



Networking: Socket

Programmazione III

Matteo Baldoni e Alberto Martelli
con integrazioni di Liliana Ardissono



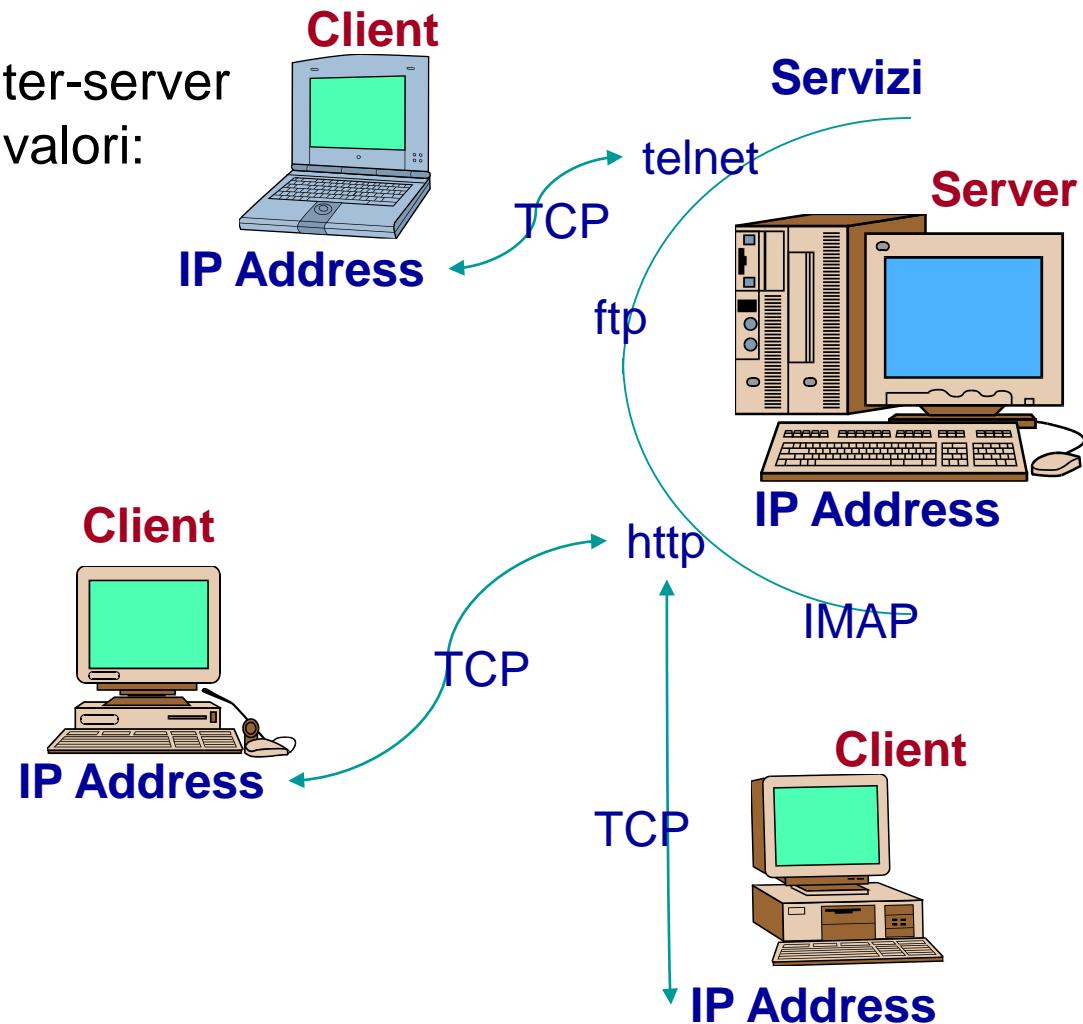
Architettura Client/Server

Un servizio presso un computer-server è **identificato** dai seguenti valori:

- **IP** (32 bit o 128 bit)
- **Port** (16 bit)

Alcuni tipici servizi:

