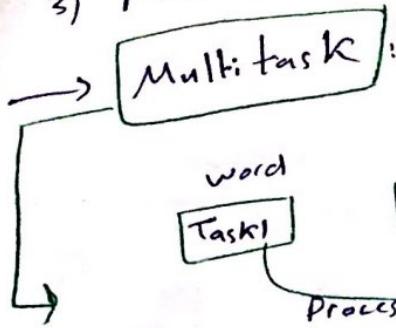
والسفر يجري وـ UDP

في الـ TCP الـ server الـ listen

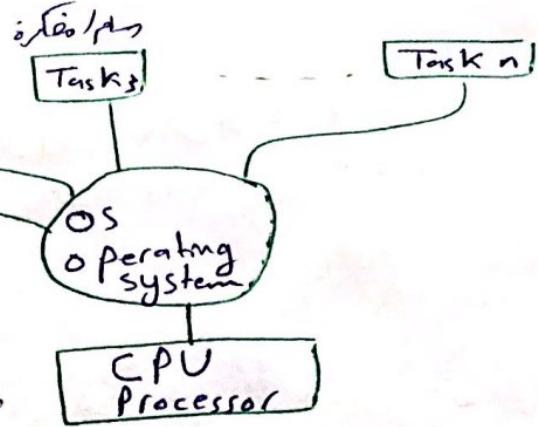
Multi-threading

تعدد المهام

- 1) Multi thread
- 2) Multi task
- 3) Multi Process



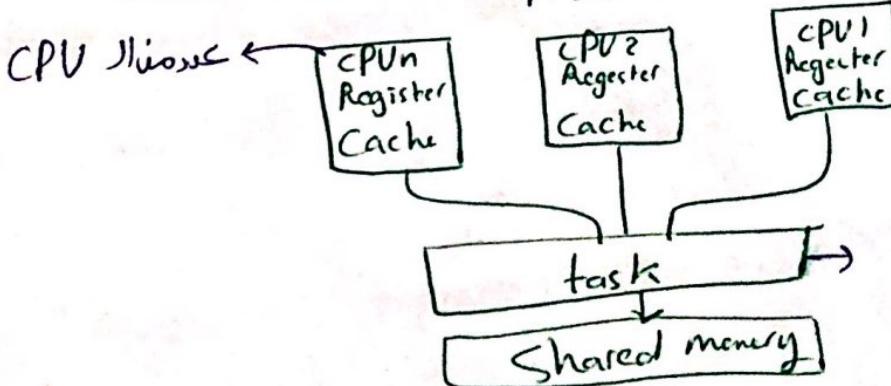
The execution of more than one task at the same time.



تَعْرِيفُ اِنْجِزْيَةٍ → Multitask: one processor performs Multiprocessor at the same time (يُنْجِزُ كُلَّ تَعْلِمَةٍ بِنَفْسِهِ وَنَوْتَهِ)

Multi-task (نُجزٌ) = multiprocess or multithread

⇒ **Multi-processing**: a system that has two or more processors work (execute) the same task



one task or more than one task
يعُالجُونَ معاً في نفس المُوقِتِ

one or more tasks يُعَالِجُونَ في نفس المُوقِتِ → Multiprocessor

one task يُعَالِجُهُ CPU → Multitask
one task يُعَالِجُهُ Processor threads → Multithread

