Esercizi su UDP

Esercitazione di Laboratorio di Programmazione di Rete A

Daniele Sgandurra

Università di Pisa

22/10/2008

Un Tipico Client UDP

Un client UDP invia datagrammi ad un server in attesa di essere contattato. Un client UDP tipicamente esegue tre passi:

1. Costruisce un'istanza di un `DatagramSocket`.
2. Comunica col server inviando e ricevendo istanze di `DatagramPacket`, tramite i metodi `send()` e `receive()` di `DatagramSocket`.
3. Quando ha finito, dealloca il socket usando il metodo `close()` di `DatagramSocket`.

Un Tipico Server UDP

Un server UDP rimane in attesa passiva di client che si connettono. Tipicamente, un server UDP esegue tre passi:

1. Costruisce un'istanza di un `DatagramSocket`, specificando la porta locale di ascolto.
2. Riceve un'istanza di un `DatagramPacket` tramite il metodo `receive()` di `DatagramSocket`:
   - quando il metodo ritorna, il datagramma contiene l'IP e porta del mittente, per cui il server sa a chi rispondere.
3. Comunica col client inviando e ricevendo istanze di `DatagramPacket`, tramite i metodi `send()` e `receive()` di `DatagramSocket`.

La classe `DatagramSocket`

La classe `DatagramSocket` rappresenta un socket per inviare e ricevere datagrammi UDP.

- Tipico costruttore per il client:

  ```
  DatagramSocket()
  ```

  costruisce un socket UDP e lo lega ad una porta libera dell'host.

- Tipico costruttore per il server:

  ```
  DatagramSocket(int port)
  ```

  costruisce un socket UDP e lo lega alla porta `port`.

- Per inviare pacchetti:

  ```
  send(DatagramPacket p)
  ```

- Per ricevere pacchetti:

  ```
  receive(DatagramPacket p)
  ```

La classe `DatagramPacket`

- In Java, i messaggi UDP sono rappresentati da istanze della classe `DatagramPacket`.
- Per inviare datagrammi, si costruisce un `DatagramPacket`, contenente i dati da inviare, e si passa questo oggetto come argomento del metodo `send()` di un `DatagramSocket`.
- Per ricevere datagrammi, si costruisce un `DatagramPacket` associato a un buffer preallocato (un `byte[]`), e si passa il datagramma come argomento del metodo `receive()` di un `DatagramSocket`.

La classe `DatagramPacket`

- Un costruttore per inviare pacchetti:

```
DatagramPacket(byte[] buf, int offset, int length,
               InetAddress address, int port)
```

  - `buf`: contiene i dati da inviare nel datagramma;
  - `offset`: la locazione dei dati nel buffer;
  - `length`: la lunghezza dei dati nel buffer;
  - `address`: l'IP di destinazione;
  - `port`: la porta di destinazione.

- Un costruttore per ricevere pacchetti:

```
DatagramPacket(byte[] buf, int length)
```

  - `buf`: il buffer per contenere i dati ricevuti;
  - `length`: la lunghezza dei byte da leggere.

Inviare Dati

- Quando si invia un datagramma, è compito dell'applicazione formattare i dati in un array di byte, tramite le classi `DataOutputStream` o `ObjectOutputStream` collegate con un `ByteArrayOutputStream`. Ad es:

```
DatagramPacket packet = new DatagramPacket();
ByteArrayOutputStream baos =
    new ByteArrayOutputStream();
DataOutputStream dos = new DataOutputStream(baos);
dos.writeInt(...);
dos.flush();
packet.setData(baos.toByteArray(), baos.size());
//send UDP packet
```

Ricevere Dati

- Quando si riceve un datagramma, Java inserisce i dati nell'array di byte associato al `DatagramPacket` a partire da `offset` fino a `length - 1`. Ad es.:

```
byte[] buffer = new byte[...];
DatagramPacket packet =
    new DatagramPacket(buffer, buffer.length);
//receive UDP packet
ByteArrayInputStream bais =
    new ByteArrayInputStream(packet.getData(),
    packet.getOffset(), packet.getLength());
DataInputStream dis = new DataInputStream(bais);
int i = dis.readInt();
```

Esercizio 1

- Scrivere un'applicazione composta da un processo Sender ed un processo Receiver. Il Sender riceve da linea di comando una stringa, l'indirizzo del Receiver (IP e porta) ed invia al Receiver la stringa tramite UDP. Il Receiver riceve il messaggio e lo visualizza.

Esercizio 2

- Programmare un server CountDownServer che fornisce un semplice servizio: ricevuto da un client un valore intero n, il server spedisce al client in sequenza i valori $n-1, n-2, \cdots, 1$. L'interazione tra i client e CountDownServer avviene su UDP.
- Si richiede di implementare due versioni di CountDownServer:
  - come server iterativo, l'applicazione riceve la richiesta di un client, gli fornisce il servizio e solo quando ha terminato va a servire altre richieste;
  - come server concorrente, l'applicazione definisce un thread che ascolta le richieste dei client dalla porta UDP a cui è associato il servizio ed attiva un thread diverso per ogni richiesta ricevuta. Ogni thread si occupa di servire un client.

Soluzioni

Inviare la soluzione degli esercizi (solo i file .java) a :

`ricci@di.unipi.it`
`sgandurra@di.unipi.it`

Tra due settimane saranno disponibili le soluzioni.