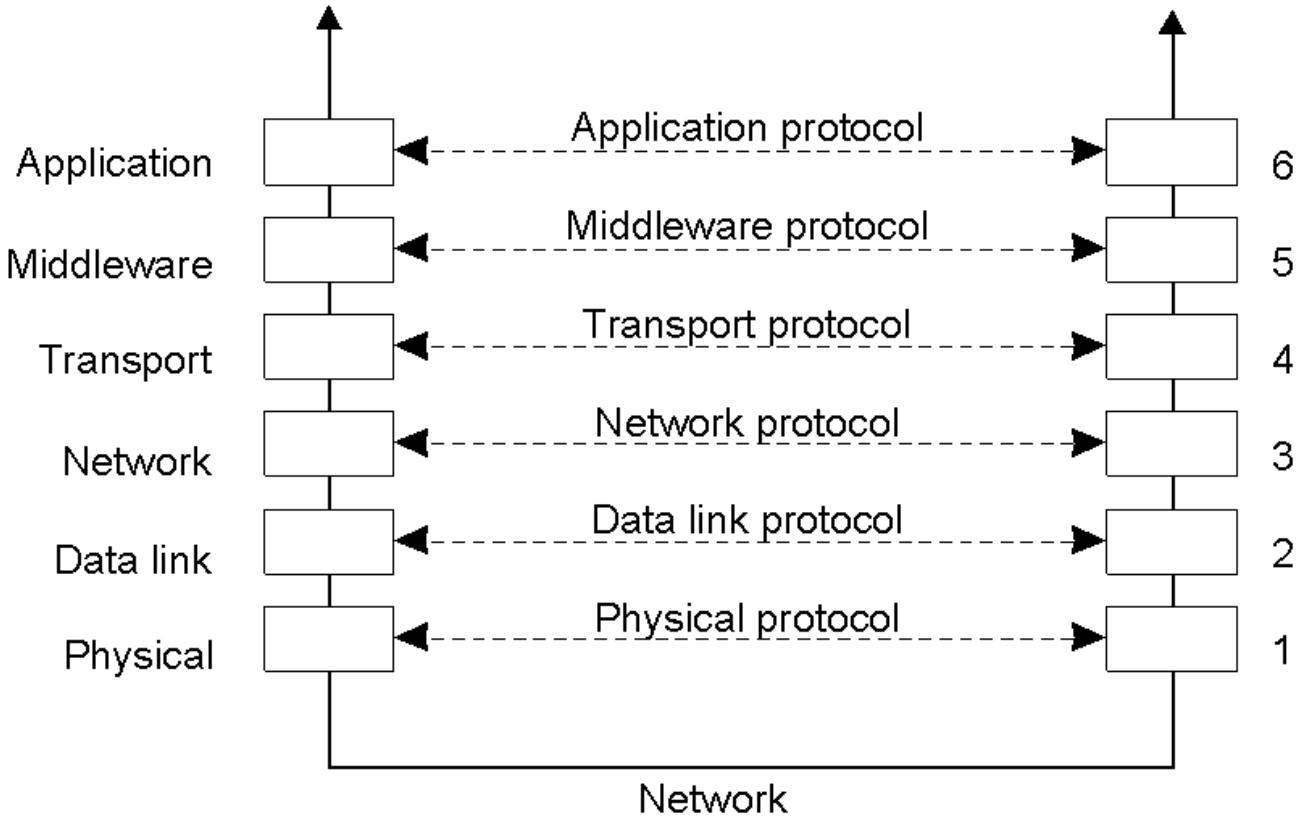
Communications


Application
Middleware
Transport
Network
Data link
Physical
Application protocol
Middleware protocol
Transport protocol
Network protocol
Data link protocol
Physical protocol
6
5
4
3
2
1
Network

طبقات شبكة الانترنيت

- Application = مجموعة التطبيقات المدعومة من الشبكة
  - HTTP, SMTP, FTP, Telnet, ...
- Middleware = البرمجيات الوسيطة و التي تدعم بناء تطبيقات موزعة و تخفي تفاصيل الاتصال على الشبكة
  - RMI, CORBA, DCOM, ..
- Transport = التحويل من نقطة لنقطة
  - TCP, UDP, …
- Network = تحويل الرزم من الـ Source إلى الـ Destination
  - IP, Routing Protocol
- Data Link = تحويل المعطيات بين النقاط المتجاورة و اكتشاف الأخطاء
  - PPP, Ethernet
- Physical = البيتات على الكابل (مستوى الجهد للـ ٠ و ١ / معدل إرسال البيتات / الاتصال بالاتجاهين أو لا .....)

Communications (1)

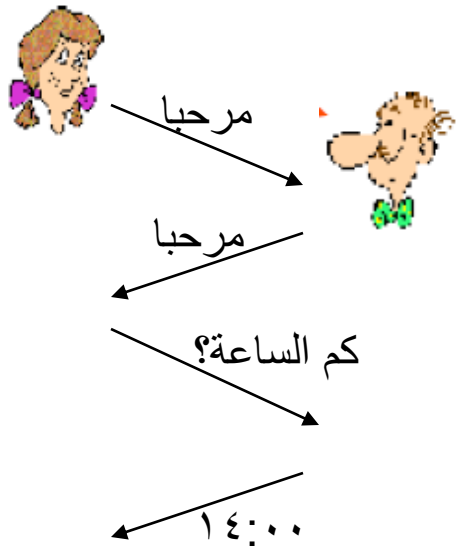
- Communication
  - Send operation = Add Msg to remote queue
  - Receive operation = Remove Msg from local queue
- Synchronous Communcation:
  - Send is blocking for the sender
  - Receive is blocking for the receiver
  - Send is blocked until Receive is issued
- Asynchronous Comm:
  - Send is non-blocking for the sender
    - Sender continues when msg is copied to local buffer
  - Receive may be :
    - blocking or
    - non-blocking ➔
      - Receiver continues after the receive operation
      - Receiver provides a buffer for incoming msg
      - + notification mechanism (pooling or interupt)