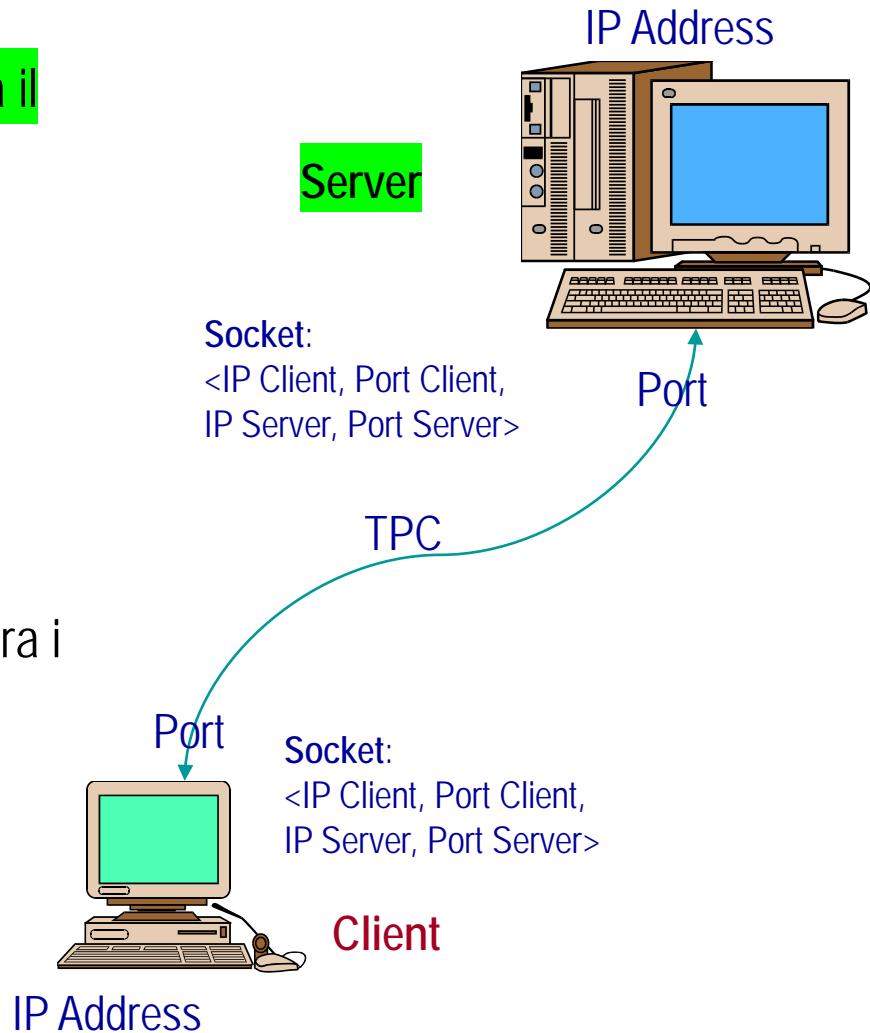
- telnet
- http
- ftp
- SMTP, IMAP4
- NFS, DNS, NIS, ...





Socket

- È una *astrazione software* che rappresenta il terminale di una connessione tra due computer
- Per ogni connessione esiste un *socket* in ognuno dei computer coinvolti
- Il client effettua la richiesta di una connessione ad un server per un servizio collegato ad una determinata porta
Se la richiesta è accettata la connessione tra i due applicativi dei due computer è stabilita





Socket e Stream in Java

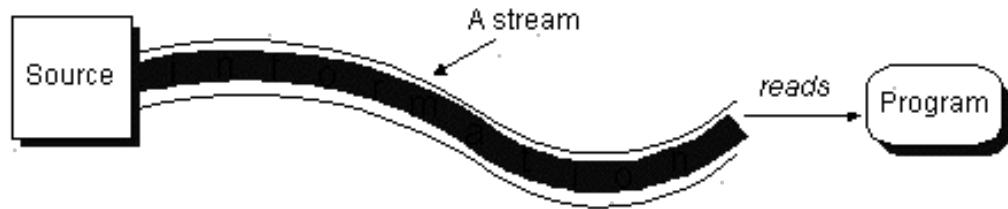
- Da ogni socket è possibile ottenere uno *stream* di *input* ed uno di *output*
- L'uso dei socket è trasparente: **readLine** e **println**

