Laboratorio di Reti, Corsi A e B

Social Gossip: un servizio di chat multifunzionale
Progetto di Fine Corso – A.A. 2017/18

1. Descrizione del problema

Scopo del progetto è la progettazione ed implementazione di Social Gossip, un servizio sviluppato secondo il modello client server che offra agli utenti registrati la possibilità di far parte di una rete sociale e di utilizzare diversi tipi di chat con altri utenti della stessa rete.

Il primo insieme di servizi offerti da Social Gossip contiene servizi relativi alla gestione della rete sociale, ad esempio la registrazione ed il login sulla rete, la ricerca di amici, la possibilità di stabilire una nuova amicizia e di ottenere la lista dei propri amici di settare il proprio stato di presenza in rete.

I servizi del secondo insieme consentono agli utenti di utilizzare, all'interno della rete sociale, diversi tipi di chat. In particolare, vengono offerti i seguenti servizi di chat:

- *chat amici* consente di selezionare e quindi chattare con un amico appartenente alla propria rete sociale. L'utente seleziona l'amico con cui intende chattare e quindi viene creato un canale di comunicazione che collega i due elementi della social network. Ad esempio, nella rete sociale in Fig. 1, l'utente C può chattare con uno dei suoi amici, F oppure G.
- *chat room* consente di chattare su un particolare tema, con utenti diversi, anche non appartenenti alla propria rete sociale, purché siano iscritti al servizio. Ad esempio, nella rete sociale in Fig. 1, l'utente B può aprire la chat room "Serie A", a cui partecipano sia l'utente D, che è un suo amico, sia l'utente A, che partecipa alla rete sociale, ma non è un suo amico.

Unicamente per la chat tra amici, viene offerto anche un servizio di traduzione che consente la comunicazione anche tra due utenti che non parlano la stessa lingua.
Inoltre è possibile inviare un file ad un utente del servizio. E' possibile inviare file solo ad amici.

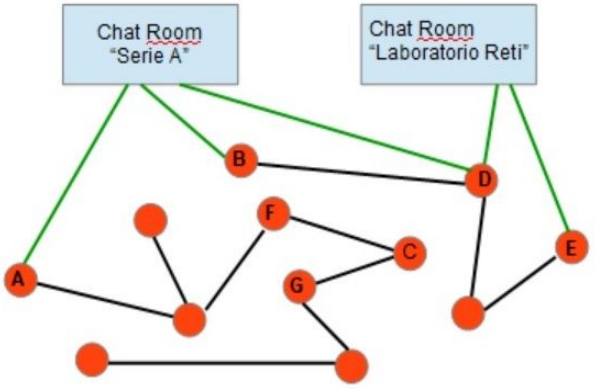

Fig 1. Social Gossip:

2. Funzionalità

2.1 Operazioni sulla Social Network

Le operazioni relative alla social network sono le seguenti:

- *register (<nickname><lingua>)*

Il client richiede che l'utente con un certo nickname venga registrato nella social network. Se esiste già un utente con quel nickname, il server comunica che non è possibile inserire l'utente. Il secondo parametro lingua può assumere i valori ISO 639 reperibili, ad esempio, nella seconda colonna della tabella presente all'indirizzo http://www.loc.gov/standards/iso639-2/php/code_list.php;

- *login (<nickname>)*

Il client richiede per l'utente la login al servizio. Se l'utente è registrato, viene loggato e il server comunica l'avvenuta identificazione, altrimenti comunica che l'utente non è registrato alla social network ed il login non viene effettuato. Questa operazione provoca anche un cambio di stato (da 'offline' a 'online') dell'utente. Ogni client viene notificato del cambio stato di un suo amico;

- *look-up (<nickname>)*

L'utente richiede di ricercare un altro utente. Il server informa il client circa il risultato della ricerca;

- *friendship (<nickname>)*

Mediante questa comando l'utente richiede che venga creata una relazione di amicizia con un utente specificato. A differenza di altri social network, in Social Gossip, il server non attende che l'utente accetti l'amicizia, ma inserisce un link nel grafo tra l'utente che richiede l'amicizia e quello specificato come argomento. Il server informa il client del risultato della operazione. Inoltre, ogni utente viene informato quando un altro utente crea una relazione di amicizia che lo coinvolge.

- *listfriend* ()

Restituisce una lista degli attuali amici dell'utente.

2.2 Gestione chat

Una chat-room può essere creata solo da un utente ed è definita univocamente da un nome, che ne identifica un tema, non possono esistere due chat-rooms con lo stesso identificativo. Il client ha a disposizione alcune operazioni per la gestione di una chat room.

- *create (<chatId>)* Crea una chat-room con identificativo chatId. ChatId è una stringa che rappresenta il titolo della chat, cioè l'argomento di discussione (ad esempio "Serie A"). Nel caso che una chat-room con chatId specificato esista già, il server comunica un errore all'utente.

- *addme(<chatId>)* Aggiunge alla chat-room specificata dal chatId l'utente che invoca l'operazione.

- *chatlist()* Restituisce una lista di tutte le chat-rooms, comprese quelle a cui l'utente è iscritto, con indicazione di quelle a cui l'utente è iscritto.

- *closechat()* Chiude una chat room. Tutti gli utenti che ne facevano parte vengono informati dell'evento.

2.3 Invio messaggi e file

Il servizio di chat fornisce all'utente i seguenti comandi, per inviare messaggi, rispettivamente, sulla chat amici e sulle chat room.

