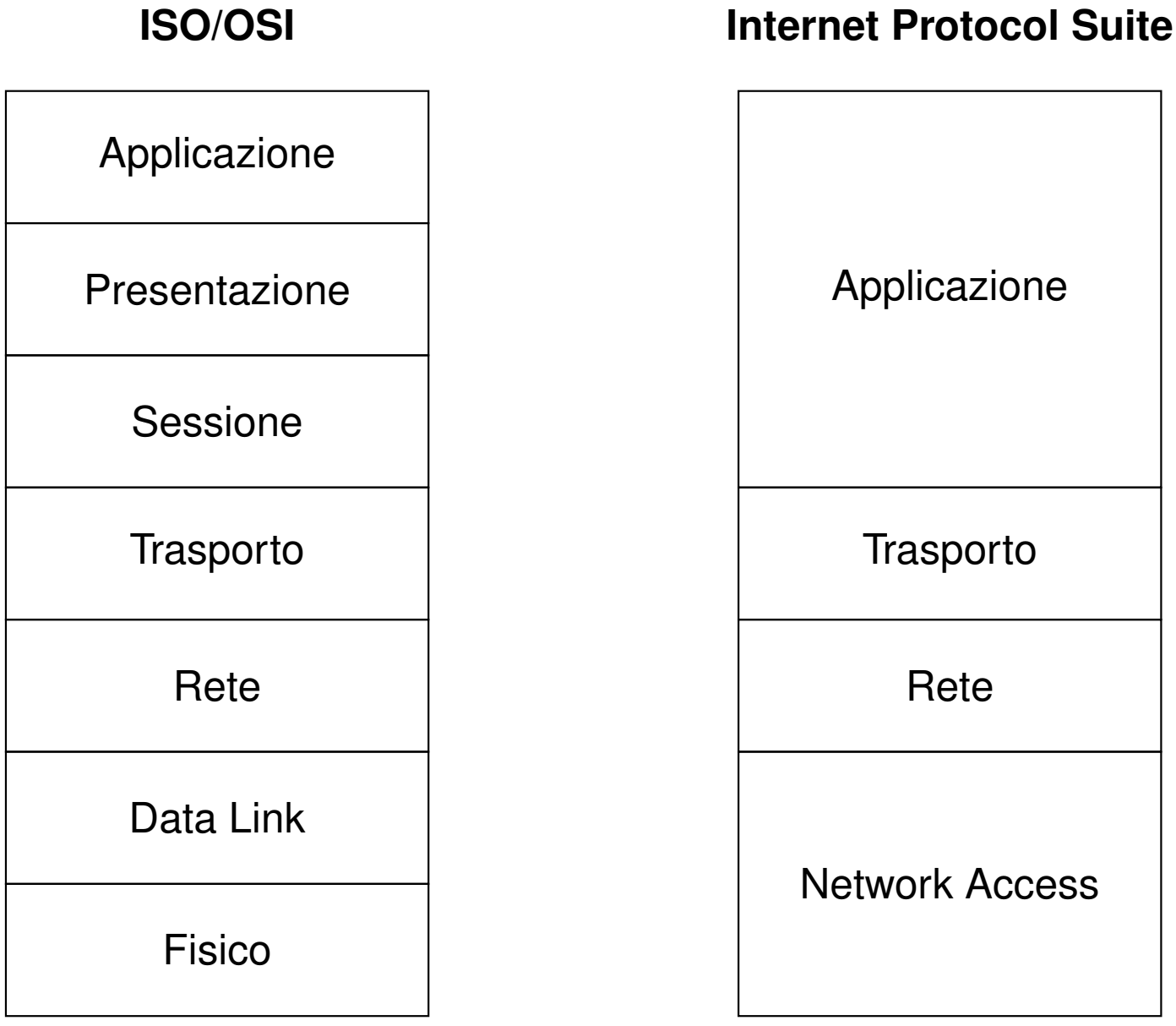
# Introduzione ai Socket in Java

Matteo Trentin
matteo.trentin2@unibo.it

# Recap - Modello ISO/OSI - Internet Protocol Suite



# Socket - In teoria

- Ad ogni *dispositivo* collegato ad una rete è associato un indirizzo
- Ad ogni *servizio* sul dispositivo è associato un numero di porta
- Ad ogni servizio corrisponde un processo
- La comunicazione tra due dispositivi può essere descritta come una quintupla: $\langle$ protocollo, indirizzo locale, porta locale, indirizzo remoto, porta remota $\rangle$
- Il socket è uno dei due endpoint di questo canale di comunicazione
- I socket rientrano nel livello Sessione del modello OSI
- I socket agiscono come interfaccia tra il livello Applicazione e quello di Trasporto nella Internet Protocol Suite

