Lezione n.9
LPR
UNIFORM RESOURCE LOCATOR
MULTICAST

05/05/2008

Laura Ricci

RIASSUNTO DELLA PRESENTAZIONE

- URL: concetti generali, accesso
- Comunicazione tra gruppi di processi
- Gruppi di multicast
- Multicast affidabile a livello applicativo
- Java Multicast API

URI: UNIFORM RESOURCE IDENTIFIER

URI (Uniform Resource Identifier)

- meccanismo che consente l' identificazione univoca di una risorsa, ma non indica necessariamente come individuare la risorsa
- la risorsa non deve necessariamente essere accessibile via Internet
- risorse : documenti, servizi, numeri di telefono, libri,.....
- sintassi URI: scheme: scheme-specific-part

```
ftp://ftp.is.co.za/rfc/rfc1808.txt

http://www.ietf.org/rfc/rfc2396.txt

ldap://[2001:db8::7]/c=GB?objectClass?one

mailto:John.Doe@example.com

news:comp.infosystems.www.servers.unix

tel:+1-816-555-1212

telnet://192.0.2.16:80/

urn:oasis:names:specification:docbook:dtd:xml:4.1.2
```

URL: UNIFORM RESOURCE LOCATOR

- Può identificare diversi tipi di risorsa
  - un file in formato HTML memorizzato in un server HTTP
  - un generico file memorizzato su disco locale o su un host remoto
  - un qualsiasi servizio offerto da un server
  - un programma (es: una query per un database). In questo caso può contenere valori che costituiscono parametri di ingresso cel programma stesso
- Esempi di URL valide
  - http: //www.di.unipi.it/ ~ricci/ricerca.html
  - file: ///C:/Documents%20and%20Settings/Documenti
  - telnet: 129.135.2.6
  - http://en.wikepedia.org/w/index.php/?title=Main_page&action=raw

URI: UNIFORM RESOURCE LOCATOR

- Esistono due diversi tipi di URI
  - Uniform Resource Locators (URLs)
  - Uniform Resource Names (URNs)
- URN
  - identificano in modo univoco una risorsa,indipendentemente dalla sua locazione
  - utili quando
    - una risorsa viene spostata da un sito ad un altro
    - una risorsa sia replicata in più siti

URL: UNIFORM RESOURCE LOCATOR

- E' composta di due parti
  - la prima specifica come accedere alla risorsa
  - la seconda indica dove si trova la risorsa
- Esempio:

  http: /www.di.unipi.it/~ricci/ricerca.html
  - http: indica che per accedere alla risorsa si utilizza il protocollo http e che la risorsa ricercata è gestita da un server http
  - www. di.unipi.it indica il nome logico del server che gestisce la risorsa
  - ~ricci/ricerca.html specifica la directory ed il nome del file a cui vogliamo accedere (che vogliamo visualizzare)
- La parte del protocollo può specificare: file, ftp, http, https, news, telnet.....

URL: UNIFORM RESORCE LOCATOR

- Forma generale di una URL:

  protocol://hostname:port/path/filename?query#fragment

- protocol: protocollo
- hostname: nome (o indirizzo IP) del server che fornisce la risorsa.
- port: porta associata al server. Può essere omesso se il server è in ascolto sulla porta 80
- path: individua una specifica directory del server.
  - in generale non tutto il filesystem sul server è accessibile dall'esterno
  - il path indica la directory radice della parte pubblica del file system
- filename: indica un file nella directory specificata dal path. Se non viene specificato il server restituisce un file predefinito, in genere per default si considera il file index.html o welcome.html

URL: UNIFORM RESORCE LOCATOR

- Forma generale di una URL:

  protocol://hostname:port/path/filename?query#fragment

- query: se la URL riferisce un programma (ad esempio un programma CGI) il campo query può contenere parametri di ingresso del programma

- #fragment: indica una parte della risorsa remota (ad esempio un anchor in un documento HTML)

URL: UNIFORM RESOURCE LOCATOR

- In JAVA per creare un oggetto URL:

  URL u = **new** URL("http://www.cli.di.unipi.it/");

  u è un esempio di URL assoluto
- E' anche possibile creare una URL relativa, che ha la forma

  URL (URL baseURL, String relativeURL),
- Esempio:

