Esercitazione n.6
LPR-A-09
TCP: Stream Sockets

16/11/2009

Vincenzo Gervasi

ESERCIZIO; COMPRESSIONE DI FILE

- Progettare un'applicazione client/server in cui il server fornisca un servizio di compressione di dati.
- Il client legge chunks di bytes da un file e li spedisce al server che provvede alla loro compressione. Il server restituisce i bytes in formato compresso al client che provvede a creare un file con lo stesso nome del file originario e con estensione gz, che contiene i dati ricevuti dal server.
- La comunicazione tra client e server utilizza il protocollo TCP.
- Per la compressione si può utilizzare la classe JAVA GZIPOutputStream.
- Individuare le condizioni necessarie affinché il programma generi una situazione di deadlock. Si dimostri sperimentalmente che tale situazione si verifica realmente quando tali condizioni sono soddisfatte.

COMPRESSIONE DI FILE (2)

- Estensioni all'esercizio precedente:
  - Si realizzi il client con due thread, uno per la lettura del file originale e l'invio al server, l'altro per la lettura dei dati compressi dal server e la scrittura nel file .gz (si curi, ove necessario, la sincronizzazione fra i due e la gestione degli errori)
  - Si realizzi il server in modo da poter gestire in maniera efficiente connessioni multiple da parte di più client – si ipotizzi anche che il numero di richieste di servizio possa essere elevato
- Quali costrutti per il multithreading sono maggiormente indicati in ciascuno dei due casi di cui sopra?

STREAM MODE SOCKET API: INTERAZIONE CON SERVERS PREDEFINITI

Esercizio: considerare un servizio attivo su una porta pubblicata da un Server (es 23 Telnet, 25 SMTP, 80 HTTP). Definire un client JAVA che utilizzi tale servizio, dopo aver controllato che sia attivo.

Provare anche con i seguenti servizi (vedere JAVA Network Programming)

Daytime(porta 13): il client richiede una connessione sulla porta 13, il server invia la data e chiude la connessione

Echo (port 7): il client apre una connesione sulla porta 7 del server ed invia un messaggio. Il server restituisce il messaggio al client

Finger (porta 79): il client apre una connessione ed invia una query, il Server risponde alla query

Whois (porta 43): il client invia una stringa terminata da return/linefeed. La stringa può contenere, ad esempio, un nome. Il server invia alcune informazioni correlate a quel nome