Socket UDP

Marco Danelutto
LPRb A.A. 2007-2008

Modello di comunicazione UDP

- modello datagram
  - messaggio con indirizzo
  - payload byte[] (max 64K - HDR : 65507)
  - creato indipendentemente
  - spedito/ricevuto utilizzando DatagramSocket

Datagram

- public Datagram(byte[] buf, int len, InetAddress destla, int port)
  - crea un datagram
  - indirizzato a destla, porta port
  - con payload buf, di lunghezza len

Datagram (2)

- public DatagramPacket(byte [] buf, int len)
- dp = new DatagramPacket(...);
- dp.setAddress(InetAddress ia);
- dp.setPort(port);

DatagramSocket

- public DatagramSocket(int port) throws SocketException
- public void send(DatagramPacket dp) throws IOException
- public void receive(DatagramPacket dp) throws IOException

Receive

- Allocazione del buffer di ricezione a carico del destinatario
  - devo dimesionarlo sulla dim massima dei pacchetti che posso ricevere
  - ricezione bloccante
  - si riceve sempre il DatagramPacket spedito (intero)

Receive

- metodi ispettori
  - public InetAddress getAddress();
  - restituisce l'indirizzo cui è destinato il DatagramPacket (prima della spedizione)
  - restituisce l'indirizzo da cui è stato spedito il DatagramPacket (dopo la ricezione)

Comunicazione tipica

- non esiste concetto di server e di client
- tutti hanno DatagramSocket uguali
- chi spedisce, di solito riceve sullo stesso DatagramSocket
- chi riceve e manda una risposta prende indirizzo di ritorno dal pacchetto

Cliente (chiedo qualcosa)

- byte [] buffer = ...
  DatagramPacket dp = new
  DatagramPacket
  (buffer,len,destIa,destPort);
  DatagramSocket ds = new DatagramSocket
  ();
  ds.send(dp);
  DatagramPacket answer = ... ;
  ds.receive(answer);

Server (rispondo)

- DatagramPacket rich = ... ;
  DatagramSocket ds = new
  DatagramSocket(wellKnownPort);
  ds.receive(rich);
  // prepara riposta
  DatagramPacket risp =
  new DatagramPacket(buf,len);
  risp.setAddress(rich.getAddress());
  risp.setPort(rich.getPort());

Opzioni DatagramSocket

- public void setSoTimeout(int millis)
- stessa funzione dei Socket TCP
- receive interrotta dopo millis millisecondi
- da una InterruptedIOExcetion

Multicast

- operazione di trasmissione da uno a molti
- un mittente, tanti destinatari
  - multicast : tutti quelli che hanno fatto richiesta
  - broadcast : tutti quanti i destinatari possibili

Multicast: spedizione

- public MulticastSocket(int port)
- public MulticastSocket()
- ms.send(dp)
  - metodo ereditato dal DatagramSocket

Multicast: ricezione

- public void join(InetAddress multicastAddr)
- public void receive(DatagramPacket dp)
  - dopo aver fatto la join, ricevo tutti i messaggi diretti al gruppo
- public void leaveGroup(InetAddress ia)

TTL

- public void setTimeToLive(int ttl)
  - numero di hop (router) prima che il pacchetto venga scartato
  - serve per evitare flooding massiccio sulla rete
- deprecato: **send(dp,ttl)**

Gruppi di multicast

- identificati da indirizzi IP
  - 224.0.0.0 -:- 239.255.255.255
- indirizzi "riservati"
  - 224.0.0.0
  - 224.0.0.1 tutti i sistemi della rete loc
  - 224.0.0.2 tutti i router della rete loc

Indirizzi “liberi”

- da 225.0.0.0 a 238.255.255.255
  - ipotesi restrittiva / garantista
- per gli esercizi
  - usate pezzi del numero di matricola
  - 067191 -> 225.6.71.91
  - 234654 -> 225.23.46.54