  URL cli = **new** URL (" http ://www.cli.di.unipi.it");

  URL faq = **new** URL (cli, "faq.php");
- la URL risultante puntare a http:www.cli.di.unipi.it/faq.php
- i costruttori possono lanciare una MalformedURLException

FATTORIZZAZIONE DI URL

```
import java.net.*;
import java.io.*;
public class provaurl050508
{public static void main(String[] args) throws Exception
      {String url = "http://www.cli.di.unipi.it:80/faq.php?item=389";
      URL cli = new URL(url);
      System.out.println("protocol = " + cli.getProtocol());
      System.out.println("authority = " + cli.getAuthority());
      System.out.println("host = " + cli.getHost());
      System.out.println("port = " + cli.getPort());
      System.out.println("path = " + cli.getPath());
      System.out.println("query = " + cli.getQuery());
      System.out.println("filename = " + cli.getFile());
      System.out.println("ref = " + cli.getRef());  }    }
```

FATTORIZZAZIONE DI URL

Nell'esempio precedente, usando come URL

"http://www.cli.di.unipi.it:80/faq.php?item=389"

protocol = http
authority = www.cli.di.unipi.it:80
host = www.cli.di.unipi.it
port = 80
path = /faq.php
query = item=389
filename = /faq.php?item=389
ref = null

URL: REPERIMENTO DI RISORSE

- una volta creato un oggetto URL si può invocare il metodo openStream() per
- ottenere uno stream da cui poter leggere il contenuto dell'URL.
- il metodo openStream()
  - si connette al server che memorizza la risorsa
  - effettua l'handshaking iniziale
  - restituisce un oggetto di tipo InputStream.
- leggere da un URL è analogo a leggere da un qualsiasi InputStream.

URL:REPERIMENTO DI RISORSE

```
import java.net.*;
import java.io.*;
public class udpexample
{ public static void main(String[] args) throws Exception
   {String url = "http://www.cli.di.unipi.it/";
   URL cli = new URL(url);
   BufferedReader in = new BufferedReader(
                        new InputStreamReader(cli.openStream()));
   String inputLine;
   while((inputLine = in.readLine()) != null)
        System.out.println(inputLine);
   in.close();}}
```

URL: REPERIMENTO DI RISORSE

Nell'esercizio precedente, leggendo l'UTL http:\\www.cli.di.unipi.it, viene stampato