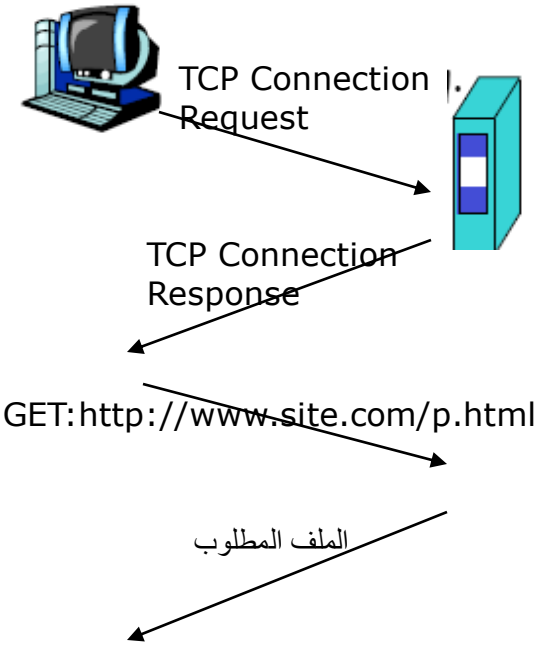
Communications (2)

- Java & Asynchronous Communication
  - Threads for asynchronous send & receive
- Reliable Communication
  - Guaranteed delivery with dropped & lost packets
- Ordering Communication
  - Msgs delivery according to the sender order
- Msg Sending
  - IP + Port
  - Location transpaency ➔ name <> IP
- Process send Msg to another Process
  - Socket in the sender & socket in the another
- Process + several Sockets

Communication Protocols

- البرتوكول = صيغة وترتيب الرسائل المرسلة / المستقبلة عبر مكونات الشبكة و الأفعال الواجب اتخاذها عند استقبال الرسائل



