Programmare con le Socket TCP

Il Client deve contattare il server

Il processo server deve già essere attivo

Il server deve aver creato una socket che accetta la connessione del client ("*socket di benvenuto*")

Il Client:

Crea una socket TCP locale

Specifica l'indirizzo IP e numero di porta del processo server

Quando il client crea la socket: il lato client TCP stabilisce una connessione con il server TCP

Programmare con le Socket TCP

Quando viene contattato dal client, il server TCP crea una **nuova socket** per far comunicare il processo server con il client

- Un server può interagire con molti client contemporaneamente
- Il numero di porta sorgente è usato per distinguere diversi client

Dal punto di vista dell'applicazione TCP fornisce un trasferimento di byte (un "tubo") affidabile e ordinato tra client e server

Dopo che è stata stabilita la connessione, le socket usate dal client e dal server sono equivalenti

Gli Stream - gergo

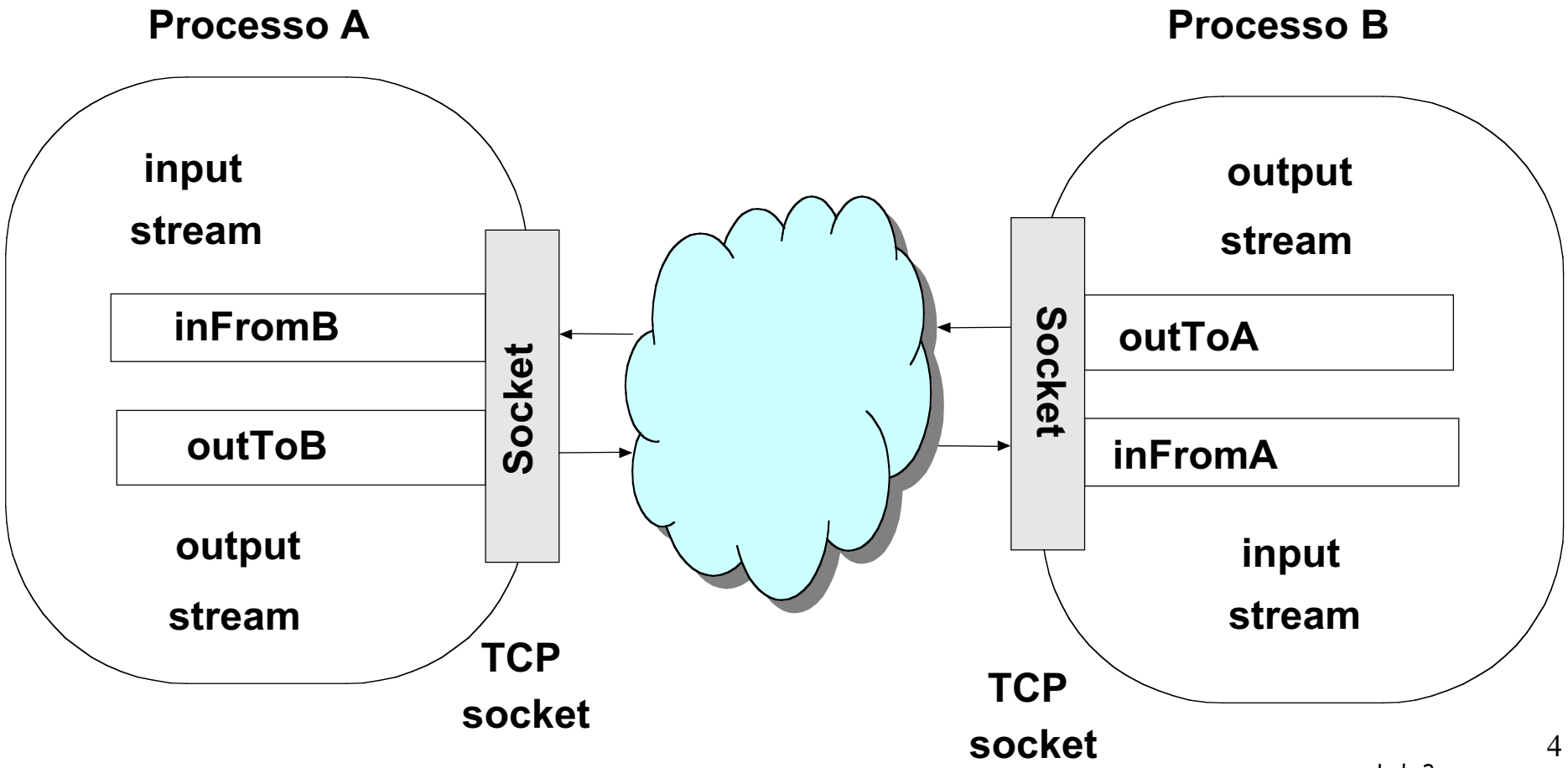
Uno stream (flusso) è una sequenza di caratteri che fluisce verso un processo o da un processo

Un input stream è collegato a qualche sorgente di input del processo, come la tastiera, un file o una socket.