BufferedReader

```
in = new BufferedReader(  
    new InputStreamReader(  
        socket.getInputStream()));
```

PrintWriter

```
out = new PrintWriter(  
    new BufferedWriter(  
        new OutputStreamWriter(  
            socket.getOutputStream()),  
        true));
```

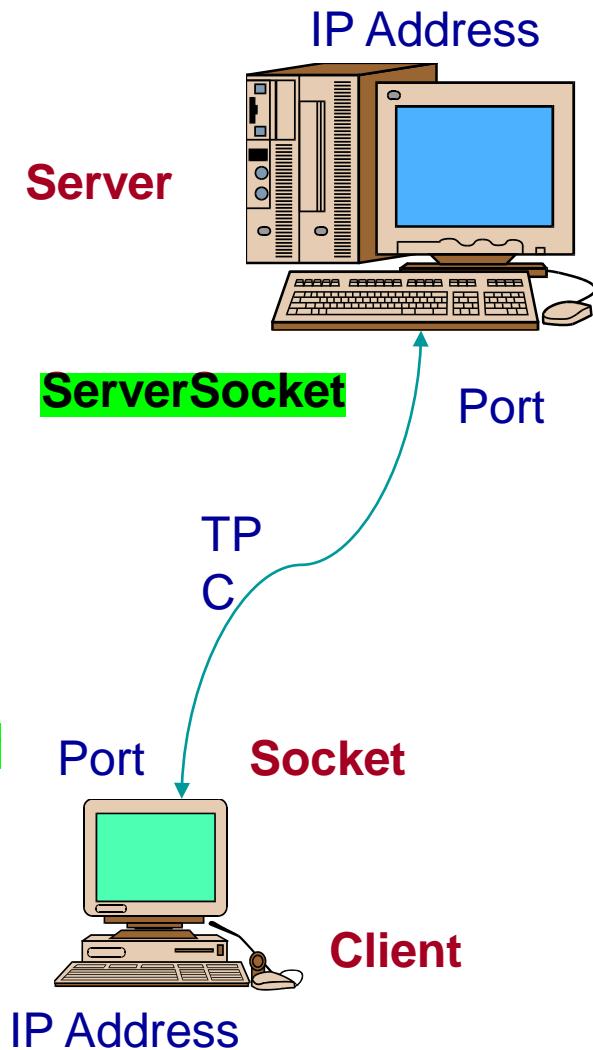


Abilita il *flush* immediato



Socket in Java

- Package da importare `java.net`
- La classe `java.net.Socket` implementa un “*client socket*”
 - ◆ richiede l’indirizzo IP e il numero della porta del servizio a cui vogliamo connetterci
- La classe `java.net.ServerSocket` implementa un “*server socket*”
 - ◆ richiede il numero della porta a cui vogliamo associare il servizio
 - ◆ crea un “*server*” che attende le richieste di connessione
 - ◆ restituisce un socket nel momento in cui accetta un collegamento completamente istanziato nei parametri di collegamento (IP e Porta locale e remota).





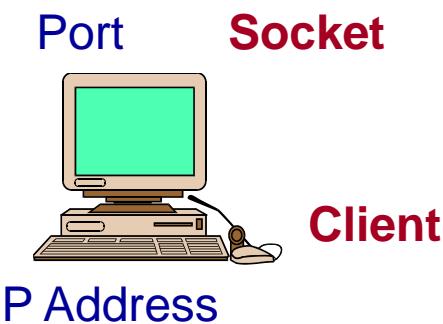
Socket (Client)

```
Socket socket = new Socket(IP, PORT);  
[...]  
socket.close();
```

- Crea una connessione con un'applicazione server (in attesa in una `accept()`) e restituisce il relativo socket
- Le eccezioni che possono essere sollevate sono del tipo `IOException` (da catturare e gestire!)