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="it">
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
 <link rel="stylesheet" href="/cdc.css" type="text/css">
<link rel="alternate" type="application/rss+xml" title="Ultime notizie"
href="/feed.php">
 <title>Home CdC </title>
</head>
<body bgcolor="#ced8e0">                    ..............
```

URL:REPERIMENTO DI RISORSE

Scrivere un programma che acceda ad un documento HTML di cui si conosce l'indirizzo mediante i metodi della classe URL. Dopo aver reperito il sorgente HTML del documento, mediante il programma JAVA, accedere allo stesso documento mediante un browser e confrontare i due sorgenti HTML ottenuti

GRUPPI DI PROCESSI: COMUNICAZIONE

comunicazioni di tipo unicast = coinvolgono una sola coppia di processi

diverse applicazioni di rete richiedono un tipo di comunicazione che coinvolga un gruppo di hosts.

Applicazioni classiche:

- usenet news: pubblicazione di nuove notizie ed invio di esse ad un gruppo di hosts interessati
- videoconferenze: un segnale audio video generato su un nodo della rete deve essere ricevuto dagli hosts associati ai partecipanti alla videoconferenza

GRUPPI DI PROCESSI: COMUNICAZIONE

Altre applicazioni:

- massive multiplayer games: alto numero di giocatori che intergiscono in un mondo virtuale
- chats
- DNS (Domain Name System): aggiornamenti delle tabelle di naming inviati a gruppi di DNS
- instant messaging
- applicazioni P2P

GRUPPI DI PROCESSI: COMUNICAZIONE

Comunicazione tra gruppi di processi realizzata mediante multicasting (one to many communication).

Comunicazione di tipo multicast

- un insieme di processi formano un gruppo di multicast
- un messaggio spedito da un processo a quel gruppo viene recapitato a tutti gli altri partecipanti appartenenti a *G*
- Un processo può lasciare un gruppo di multicast quando non è più interessato a ricevere i messaggi del gruppo

COMUNICAZIONE TRA GRUPPI DI PROCESSI

Multicast API: deve contenere primitive per

- unirsi ad un gruppo di multicast (join). E' richiesto uno schema di indirizzamento per identificare univocamente un gruppo.
- lasciare un gruppo di multicast (leave).
- spedire messaggi ad un gruppo. Il messaggio viene recapitato a tutti i processi che fanno parte del gruppo in quel momento
- ricevere messaggi indirizzati ad un gruppo

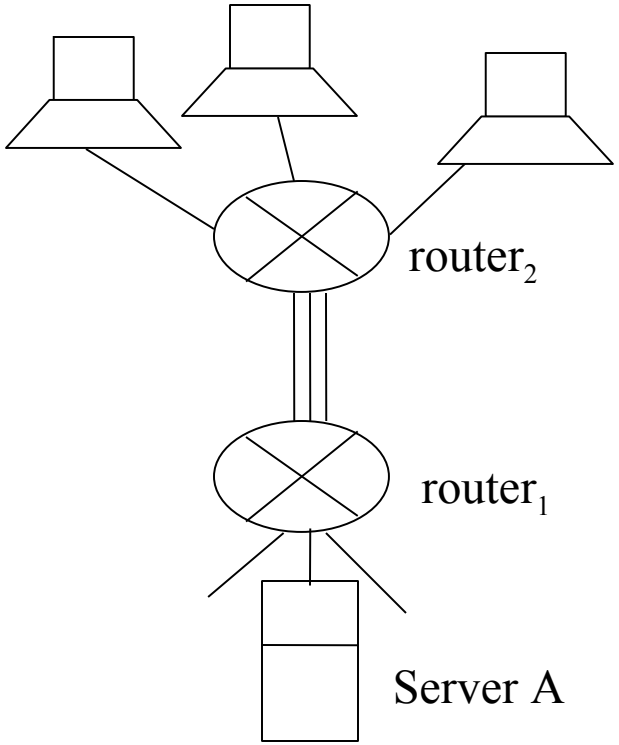
COMUNICAZIONE TRA GRUPPI DI PROCESSI: IMPLEMENTAZIONE

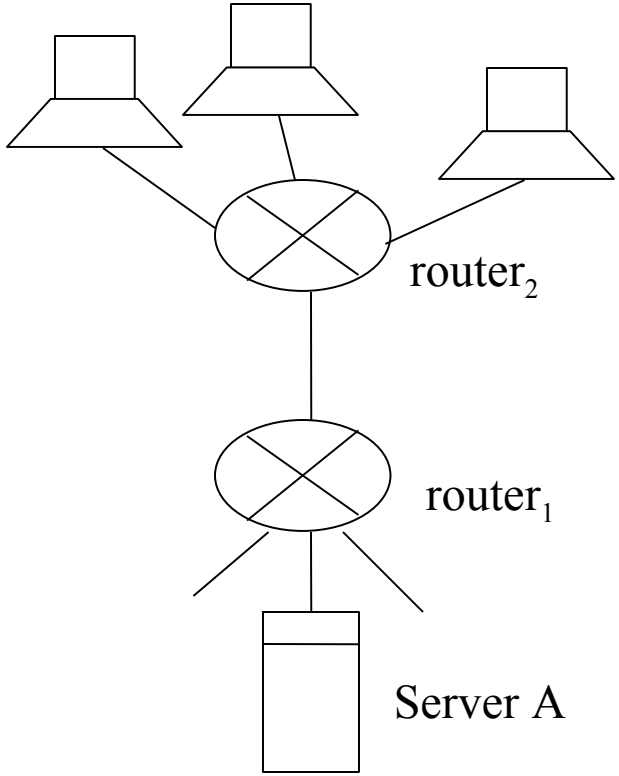
L'implementazione del multicast richiede:

- uno schema di indirizzamento dei gruppi
- un supporto che registri la corrispondenza tra un gruppo ed i partecipanti