Multithread: executing multiple threads of a single process at the same time

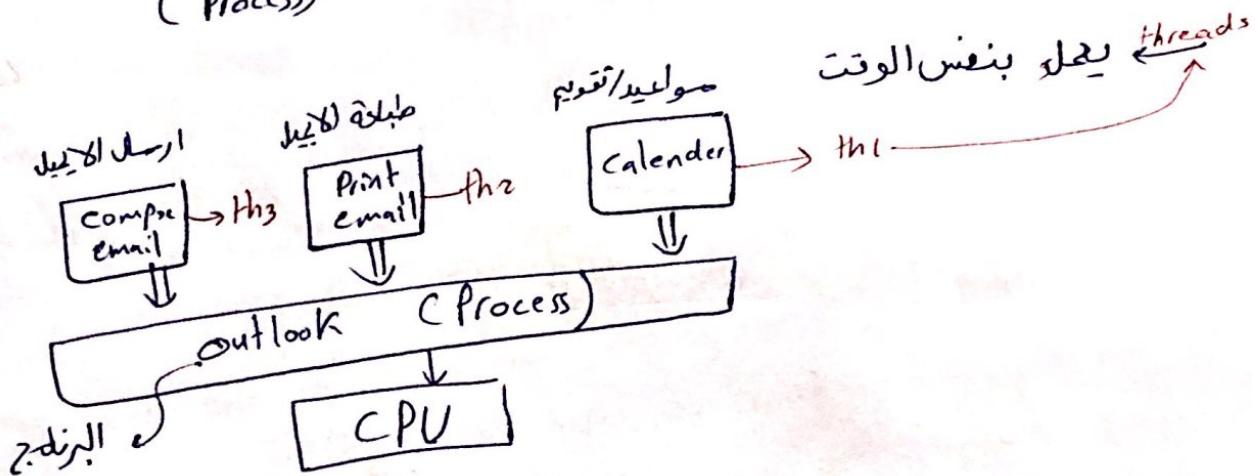
Threads are light weight processes

(A process divided into number of sub processes)

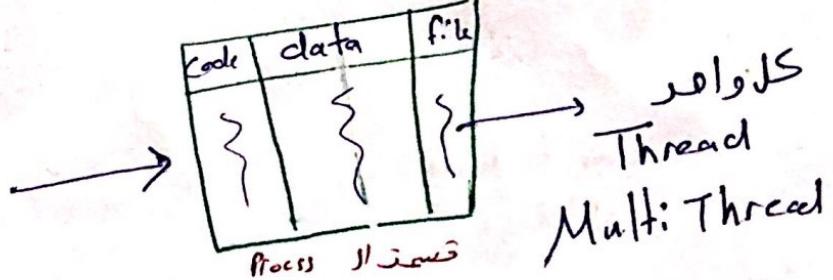
* Thread: is light weight process

"آلية مخففة"

(Thread) is a unit of work - رسائل : كل وحدة عبارة عن رسائل
(Process) وحيث كلها عبارة عن عملية كاملة



Process:



one process has many threads

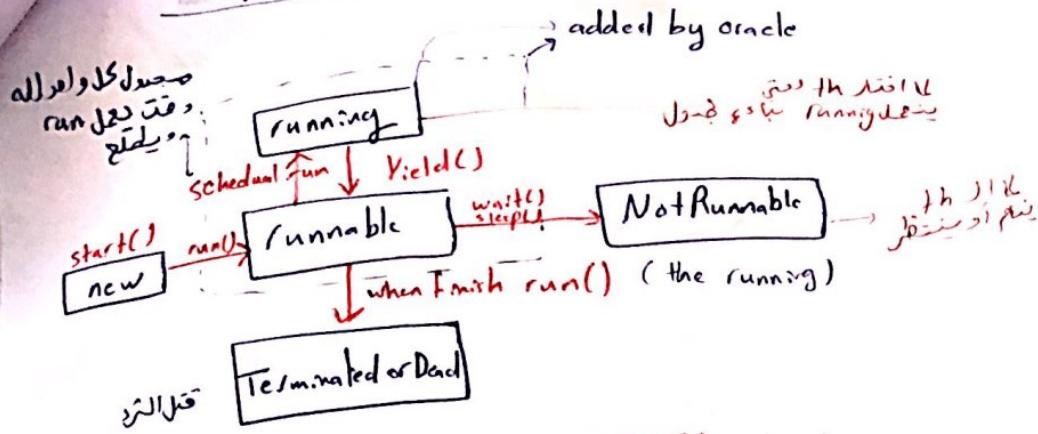
* why use Multithread ??

- 1) executing thread at the same time (Save time) → fast process
- 2) Handling error independently (each thread don't effected when error occurs in the other thread) كلا يتحمل مسؤولياته

Thread life cycle

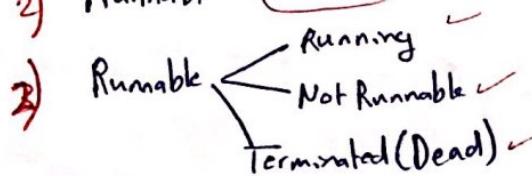
محاضرة الثلاثاء: برمجة متعددة النواة

28.12



1) new: Create thread using **start()** method .

2) Runnable: **start()** method move thread from new state to runnable state.



3) Running: When select thread to running , using **schedual table**

Name	state	Id	Time
thread1	run	15	400 - 500 ms
thread2	run	16	500 - 700 ms
thread3	run	17	900 - 1000 ms

متتبلاً
متتابع
متزامن
متزامن

* Thread 11 is selected because it has the largest time interval between the start of the thread and the end of the sleep period.