E`un valore intero, in genere non inferiore a 1024 (riservati al sistema) e rappresenta il numero della porta a cui Vogliamo connetterci

E`un oggetto del tipo `java.net.InetAddress` contenente l'indirizzo IP della macchina a cui vogliamo connetterci ad esempio `speedy/130.192.241.1`





Inet Address

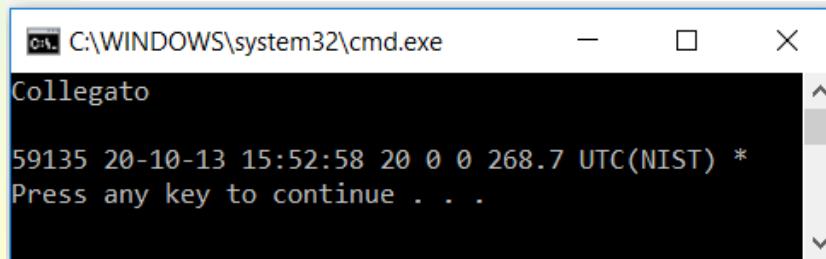
```
public class SocketTest1 {  
    public static void main(String[] args) {  
        String host = "www.unito.it";  
        try {  
            InetAddress local = InetAddress.getLocalHost();  
            InetAddress addr = InetAddress.getByName(host);  
            System.out.println("Locale: indirizzo IP: " + local);  
            System.out.println("Remoto: indirizzo IP: " + addr);  
        }  
        catch(UnknownHostException e)  
        {System.out.println(host +"sconosciuto");}  
    }  
}
```

The screenshot shows a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The window contains the following text:
Locale: indirizzo IP: lilianasantech/192.168.1.5
Remoto: indirizzo IP: www.unito.it/130.186.27.242
Press any key to continue . . .



Un esempio: connessione a un server

```
public class SocketTest {  
    public static void main(String[] args){  
        try {  
            Socket s =  
                new Socket("time-c.nist.gov", 13);  
            System.out.println("Collegato");  
            try {  
                InputStream inStream = s.getInputStream();  
                Scanner in = new Scanner(inStream);  
                while (in.hasNextLine()) {  
                    String line = in.nextLine();  
                    System.out.println(line);  
                }  
            }  
            finally  
            { s.close();}  
        }  
        catch (IOException e)  
        { e.printStackTrace();}  
    }  
}
```



- Si collega a un servizio che fornisce l'ora esatta

- La stringa **time-c.nist.gov** viene convertita nell'indirizzo IP del server

Il servizio di ora è disponibile sulla porta 13



Astrazione: connessione URL

```
public class URLReader {  
    public static void main(String[] args) throws Exception {  
        URL yahoo = new URL("http://www.yahoo.com/");  
        BufferedReader in = new BufferedReader(  
            new InputStreamReader(yahoo.openStream()));  
        PrintWriter out = new PrintWriter(  
            new FileWriter("esempio.html"));  
        String inputLine;  
        while ((inputLine = in.readLine()) != null)  
            out.println(inputLine);  
        in.close();  
        out.close();  
    }  
}
```



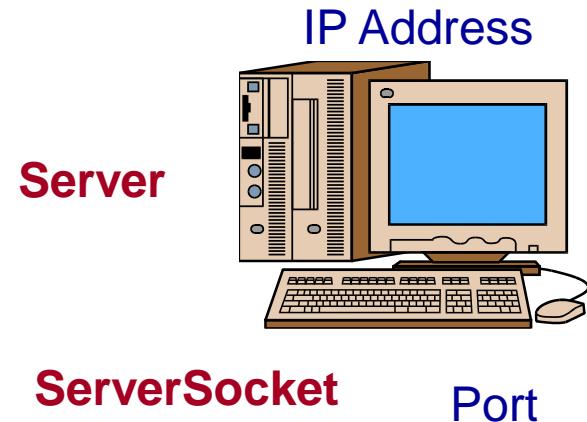
Scarica un file HTML dal sito di yahoo

- Per effettuare delle connessioni sia ad un server di posta che ad un server web è possibile utilizzare delle classi che realizzano delle astrazioni sul tipo di protocollo utilizzato
- L'uso dei socket è nascosto come quello del protocollo HTTP



