Esercitazione n.5
LPR-A-08
Il protocollo UDP: Socket e Datagram

26/10/2008

Vincenzo Gervasi

PER ESEGUIRE GLI ESERCIZI SU UN UNICO HOST

- Attivare il client ed il server in due diverse shell
- Se l'host è connesso in rete: utilizzare come indirizzo IP del mittente/destinatario l'indirizzo dell'host su cui sono in esecuzione i due processi (reperibile con InetAddress.getLocalHost( ))
- Se l'host non è connesso in rete utilizzare l'indirizzo di loopback ("localhost" o 127.0.0.1)
- Tenere presente che mittente e destinatario sono in esecuzione sulla stessa macchina ⇒ devono utilizzare porte diverse
- Mandare in esecuzione per primo il server, poi il client

ESERCIZIO 1: INVIO DI DATAGRAM UDP

Esercizio:

Scrivere un'applicazione composta da un processo Sender ed un processo Receiver. Il Receiver riceve da linea di comando la porta su cui deve porsi in attesa. Il Sender riceve da linea di comando una stringa e l'indirizzo del Receiver (indirizzo IP + porta), e invia al Receiver la stringa. Il Receiver riceve la stringa e stampa, nell'ordine, la stringa ricevuta, l'indirizzo IP e la porta del mittente.

Considerare poi i seguenti punti:

- cosa cambia se mando in esecuzione prima il Sender, poi il Receiver rispetto al caso in cui mando in esecuzione prima il Receiver?
- nel processo Receiver, aggiungere un time-out sulla receive, in modo che la receive non si bocchi per più di 5 secondi. Cosa accade se attivo il receiver, ma non il sender?

ESERCIZIO 1: INVIO DI DATAGRAM UDP

- Modificare il codice del Sender in modo che usi lo stesso socket per inviare lo stesso messaggio a due diversi receivers. Mandare in esecuzione prima i due Receivers, poi il Sender. Controllare l'output dei Receiver.
- Modificare il codice del Sender in modo che esso usi due sockets diversi per inviare lo stesso messaggio a due diversi receivers. Mandare in esecuzione prima i due Receivers, poi il Sender.
- Modificare il codice ottenuto al passo precedente in modo che il Sender invii una sequenza di messaggi ai due Receivers. Ogni messaggio contiene il valore della sua posizione nella sequenza. Il Sender si sospende per 3 secondi tra un invio ed il successivo. Ogni receiver deve essere modificato in modo che esso esegua la receive in un ciclo infinito.
- Modificare il codice ottenuto al passo precedente in modo che il Sender non si sospenda tra un invio e l'altro. Cosa accade?
- Modificare il codice iniziale in modo che il Receiver invii al Sender un ack quando riceve il messaggio. Il Sender visualizza l'ack ricevuto.

ESERCIZIO 2: COUNT DOWN SERVER

Si richiede di programmare un server CountDownServer che fornisce un semplice servizio: ricevuto da un client un valore intero n, il server spedisce al client i valori n-1,n-2,n-3,….,1, in sequenza.

La interazione tra i clients e CountDownServer è di tipo connectionless.

Si richiede di implementare due versioni di CountDownServer

- realizzare CountDownServer come un server iterativo. L'applicazione riceve la richiesta di un client, gli fornisce il servizio e solo quando ha terminato va a servire altre richieste
- realizzare CountDownServer come un server concorrente. Si deve definire un thread che ascolta le richieste dei clients 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.

Opzionale: Il client calcola il numero di pacchetti persi e quello di quelli ricevuti fuori ordine e lo visualizza alla fine della sessione.

Utilizzare le classi ByteArrayOutput/InputStream per la generazione/ricezione dei pacchetti.