Reti di Calcolatori:
Internet, Intranet e Mobile Computing
a.a. 2007/2008

http://www.di.uniba.it/~lisi/courses/reti/reti0708.htm

dott.ssa Francesca A. Lisi
lisi@di.uniba.it

Orario di ricevimento: mercoledì ore 10-12

dib

Programmazione delle socket

Scopo: comprendere come le applicazioni client/server comunicano attraverso le interfacce *socket*

Socket API

- introdotte in BSD4.1 UNIX, 1981
- esplicitamente create, usate, rilasciate dalle applicazioni
- paradigma client/server
- due tipi di servizio di trasporto via socket API:
  - affidabile, byte stream-oriented (cfr. TCP)
  - inaffidabile, datagram-oriented (cfr. UDP)

socket

Interfaccia
- *host-local*,
- *application-created/owned*,
- *OS-controlled*

(una "porta") in cui un processo applicativo può sia inviare che ricevere messaggi da/a un altro processo applicativo (remoto o locale)

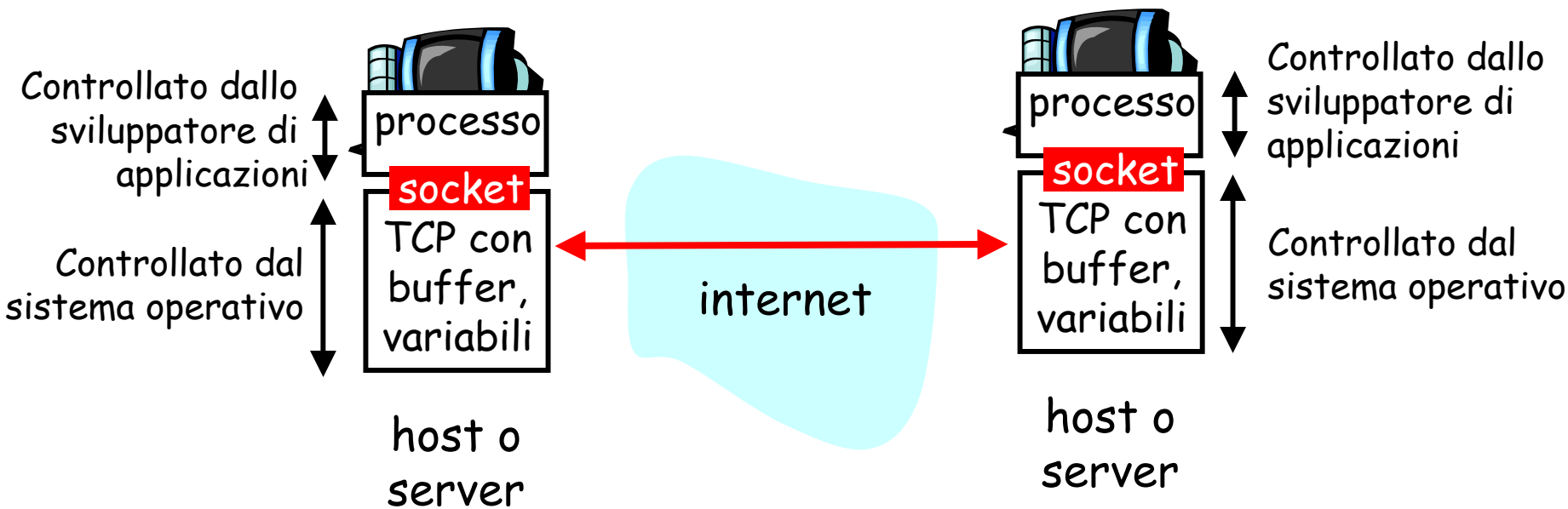
Programmazione delle socket: un esempio di applicazione client/server

- il client legge una linea dal suo *ingresso standard* (tastiera) e la invia attraverso la sua socket al server
- il server legge una linea dalla sua socket
- il server converte la linea in maiuscolo
- il server invia al client la linea modificata facendola passare attraverso la sua socket
- il client legge la linea (modificata) dalla sua socket e la stampa sulla sua *uscita standard* (monitor)

Programmazione delle socket con TCP

Socket: una porta fra il processo applicativo ed il protocollo TCP per il trasporto end-to-end

Servizio TCP: trasferimento affidabile di byte da un processo ad un altro



Programmazione delle socket con TCP

Il client deve contattare il server

- Il processo server deve innanzitutto essere in esecuzione
- il server deve aver creato una socket che accolga il contatto da parte del client