ServerSocket

- Il metodo `accept()` si mette in attesa di una richiesta di connessione da un socket client
- Le eccezioni che possono essere sollevate sono del tipo `IOException` (da catturare e gestire!)



```
ServerSocket s = new ServerSocket(PORT);
[...]
Socket socket = s.accept();
System.out.println("Accettato socket " + socket);
[...]
socket.close();
[...]
s.close();
```

Crea un nuovo socket sul server per accettazione di una richiesta di connessione

Chiude il socket di una connessione precedentemente accettata

Chiude il server



ServerSocket e Socket: gestione errori

- E` importante che i socket siano propriamente chiusi al termine di un'applicazione in quanto questi sono una risorsa di rete condivisa del sistema
- La chiusura dei socket deve essere eseguita indipendentemente dal flusso di esecuzione per questo e` bene utilizzare il costrutto *try-finally* per garantirne la chiusura
- La stessa considerazione vale anche per il ServerSocket

```
[...]
try {
    [...]
    String str = in.readLine();
    [...]
    out.println("OFFERTA");
    [...]
}
catch(IOException e) {
    System.err.println("IO Exception" + e);
}
finally {
    try {
        socket.close();
    } catch(IOException e) {
        System.err.println("Socket not closed");
    }
}
[...]

try {
    Socket socket = s.accept();
    [...]
} finally {
    s.close();
}
```





Un esempio: un server Eco (ripetizione)

Si vuole realizzare un server che invia al client l'eco di quanto riceve dal client.

Il server si mette in attesa della connessione di un client sulla porta 8189.

ServerSocket s = new ServerSocket(8189);

Dopo aver stabilito una connessione, il metodo **accept** del **ServerSocket** restituisce un **Socket** che rappresenta la connessione.

Socket incoming = s.accept();

Da questo oggetto si possono ricavare gli stream di input e output che collegano client e server

InputStream inStream = incoming.getInputStream();
OutputStream outStream = incoming.getOutputStream();

Un esempio: un server Eco (ripetizione)



```
public class EchoServer {  
    public static void main(String[] args ) {  
        try{  
            ServerSocket s = new ServerSocket(8189);  
            Socket incoming = s.accept( );  
            try {  
                InputStream inStream = incoming.getInputStream();  
                OutputStream outStream = incoming.getOutputStream();  
                Scanner in = new Scanner(inStream);  
                PrintWriter out = new PrintWriter(outStream, true);  
                out.println( "Hello! Enter BYE to exit." );  
  
                boolean done = false;  
                while (!done && in.hasNextLine()) {  
                    String line = in.nextLine();  
                    out.println("Echo: " + line);  
                    if (line.trim().equals("BYE"))  
                        done = true;  
                }  
            } finally {incoming.close();}  
        } catch (IOException e){e.printStackTrace();}  
    }  
}
```



Come provare il server

Per usare il server si può scrivere un client che si connette alla porta 8189 all'indirizzo IP a cui si trova il server.

Se il server si trova sullo stesso computer del client, si può usare l'indirizzo 127.0.0.1, che indica il computer locale.

Si veda l'esempio **EchoClient**.





Come servire più client

Il server descritto prima può servire solo un client. Se un altro client cerca di connettersi, rimane bloccato perché il server sta già gestendo un altro client.

Per servire più client si posso usare i thread. Ogni volta che il server ha stabilito una nuova connessione con un client, viene avviato un thread che si prende cura della connessione fra il server e quel client. Il programma principale del server può mettersi in attesa della connessione successiva.