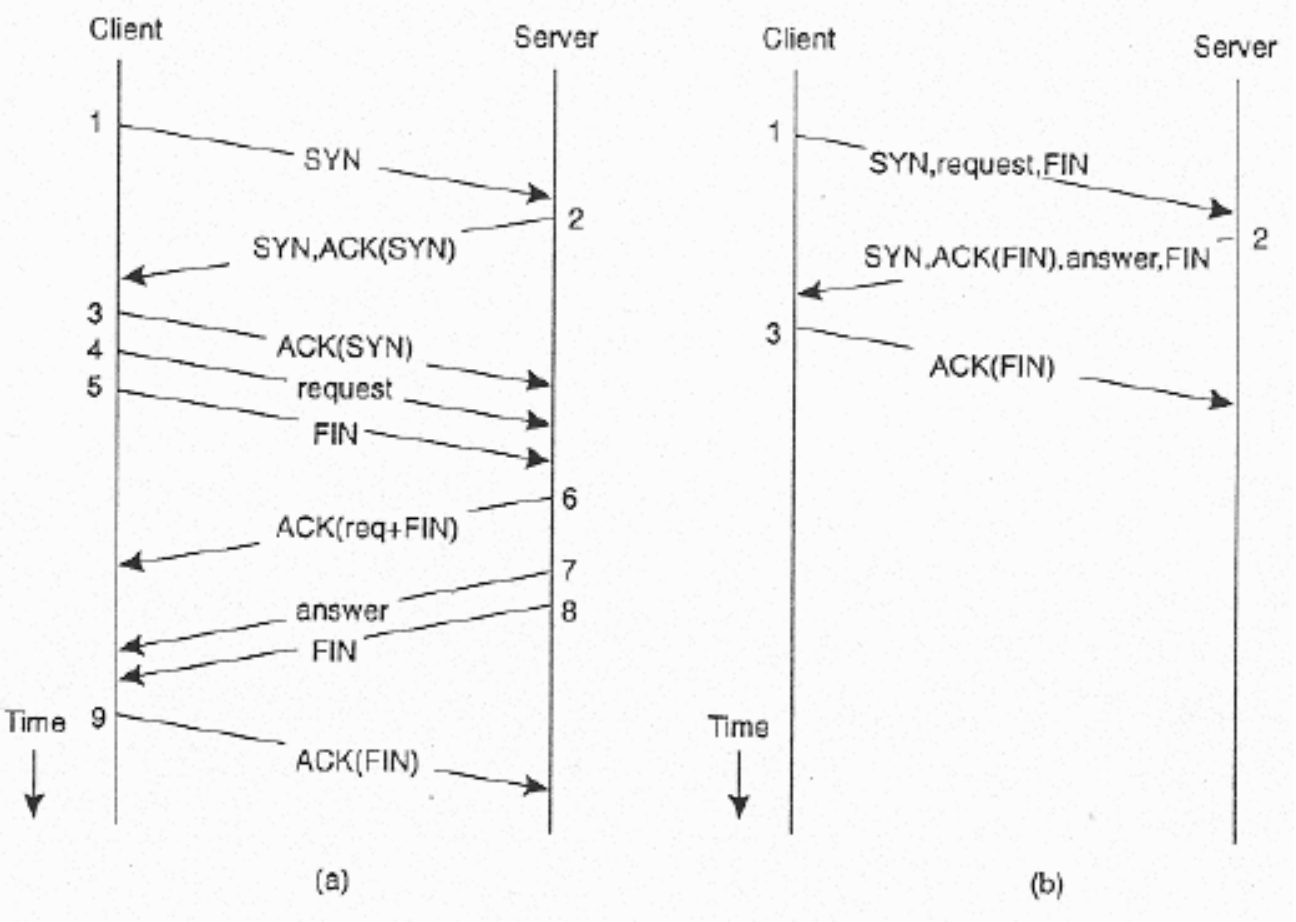


Transport Communication Protocols

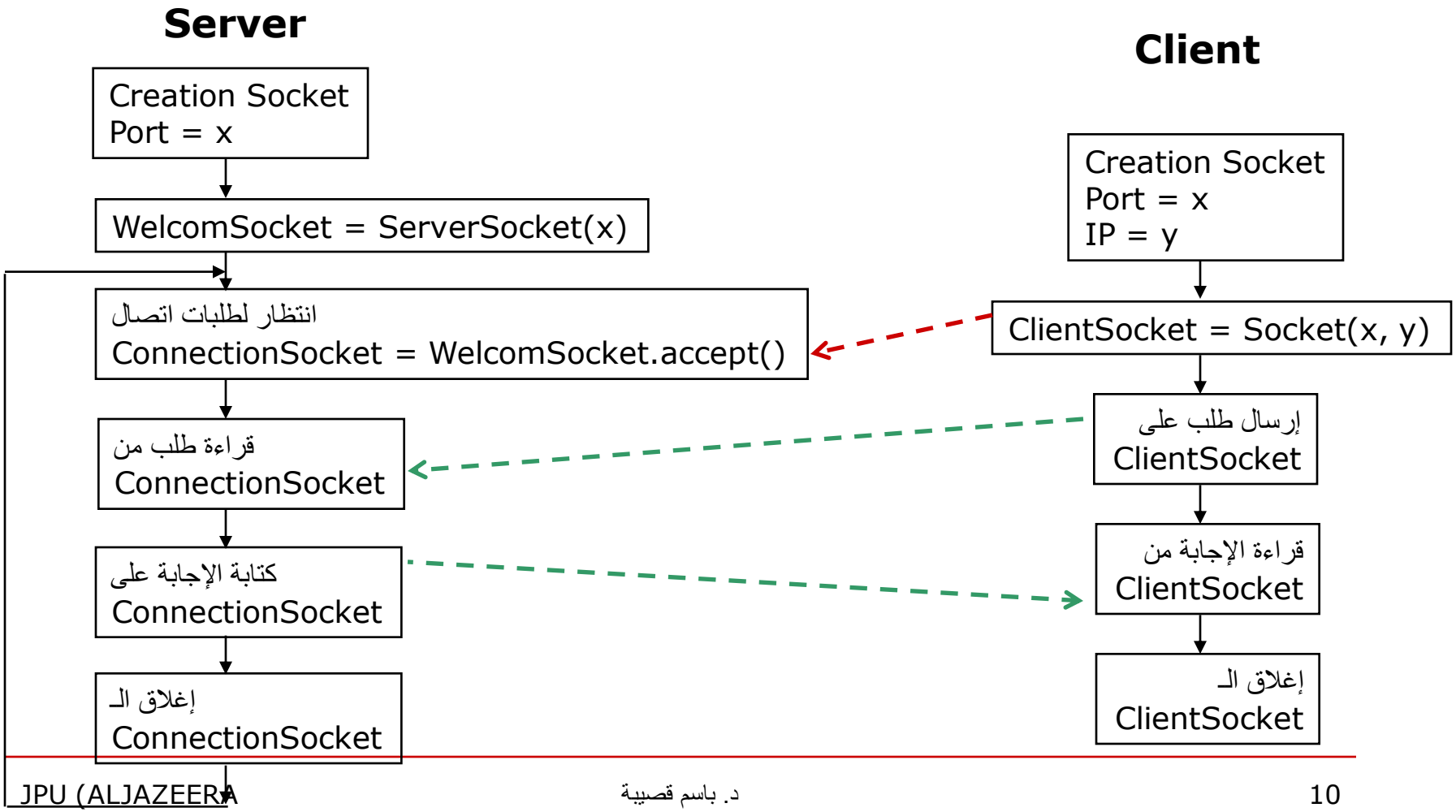
- Connection-Oriented Protocols
  - TCP
  - Secure communication protocols
- Connection-less Protocols
  - UDP
  - Group communication protocols

Connection-Oriented Protocols

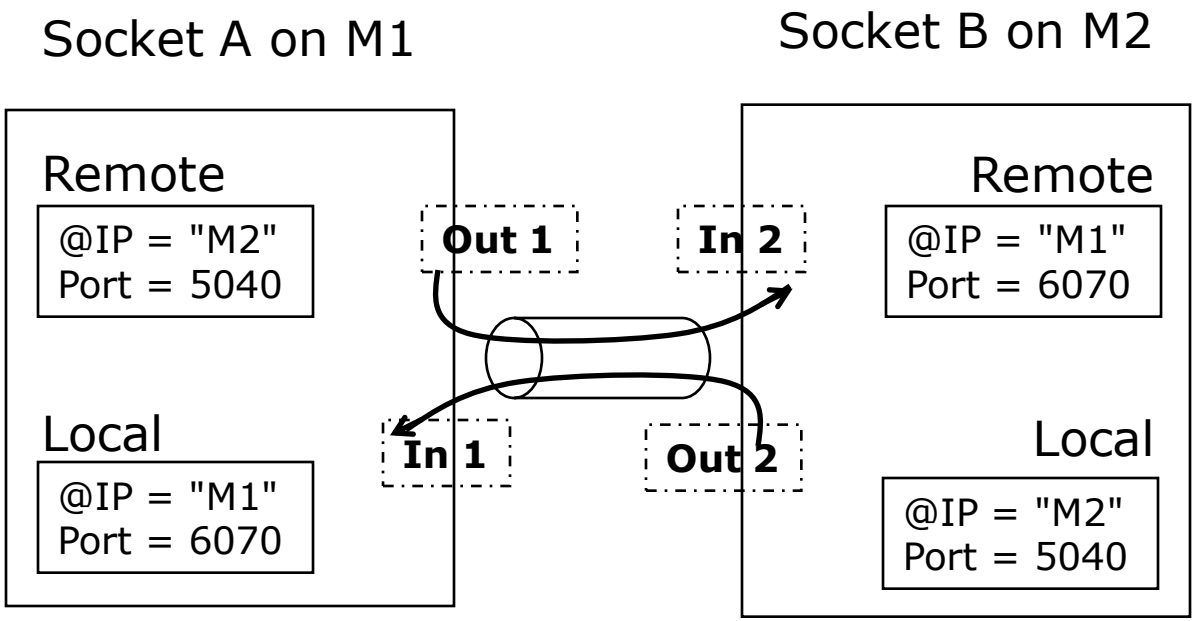
- الهدف = تحويل المعطيات من نقطة لنقطة
- Connection = (المصافحة Handshacking) :
  - تهيئة حالتي الحاسبين و التحضير لتحويل المعطيات
- TCP (Transmission Control Protoco)
  - يقدم قناة اتصال بين فعاليتين (Producer-Consumer) و بكلا الاتجاهين
    - المعطيات المرسلة تخزن في Queue حتى يصبح المستقبل جاهز
    - المستقبل ينتظر المعطيات
    - المرسل ينتظر عندما يمتلئ مكان التخزين
  - يحول المعلومات بطريقة موثوقة و مرتبة
    - في حالة فقدان معطيات : لدينا إعلام و إعادة إرسال
  - التحكم بتدفق المعطيات عبر الشبكة
    - المرسل يكيف نفسه حسب سرعة المستقبل
  - التكيف في حالات اغتصاص الشبكة
    - المرسل يخفف من سرعته عند ملاحظة اغتصاص في الشبكة
  - المبرمج لا يهتم بحجم الرسالة = TCP يقرر حجم و عدد الرزم للرسالة
- التطبيقات التي تستخدم TCP :
  - HTTP, FTP, SMTP, Telnet, ...