- un'implementazione che ottimizzi l'uso della rete nel caso di invio di pacchetti ad un gruppo di multicast

MULTICAST: IMPLEMENTAZIONE

Server A invia un messaggio su un gruppo di multicast composto da 3 clients connessi allo stesso router ($router_2$)





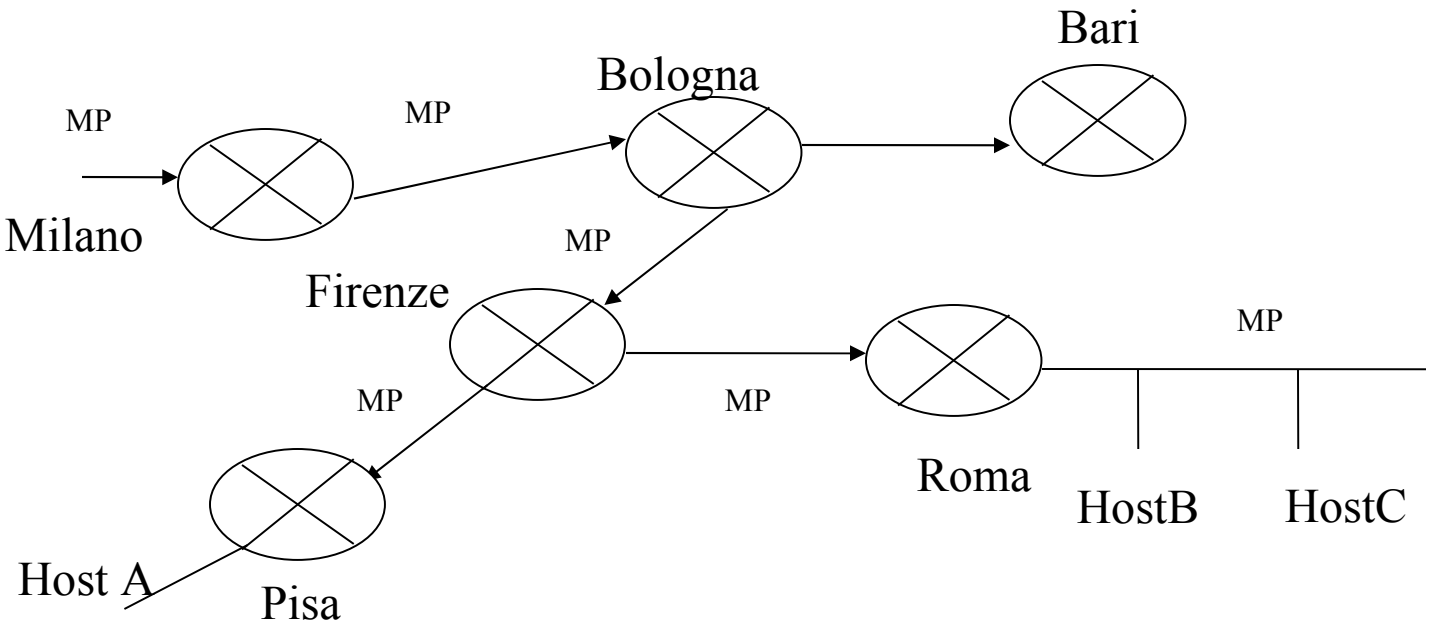
Soluzione 1: $router_1$ invia 3 messaggi distinti con collegamenti di tipo unicast

Soluzione 2: $router_1$ invia un unico messaggio $router_2$ replica il messaggio per i tre clients

MULTICAST: IMPLEMENTAZIONE

Ottimizzazione della banda di trasmissione: il router che riceve un pacchetto di multicast MP invia un unico pacchetto sulla rete. Il pacchetto viene replicato solo quando è necessario.

Esempio: pacchetto di multicast spedito da Milano agli hosts HostA, HostB, HostC



INDIVIDUAZIONE GRUPPI DI MULTICAST

- Indirizzo di multicast: indirizzo IP di classe D, che individua un gruppo di multicast
- Indirizzo di classe D- intervallo 224.0.0.0 – 239-255-255-255

- l'indirizzo di multicast è condiviso da tutti i partecipanti al gruppo
- è possibile associare un nome simbolico ad un gruppo di multicast
- Esempio: 224.0.1.1 ntp.mcast.net (network time protocol distributed service)

INDIVIDUAZIONE GRUPPI DI MULTICAST

- Il livello IP (nei routers) mantiene la corrispondenza tra l'indirizzo di multicast e gli indirizzi IP dei singoli hosts che partecipano al gruppo

- Gruppo di multicast = Insieme di hosts che condividono un indirizzo di multicast

  Permanenti : l'indirizzo di multicast viene assegnato da IANA. L'indirizzo rimane assegnato a quel gruppo, anche se, in un certo istante non ci sono partecipanti

  Temporanei : Esistono solo fino al momento in cui esiste almeno un partecipante

INDIVIDUAZIONE GRUPPI DI MULTICAST

- gli indirizzi di multicast appartenenti all'intervallo

224.0.0.0 - 224.0.0.255

sono riservati per i protocolli di routing e per altre funzionalità a livello di rete

ALL-SYSTEMS.MCAST.NET 224.0.0.1 tutti gli host della rete locale

ALL-ROUTERS-MCAST.NET 224.0.0.2 tutti i routers della rete locale

……

- i routers non inoltrano mai i pacchetti che contengono questi indirizzi multicast

- la maggior parte degli indirizzi assegnati in modo permanente hanno come prefisso 224.0,224.1, 224.2, oppure 239
- lista completa degli indirizzi di multicast riservati:

http://www.iana.org/assignments/multicast-address

http://www.iana.org/assignments/multicast-address

MULTICAST ROUTERS

- per poter spedire e ricevere pacchetti di multicast oltre i confini della rete locale, occorre disporre di un router che supporta il multicast (mrouter)

- problema: disponbilità limitata di mrouters

- per testare se la vostra rete è collegata ad un mrouter, dare il comando