Il client contatta il server:

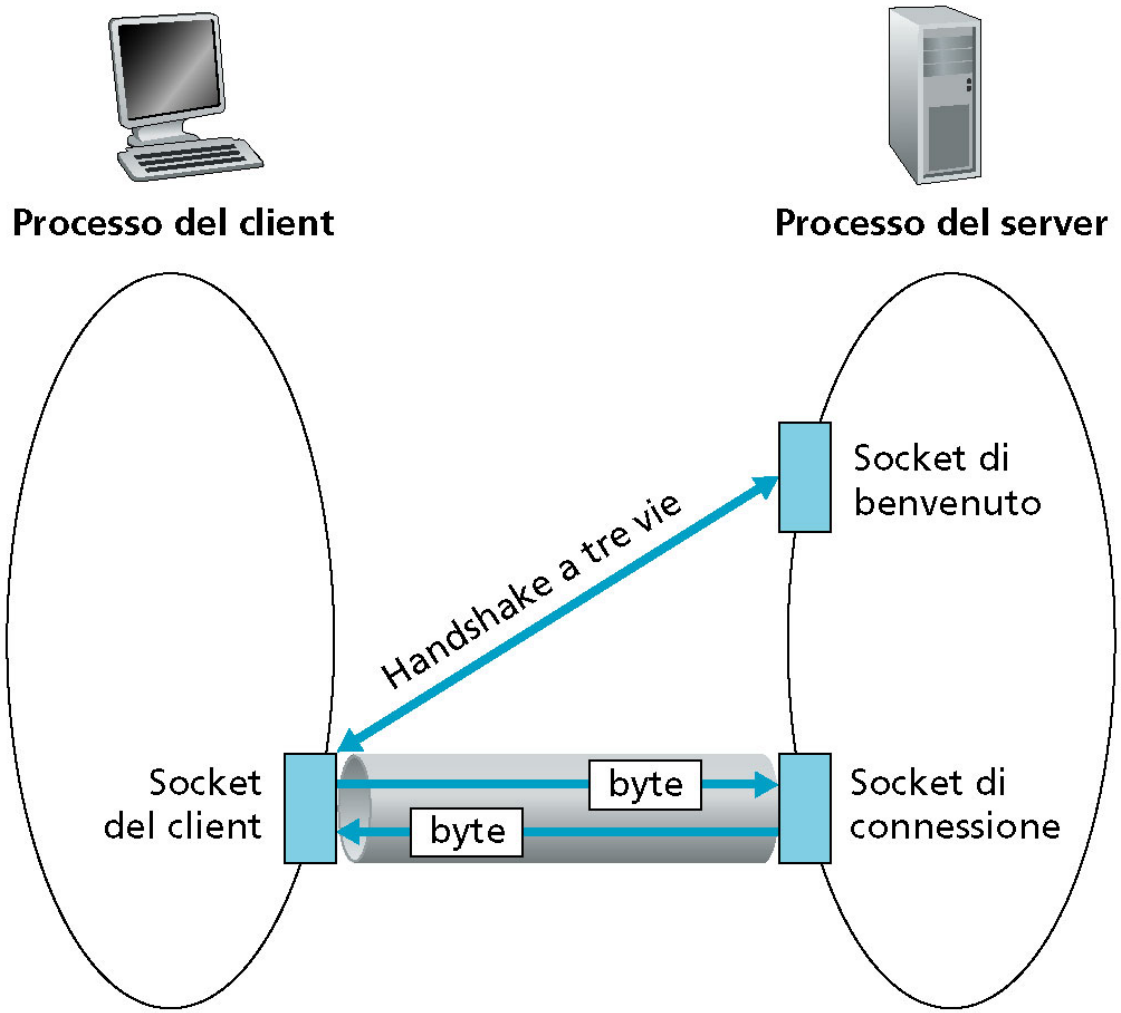
- creando una socket TCP locale al client
- specificando indirizzo IP e numero di porta del processo server

- Quando il client crea la socket: il client TCP stabilisce una connessione al server TCP
- Se contattato dal client, il server TCP crea una nuova socket per il processo server per comunicare con il client
  - consente al server di parlare con client multipli