Client
Server
1
SYN
2
SYN,ACK(SYN)
3
ACK(SYN)
4
request
5
FIN
6
ACK(req+FIN)
7
answer
8
FIN
Time
9
ACK(FIN)
(a)
Client
Server
1
SYN,request,FIN
2
SYN,ACK(FIN),answer,FIN
3
ACK(FIN)
Time
(b)

Java & TCP


Server
Creation Socket
Port = x
WelcomSocket = ServerSocket(x)
انتظار لطلبات اتصال
ConnectionSocket = WelcomSocket.accept()
قراءة طلب من
ConnectionSocket
كتابة الإجابة على
ConnectionSocket
إغلاق الـ
ConnectionSocket
Client
Creation Socket
Port = x
IP = y
ClientSocket = Socket(x, y)
إرسال طلب على
ClientSocket
قراءة الإجابة من
ClientSocket
إغلاق الـ
ClientSocket

TCP Socket



A.getInetAddress() = B.getLocalInetAddress()
A.getPort() = B.getLocalPort()
A.getLocalInetAddress() = B.getInetAddress()
A.getLocalPort() = B.getPort()

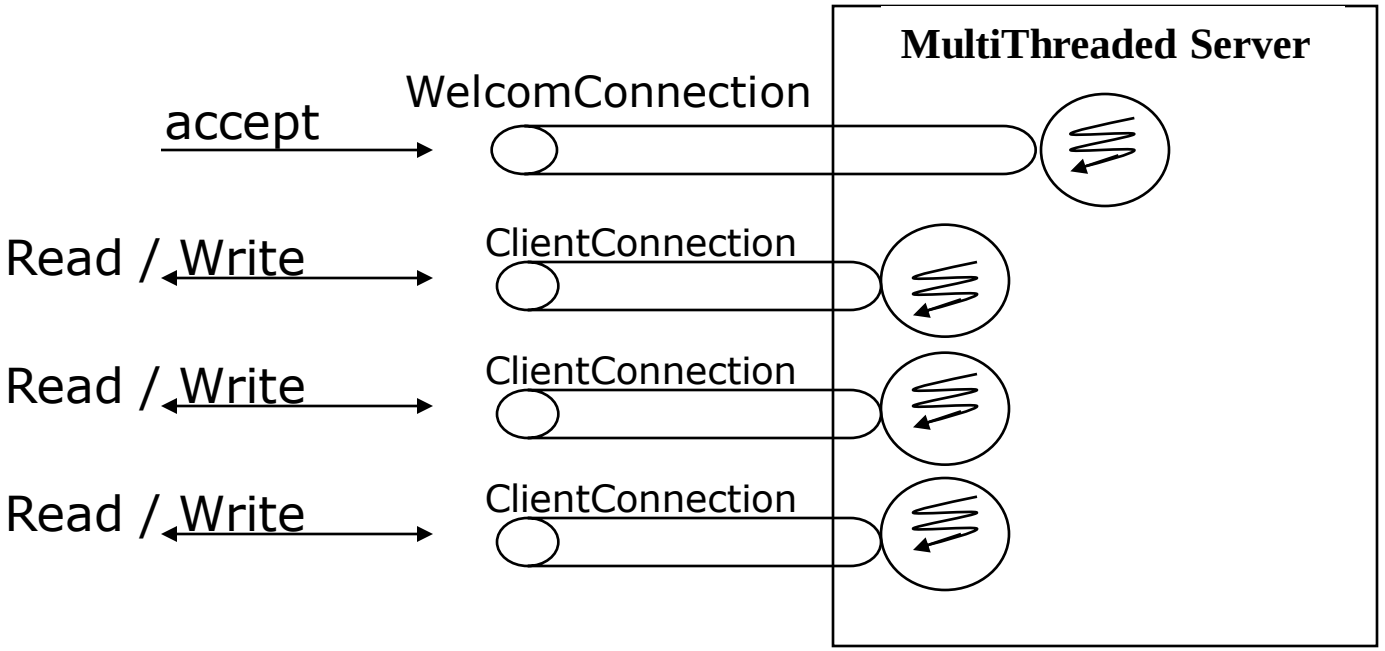
JAVA – Sum TCP Server

```
import java.io.*;
import java.net.*;
public class sumServer {
        public static void main(String[] args) throws Exception {

                ServerSocket welcomSocket = new ServerSocket(6789);
                while(true)
                {
                        System.out.println("Server wait a connection ....");
                        Socket connectionSocket = welcomSocket.accept();

                        DataInputStream inFromClient = new  DataInputStream(connectionSocket.getInputStream());
                        DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());

                        int n1 = inFromClient.readInt();
                        int n2 = inFromClient.readInt();

                        int sum = n1 + n2;
                        System.out.println(sum);

                        outToClient.writeInt(sum);
                }
        }
}
```

JAVA – Sum TCP Client

```
import java.io.*;
import java.net.*;
public class sumClient {
        public static void main(String[] args)throws Exception {
                BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
                try {
                                InetAddress lclHost = InetAddress.getLocalHost();
                                Socket clientSocket = new Socket(lclHost, 6789);

                                DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
                                DataInputStream inFromServer = new DataInputStream(clientSocket.getInputStream());

                                String nb1 = inFromUser.readLine();
                                String nb2 = inFromUser.readLine();

                                int n1 = (new Integer(nb1)).intValue();
                                int n2 = (new Integer(nb2)).intValue();

                                outToServer.writeInt(n1);
                                outToServer.writeInt(n2);

                                int sum = inFromServer.readInt();

                                System.out.println("From Server : " + sum);
                                clientSocket.close();
                }
                catch(Exception e) {e.printStackTrace();}
        }
}
```

MultiThreaded Server



Thread foreach Connection

```
import java.net.*;
class ClientManager extends Thread{
        Socket client;
        public ClientManager(Socket s)
        {
                client = s;
        }
        public void run() {
                // … process the client request
                client.close();
                }
        }
}
```

```
import java.net.*;
public class Telnet{
        public static void main(String[] args) {
                ServerSocket server = new ServerSocket(port);

                while(true) {
                        S.O.P("Server is ready …");
                        Socket client = server.accept();
                        (new ClientManager(client)).start();
                }
                server.close();
        }
}
```

Multi-threaded TCP

```
import java.net.*;
class RequestProcessor extends Thread{
        Socket client;
        public RequestProcessor(Socket s)
        {
                client = s;
        }
        public void run() {
                // … process the client request
                // …get inputstream
                // …get outputstream
                // …do reques processing
                client.close();
                }
        }
}
```

```
import java.net.*;
public class TCPServer{
        public static void main(String[] args) {
                ServerSocket server = new ServerSocket(port);

                while(true) {
                        S.O.P("Server is ready …");
                         Socket request = server.accept();
                        (new RequestProcessor(request)).start();
                }
                server.close();
        }
}
```