```
% ping all-routers.mcast.net
```

  - se si ottiene una risposta, è disponibile un mrouter sulla sottorete locale.
  - routers che non supportano il multicast, possono utilizzare la tecnica del *tunnelling* = trasmissione di pacchetti di multicast mediante unicast UDP

CONNECTIONLESS MULTICAST

Comunicazione Multicast utilizza il paradigma connectionless

Motivazioni:

- gestione di un alto numero di connessioni
- richieste n(n-1) connessioni per un gruppo di n processi
- comunicazione connectionless adatta per il tipo di applicazioni verso cui è orientato il multicast (trasmissione di dati video/audio).

Esempio: invio dei frames di una animazione. E' più accettabile la perdita occasionale di un frame piuttosto che un ritardo costante tra la spedizione di due frames successivi

UNRELIABLE VS. RELIABLE MULTICAST

Unreliable Multicast (multicast non affidabile):

non garantisce la consegna del messaggio a tutti i processi che partecipano al gruppo di multicast.

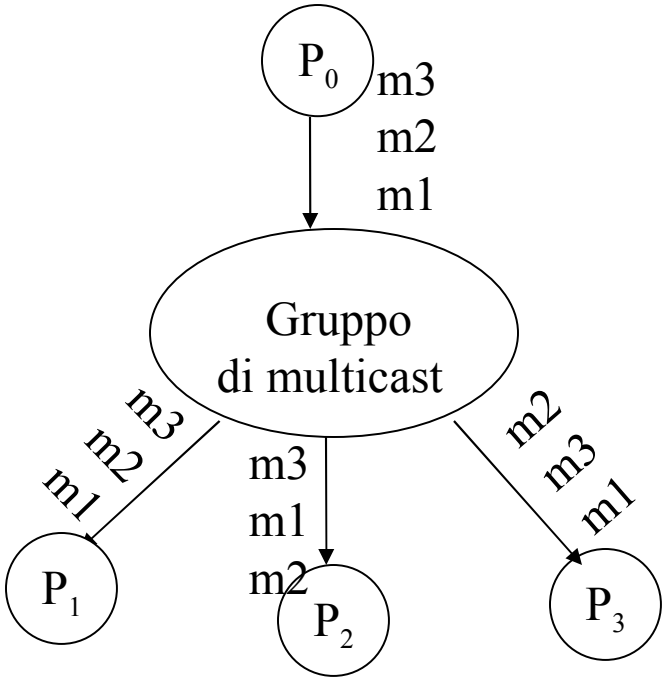
unico servizio offerto dalla multicast JAVA API standard (esistono package JAVA non standard che offrono qualche livello di affidabilità)

Reliable Multicast (multicast affidabile):

- garantisce che il messaggio venga recapitato una sola volta a tutti i processi del gruppo
- può garantire altre proprietà relative all'ordinamento con cui i messaggi spediti al gruppo di multicast vengono recapitati ai singoli partecipanti

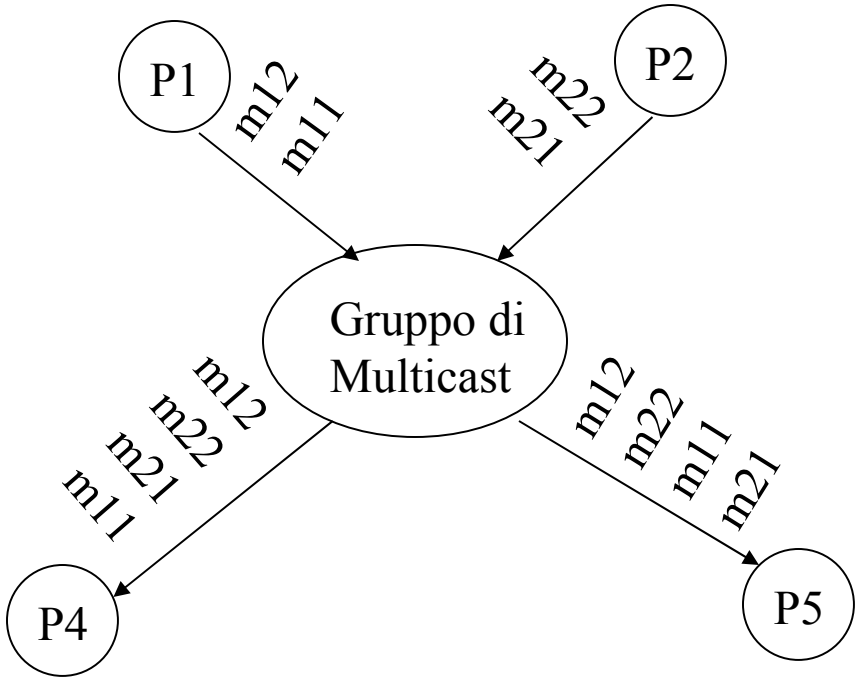
UNORDERED MULTICASTING

Unordered Multicasting: nessuna garanzia circa l'ordine con cui i messaggi vengono recapitati ai destinatari. Esempio: p_0 invia, nell'ordine, m_1,m_2,m_3 al gruppo di multicast. I messaggi vengono consegnati ai partecipanti secondo ordini diversi



FIFO MULTICASTING

Se un processo P invia, nell'ordine, i messaggi m_i ed m_j ad un gruppo di multicast, allora tutti i processi del gruppo ricevono i messaggi secondo quell'ordine. Non viene garantito nessun ordinamento per messaggi spediti da processi diversi



CAUSAL ORDER MULTICAST

Relazione di ordinamento causale tra coppie di messaggi: mi $\rightarrow$ mj

mi precede mj se l'invio di mj da parte di un processo P dipende dalla ricezione di mj da parte dello stesso P.

Esempio: P_1,P_2,P_3 appartengono allo stesso gruppo di multicast. P_1 invia un messaggio m_i a P_2. P_2, dopo aver ricevuto il messaggio di P_1, spedisce un messaggio m_j sul gruppo di multicast. In questo caso: $m_i \rightarrow m_j$

Causal Order Multicast: tutti i processi ricevono i messaggi spediti al gruppo di multicast secondo l'ordinamento causale

ATOMIC ORDER MULTICAST

Atomic order multicast: garantisce che tutti i messaggi spediti ad un gruppo di multicast vengano consegnati ad ogni partecipante al gruppo secondo lo stesso ordine.

Esempio: P_1 invia m_1, P_2 invia m_2, P_3 invia m_3

l'ordine con cui i partecipanti riceveranno i messaggi può essere uno qualsiasi tra

m1-m2-m3, m1-m3-m2, m2-m1-m3, m2-m3-m1, m3-m1-m2, m3-m2-m1

ma l'ordine è lo stesso per ogni partecipante

MULTICAST: L'API JAVA

MulticastSocket = socket su cui spedire/ricevere i messaggi verso/da un gruppo di multicast