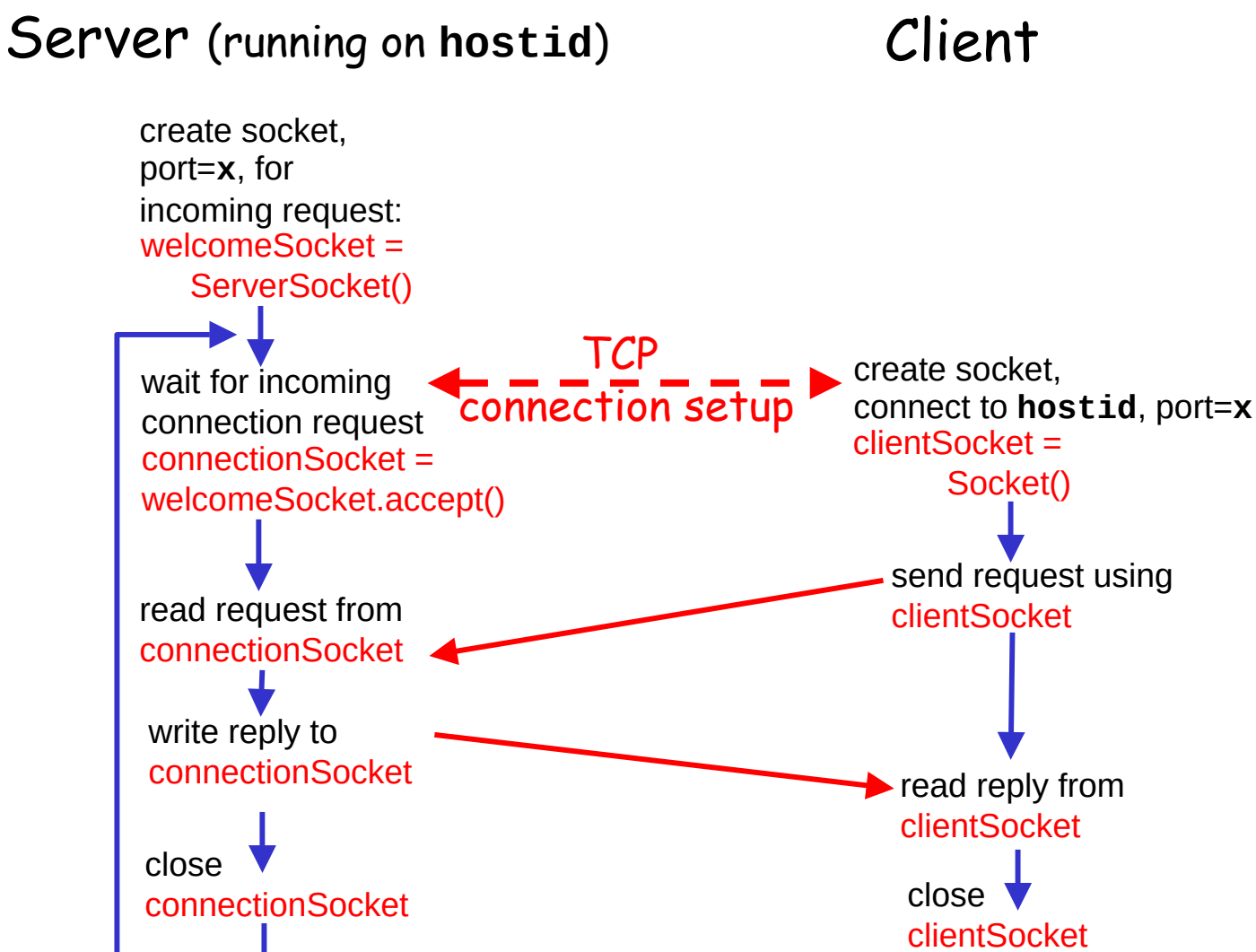
Punto di vista applicativo

Il TCP fornisce un trasferimento affidabile, in-order di byte ("pipe") fra client e server

