Laboratorio di Reti, Corsi A e B

Simple-Social: implementazione di una Online Social Network

Progetto di Fine Corso – A.A. 2015/16

1. Descrizione del problema

Il progetto consiste nello sviluppo di una rete sociale caratterizzata da un semplice insieme di funzionalità. Per utilizzare queste funzionalità, gli utenti si devono registrare, quindi possono:

- effettuare il login
- ricercare un utente specificandone il nome
- stabilire amicizie con altri utenti
- pubblicare contenuti (testo)
- ricevere contenuti pubblicati dai loro amici, se hanno manifestato interesse in tali contenuti
- effettuare logout

Componenti

La rete sociale deve essere implementata mediante 2 componenti principali, che interagiscono usando diversi protocolli e paradigmi di comunicazione di rete (TCP, UDP e RMI). Ogni componente sarà composta di un insieme di classi definite in fase di design del programma. Le componenti sono le seguenti:

SocialClient
Gestisce l'interazione con l'utente, tramite una user interface, comunica con il SocialServer per eseguire le azioni richieste dall'utente, inoltre partecipa ad un gruppo di multicast per la gestione dei messaggi di keep-alive. La sua struttura è quella di un "thin-client", in quanto la realizzazione delle principali funzionalità è demandata al server.
In seguito indicheremo con SocialClient(u) la componente SocialClient associata all'utente *u*.

SocialServer
Gestisce la fase di registrazione, memorizza tutti gli utenti registrati e le loro reti di amicizie, crea nuove amicizie quando richiesto dagli utenti, gestisce i contenuti prodotti dagli utenti, verifica la presenza degli utenti online. Ha quindi una visione completa della rete. Poichè il client ha una struttura “thin”, tutte le strutture dati utilizzate per queste funzionalità sono gestite dal server stesso.

2. Funzionalità principali

a. Registrazione utenti

Quando un utente u apre l’applicazione per la prima volta, SocialClient(u) richiede il nome utente e la password, che vengono quindi spediti, tramite una connessione TCP, al SocialServer. Il SocialServer verifica se il nome dell’utente esiste già nel sistema.

- se non esiste, lo aggiunge alla lista di utenti e invia un messaggio di conferma.
- se esiste, invia un messaggio di errore

Alla fine della procedura di registrazione, la connessione viene chiusa.

b. Login

SocialClient chiede all’utente di inserire la password, e invia *username* e *password* al SocialServer, usando una connessione TCP. Il SocialServer verifica la correttezza e genera un nuovo UID come token di autenticazione (oAuth) che viene spedito al SocialClient che ha inviato i dati. Il SocialClient memorizza il token che sarà valido per 24 ore e che verrà inviato con ogni messaggio trasmesso da SocialClient al SocialServer.
Quando l’utente inizia una nuova azione, SocialClient verifica se il token è ancora valido, altrimenti effettua nuovamente la procedura di login. Alla fine della procedura, la connessione TCP viene chiusa.

c. Richiesta nuova amicizia

Un utente A inserisce il nome di un utente B e SocialClient(A) invia tale nome al SocialServer, usando una connessione TCP. SocialServer verifica se l’utente B è online, aprendo una connessione TCP. Se la connessione viene stabilita, SocialServer invia un messaggio a SocialClient(B), dopo aver memorizzato la richiesta di A in una sua lista interna ed aver confermato all’utente A che la sua richiesta di amicizia è stata inoltrata (il che non implica necessariamente che sia stata accettata)

La richiesta stessa potrà essere eventualmente confermata dall'utente B più tardi, quando questo sceglierà di usare la funzionalità "Conferma nuova amicizia". Se l'utente B non è online o non esiste nella lista di utenti, SocialServer invia un messaggio di errore all'utente A. Alla fin, le connessioni TCP vengono chiuse.

d. Conferma nuova amicizia

La componente SocialClient(u) mostra all'utente u la lista contenente tutte richieste di amicizia pervenute e non ancora riscontrate, ed u inserisce il nome di uno degli utenti (v) presenti nella lista, che vuole accettare come amico/per cui non intende accettare l'amicizia. SocialClient(u) invia su una connessione TCP al SocialServer il nome v e la scelta dell'utente. SocialServer verifica l'esistenza della richiesta iniziale ed eventualmente modifica la rete di amicizia di u e di v, rispondendo con un messaggio di conferma a u. Se la richiesta iniziale non è presente, SocialServer risponde con un messaggio di errore. La connessione viene chiusa alla fine della procedura. SocialServer elimina le richieste di amicizia non riscontrate dagli utenti dopo un intervallo di tempo il cui valore può essere impostato in fase di configurazione.