```
import java.net.*;
import java.io.*;
public class multicast
    {public static void main (String [  ] args)
    {try {  MulticastSocket ms =new MulticastSocket( );}
        catch (IOException ex) {System.out.println ("errore"); }
    }}
```

- è possibile specificare la porta a cui collegare il multicast socket.
- la classe MulticastSocket estende la DatagramSocket

MULTICAST: L'API JAVA

```
import java.net.*;
import java.io.*;
public class multicast
{public static void main (String [ ] args)
{   try {  MulticastSocket ms =new MulticastSocket(4000);
        InetAddress ia=InetAddress.getByName("226.226.226.226");
       ms.joinGroup (ia);      }
   catch (IOException ex) {System.out.println ("errore"); }}}
```

- operazione necessaria nel caso si vogliano ricevere messaggi dal gruppo di multicast
- lega il multicast socket ad un gruppo di multicast ⇒ tutti i messaggi ricevuti tramite quel socket provengono da quel gruppo
- IOException sollevata se ia non è un indirizzo di multicast

MULTICAST: L'API JAVA

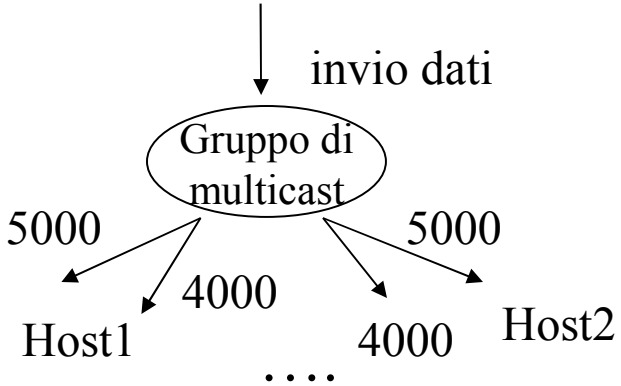
Uso delle porte per multicast sockets:

Unicast

- IP Address individua un host,
- porta individua un servizio

Multicast

- IP Address individua un gruppo di hosts,
- porta consente di partizionare dati di tipo diverso inviati allo stesso gruppo



Esempio: porta 5000 **traffico audio,** porta 4000 **traffico video**

MULTICAST: L'API JAVA

Una porta non individua un servizio (processo) su un certo host

