



LABORATORIO DI RETI DI CALCOLATORI

**Socket: concetti fondamentali,
creazione connessione TCP**

Bibliografia

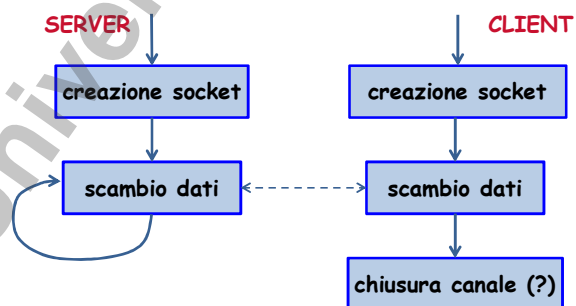
- ❖ slide della docente
- ❖ *testo di supporto*: D. Maggiorini, “Introduzione alla programmazione client-server”, Pearson Ed., 2009
 - ❑ cap.4 (tutto)
 - ❑ cap.5 (tutto)
 - ❑ cap.7 (tutto)
 - ❑ cap.8 (tutto)
- ❖ *Link utili*:
 - ❑ <http://docs.oracle.com/javase/tutorial/networking/index.html>
 - ❑ <http://docs.oracle.com/javase/6/docs/>

socket: tipo di servizio

- ❖ in Internet determina protocolli
- ❖ **connection-oriented**: trasferimento di *stream di byte*
 - ❑ garanzia di ordine nell'arrivo dei byte
 - ❑ non preservato confine di messaggio
 - da origini della rete (comandi Unix)
 - ❑ può anche essere affidabile → TCP
- ❖ **connection-less**: trasferimento di *datagram*
 - ❑ *best effort*:
 - non garantito ordinamento né arrivo
 - ❑ preservato confine messaggi
 - ❑ può anche essere affidabile

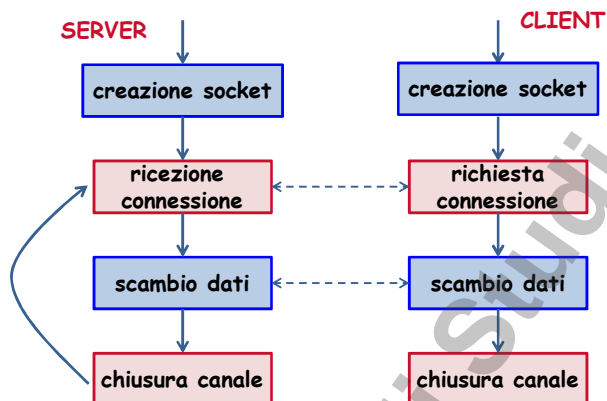
UDP / IP

connectionless: fasi comunicazione



- ❖ creazione socket in server include scelta porta (well known)
- ❖ creazione socket in client include scelta porta, eventualmente fatta da sistema operativo
 - ❑ stessa socket può essere usata anche per altri server

conn-oriented: fasi comunicazione



- ❖ ...con interleaving per server concorrente
- ❖ ...con delega a server secondari per server multi-processo

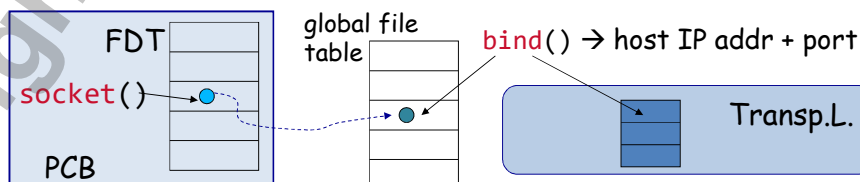
Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

5/15

Primitive di servizio

- ❖ *dove operano? tra quali livelli?*
- ❖ differenti a seconda del servizio richiesto
 - ❑ diversa visibilità sistema fornita all'utente
 - ❑ diversa complessità proto di livello superiore
- ❖ *inizializzazione locale* /* libreria C */
 - ❑ `socket()`: creazione end-point
 - ❑ `bind()` a indirizzo noto all'esterno



Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

6/15

Primitive di servizio TCP



❖ **associazione** <proto, IPsrc, port-src, IPdst, port-dst>

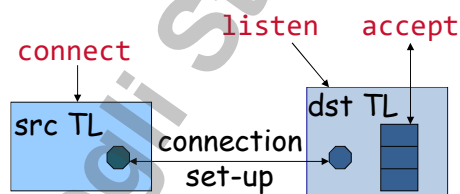
❖ connection set-up [conn.oriented]

❑ **listen** : *passive open*

❑ **connect** : *active open*

❑ **accept**

❖ per ogni connessione TCP ricorda un **connection record**



Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

7/15

limitazioni Java socket

cosa **non** possiamo fare?

❖ non supportato il dominio AF_UNIX

❖ non è possibile interagire direttamente con IP in Java

❖ connection-oriented, connectionless, domini di indirizzi...

❑ Java considera che gli indirizzi possano essere solo IP

❑ e che i servizi/protocolli siano solo TCP e UDP

❖ meccanismi particolari per costruire server concorrenti...

❖ cosa possiamo fare?

❑ possiamo lavorare anche con IPv6

Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

8/15

servizio connection-oriented

❖ ripasso fasi...