- 4) NotRunnable: means the thread are stopped or **(sleep())** وقت ينفك
- 5) Terminated : The Thread is Dead or (finished) using **run()** method

28-12
2-1
4-1
+ 6-1

كيف نكتب Thread ?? Thread Class

* How we Built thread ?? Thread Class (وراثة) OOP

- 1) By extends it from **thread class**
- 2) Implements it from **Runnable interface**

Public class Multithread extends Thread {

 Public run () { "override"

 for (int i=5; i<=15; i++) { number of Process

 System.out.println(Thread.currentThread().getName() + " Id - " + Thread.currentThread().getId()) +

 ": " + i); process = i

// Thread-0 Id - 10 : | 2 threads on same process
ex: Thread-1 Id - 11 : |

try {

 Thread.sleep(500); ?? time

} catch (InterruptedException e) {

 e.printStackTrace();

}, , run --> Thread() مفهوم

Public static void main () {

 Multithread th1 = new Multithread();

 Multithread th2 = new Multithread();

 th1.start(); --> run() دخل في run()

 th2.start(); --> يكون متوقف

} // main

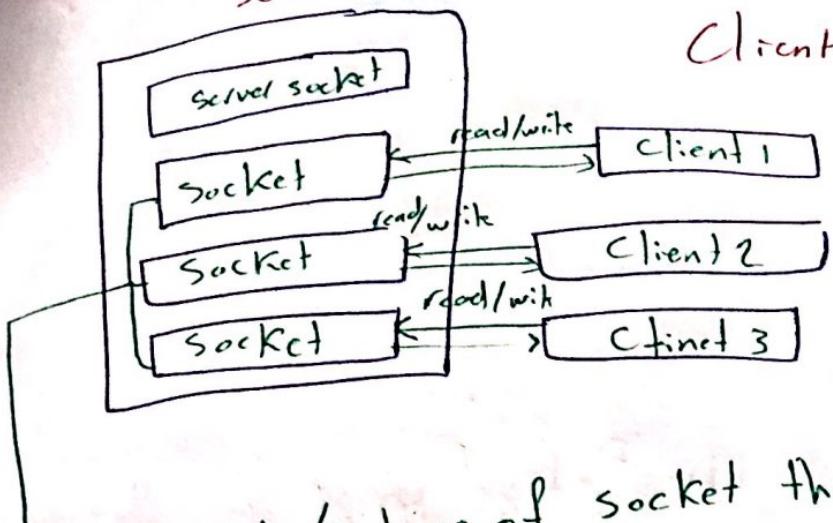
} // class

① Thread() الارجمنت المطلوب بنسخته
② th1 دخل في start run()
③ بداخل اذل بنام
④ start في بروتوكول بسرور في run

- خاص ونام -
الثاني بدأ -

او يكمل جديدها المسماة

مكتبة



→ Thread instance of socket threads ?

Server side *

public class Multi-thread TCP server {

private static ServerSocket SerSock;

private static final int PORT = 1234;

public static void main () { generic

Thread server	thread server2	-----
---------------	----------------	-------

list // ArrayList < Client Handler > threadlist = new ArrayList <> ();
لست منفرزة ترددان او سيرفر

S.O.P ("connect to port");

ارسال اريه عنوان اخترن منها شردان البروتوكول
البروتوكول منفرز منها thread Array ①

Try {

if // sersock = new Server Socket (PORT); Port

? Catch (IOException e) {

S.O.P ("unable to connect to PORT");

System.exit(1);

}

```

try {
    do {
        step 3 // Socket client = SerSock.accept(); // بثبات الميرفر
        S.o.p("new Client accepted"); // client JI to
    } while (true); // client JI to
}

client Handler Server thread = new Client Handler(
    client, thread list);
    ↴ socket
    ↴ Thread list
Thread list.add (Server thread); // (Server io inactive list)
                                    (client) على عدد اجهزة العملاء
                                ↴
Server thread.start(); // بثوابع
                        ↴ بلصق تسلق
} while (true); // server on
} // try
catch (IOException) {
    S.o.p ("client is unable to be accepted");
}
}

```

البراجم

```

import Java.io.IOException
import Java.net.ServerSocket;
import Java.net.Socket;
import Java.util.ArrayList;

```

"Multithread TCP server"

import
.
.
.
.
servethd || also class, generic

public class ClientHandler extends Thread {

 private static Socket Client;
 private static InputStream input;
 private static InputStreamReader isr, input
 private static BufferedReader bur;
 private static PrintWriter output;
 private static ArrayList <Client Handler> threadlist =
 new ArrayList <>();

Constructor

Public ClientHandler (Socket link, ArrayList <client Handler> threads);

22/1

ClientHandler • Client = link;