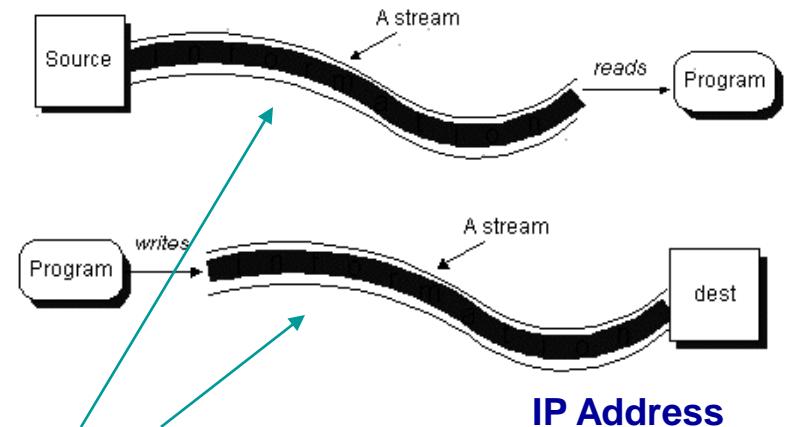
```
while (true)
{
    Socket incoming = s.accept();
    Runnable r = new ThreadedEchoHandler(incoming);
    Thread t = new Thread(r);
    t.start();
}
```



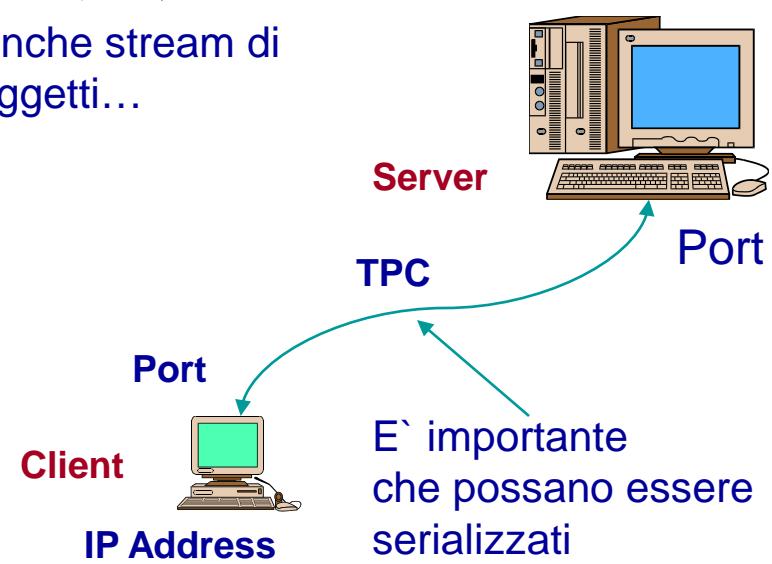


Stream di oggetti e Socket

- Nell'esempio precedente i dati venivano trasmessi come stringhe da un client ad un server e viceversa
- In Java si puo` pero` utilizzare anche stream di oggetti (ObjectInputStream e ObjectOutputStream)
- In Java e` possibile utilizzare stream di oggetti tramite socket
- Gli oggetti devono implementare l'interfaccia java.io.Serializable



Anche stream di oggetti...