e. Keep-alive

SocialServer mantiene una lista di utenti online. Ad ogni nuova richiesta (di qualsiasi tipo) ricevuta, la lista viene aggiornata. Un cliente che si disconnette (logout), viene rimosso dalla lista. Inoltre, al fine di gestire le disconnessioni improvvise, ogni 10 secondi SocialServer invia un messaggio di keep-alive su un gruppo di multicast di cui fanno parte tutti gli utenti, e aspetta da loro una risposta. Gli utenti da cui non arriva nessun riscontro entro 10 secondi, vengono rimossi dalla lista di utenti online, quindi il SocialServer stampa il numero di utenti rimasti online. La componente SocialClient deve essere in grado di ricevere e rispondere ai messaggi di keep-alive.

f. Richiesta lista amici.

SocialClient(u) invia una richiesta TCP al SocialServer che risponde con la lista di nomi degli amici di u, riportando anche il loro stato attuale (online/offline).

g. Ricerca utenti

Il sistema offre la possibilità di ricercare un nome nella lista di utenti. L'utente inserisce una stringa, SocialClient la invia al SocialServer che risponde con una lista di nomi di utenti (possibilmente vuota) che contengono la stringa ricercata. La comunicazione avviene mediante TCP.

h. Pubblicazione contenuti

L'utente u inserisce un contenuto da pubblicare e SocialClient(u) lo invia su una connessione TCP al SocialServer, il quale mantiene, per ogni utente, una lista di utenti che hanno manifestato interesse per i suoi contenuti (i suoi follower). Ricevuto un contenuto nuovo dall'utente u, SocialServer lo invia a tutti i suoi follower usando la callback registrata da u su SocialServer (come dettagliato nella prossima sezione).

i. Registrazione interesse per contenuti di un utente

L'utente u inserisce il nome di un amico che è interessato a seguire, e SocialClient(u) notifica il suo interesse registrando su Social Server, una callback implementata mediante RMI. Tale callback sarà utilizzata per notificare a SocialClient(u) la pubblicazione di nuovi contenuti da parte del suo amico. Questo metodo callback inserisce il messaggio nella lista di messaggi ricevuti del SocialClient(u), che verranno visualizzati successivamente da u usando la funzionalità del punto **j**.

j. Visualizzazione contenuti

Tutti i contenuti ricevuti dal SocialServer vengono memorizzati dal SocialClient in una lista di messaggi e mostrati all'utente quando questo lo richiede. Una volta visualizzato, un messaggio viene cancellato dalla lista.

k. Logout

SocialClient(u) invia un messaggio TCP al server che cancella il token di autorizzazione per u. L'utente dovrà effettuare il login, nel caso in cui usi successivamente l'applicazione.

3. Interfaccia utente

Offre opzioni all'utente per eseguire le varie funzionalità:

- Fornire il nome utente e la password per il login e ricevuta del token di autorizzazione.
- Rinnovare il token.
- Richiedere nuova amicizia
- Confermare una nuova amicizia.
- Consultare la lista di amici.
- Ricercare utenti.
- Pubblicare contenuti.
- Registrare interesse per contenuti di un amico.
- Visualizzare i contenuti nuovi.
- Logout

La realizzazione della interfaccia può essere:

- testuale mediante riga di comando, nella soluzione base,
- (opzionale) grafica: quest'ultima soluzione risulta più avanzata e consente di gestire meglio le operazioni asincrone.

4. Modalità di svolgimento del Progetto

Il progetto deve essere svolto singolarmente. Il materiale consegnato deve comprendere:

- il codice dell'applicazione e di eventuali programmi utilizzati per il test delle sue funzionalità
- la relazione in formato pdf che deve contenere:
    - una descrizione generale dell'architettura del sistema. Motivare le scelte di progetto;
    - uno schema generale dei threads attivati, con particolare riferimento al controllo della concorrenza, e delle strutture dati utilizzate
    - per ogni componente e struttura dati utilizzate, una descrizione delle classi definite (preferibilmente usando anche UML)
- il codice eseguibile (2 archivi eseguibili .jar, uno per il client, l'altro per il server) con indicazioni precise sulle modalità di esecuzione.

L'organizzazione e la chiarezza della relazione, nonché l'organizzazione del codice e i commenti, influiranno sul voto finale. Relazione e codice sorgente devono essere consegnati sia in formato cartaceo, presso la portineria del Dipartimento, sia in formato elettronico, via e-mail al docente del corso. Gli eseguibili devono essere consegnati in formato elettronico.