# Socket - In pratica

- Le applicazioni utilizzano le **Socket API** per inviare e ricevere messaggi tramite socket
- Lato server l'applicazione inizialmente ha comportamento *passivo*:
  1. Creazione del socket (con specifica porta e opzionalmente indirizzo)
  2. "Ascolto" sulla porta all'indirizzo specificato, in attesa di una connessione
  3. Accettazione della richiesta di connessione
  4. La coppia di socket server-client forma il canale di comunicazione, attraverso cui vengono scambiati i dati
- Lato client l'applicazione ha comportamento *attivo*:
  1. Creazione del socket (specificando porta e indirizzo *del server*)
  2. La porta locale viene assegnata generalmente in automatico
  3. Richiesta di connessione al server
  4. La coppia di socket client-server forma il canale di comunicazione, attraverso cui vengono scambiati i dati

# Socket - Più in pratica

```
Socket s = new Socket(host, port);
Scanner from = new Scanner(s.getInputStream());
PrintWriter to = new PrintWriter(s.getOutputStream(),
    true);

...
to.println("some request");
String response = from.nextLine();
...

s.close();
```

# Socket - Client/Server

**Client**

```
String host = "127.0.0.1";
int port = 9000;
Socket s = new Socket(host,
     port);
...
s.close();
```

**Server**

```
int port = 9000;
ServerSocket server = new
    ServerSocket(port);
Socket s = server.accept();
server.close();
...
s.close();
```

# Socket in Java - Info Server

- Vogliamo definire un'applicazione client-server, con un solo client
- Il client si connette al server e richiede informazioni su un'entità
- Il server restituisce l'informazione se esiste, e un errore altrimenti

# Socket in Java - Classi utili - `InetAddress`

- Classe utilizzata per rappresentare indirizzi IP (v4 e v6)
- Alcuni metodi per creare un oggetto di questa classe:

```
static InetAddress[] getAllByName(String host);
static InetAddress   getByAddress(byte[] addr);
static InetAddress   getByName(String host);
static InetAddress   getByAddress(String host, byte[] addr);
static InetAddress   getLocalHost();
```

- Può essere usato nella creazione di un socket:

```
InetAddress addr = InetAddress.getLocalHost();
Socket s = new Socket(addr, 9000);
```

# Socket in Java - Cenni su UDP

- I costrutti mostrati fino ad ora (`Socket` e `ServerSocket`) sono per socket TCP
- In Java è possibile definire socket UDP, e inviare su di essi dei `DatagramPacket`
- Lato client:

```
DatagramSocket s = new DatagramSocket();
DatagramPacket p = new DatagramPacket(msg, msg.length, host, port);
s.send(p);
```

- Lato server:

```
DatagramSocket s = new DatagramSocket(9000);
DatagramPacket p = new DatagramPacket(new byte[1000], 1000);
s.receive(p);
String msg = new String(p.getData()).substring(0, p.getLength());
```

# Socket in Java - Cenni su UDP - `connect()`

- I socket UDP in Java forniscono anche un metodo `connect()`
- Questa “connessione” non è una connessione nel senso di TCP (no 3-way handshake, no retransmission)
- Semplicemente viene controllato che i pacchetti vengano inviati e ricevuti solo verso l’indirizzo specificato
- Esempio:

```
DatagramSocket s = new DatagramSocket();
s.connect(InetAddress.getByName("localhost"), 9000);
DatagramPacket p = new DatagramPacket(msg, msg.length, "localhost",
     9001);
s.send(p); // viene sollevata una IllegalArgumentException
```