Thread Pools

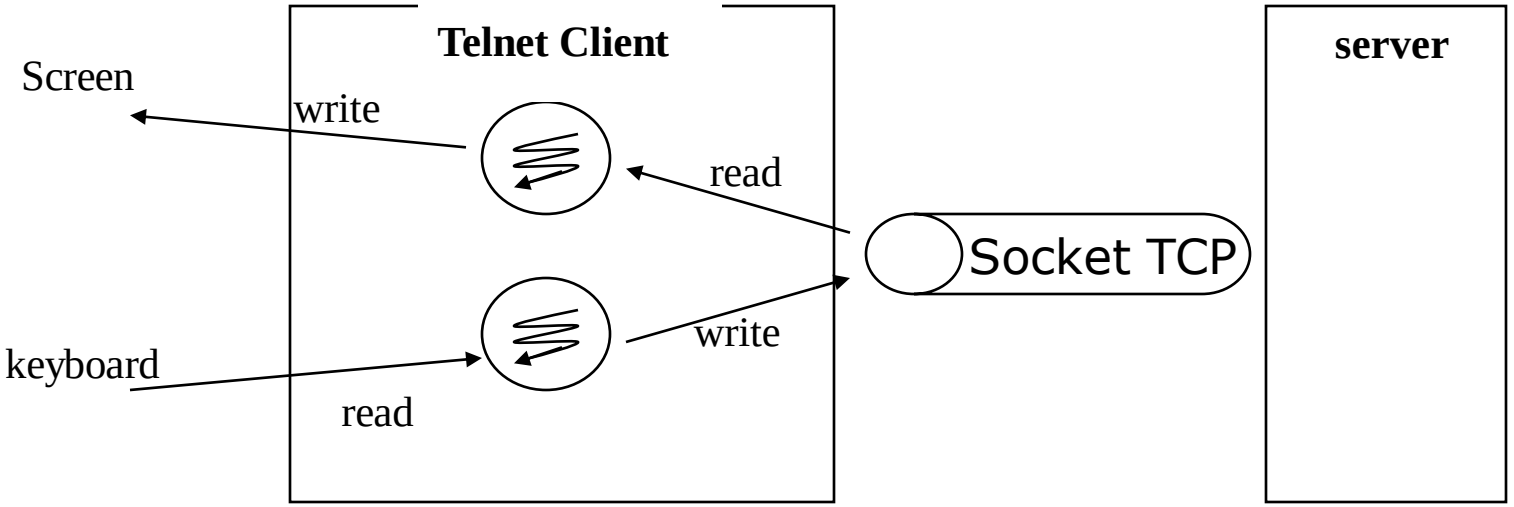
- Threads تحسن من أداء النظام خصوصاً مع I/O
- Threads لها كلفة (لدى الإنشاء و الإزالة و المبادلة) خصوصاً عندما : تنتهي بسرعة و/أو نخصص الآلاف منها بالدقيقة
- =Thread Pools
  - الـ threads لا تموت بعد إنهاء عملها (لا حاجة لإقلاع الـ Threads)
  - المهام في رتل و الـ Threads تبحث عن مهمة في الرتل لإنجازها
  - = Method 1
    - إقلاع عدد ثابت من الـ Threads
    - task pool فارغة فإن الـ Threads تنتظر على الـ pool
    - Thread تنهي عملها تعود إلى Task pool للبحث عن مهمة جديدة
  - = Method 2
    - Threads في الـ pool و البرنامج الرئيسي يسحب الـ Threads و يربطهم بالمهام
    - يمكن إضافة Thread جديدة عندما لا Threads بالـ pool و لدينا مهمة ضرورية

Thread Pools

```
Class WorkQueue{
        list queue = new linkedList();
        public synchronized void add (Socket s) {
                queue.addLast(r);
                queue.notifyAll();
        }
}
Class TCPThreadPool{
    public static void main(String[] args){
        WorkQueue q = new WorkQueue();
        threads = new PoolWorker[nThreads];
        for(int i=0; i<nThreads; i++){
                threads[i] = new PoolWorker(q);
                threads[i].start();
        }
    ServerSocket ws = new ServerSocket(6789);
    while(true){
        Socket s = ws.accept();
        q.add(s);
        }
    }
}
```

```
class PoolWorker extends Thread{
    WorkQueue queue;
    public PoolWorker(WorkQueue queue){
        this.queue =queue;
    }
    public void run() {
        Socket r;
        while(true){
            synchronized(queue){
                while(queue.isEmpty()){
                    try{queue.wait();}
                    catch(InterruptedException e) {}
                }
                r = (Socket) queue.removeFirst();
                // process the request r.
            }

        }
    }
}
```

Telnet Client

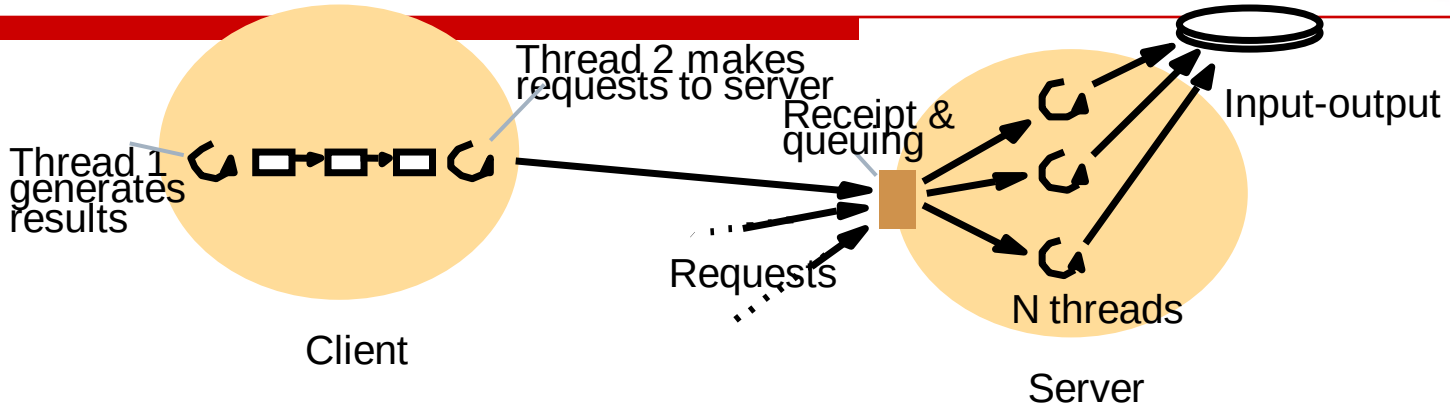


Telnet Client

```
import java.io.*;
class RedirectStream extends Thread{
        BufferedReader rs = null;
        PrintStream ws = null;
        public RedirectStream(InputStream is, OutputStream os)
        {
                rs = new BufferedReader(is);
                ws = new PrintStream(os);
                start();
        }
        public void run() {
                String s;
                while((s=rs.readLine()) != null) {
                                ws.println(s);
                }
        }
}
```