Programmazione delle socket con TCP: interazione client/server

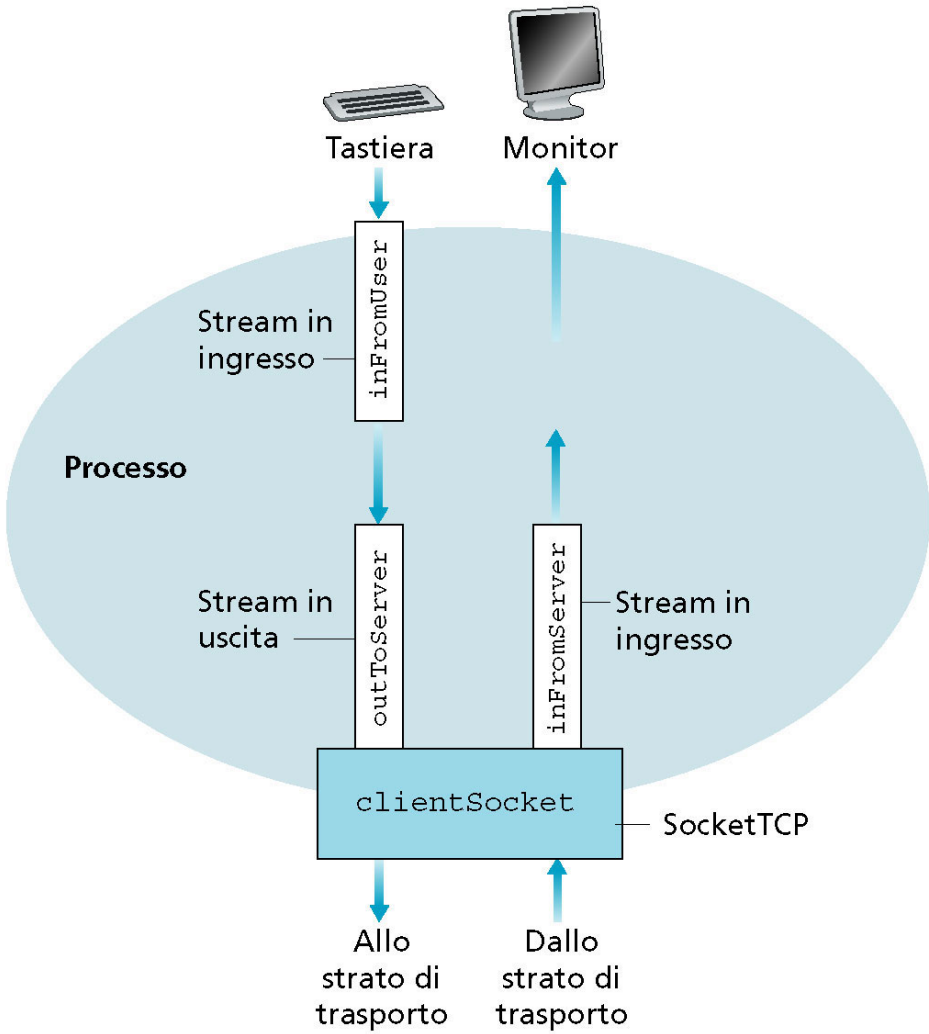


Programmazione delle socket con TCP: interazione client/server (cont.)



Programmazione delle socket con TCP: lato client

- il client legge una riga (stream in ingresso **`inFromUser`**) dallo standard input (tastiera) e la invia al server in byte (stream in uscita **`outToServer`**) attraverso la sua socket
- il client legge la linea modificata dalla sua socket (stream in ingresso **`inFromServer`**) e la manda in stampa sullo standard output (monitor)
- N.B. Un flusso (*stream*) è una sequenza di caratteri che fluisce verso/da un processo.



Programmazione delle socket con TCP: lato client in Java

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Create input stream

Create client socket, connect to server

Create output stream attached to socket

Programmazione delle socket con TCP: lato client in Java (cont.)

Create input stream attached to socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

    }
}
```

Send line to server

Read line from server

Programmazione delle socket con TCP: lato server in Java

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

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

        Socket connectionSocket = welcomeSocket.accept();

        BufferedReader inFromClient =
          new BufferedReader(new
          InputStreamReader(connectionSocket.getInputStream()));
```

Create welcoming socket at port 6789

Wait, on welcoming socket for contact by client

Create input stream, attached to socket

Programmazione delle socket con TCP: lato server in Java (cont.)

