Università degli Studi della Calabria
Corso di Laurea in Ingegneria Informatica
A.A. 2010/2011

Corso di Reti di Calcolatori

Lucidi delle Esercitazioni

Raffaele Giordanelli

Richiami sulle Reti

Indirizzo numerico o fisico.

- Un indirizzo IP consente di identificare univocamente un computer sulla rete. L'indirizzo IP è formato da una sequenza numerica costituita da quattro raggruppamenti separati da un punto.

XXX.TTT.YYY.ZZZ

- Ogni raggruppamento identifica una sottorete o un host attraverso un numero compreso tra 0 e 255.

Indirizzo simbolico o logico.

- Consente di riferirsi ad un computer in rete attraverso un nome a cui corrisponde un indirizzo di tipo numerico.

host.localdomain.globaldomain

Richiami sulle Reti

DNS – Domain Name System

E’ un servizio adibito alla risoluzione dei nomi degli indirizzi. In particolare consente di tradurre un indirizzo simbolico nell’equivalente indirizzo numerico e viceversa.

160.168.29.5	deis.unical.it
150.45.63.33	icarus.cs.icar.cnr.it
160.168.29.111	jupiter.deis.unical.it
131.154.99.131	testbed001.cnaf.infn.it

Richiami sulle Reti

I **Socket** consentono di aprire un canale di comunicazione tra due computer collegati in rete. Una volta aperto il canale la responsabilità della comunicazione tra i due programmi che partecipano nella connessione ricade interamente sul programmatore.

Protocolli di comunicazione
Un protocollo è l'insieme di regole su cui è basata la comunicazione all'interno di una rete. Nelle reti moderne la stragrande maggioranza del traffico dati prevede l'utilizzo dei *pacchetti.*

I principali protocolli per l'instradamento dei pacchetti sono TCP e UDP

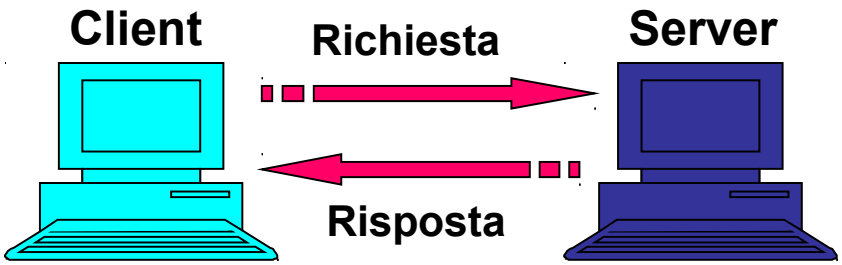
- **UDP** (User Datagram Protocol) basata su messaggi
- **TCP** (Transmission Control Protocol) basata su un canale di comunicazione vero e proprio

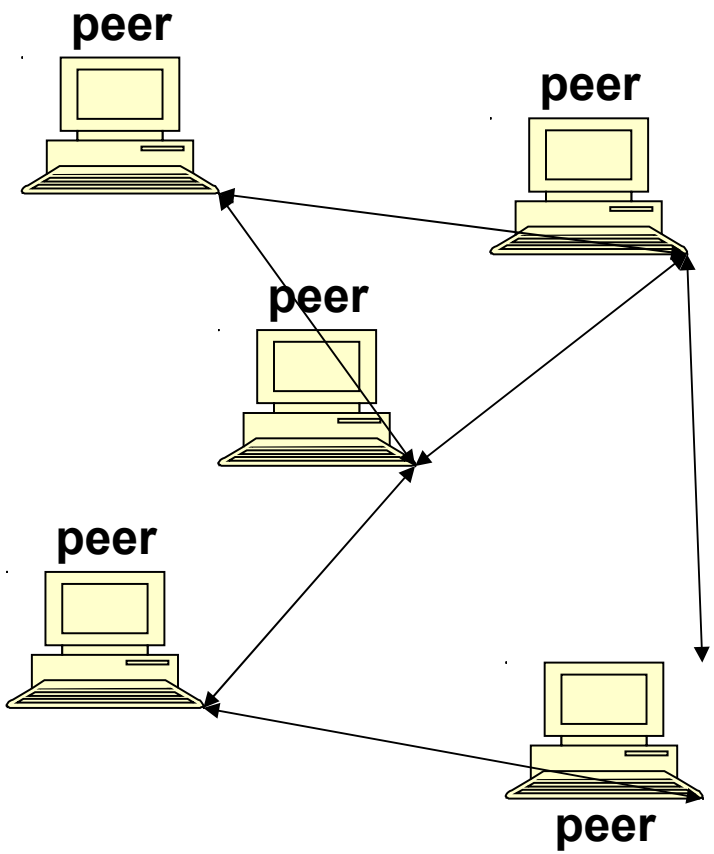
Il protocollo TCP/IP offre un'ampia serie di servizi gestiti da protocolli più specifici quali HTTP, SMTP, FTP, ecc.

