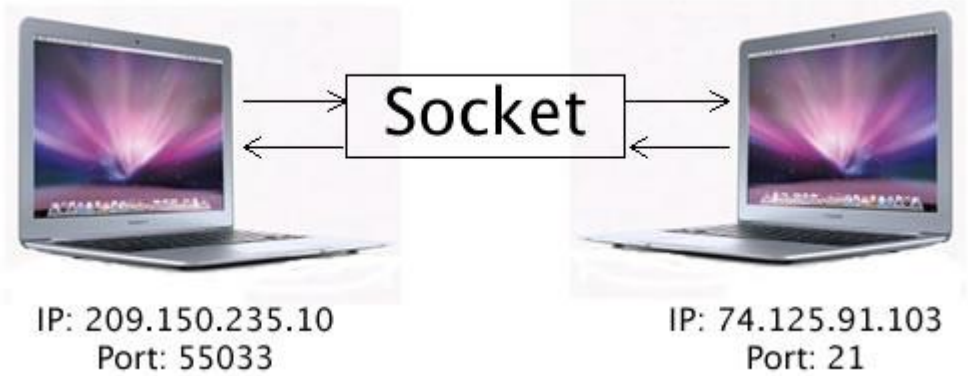


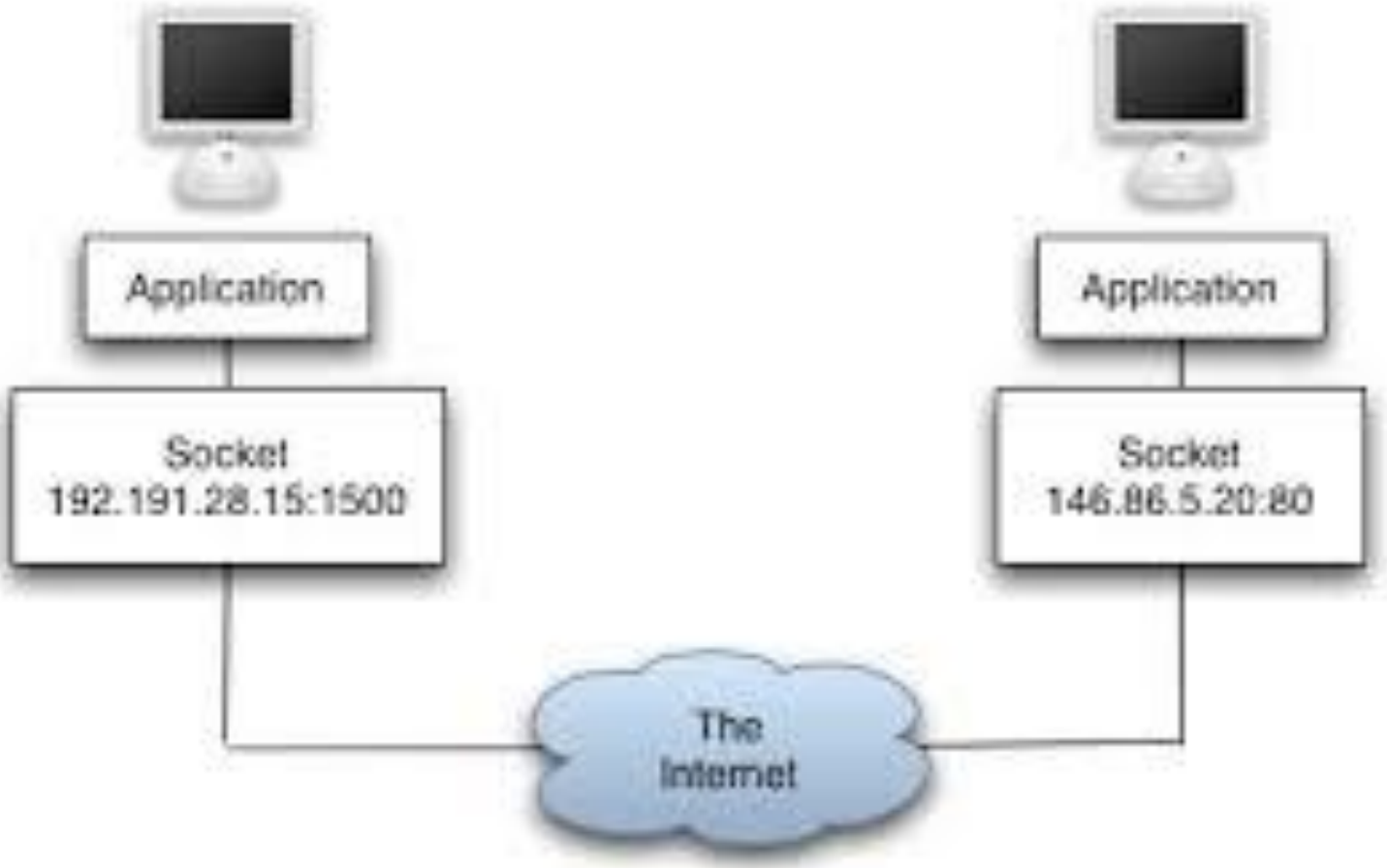
SOCKET

Prof. Ranalli

CHE COS'È?

Un'interfaccia software che consente a due processi in esecuzione su host diversi (o sullo stesso host) di comunicare tra loro.

Ciascun processo può essere visto come una casa e la socket rappresenta la porta di questa casa


Application
Socket
192.191.28.15:1500
Application
Socket
146.86.5.20:80
The
Internet

MODELLO CLIENT- SERVER

- Nel modello Client – Server la comunicazione tra il Client e il Server può essere gestita attraverso i protocolli di trasporto utilizzando le Socket.
- Per usare le socket, oltre a conoscere l'indirizzo IP dell'host a cui connettersi, bisogna disporre dell'informazione sufficiente per collegarsi al processo server corretto.

MODELLO CLIENT- SERVER

- Per questo motivo esistono i numeri di porta (port number) che permettono di associare un servizio (un processo server che risponde alle richieste) ad un ben determinato numero.