Un output stream è collegato a qualche destinazione di output, come il monitor, un file o una socket.

Gli Stream delle socket TCP

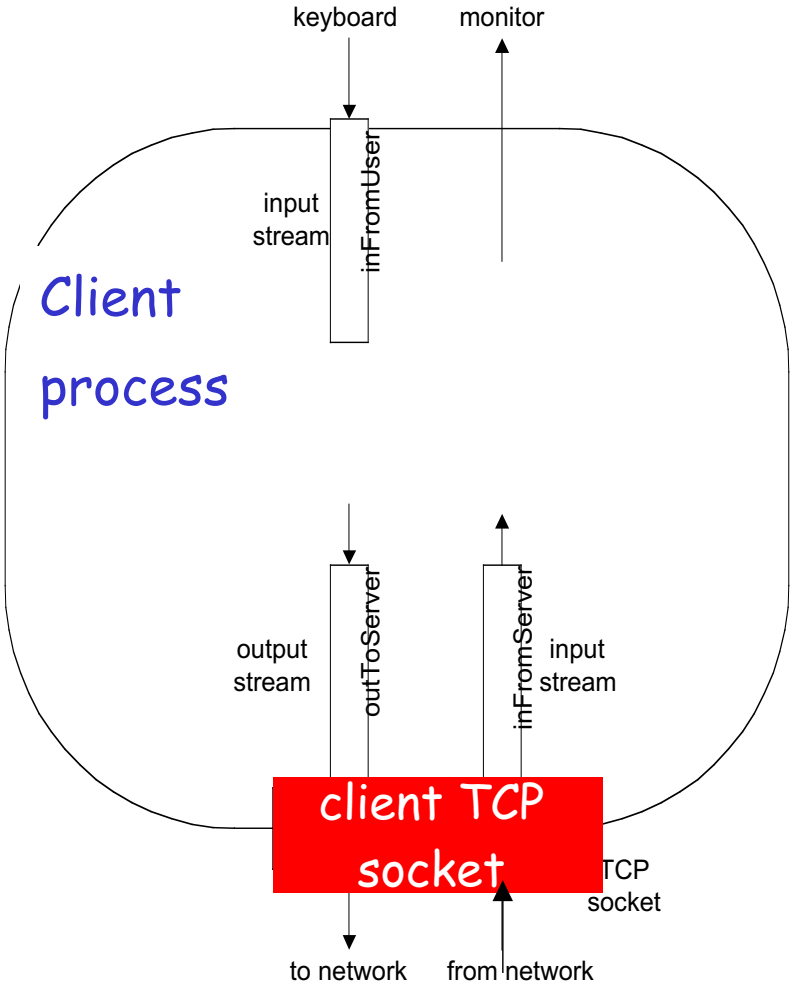
Quando due processi hanno stabilito una connessione TCP usando le socket, l'input stream della socket nel processo A è collegato all'output stream della socket corrispondente nel processo B, e viceversa