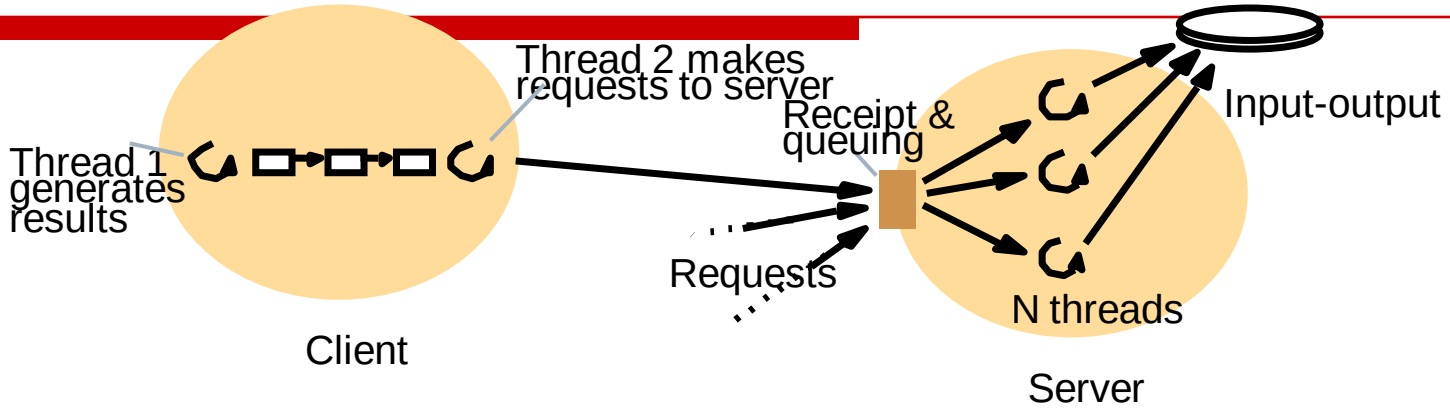
```
import java.io.*;
import java.net.*;
public class Telnet{
        public static void main(String[] args) {
                Socket s = new Socket(TelnetServer, port);

                RedirectStream SktToScreen = new
                RedirectStream(s.getInputStream(), System.out);

                RedirectStream KeyboardToSkt = new
                RedirectStream(System.in, s.getOutputStream());

                SktToScreen.join();
                keyboardToSkt.Interrupt();
                }
}
```

Multi-threading (Ex 1/2)



- Request = 2ms processing + 8 ms I/O
  - Mono-thread Server ➔ throughput = 100 client / sec
  - Multi-threaded Server (**2** threads) = when 1 thread blocked on I/O. Then, the another can process another request ➔ throughput = 125 client / sec
  - Multi-threaded Server (**2** threads) + (caches, hit rate = 75% → 2ms disk access, 0.5 ms search in cashes) ➔ throughput = 400 client / sec

Multi-threading (Ex 2/2)