- Le connessioni avvengono sempre specificando un indirizzo IP ed un numero di porta.

`Socket → {IP_Addr, Port_Num)`

NUMERO DI PORTA

N. di porta → numero a 16 bit. [0, 65535]

Porte riservate → [0, 1023]

Porte processi utente → [1024, 65535]

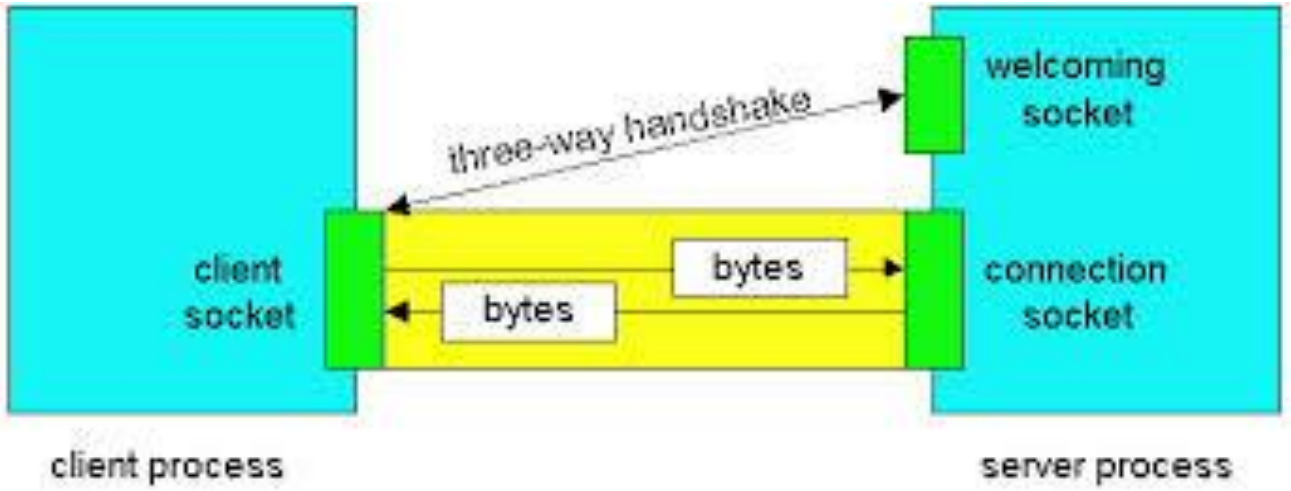
Tipologie di comunicazioni tramite Socket

- Comunicazione tramite protoccolo di trasporto TCP (Stream).
- Comunicazione tramite protocollo di trasporto UDP (Datagrammi)

Comunicazione tramite protocollo TCP

Viene realizzato un canale di comunicazione tra la porta del client e la porta del server con le seguenti caratteristiche:

- il canale di comunicazione che si viene a creare è bidirezionale.
- la comunicazione termina quando il client chiude la connessione.
- La comunicazione è affidabile. I pacchetti inviati arrivano correttamente a destinazione.