1. creazione socket
2. binding → gestione indirizzi host + #porta
3. connessione client e server
4. scambio dati (*byte stream*)
5. chiusura

client

server

❖ classi utilizzate: *package java.net*

- ❑ class `Socket` (client); `ServerSocket` (...server)
- ❑ class `InetAddress` (indirizzi host); `InetSocketAddress`

Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

9/15

1. creazione socket lato client

```
EP_Socket > esempio1.java > M main(String[] args)
1 import java.net.Socket;
2
3 // codice client per servizio connection-oriented (TCP)
4
5 public class esempio1
6 {
7     public void main(String[] args)
8     {
9         Socket sClient;
10
11         sClient = new Socket();
12
13         // altro codice...
14     }
15 }
```

diversi altri costruttori
disponibili

❖ abbiamo creato una struttura *del processo* per gestire il *punto terminale del canale* di comunicazione

❖ adesso dobbiamo indicare a quale indirizzo di rete corrisponde

Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

10/15

2. binding esplicito

- ❖ metodo void `Socket.bind(SocketAddress bindpoint)`
 - ❑ colleghiamo struttura processo a informazioni per S.O.
 - ❑ `SocketAddress` è superclasse di `InetSocketAddress`

```
19 sClient = new Socket();
20 try {
21     ia = InetAddress.getLocalHost();
22     isa = new InetSocketAddress(ia, 0); // S.O. sceglie #port libero
23     sClient.bind(isa);
24     System.out.println("Porta allocata: " + sClient.getLocalPort());
25     Thread.sleep(120 * 1000);
26 } catch (Exception e) {
27     e.printStackTrace(); }
    dopo associazione
```

- ❖ `#port 0` lascia scelta porta libera al S.O. (*effimere*)
 - ❑ non va tanto bene per il server...
- ❖ comandi *netstat* oppure *lsof* mostrano stato socket
 - ❑ CLOSED : non è connessa ad alcun server

Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

11/15

implementazione server e connessione

- ❖ creazione socket con due costruttori di `ServerSocket`:
 - ❑ `ServerSocket()` oppure `ServerSocket(int port)`
 - ❑ il primo crea socket non connessa → *serve bind successiva*
 - manipolazione indirizzi come per caso client
 - ❑ nel secondo caso, `#port` può essere 0
 - si crea già *coda* per ospitare richieste connessione pendenti
 - stato socket risulta LISTEN
- ❖ connessione: il server si mette in attesa di richieste
 - ❑ `Socket` `ServerSocket.accept()`
 - bloccante in attesa di clienti
 - crea nuova Socket per comunicare con specifico client
 - ricordate discorso su *associazione*?

Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

12/15

3. creazione connessione (server)

```
EP_Jsocket > j es1SrvIter.java > M main(String[] args)
1 import java.net.ServerSocket;
2 import java.net.Socket;
3 import java.io.IOException;
4
5 // codice server per servizio connection-oriented (TCP)
6
7 public class es1SrvIter
8 {
9     public static void main(String[] args)
10    {
11        ServerSocket sSrv;
12        Socket toClient;
13        try {
14            sSrv = new ServerSocket(0);
15            System.out.println("Indirizzo: " + sSrv.getInetAddress()
16                               + "; porta: " + sSrv.getLocalPort());
17            toClient = sSrv.accept();
18            System.out.println("Ind Client: " + toClient.getInetAddress()
19                               + "; porta: " + toClient.getPort());
20            Thread.sleep(240 * 1000);
21        } catch (Exception e) {
22            e.printStackTrace();
23        }
24    }
25 }
```

visualizza indirizzo (di trasporto) locale che è stato associato alla socket

visualizza indirizzo (di trasporto) del client

Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

13/15

3. creazione connessione (client)

```
EP_Jsocket > j esempio1.java > class esempio1
1 import java.net.Socket;
2 import java.net.InetAddress;
3 import java.net.InetSocketAddress;
4
5 import java.net.UnknownHostException;
6 import java.io.IOException;
7
8 // codice client per servizio connection-oriented (TCP)
9
10 public class esempio1
11 {
12     public static void main(String[] args)
13     {
14         Socket sClient;
15         InetAddress ia; // IP address SERVER
16         InetSocketAddress isa; // socket address SERVER
17
18         sClient = new Socket();
19         try {
20             ia = InetAddress.getLocalHost();
21             isa = new InetSocketAddress(ia, 57195); // porta server da inserire...
22             sClient.connect(isa);
23             System.out.println("Porta locale: " + sClient.getLocalPort());
24             System.out.println("Indirizzo: " + sClient.getInetAddress()
25                                + "; porta: " + sClient.getPort());
26             Thread.sleep(120 * 1000);
27         } catch (Exception e) {
28             e.printStackTrace();
29         }
30     }
31 }
```

CLIENT CHE DIALOGA CON SERVER SU STESSO HOST (ma usando la rete...)

deve essere IP address server

deve essere la porta stampata come locale dal server

stampa porta locale e indirizzo server

Elena Pagani

LABORATORIO di Reti di Calcolatori – A.A. 2023/2024

14/15

3. creazione connessione

- ❖ metodo void `Socket.connect(SocketAddress peer)`
 - ❑ esecuzione *three-way handshake* (→ Teoria)
 - ❑ esegue contestualmente anche bind implicito
- ❖ indirizzo locale server è 0.0.0.0 che indica *any*
 - ❑ *attenzione che `getInetAddress` su `ServerSocket` mostra indirizzo locale; su `Socket` mostra indirizzo remoto*
- ❖ lo output di `lsof` mostra 3 socket sul sistema:

```
{ java pid_srv user IP_srv TCP *:57220 (LISTEN)
  java pid_srv user IP_srv TCP nome_srv:57220->
  nome_cli:57223 (ESTABLISHED)
  java pid_cli user IP_cli TCP nome_cli:57223->
  nome_srv:57220 (ESTABLISHED)
```