Create output stream, attached to socket

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());
```

Read in line from socket

```
clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

Write out line to socket

```
outToClient.writeBytes(capitalizedSentence);
            }
        }
    }
```

End of while loop, loop back and wait for another client connection

Programmazione delle socket con UDP

UDP: nessuna "connessione" fra client e server

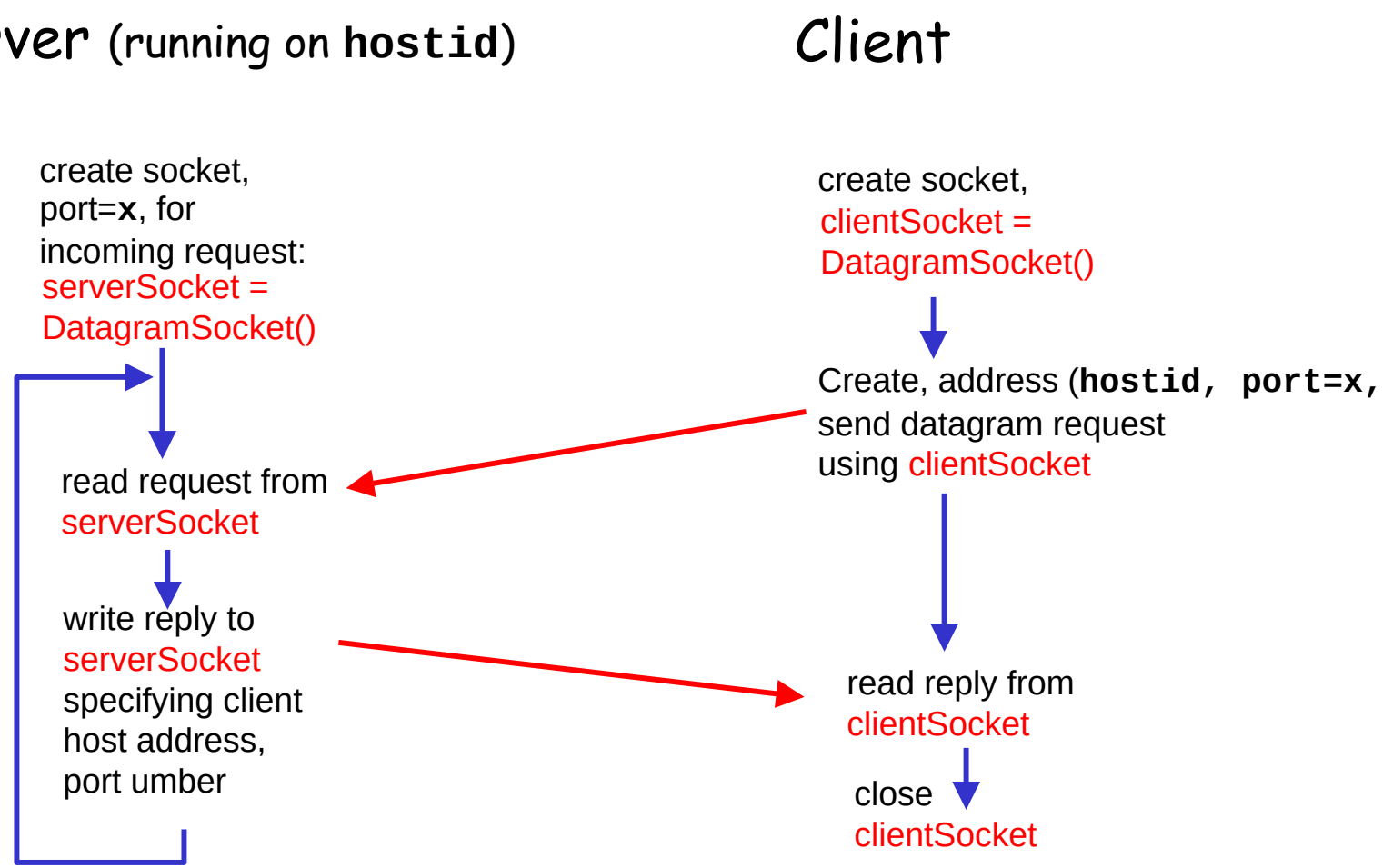
- no handshaking
- il mittente attacca esplicitamente l'indirizzo IP e il numero di porta del destinatario
- il server deve estrarre l'indirizzo IP e numero di porta del mittente dal datagramma ricevuto