Handshake a tre vie (*Three-way Handshake*):

1.Il client bussa alla porta del server per avviare una comunicazione.

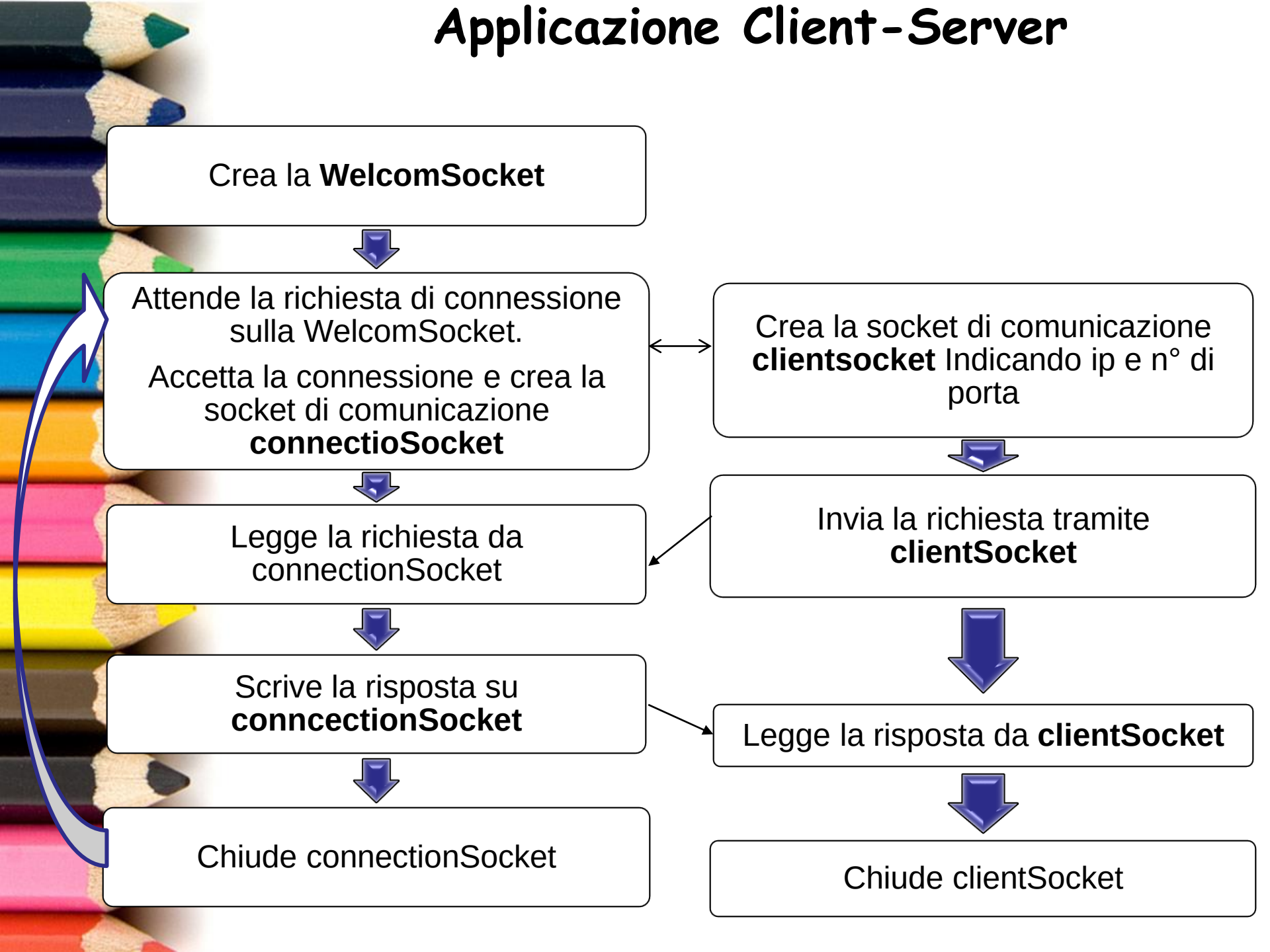
2.Il server comunica al client di aver accettato la comunicazione e predispone un'opportuna porta dedicata per la comunicazione Client-Server