Richiami sulle Reti

La comunicazione tra due computer o due applicativi in esecuzione su due macchine può avvenire secondo diversi modelli. I principali sono:

- **Client-Server**
- **Peer-to-Peer**





Package java.net

Classi

Authenticator
ContentHandler
DatagramPacket
DatagramSocket
DatagramSocketImpl
HttpURLConnection
InetAddress
JarURLConnection
MulticastSocket
NetPermission
PasswordAuthentication
ServerSocket
Socket
SocketImpl
SocketPermission
URL
URLClassLoader
URLConnection
URLDecoder
URLEncoder
URLStreamHandler

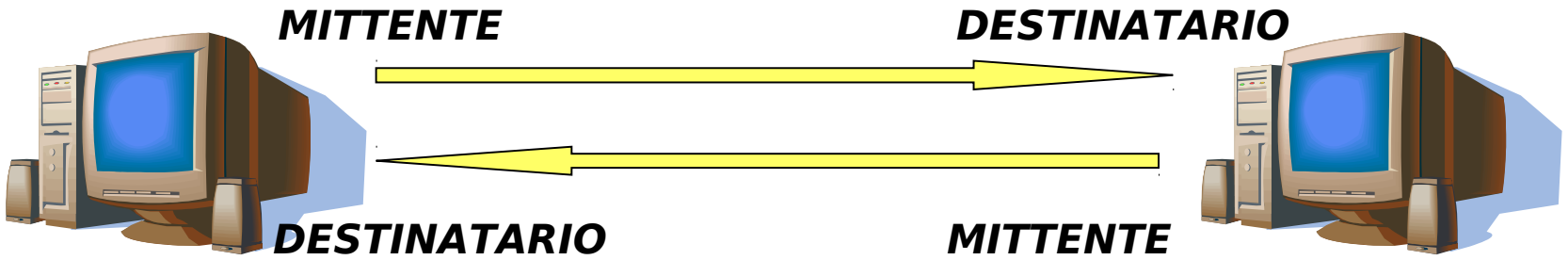
Interfacce

ContentHandlerFactory
FileNameMap
SocketImplFactory
SocketOptions
URLStreamHandlerFactory

Eccezioni

BindException
ConnectException
MalformedURLException
NoRouteToHostException
ProtocolException
SocketException
UnknownHostException
UnknownServiceException

Indirizzi in Java



127.0.0.1

150.145.63.128

processo di *risoluzione del nome*

deis.unical.it

minos.icar.cnr.it

Il package java.net offre una classe dedicata alla rappresentazione degli indirizzi: la classe **InetAddress**.

Questa classe contiene molti metodi utili per la manipolazione degli indirizzi.

Il semplice indirizzo di un computer non è sufficiente per realizzare una comunicazione: è necessario specificare anche il servizio. In TCP/IP questo si fa specificando un intero tra 0 e 65535 conosciuto col nome di ***porta***.

Ai vari servizi sono generalmente associate porte convenzionali

(HTTP → 80, SMTP → 25, FTP → 21, ecc.)

java.net.InetAddress

static InetAddress getByName(String host): restituisce l'indirizzo IP di un dato host.

static InetAddress[] getAllByName(String host): restituisce tutti gli indirizzi IP di un dato host.

static InetAddress getLocalHost(): restituisce l'indirizzo IP di localhost.

byte[] getAddress(): restituisce l'indirizzo IP sotto forma array di byte.

String getHostAddress(): restituisce l'indirizzo IP sotto forma di stringa (p.es. "192.168.35.75").

String getHostName(): restituisce l'hostname associato all'indirizzo IP.

Lookup di indirizzi locali e remoti

Il comando **nslookup** effettua la risoluzione (diretta ed inversa) dell'host specificato.