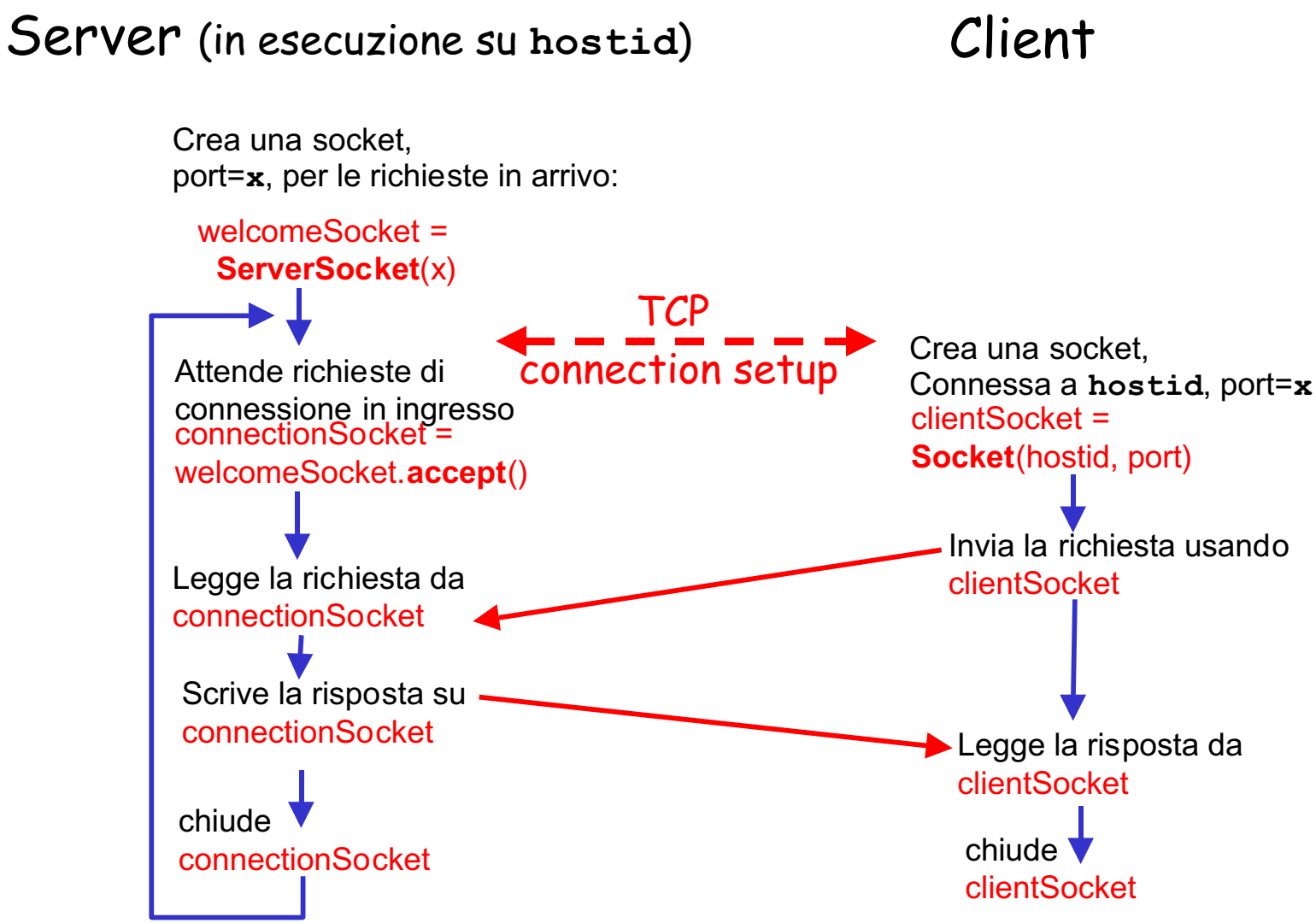
Programmazione con le socket TCP

Esempio di applicazione client-server:

1) Il client legge una riga dallo reads standard input (stream **`inFromUser`**), e la manda al server usando la socket (stream **`outToServer`**)
2) Il server legge la riga dalla socket
3) Il server converte la riga in caratteri maiuscoli, e la rimanda indietro al client
4) Il client legge dalla socket (stream **`inFromServer`**) la riga modificata, e la stampa



Interazione Client/server con le socket : TCP



Esempio: Java client (TCP)

```
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());
```

Crea un input stream

Crea la socket client, Connessa al server

Crea un output stream Connesso alla socket

Esempio: Java client (TCP), cont.

Crea un input stream Connesso alla 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();

    }
}
```

Manda una riga al server

Legge una riga dal server

Esempio: Java server (TCP)