UDP: i dati trasmessi potrebbero essere ricevuti non in ordine, o andare persi

application viewpoint

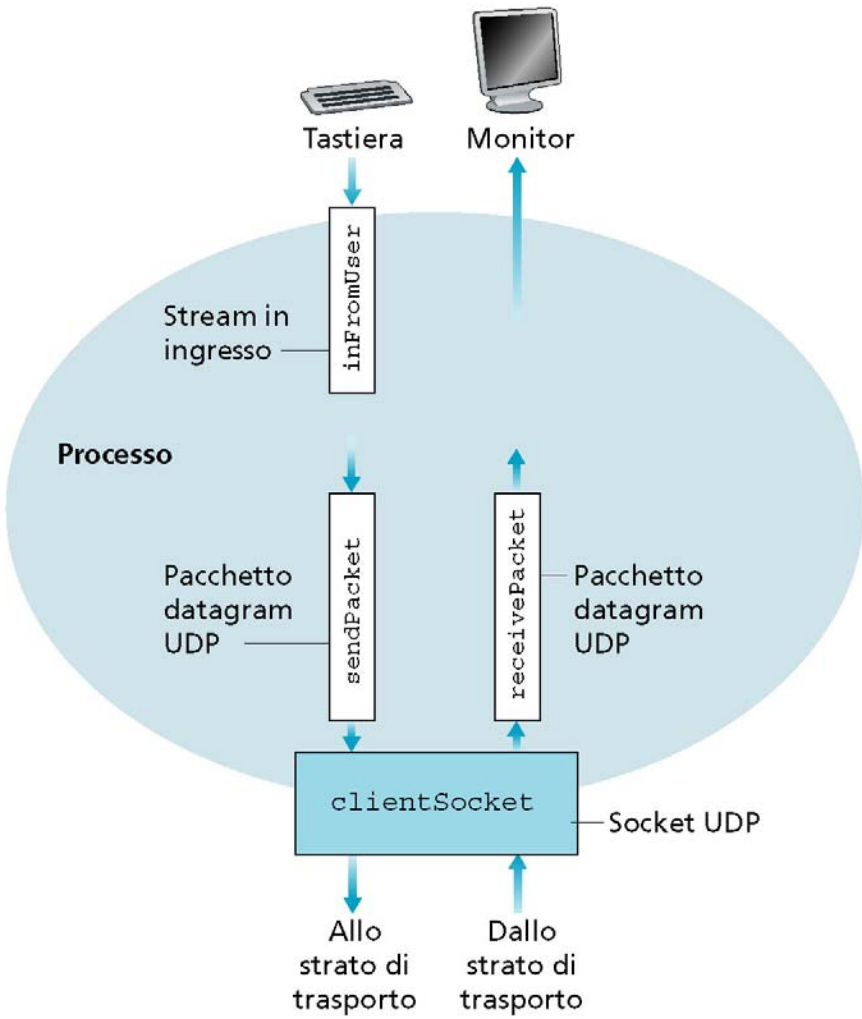
UDP fornisce trasferimento inaffidabile di gruppi di byte ("datagrammi") fra client e server

Programmazione delle socket con UDP: interazione client/server



Programmazione delle socket con UDP: lato client

- il client legge una riga (stream in ingresso **`inFromUser`**) dallo standard input (tastiera) e la invia in pacchetti datagram UDP (**`sendPacket`**) al server attraverso la socket
- il client legge il pacchetto datagram UDP contenente la linea modificata (**`receivePacket`**) attraverso la sua socket e la manda in stampa sullo standard output (monitor)



Programmazione delle socket con UDP: lato client in Java

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

class UDPClient {
   public static void main(String args[]) throws Exception
   {
      BufferedReader inFromUser =
      new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();
      sendData = sentence.getBytes();
```

Create input stream

Create client socket

Translate hostname to IP address using DNS

Programmazione delle socket con UDP: lato client in Java (cont.)

Create datagram with data-to-send, length, IP addr, port

```
DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length,
                                IPAddress, 9876);
```

Send datagram to server