C:\>nslookup java.sun.com

Server: dns.icar.cnr.it
Address: 194.119.198.151

Nome: java.sun.com
Address: 192.18.97.71

C:\>nslookup 192.18.97.71

Server: dns.icar.cnr.it
Address: 194.119.198.151

Nome: flres.java.Sun.COM
Address: 192.18.97.71

Lookup di indirizzi locali e remoti

```
static void printLocalAddress () {
   try {
     InetAddress myself = InetAddress.getLocalHost ();
     System.out.println ("My name : " + myself.getHostName ());
     System.out.println ("My IP : " + myself.getHostAddress ());
   } catch (UnknownHostException ex) {
     System.out.println ("Failed to find myself:");
   }
}

static void printRemoteAddress (String name) {
   try {
     InetAddress machine = InetAddress.getByName (name);
      System.out.println ("Host name : " + machine.getHostName
());
       System.out.println ("Host IP : " + machine.getHostAddress
());
   } catch (UnknownHostException ex) {
     System.out.println ("Failed to lookup " + name);
   }
}
```

I Socket in Java

La comunicazione basata su canali consente a due programmi in esecuzione su due computer differenti di scambiarsi flussi di dati. Una volta stabilita la connessione i due programmi possono inviare dati lungo il canale come se si trattasse della scrittura di dati in un file. Ciascuno dei due programmi ha a disposizione quindi di un **InputStream** da cui ricevere i dati e di un **OutputStream** in cui inserire i dati da inviare. Contrariamente alla comunicazione basata su messaggi quella basata su canale garantisce che i dati siano ricevuti nello stesso ordine in cui sono stati inviati e che non vi siano perdite di informazione.

Socket

ServerSocket

java.net.Socket (1)

Socket(String host, int port): crea un socket e lo connette all'host ed alla porta specificati; host è un hostname (per es. "www.deis.unical.it") oppure un indirizzo IP (per es. "160.97.27.7").

Socket(InetAddress address, int port): crea un socket e lo connette all'indirizzo ed alla porta specificati.

void close(): chiude il socket.

InetAddress getInetAddress(): restituisce l'indirizzo a cui è connesso il socket.

int getPort(): restituisce la porta remota a cui è connesso il socket.

int getLocalPort(): restituisce la porta locale del socket.

java.net.Socket (2)

InputStream getInputStream(): restituisce lo stream di input del socket; questo stream è utilizzato per leggere i dati provenienti dal socket.

OutputStream getOutputStream(): restituisce lo stream di output del socket; questo stream è utilizzato per scrivere dati nel socket.

void setSoTimeout(int timeout): imposta il timeout per operazioni di lettura dal socket; se il tempo specificato trascorre genera una InterruptedIOException.

String toString(): restituisce una rappresentazione del socket del tipo "Socket[addr=hostname/192.168.90.82,port=3575,localport=1026]"

Schema tipico di un client TCP

- Creare un socket verso il computer e la porta del server col quale si vuole comunicare.
- Attaccare al socket una catena di stream di input e di output.
- Per comunicare leggere e scrivere su questa catena di stream.
- Chiudere le catene di stream
- Chiudere il socket.

Connessione ad un time-server con Java

```
import java.io.*;
import java.net.*;
public class SocketTest {
  public static void main (String args[]) {
    try {
      Socket s = new Socket ("time-A.timefreq.bldrdoc.gov",13);
      BufferedReader in = new BufferedReader
                                (new InputStreamReader (s.getInputStream()));
      while (true) {
        String line = in.readLine();
        if (line == null)
          break;
        else
          System.out.println (line);
      }
    } catch (IOException e) { System.out.println(e); }
  }
}
```

Schema tipico di un Server TCP

- Apre un ServerSocket sulla porta prestabilita.
- Esegue un ciclo senza fine richiamando il metodo *accept()* di ServerSocket. Questo ritorna un normale socket in seguito alla connessione di un client.
- Crea uno *stream* di input e uno di output e li aggancia al socket.
- Esegue operazioni di lettura e scrittura su questi stream.
- Chiude gli stream (problema del flushing)
- Chiude il socket
- Si rimette in ascolto.

java.net.ServerSocket

ServerSocket(int port): crea un server socket che controlla una porta.

Socket accept(): rimane in attesa di una connessione, e restituisce un socket tramite il quale si effettua la comunicazione.

void close(): chiude il server socket.

InetAddress getInetAddress(): restituisce l'indirizzo locale di questo server socket.

int getLocalPort(): restituisce la porta locale di questo server socket.

Timeout di Socket

```
Socket s = new Socket (…);
s.setSoTimeout(10000);
```

Le operazioni di lettura che seguono generano una InterruptedIOException quando è stato raggiunto il timeout.

```
try {
  String line;
  while (line = in.readLine()) != null) {
     process line
  }
} catch (InterrupedIOException e)
{
  react to timeout
}
```

Questo timeout può essere settato su un socket già istanziato. Tuttavia si potrebbe verificare un blocco indefinito già in fase di creazione del socket, fino a quando non si stabilisce la connessione con l’host.