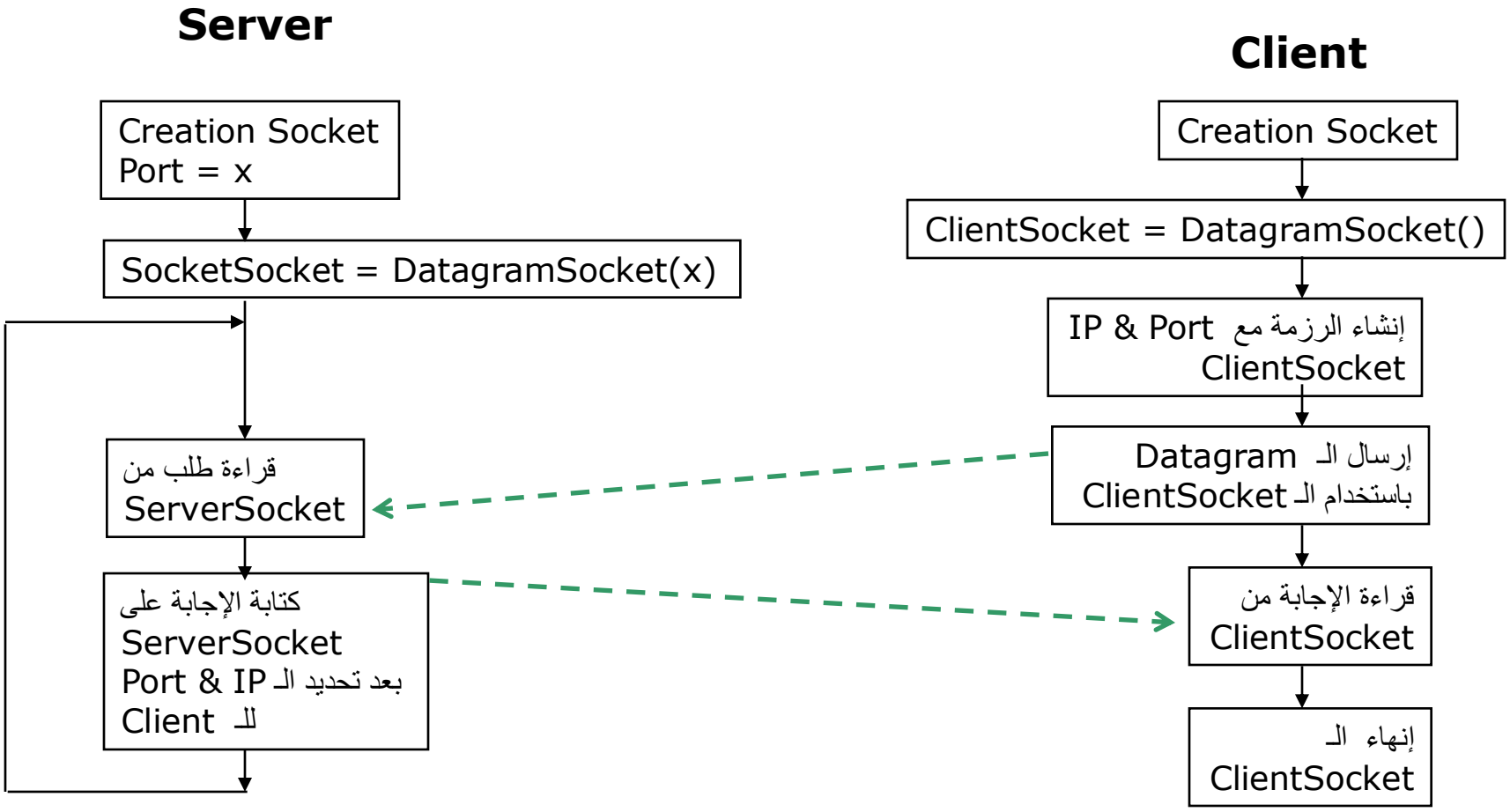


- Request = 2ms processing + 8 ms I/O
  - Multi-threaded Server (**2** threads) + (caches, hit rate = 75% → 2ms disk access, 0.5 ms search in cashes) + 2 processors ➔ throughput = 444 client / sec
  - Multi-threaded Server (more threads of 3) + (caches, hit rate = 75% → 2ms disk access, 0.5 ms search in cashes + 2 processors ➔ throughput = 500 client / sec

Connection-less Protocols

- الهدف = تحويل المعطيات من نقطة لنقطة
- UDP (User Datagram Protoco)
  - يرسل الرزم datagram packets من المرسل للمستقبل بشكل منفصل بدون إعلام و بدون تكرار
  - يحول المعلومات بطريقة غير موثوقة و غير مرتبة
    - الرزم يمكن أن تضيع ببساطة لعدم وجود مساحة تخزين
  - لا تحكم بتدفق المعطيات عبر الشبكة
  - لا تكيف في حالات اغتصاص الشبكة
  - عملية الإرسال non-blocking
  - عملية الاستقبال blocking
    - عملية الاستقبال + Thread = non-blocking
  - المبرمج يهتم بتخصيص مصفوفة من البايتات لاستقبال الرسالة
- التطبيقات التي تستخدم UDP :
  - Telephone on Internet, TeleConference, ..DNS
- **ملاحظة** : استخدم UDP + آلية لاكتشاف و معالجة الأخطاء خاصة بالتطبيق ذاته <> TCP
  - ميزة = تحسين من أداء الـ TCP في شبكات موثوقة
  - سيئة = كلفة في التطوير + أثر سلبي على مفتوحية الأنظمة

JAVA & UDP


Server
Client
Creation Socket
Port = x
SocketSocket = DatagramSocket(x)
قراءة طلب من
ServerSocket
كتابة الإجابة على
ServerSocket
Port & IP بعد تحديد الـ
Client للـ
Creation Socket
ClientSocket = DatagramSocket()
إنشاء الرزمة مع IP & Port
ClientSocket
إرسال الـ Datagram
باستخدام الـ ClientSocket
قراءة الإجابة من
ClientSocket
إنهاء الـ
ClientSocket

JAVA – UDP Client

```
import java.io.*;
import java.net.*;
public class UDPClient {
        public static void main(String[] args)throws Exception {
                BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
                try
                {
                        DatagramSocket clientSocket = new DatagramSocket();

                        InetAddress lclHost = InetAddress.getLocalHost();
                        System.out.println(lclHost);

                         byte[] sendData = new byte[256];
                         String s = inFromUser.readLine();
                         sendData = s.getBytes();
                         DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, lclHost, 7777);
                         clientSocket.send(sendPacket);

                         byte[] receiveData = new byte[512];
                         DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
                         clientSocket.receive(receivePacket);
                         String result = new String(receivePacket.getData());

                         System.out.println(result);
                }
                catch(Exception e)
                {e.printStackTrace();}
        }
}
```

JAVA – UDP Server

```
import java.io.*;
import java.net.*;
public class UDPServer {
        public static void main(String[] args) throws Exception {
                DatagramSocket ServerSocket = new DatagramSocket(7777);
                byte[] sendData = new byte[512];
                byte[] receiveData = new byte[256];

                while(true)
                {
                      System.out.println("server wait .... ");
                      try{
                         DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
                         ServerSocket.receive(receivePacket);
                         String s = new String(receivePacket.getData());
                         s = "I received " + s;
                         System.out.println(s);

                         InetAddress ia = receivePacket.getAddress();
                         int port = receivePacket.getPort();
                         sendData = s.getBytes();
                         System.out.println("response send to " + ia + " on " + port);
                         DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, ia, port);
                         ServerSocket.send(sendPacket);
                         }
                      catch(Exception e) {e.printStackTrace();}
                }
        }
}
```

Applications & Transport Services

- فقدان المعطيات :
  - Audio app متسامحة مع فقدان المعطيات
  - FTP, Telnet تتطلب تحويل موثوق للمعطيات ١٠٠%
- تأخر الإرسال :
  - Net telephone, games تتسامح مع تأخير قصير لتكون فعالة
- عرض المجال :
  - Multimedia app تتطلب عرض مجال أصغري لتكون فعالة
  - Apps يمكنها العمل مع عرض المجال المتوفر

Applications & Transport Services

APP	Data lose	Band width	delay
FT Email Web	No lose No lose Toler Lose	Elastic Elastic Elastic	X X X
Audio/video Real-Time	Toler Lose	Audio 5k-1M Video 10k-5M	100 msec
Audio/video Stock	Toler Lose	Audio 5k-1M Video 10k-5M	Some seconds
Interactive games	Toler Lose	Some kbps	100 ms
Finance app	No lose	elastic	yes/no

Group Communication Protocols (1)

- الهدف = تحويل المعطيات من نقطة لعدة نقاط (كل عضو من المجموعة)
- تنفيذ خدمة بين مجموعة من الفعاليات على عدة حواسيب
  - **Fault Tolerance** = الزبون يرسل نفس الطلب إلى N Server(s)
  - **Performance Enhancing** = Caching + إرسال التحديثات لكل الفعاليات التي تدير الـ replicas
  - **Finding discovery servers** = الزبائن يستخدمونها لتحديد خدمات الـ discovery (لتسجيل واجهاتهم أو البحث عن واجهات لخدمات)
  - **Event notification** = إعلام الفعاليات لحدث ما (مثلاً وصول رسالة جديدة)
- IP multicast = يرسل رزمة (UDP datagram packet) إلى حواسيب Multicast Group على Port محدد
- عناوين الـ Multicast group من class D
  - 224.0.0.0 → 239.255.255.255
  - عنونة ديناميكية = يمكن الانضمام أو ترك المجموعة بشكل آلي
- أخطاء الـ Multicast = أخطاء الـ UDP