// step 1

ClientHandler • threadlist = threads; // step 2

client
link input = Client.getInputStream();
isr = new InputStreamReader(input); }
bur = new BufferedReader(isr); }
client
output = new PrintWriter (client.getOutputStream()); }

input

client
link

(client.getOutputStream()); }

```

try {
    input // step2
    output // step3
} catch (IOException e) {
    s.o.p("No threads found");
}
}

```

```

public void run() {
    int numMessages = 0;
    ornull string message = "";

    try {
        do {
            message = bui.readLine(); // step 4
            if (message.equals("message received")) {
                numMessages++;
                s.o.p("Message " + numMessages + "=" + message);
                * عداد بعد كل سبعة رسائل
                * يعاد لها
                Client.printToAllClients(message); // step 5
                * رجع الرسالة
                * لرجاع الرسالة الى الراتخ Client الذي بعث
            }
            output.flush();
        } while (!message.equalsIgnoreCase("close"));
    } catch (IOException e) {
        s.o.p("message not received");
    }
}

```

```

try {
    S.o.p ("closing connection");
    client.close(); // step 6 - -> إغلاق الـ socket
} catch (IOException e) {
    S.o.p ("unable to close socket"); // client نحن نعمل على كل
} // catch
} // run ... - -> بستقبل الرسالة ويرجعها

```

Private void printToAllClient (String message) {
 for (ClientHandler st : threadList) {
 S.o.p (message); //
 }
}

طبعاً من client إلى server (برجعه)
 كل Client يرجع رسالة
 Client عادي على كل Client

نرجع الرسالة

طرف السيرفر كيف يستقبل الرسالة ويرجعها
 TCP Multi-client Thread
 one Server Multi-client =>

مكتبات البرنامج السابق :-

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.ArrayList;

```

Multithread TCP server

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

Client will be

NOTE

Thread is light weight
process

Public class **Multi thread TCP Client** {

 private static final int PORT = 1234;

 private static InetAddress host;

 private static Socket client = null;

 public static void main() {

 local file "Try"
 host

 host = InetAddress.getLocalHost();

 client = new Socket(host, PORT); // Step 1

 clientThread = new ServerAccesser(client); // Step 2

 clientThread.start(); // Client Thread will start

 }

 class ServerAccesser implements Runnable {

 clientThread.start(); // Client Thread will start

 // Server will go to sleep Thread.join(); client is not working

 }

 }

 clientThread.start(); // Client Thread will start

 } // catch (IOException e) {

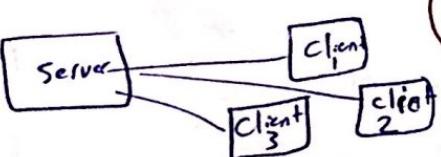
 System.out.println("Host Not found");

 System.exit(1);

 } // catch

} // main

} // class



```
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
```

```
try {
    public class Server Accesser extends Thread {
```

```
    private static InputStream input;
```

```
    -> InputStreamReader isr;
```

```
    -> BufferedReader br;
```

```
    -> PrintWriter output;
```

```
    -> Scanner UserEntry;
```

```
    -> Socket client;
```

```
    -> Socket link;
```

```
}
```

```
public Server Accesser (Socket link) {
```

```
    client = link; // step 1
```

```
user = new Scanner (System.in); // step 2
```

```
input = client.getInputStream(); // step 3
```

```
isr = new InputStreamReader (input);
```

```
br = new BufferedReader (isr);
```

```
output = new PrintWriter (client.getOutputStream());
```

```
}
```

```
catch (IOException e) {
```

```
    System.out.println ("message not found"); } }
```

```
public void run() {
```

```
    String message, response;
```

```
    System.out.println ("connected");
```

```
try {
```

```
do {
```

```

        s.o.p("enter message");
        message = userEntry.nextLine(); // Step 5
        s.o.p("message"); // طاقة از رسالة
        output.println(message); // ارجاع ارسالة
        client.println(message); // للبروز
        ضرورة لتنظيف مجرى الاذبوبة
        output.flush(); // نظف
    } // نظف من

    response = br.readLine();
    s.o.p("server receive > " + response);
    while(!message.equalsIgnoreCase("close")) {
        client.close(); // يبعث رسالة close الى client
        UserEntry.close();
        br.close();
        output.close();
    }
    catch (IOException e) {
        s.o.p("No Response");
    }
    try {
        s.o.p("Close connection");
        client.close();
    }
    catch (IOException e) {
        s.o.p("unable to close Socket");
    }
}
// run
} // close

```

السطح المستخدم
 انه يدخل الرسالة
 اخواتي ويعتني بالبيانات
 (client وresponse للبروز input