```
import java.io.*;  import java.net.*;
public class provemulticast {
public static void main (String args[]) throws Exception
      { byte[] buf = new byte[10];
      InetAddress ia = InetAddress.getByName("228.5.6.7");
      DatagramPacket dp = new DatagramPacket (buf, buf.length);
      MulticastSocket ms = new MulticastSocket (4000);
      ms.join(ia);
      ms.receive(dp);    } }
```

se attivo due istanze di provamulticast sullo stesso host, non viene sollevata una BindException (che viene invece sollevata se MulticastSocket è sostituito da un DatagramSocket)

MULTICAST SNIFFER

```
import java.net.*; import java.io.*;
public class multicastsniffer {
public static void main (String[] args)
{InetAddress group = null;
int port = 0;
try{
    group = InetAddress.getByName(args[0]);
    port = Integer.parseInt(args[1]);
} catch(Exception e){System.out.println("Uso:java multicastsniffer
    multicast_address port");
System.exit(1); }
```

MULTICAST SNIFFER

```
MulticastSocket ms=null;
try{ ms = new MulticastSocket(port);
      ms.joinGroup(group);
      byte [ ] buffer = new byte[8192];
      while (true)
      {DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
      ms.receive(dp);
      String s = new String(dp.getData());
      System.out.println(s);}
      } catch (IOException ex){System.out.println (ex);}
```

MULTICAST SNIFFER

```
finally{
        if (ms!= null) {
                try{
                ms.leaveGroup(group);
                ms.close();
                } catch (IOException ex){}
                }}}}
```

- Il programma riceve in input il nome il nome simbolico di un gruppo di multicast si unisce al gruppo e 'sniffa' i messaggi spediti su quel gruppo, stampandone il contenuto
- Remainder: il blocco finally associato ad una **try...catch** viene eseguito qualsiasi sia il modo con cui il blocco try.. termina. Esempio: un return, una exxezione,.....

MULTICAST: L'API JAVA

Per spedire messaggi ad un gruppo di multicast:

- creare un multicastSocket su una porta anonima
- non è necessario collegare il multicast socket ad un gruppo di multicast
- creare un pacchetto inserendo nell'intestazione l'indirizzo del gruppo di multicast a cui si vuole inviare il pacchetto
- Spedire il pacchetto tramite il socket creato

  public void send (DatagramPacket p) throws IOException

MULTICAST: L'API JAVA

Spedizione di un pacchetto tramite un multicast socket