```
clientSocket.send(sendPacket);
```

```
DatagramPacket receivePacket =
   new DatagramPacket(receiveData, receiveData.length);
```

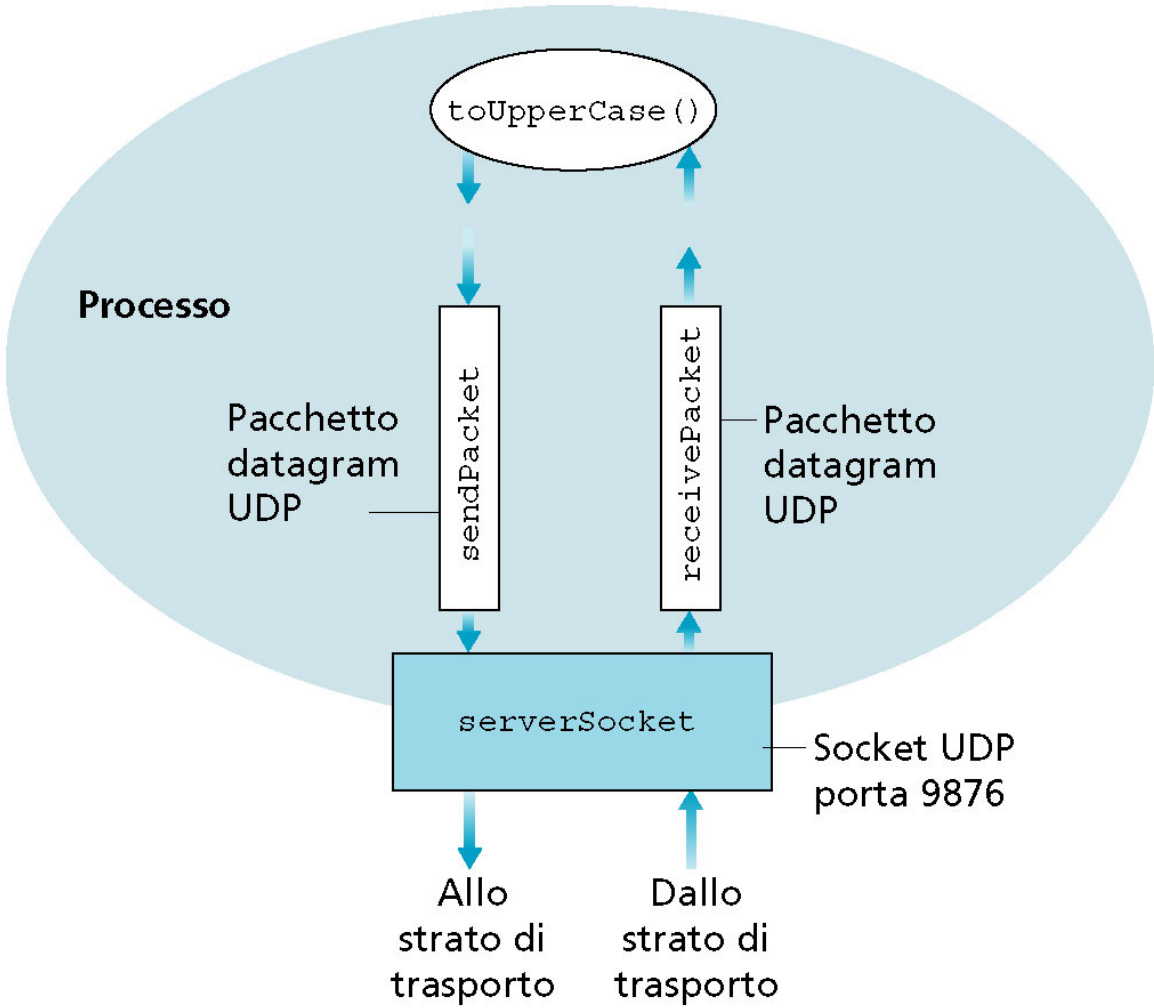
Read datagram from server

```
clientSocket.receive(receivePacket);

String modifiedSentence =
    new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}
}
```

Programmazione delle socket con UDP: lato server



Programmazione delle socket con UDP: lato server in Java

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

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

      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];
      byte[] sendData  = new byte[1024];

      while(true)
       {
          DatagramPacket receivePacket =
           new DatagramPacket(receiveData, receiveData.length);

           serverSocket.receive(receivePacket);
```

Create datagram socket at port 9876

Create space for received datagram

Receive datagram

Programmazione delle socket con UDP: lato server in Java (cont.)

```
String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress, port);

serverSocket.send(sendPacket);
  }
 }
}
```

Get IP addr port #, of sender

Create datagram to send to client

Write out datagram to socket

End of while loop, loop back and wait for another datagram

Programmazione delle socket: spunti di approfondimento

- Un esempio di semplice server Web in Java è riportato nella sezione 2.9 di Kurose & Ross.
- Esercizi di programmazione delle socket in Java sono presenti in coda al capitolo 2 di Kurose & Ross
  - Per una panoramica sul package java.net consultare http://java.sun.com/j2se/1.4.2/docs/api/java/net/package-summary.html
- Esperienze di programmazione delle socket in Java sono anche possibili in ambito di tesi di laurea (contattare il docente per ulteriori informazioni)