```
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()));
```

Crea una socket di benvenuto Sulla porta 6789

Aspetta sulla socket di benvenuto il contatto del client

Crea un input stream, connesso alla socket

Esempio: Java server (TCP), cont

Crea un output stream, connesso alla socket

Legge una riga dalla socket

Scrive una riga sulla socket

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

clientSentence = inFromClient.readLine();

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

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

Fine del loop `while`
Torna all'inizio in attesa
di una nuova connessione

Programmazione con le socket UDP

UDP: non c'è connessione tra client e server

Non è necessario stabilire la connessione

Il mittente inserisce esplicitamente l'indirizzo IP e la porta della destinazione in ogni pacchetto

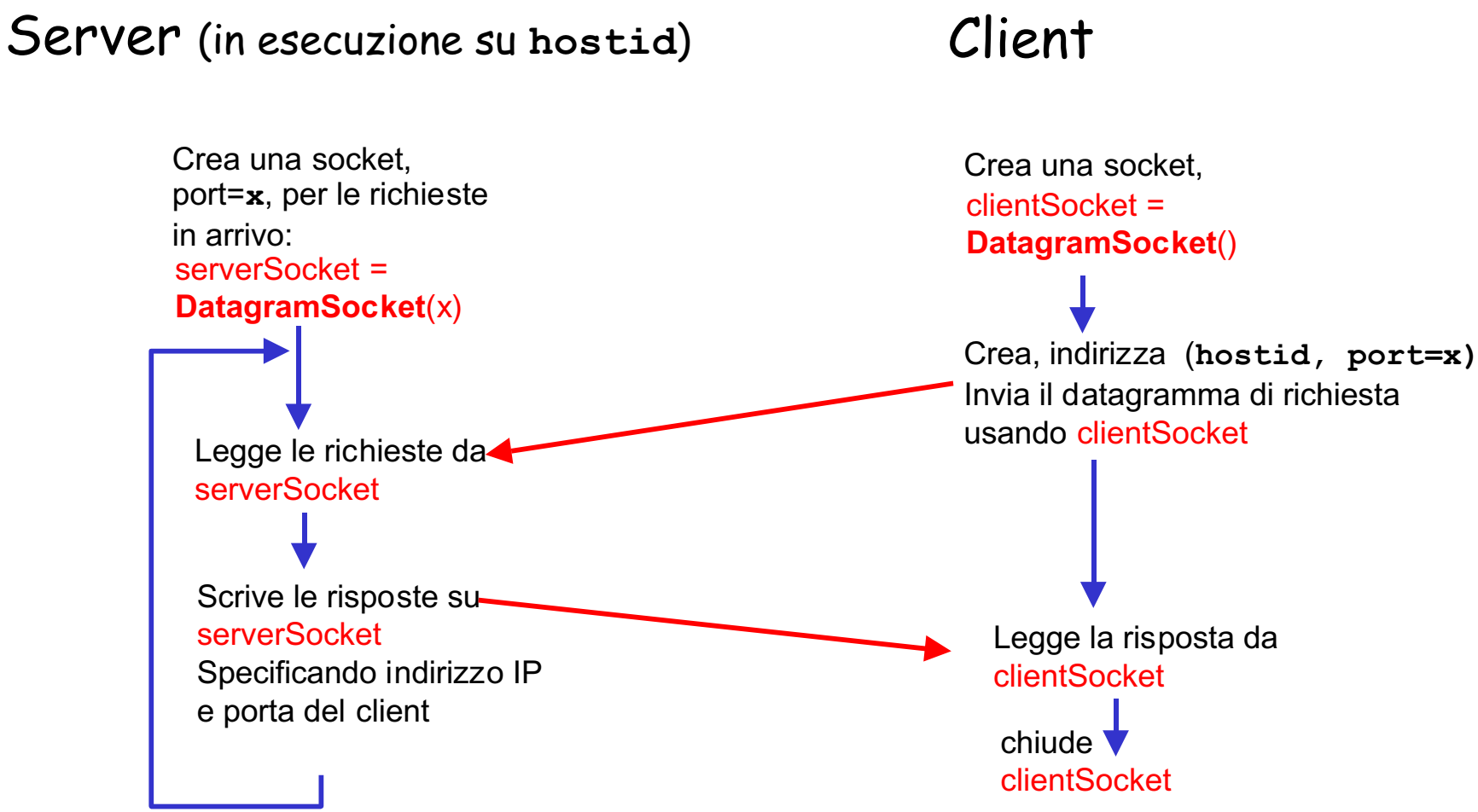
Il server deve estrarre l'indirizzo IP e la porta del mittente dal pacchetto ricevuto

UDP: i dati ricevuti possono essere persi o ricevuti fuori ordine

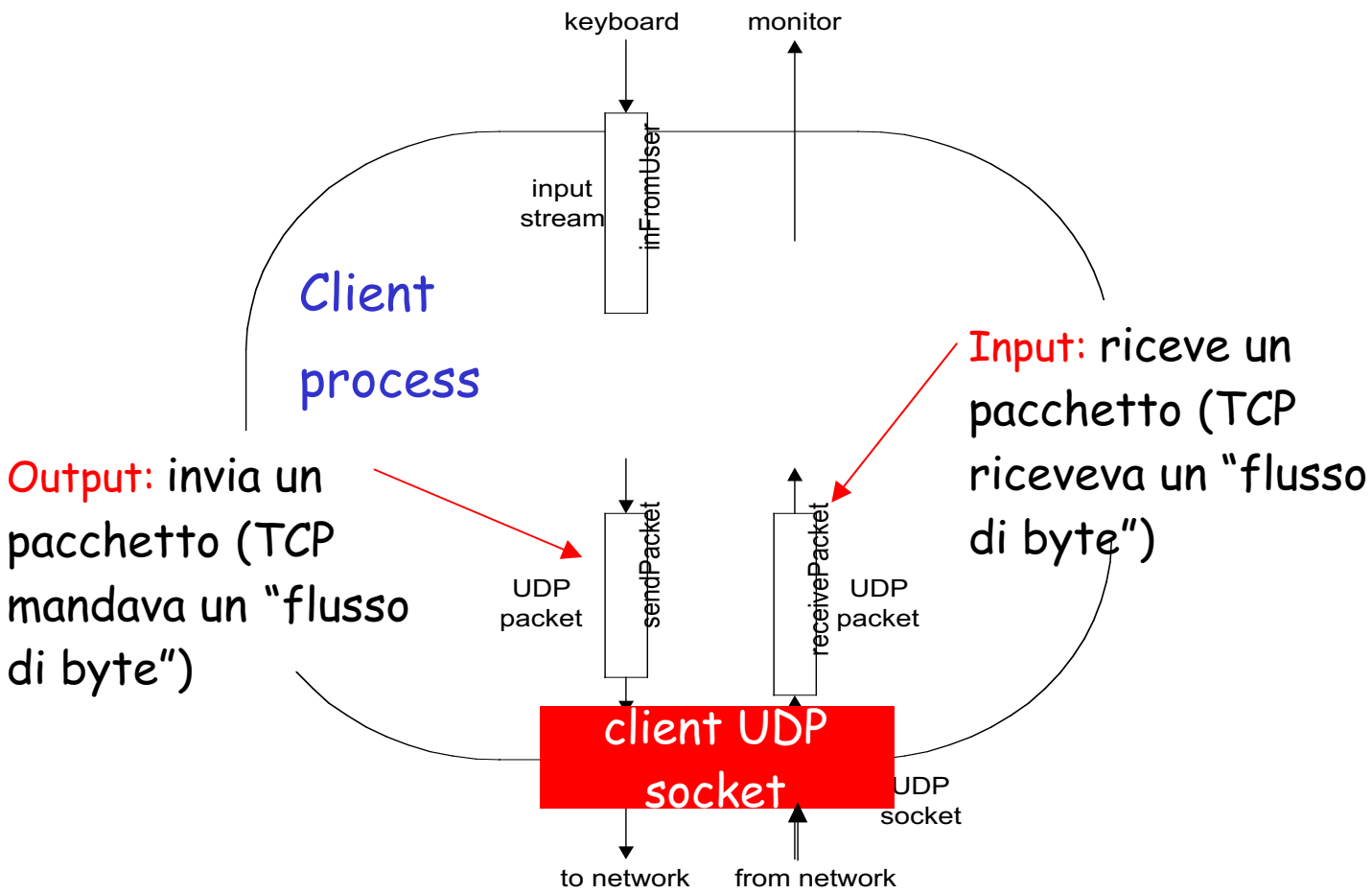
Punto di vista dell'applicazione:

UDP fornisce un trasferimento non affidabile di gruppi di byte ("datagrammi") tra client e server

Interazione Client/server con le socket: UDP



Esempio: Java client (UDP)



Esempio: Java client (UDP)

```
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();
```

Crea un input stream

Crea la client socket

Traduce hostname in un indirizzo IP usando il DNS

Esempio : Java client (UDP), cont.

Crea un datagramma con i dati, lunghezza, indirizzo IP, porta

Invia il datagramma to server

Legge il datagramma dal server

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

clientSocket.send(sendPacket);

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

clientSocket.receive(receivePacket);

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

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

}
```

Esempio : Java server (UDP)

```
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);
```

Crea una socket datagramma Sulla porta 9876

Crea spazio per il datagramma ricevuto

Rieceve il datagramma

Esempio : Java server (UDP), 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);
  }
}
}
```

Determina indirizzo IP e porta del mittente

Crea un datagramma da mandare al client

Scrive il datagramma Sulla socket

Fine del loop while torna all'inizio e attendi un altro datagramma