```
import java.io.*;
import java.net.*;
public class multicast {
public static void main (String args[])
{  try   {
      InetAddress ia=   InetAddress.getByName("228.5.6.7");
      byte [  ] data;
      data="hello".getBytes();
      int port= 6789;
      DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
      MulticastSocket ms = new MulticastSocket(6789);
      ms.send(dp);
      } catch(IOException ex){ System.out.println(ex);}}}
```

MULTICAST: TIME TO LIVE

- IP Multicast Scoping: meccanismo utilizzato per limitare la diffusione sulla rete di un pacchetto inviato in multicast
- ad ogni pacchetto IP viene associato un valore rappresentato su un byte, riferito come TTL (Time-To-Live) del pacchetto
- TTL assume valori nell'intervallo 0-255
- TTL indica il numero massimo di routers che possono essere attraversati dal pacchetto
- il pacchetto viene scartato dopo aver attraversato TTL routers
- meccanismo introdotto originariamente per evitare loops nel routing dei pacchetti

MULTICAST: TIME TO LIVE

Internet Scoping, implementazione

- il mittente specifica un valore per del TTL per i pacchetti spediti
- il TTL viene memorizzato in un campo dell'header del pacchetto IP
- TTL viene decrementato da ogni router attraversato
- Se TTL = 0 ⇒ il pacchetto viene scartato

MULTICAST: TIME TO LIVE

Valori consigliati per il ttl

Destinazione	Valori di ttl
processi sullo stesso host	0
processi sulla stessa sottorete locale	1
processi su reti locali gestite dallo stesso router	16
	....
processi allocati su un qualsiasi host di Internet	255

TIME TO LIVE: API JAVA

- Assegna il valore di default = 1 al TTL (i pacchetti di multicast non possono lasciare la rete locale)

- Per modificare il valore di default: posso associare il ttl al multicast socket

```
MulticastSocket s = new MulticastSocket( );
s.setTimeToLive(1);
```

MULTICAST: ESERCIZIO 1

Definire un Server TimeServer, che invia su un gruppo di multicast dategroup, ad intervalli regolari,la data e l'ora. L'attesa tra un invio ed il successivo può essere simulata mediante il metodo sleep( ). L'indirizzo IP di dategroup viene introdotta linea di comando.
Definire quindi un client TimeClient che si unisce a dategroup e riceve, per dieci volte consecutive, data ed ora, le visualizza, quindi termina.

MULTICAST: ESERCIZIO 2

Si consideri una applicazione composta da *m* servers $Magazzino_1$, ..., $Magazzino_m$ ed n clients $Client_1$, ...,$Client_n$. Ognuno dei client ed dei server è allocato all'interno di uno spazio virtuale bidimensionale ed può essere individuato dalle sue coordinate cartesiane all'inteno di questo spazio. Ogni Magazzino possiede lo stesso insieme di articoli, ognuno caratterizzato da un codice e dalla quantità presente in quel magazzino (scorta). Le informazioni riguardanti gli articoli vengono caricate da un file.
Ogni magazzino invia periodicamente le sue coordinate ad un gruppo di multicast M a cui ogni client si collega all'inizio della propria esecuzione. Quando un client riceve una informazione da M, la scarta se l'ha ricevuta precedentemente, altrimenti la memorizza in una struttura dati opportuna.

MULTICAST: ESERCIZIO 2 (CONTINUA)

Ogni client riceve dall'utente, in modo interattivo, il codice e la quantità di un prodotto e richiede tale prodotto al magazzino più vicino, tramite una connessione UDP. Il Magazzino invia al client un ack, nel caso possieda tale prodotto in quantità sufficiente, altrimenti invia un nack. Il client visualizza la risposta e, nel caso di nack, provvede ad inviare una ulteriore richiesta al Magazzino piu' vicino, scelto tra i rimanenti.

Il procedimento termina quando il Client riceve una risposta positiva da un magazzino, oppure quando sono stati consultati tutti i magazzini.