Group Communication Protocols (2)

- TTL(Time To Live)
  - 0 = local host
  - 1 = local subnet
  - 16 = local Campus
  - 32 = HB sites in country
  - 48 = sites in country
  - 64 = sites in continent
  - 128 = HB sites in worldwide
  - 255 = all sites worldwide

API JAVA – Multicast Receiver

```
import java.io.*;
import java.net.*;
public class GroupReceiver {
      public static void main(String[] args) throws Exception {
            MulticastSocket ms = new MulticastSocket(4000);
            InetAddress ia = InetAddress.getByName("224.2.2.2");
            ms.joinGroup(ia);
            byte[] buffer = new byte[8192];
            while(true) {
                  System.out.println("Receiver wait .... ");
                  try{
                      DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
                      ms.receive(dp);
                      String s = new String(dp.getData());
                      System.out.println(s);
                  }catch(Exception e)
                  {e.printStackTrace();}
            }
      }
}
```

API JAVA – Multicast Sender

```
import java.io.*;
import java.net.*;
public class GroupSender {
      public static void main(String[] args)throws Exception {
            try {
                MulticastSocket ms = new MulticastSocket();
                ms.setTimeToLive(0);
                InetAddress ia = InetAddress.getByName("224.2.2.2");
                int port = 4000;
                byte[] data = "some data".getBytes();
                DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
                ms.send(dp);
            }
            catch(Exception e)
            {e.printStackTrace();}
      }
}
```

Secure Communication (1)

- **Symmetric secret key encryption**
  - Sender & Receiver must have the same key to encrypt & decrypt the data
  - Sender send the key to the receiver
  - Anybody can read the key ➔ decrypt the msg
- **Asymmetric encryption**
  - Public key to encrypt the msgs
  - Private key (**secret**) to decrypt the msgs

Secure Communication (2)

- **Confidenciality**
  - **A** send to **B**. ➔ **B** send clearly his public key to **A**
  - **A** use the key to encrypt the msg & send to **B**
  - Only **B** can decrypt the msg (by privte key) ➔ secret msg
- **Authentication**
  - **A** encrypt the msg by his private key & send to **B**
  - **B** decrypt msg by B's public key ➔ msg come from **A**
- **Authentication & Message Integrity**
  - **A** encrypt msg by his private key
  - **A** encrypt msg by B's public key & send the msg to **B**

Secure Communication (3)

- Asymmetric encryption is more expensive than symmetric encryption
  - **A** send symmetric key to **B**.
  - **B** use his private key to decrypt the msg
  - **A, B** have symmetric key ➔ they can perform secret fast communication

Secure Communication (4)

- man-middle-attack problem
  - **X** can read **B**'s public key and replace it by his key (comm pb & not encryption pb).
  - So, when **A** send to **B**. **X** can use his private key to decrypt the msg
- Solution ~
  - ما الذي يضمن بأن المفاتيح العامة لكل من A و B تعود فعلياً لهما.
    - الحل يكمن بوجود شخص ثالث C موثوق و الذي يتأكد من شخصية A مثلاً و عندها فإنه ينشر مفتاح A العام بعد تشفيره باستخدام مفتاح C الخاص و الذي ندعوه بالـ certificate.
    - B يستطيع الآن التأكد من أن المفتاح العام عائد لـ A باستخدامه المفتاح العام لـ C لفك تشفيره

JAVA – Secure Sockets

- keytool = generate public keys

```
E:\Ordi_fac\RMI\Sokets>keytool -genkey -alias ourstore -keystore mykyes.keys
Enter keystore password:  secret
What is your first and last name?
  [Unknown]:  bassem kosayba
What is the name of your organizational unit?
  [Unknown]:
.................
......
Enter key password for <ourstore>
        (RETURN if same as keystore password):
```

- Certification at (http://www.verisign.com/)
  - About 15$ per year
- free at (http://java.sun.com/products/jsse/)
  - Keys file = testkeys
  - Password = "passphrase"

API JAVA – Secure Client

```
import java.io.*;
import java.security.*;
import javax.net.ssl.*;
public class SecureClient  {
  public static void main(String[] args)   {
    int port = 6666;
    try {
          SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory.getDefault( );
          InetAddress lclHost = InetAddress.getLocalHost();
          SSLSocket socket = (SSLSocket) factory.createSocket(lclHost, port);

           // enable all the suites
          String[] supported = socket.getSupportedCipherSuites( );
          socket.setEnabledCipherSuites(supported);

          Writer out = new OutputStreamWriter(socket.getOutputStream( ));
          out.write("hello secret socket .... "); out.flush( );

          // read response
          BufferedReader in = new BufferedReader( new InputStreamReader(socket.getInputStream( )));
          String s = in.readLine();
          System.out.println(s);

      out.close( );  in.close( );  socket.close( );
    }
    catch (IOException ex) { System.err.println(ex); }
  }
}
```

API JAVA – Secure Server

```
import java.net.*; import java.io.*; import java.util.*; import java.security.*; import javax.net.ssl.*;
import javax.net.*;
public class SecureServer {
  public final static String algorithm = "SSL";
  public static void main(String[] args) {
    try {
        SSLContext context = SSLContext.getInstance(algorithm);
        KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");
        KeyStore ks = KeyStore.getInstance("JKS");
        char[] password = "secret".toCharArray( );
        ks.load(new FileInputStream("mykyes.keys"), password);
        kmf.init(ks, password);
        context.init(kmf.getKeyManagers( ), null, null);
        SSLServerSocketFactory factory = context.getServerSocketFactory( );
        SSLServerSocket server = (SSLServerSocket) factory.createServerSocket(6666);
        String[] supported = server.getSupportedCipherSuites( );
        server.setEnabledCipherSuites(supported);
                while (true) {
                        System.out.println("Server wait ....");
                        Socket theConnection = server.accept( );
                        BufferedReader in = new BufferedReader(new
                                                InputStreamReader(theConnection.getInputStream()));
                        String s = in.readLine();
                        System.out.println(s);
                        Writer out = new OutputStreamWriter(theConnection.getOutputStream( ));
                        out.write(s + " has been received ...");
                        theConnection.close( );
                } // end while
        } catch (Exception ex) { ex.printStackTrace( );
    } // end main
} // end server
```

Questions ?