- *msg2friend(<nickname><messaggio>)* l'utente U indica il nickname dell'amico A con cui vuole chattare ed il messaggio da inviare. Vengono segnalati

eventuali errori se A non è amico di U, se A non esiste oppure se A non è al momento online. Il destinatario del messaggio riceverà dal server un messaggio in cui viene indicato il nickname dell'utente che ha inviato il messaggio, ed il messaggio tradotto nella lingua del destinatario.
- *chatroom-message (<idChat>-<messaggio>)* Questo comando consente alll'utente che lo invoca di chiedere che il messaggio specificato venga inviato a tutti gli utenti (amici e non) iscritti alla chatroom con identificativo idChat. Viene segnalato un errore se la chatroom non esiste o se nessun utente iscritto alla chat-room è online (ad esclusione dell'utente che invoca il comando). Gli iscritti alla chat-room riceveranno un messaggio con l'identificativo del mittente, l'identificativo della chat room ed il testo del messaggio.

Inoltre, viene definito il seguente comando per l'invio di un file:

- *file2friend(<nickname>)* indica la volontà di un utente di inviare un file ad un amico. Il file deve esistere nel filesystem accessibile all'utente mittente. Se l'utente non esiste oppure se l'utente non è al momento online, il server manda un messaggio di errore.

3. Implementazione

Il servizio di chat avviene sempre mediante il server, ovvero il server agisce sempre da intermediario tra i client interessati a chattare. Invece, l'eventuale invio di un file avviene invece in modo diretto tra i client. Il server memorizza il grafo che descrive le relazioni di amicizia tra gli utenti e lo stato degli utenti (online/offline). Il server è implementato seguendo l'architettura multithreaded: l'ultilizzo di NIO è permesso soltanto per il trasferimento dei file.

3.1 Protocolli di comunicazione

L'interazione tra i client ed il server e l'interazione diretta tra i client avviene utilizzando i seguenti protocolli:

- tutte le operazioni richieste dal client ed i rispettivi messaggi di risposta che riguardano la gestione della social network e la gestione delle chat-room vengono scambiati tra client e server mediante TCP. Si richiede di definire opportuni formati JSON sia per i messaggi di richiesta che di risposta;
- Al momento del login il client registra, tramite RMI, una callback. Il server utilizza tale callback per informare il client quando uno dei suoi amici cambia

stato (online/offline) oppure quando un altro utente ha creato una relazione di amicizia con il client;

- I messaggi della chat-amici, vengono inviati al server tramite una diversa connessione TCP ed il server li invia all'amico selezionato, sempre mediante una connessione TCP;
- Il server crea un gruppo di multicast per ogni chat-room. Ogni messaggio spedito in una chat room, viene spedito al server mediante il protocollo UDP, ed il server li spedisce a sua volta sul gruppo di multicast relativo a quella chat-room;
- Per l'invio dei file, su richiesta dell'utente mittente, il server contatta l'utente destinatario indicando di aprire un socket per la ricezione del file, e invia al mittente l'ip e la porta del destinatario. Il trasferimento del file avviene in modo peer-to-peer tra i due client utilizzando TCP e NIO, senza ulteriori operazioni da parte del server.

3.2 Struttura del client

Lo sviluppo di un servizio di chat richiede, in modo naturale, un'opportuna interfaccia grafica sul client. Tuttavia, è possibile anche utilizzare una interfaccia testuale: in questo caso, sono necessarie particolari interazioni dell'utente con il client per consentire la visualizzazione dei messaggi o di altri eventi in arrivo, come l'aver ricevuto una amicizia o il cambio di stato (online/offline) di un amico. Ad esempio, l'utente deve digitare un comando del tipo 'getIncomingNews' per poter visualizzare i messaggi che gli sono stati spediti o gli altri eventi in entrata.

3.3 Servizio di traduzione

Per quanto riguarda la traduzione dei messaggi, il server si può avvalere di alcune REST API messe a disposizione da alcuni providers. Si consiglia di utilizzare il servizio *MyMemoryTranslated. net* che offre un semplice servizio di traduzione free, La documentazione è reperibile all'indirizzo http://mymemory.translated.net/doc/spec.php.

L'interazione con il servizio REST di traduzione viene svolta dal server. Il server riceve la richiesta di traduzione, manda la richiesta REST al servizio di traduzione ed invia quindi il messaggio tradotto al richiedente.

4. Modalità di svolgimento del Progetto

Il progetto può essere svolto singolarmente o in coppia. Nel secondo caso, entrambe gli studenti sono responsabili di tutto il materiale consegnato. In particolare, tutto il codice deve essere capito e conosciuto a fondo da entrambe gli studenti. Ogni

studente dovrà comunque sostenere una prova orale individuale che riguarderà sia la discussione del progetto che tutto il materiale presentato a lezione.

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.

Alcuni fattori che influenzeranno il voto finale:

- gestione della concorrenza lato client e lato server
- modularità del codice e rispetto del paradigma ad oggetti
- gestione oculata delle eccezioni
- uso appropriato dei commenti
- pulizia e chiarezza del codice
- chiarezza della relazione

Relazione e codice sorgente devono essere consegnati sia in formato cartaceo, presso la portineria del Dipartimento, sia in formato elettronico, tramite Moodle. Gli eseguibili devono essere consegnati in formato elettronico.