3.Il client avvia la comunicazione facendo la sua richiesta al Server

L'handshake avviene in modo trasparente a livello di trasporto

Le socket stream (utilizzate per comunicare con TCP) sono affidabili, stabiliscono una connessione stabile e bidirezionale basata su stream di byte di lunghezza variabile

Applicazione Client-Server



Stream di comunicazione

Una volta stabilita la connessione, al Client e al Server vengono creati due stream di comunicazione:

- Uno dal client al server (Client_out e Server_in)
- Uno dal server al client (Server_out e Client_in)

Socket in Java

Le API invocate da client e server, per instaurare una connessione, sono diverse. Quindi Esistono classi di libreria Java diverse per la programmazione Client-Server.

Si utilizzano le classi del package java.net (`import java.net.*`):

- Socket (per il Client)
- ServerSocket (per il Server)

Classe Socket

- Utilizzata nel processo Client
- Costruttore:

  `Socket(String server, int Porta)`

  `Es. Socket("192.168.0.1",2354)`

  - Definisce il socket utilizzato per comunicare con il servitore
  - Può generare un'eccezione UnknownHostException e IOException
- Metodi comuni

  `close()`

  - Chiude la comunicazione con il Server

Classe ServerSocket

- Utilizzata nel processo Server
- Costruttore:

  `ServerSocket(int Porta)`

  `Es. ServerSocket(2354)`

  - Crea la Welcom-socket sulla quale il server si mette in ascolto.
  - Può generare un'eccezione UnknownHostException e IOException.
- Metodi comuni

  `Socket accept()`

  - Attende la richiesta di connessione dal client e quando questa avviene genera la connectionSocket attivando il canale di comunicazione

Metodi Stream

Il client e il server per comunicare utilizzano degli stream (Canali di comunicazione) che vengono generati quando si attiva la connessione.

Le classi Socket e ServerSocket hanno due metodi che forniscono i relativi stream di Input e Output

- `InputStream getInputStream()`

  Restituisce lo stream di Lettura al client(o al server).

- `OutputStream getOutputStream()`
  Restituisce lo stream di Scrittura al client(o al server).

Attraverso questi stream vengono inviati i singoli byte tra client e Server e viceversa

Metodi Stream

```
//-----------LATO SERVER----------------
Socket cs= ss.accept();


InputStream is= cs.getInputStream();
OutputStream os= cs.getOutputStream();


//-----------LATO CLIENT----------------
Socket s=new Socket("192.168.10.32",2035);
InputStream is= s.getInputStream();
OutputStream os= s.getOutputStream();
```

Classe BufferedReader e PrinterWriter

Le classi BufferedReader e PrinterWriter forniscono metodi rispettivamente per leggere stringhe. Contengono rispettivamente i metodi readLine() e print(String s)

Costruttori:

```
BufferedReader(InputStream is)

PrinterWriter(OutputStream os)
```

DataInputStream e DataOutputStream

Le classi DataInputStream e DataOutputStream forniscono metodi rispettivamente per leggere e scrivere tutti i tipi primitivi del linguaggio (int, char, String, ecc.). Questi due stream devono essere utilizzati in coppia in quanto l'uno riesce a decodificare ciò che l'altro ha inviato.

Costruttori:

```
DataInputStream(InputStream is)

DataOutputStream(OutputStream os)
```

DataInputStream e DataOutputStream

Metodi DataOutputStream

```
void writeBoolean (boolean v)
void writeByte (byte v)
void writeChar (char v)
void writeChars (String s)
void writeUTF (String s)
```

…

Metodi DataInputStream

```
boolean readBoolean ()
byte readByte ()
char readChar ()
String readUTF ()
…
```

Esempio di comunicazione Client - Server

Si vuole realizzare un sistema in cui il Client invia un numero intero e il Server restituisce il suo valore assoluto

Esercizio 0

Scrivere un server **TCPEchoServer** che:

- accetta come messaggi linee di testo dal cliente, conta i messaggi ricevuti, e manda indietro al cliente il messaggio preceduto dal suo numero progressivo.
- Scrivere il relativo cliente TCPEchoClient che termina la connessione quando riceve la stringa «FINE»

Esercizio 0

Ciao

Risp. Server:1 Ciao

Come Stai?

Risp. Server:2 Come Stai?

Arrivederci

Risp. Server:3 Arrivederci

FINE