Stream di oggetti e Socket

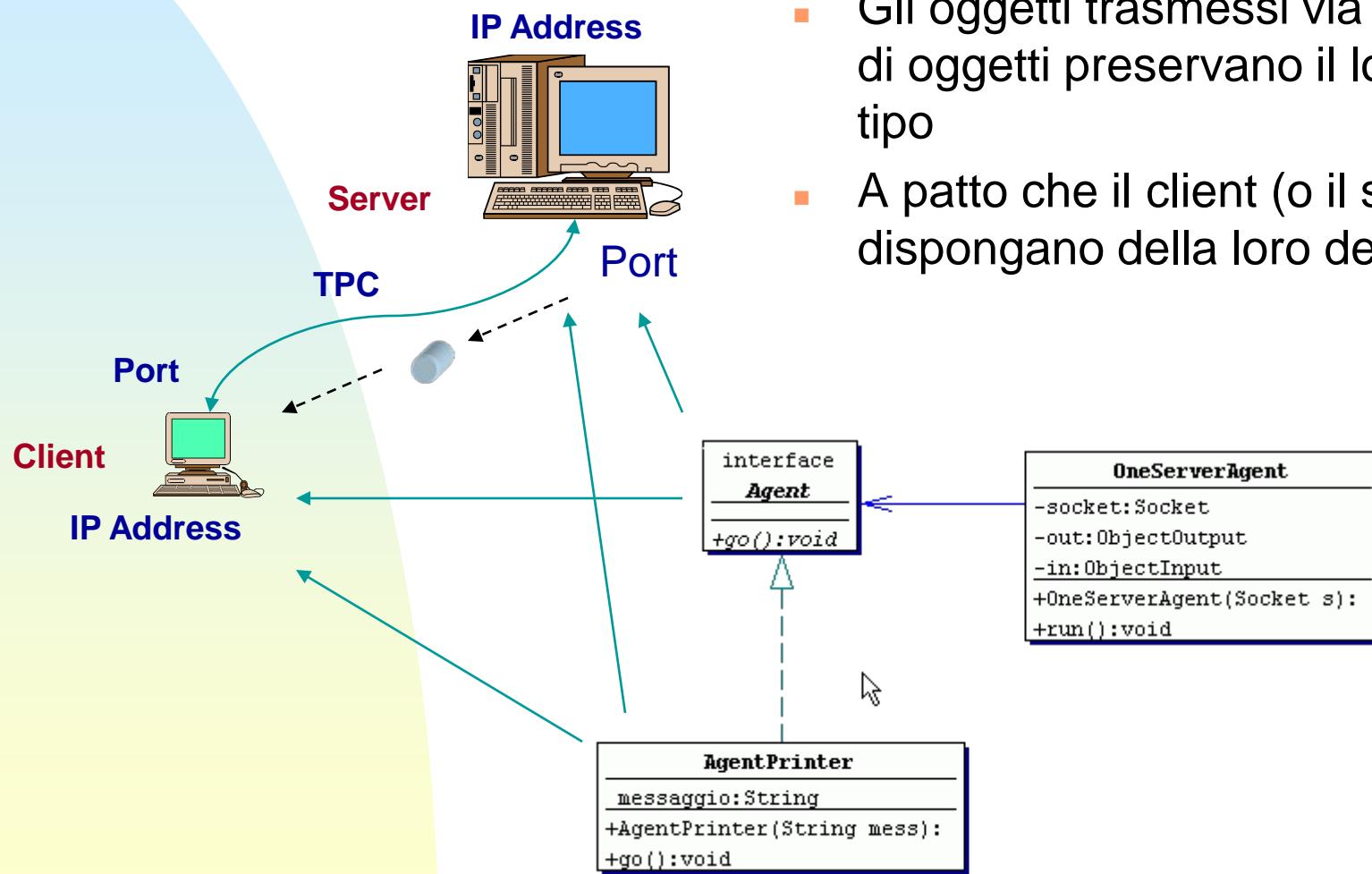
```
[...]  
ServerSocket s = new ServerSocket(PORT);  
[...]  
Socket socket = s.accept();  
[...]  
ObjectInputStream in =  
    new ObjectInputStream(socket.getInputStream());  
ObjectOutputStream out =  
    new ObjectOutputStream(socket.getOutputStream());  
[...]  
Offerta nuova = (Offerta)in.readObject();  
[...]  
out.writeObject("accettata");  
[...]
```

← Server

- **Importante:** la dichiarazione nel client è analoga MA è l'oggetto di tipo ObjectOutputStream va creato prima dell'oggetto di tipo ObjectInputStream pena il deadlock (non è un bug!)



Polimorfismo via socket?



- Gli oggetti trasmessi via stream di oggetti preservano il loro vero tipo
- A patto che il client (o il server) dispongano della loro definizione