LABORATORIO Reti di Calcolatori − appello di febbraio 2017

Corso di Laurea triennale in INFORMATICA (F1X)

Prova Socket

[Si consiglia come operazione preliminare di creare una directory in cui raccogliere tutti i file sorgente.]

Si realizzi un sistema composto da un server, più stazioni ferroviarie e più utenti[1] – potenzialmente *non* residenti sullo stesso host – che implementano una piattaforma per il monitoraggio del traffico ferroviario aderendo alle seguenti specifiche:

STAZIONE:

- È implementata in linguaggio Java, e comunica con il server sfruttando i servizi del protocollo UDP.
- Richiede da tastiera l'immissione di due numeri: il primo indica il codice del treno appena transitato per la stazione (compreso tra 1 e 9), e il secondo indica i minuti di ritardo con cui quel treno sta viaggiando. Tale informazione è inviata al server per aggiornarne la conoscenza sulla situazione del traffico in linea.
- Riceve dal server una conferma di ricezione notifica e la stampa a video. Quindi torna in attesa sull'immissione da tastiera di un nuovo aggiornamento.
- Se il carattere inserito da tastiera è '**.**' lo interpreta come segnale di terminazione del lavoro, lo invia al server e termina.

UTENTE:

- È implementato in linguaggio Java, e comunica con il server sfruttando i servizi del protocollo UDP.
- Quando riceve l'immissione di un numero da tastiera – che rappresenta il codice del treno a cui è interessato – invia il codice al server per sottoporre la query.
- Riceve dal server la risposta contenente l'identità dell'ultima stazione da cui è stato ricevuto un aggiornamento per quel treno, e il ritardo da essa notificato. Stampa l'informazione a video, quindi torna ad attendere l'immissione da tastiera di una nuova quantità.
- Se il carattere inserito da tastiera è '**.**' lo interpreta come segnale di terminazione dell'interazione col server, lo invia al server e termina.

SERVER:

- È implementato in linguaggio Java, e comunica con gli altri processi sfruttando i servizi del protocollo UDP.
- Ogni volta che riceve da una stazione la notifica di un nuovo rilevamento di passaggio di un treno, aggiorna l'informazione mantenuta in memoria per quel treno, stampa la notifica a video e invia una conferma alla stazione. Quindi passa a gestire la prossima richiesta in arrivo da una stazione o un utente.
- Quando riceve una richiesta da un utente, ripesca dalla propria memoria l'identità della stazione che ha inviato l'ultimo aggiornamento e l'entità del ritardo per il treno richiesto, e invia queste informazioni all'utente. Quindi passa a gestire la prossima richiesta in arrivo da una stazione o un utente.
- Il server non termina mai.

A scelta dello studente, l'identità di una stazione può essere rappresentata da una lettera maiuscola, piuttosto che dal suo indirizzo di rete. La definizione del formato dei messaggi, così come i dettagli implementativi non specificati sopra, sono a discrezione dello studente. Tutti i messaggi ricevuti da ogni processo devono essere mostrati a video. Si richiede la gestione delle situazioni anomale.

Modalità di consegna

[1] Il test verrà eseguito con non più di 2 stazioni e 2 utenti.

1. ***Prima*** di consegnare, assicurarsi di ***salvare*** il contenuto di tutti i file sorgente prodotti.
2. Preparare un file archivio .zip oppure .tar, nominato ***cognome_matr***.zip/.tar , contenente **tutti** i file sorgente prodotti (ovvero la compressione della directory contenente i file sorgente).
    - Per preparare un file tar si leggano le istruzioni sotto.
3. Collegarsi al sito http://upload.di.unimi.it
   (dovrebbe essere raggiungibile anche come *Home* nel browser)
4. Autenticarsi con login name e password di Ateneo (cioè @studenti.unimi.it)
5. Fare upload del file .tar oppure .zip
6. Fare logout dal sito.

NB: in caso si procedesse ad effettuare la consegna dell'elaborato più di una volta, concatenare un numero progressivo al nome file su indicato, così che sia possibile individuare l'ultima versione.

NB: per produrre il file **.tar** procedere come segue da prompt di shell:

- posizionarsi nella directory *genitore* di quella in cui sono raccolti i file che si vogliono consegnare
- produrre il file archivio con il comando

  tar cvf *cognome_matr*.tar *nome_directory_da_archiviare*

  (es. tar cvf bianchi_123456.tar esameSocket/)

  Lo output verbose permette di controllare che nell'archivio siano inclusi tutti i file desiderati.