Progetto di
Laboratorio di Programmazione di Rete
Laurea Triennale in Informatica Applicata
Anno Accademico 2009/2010

ConnectionBook:
Un sistema per la Gestione di Social Networks

1 Scopo del Progetto

Il progetto richiede la realizzazione di *ConnectionBook*, un servizio per la gestione di social networks contenente un insieme molto ridotto delle funzionalità di *Facebook*. In *ConnectionBook* ogni utente prima di usufruire del servizio deve effettuare *una registrazione* in cui fornisce al sistema il suo *indirizzo di posta elettronica*, il suo *nome*, alcuni dati personali (data di nascita, indirizzo,...) ed una *password*. Poichè possono esistere più utenti con lo stesso nome, l'indirizzo di posta elettronica è utilizzato per *identificare univocamente* ogni utente.

Al momento della registrazione il sistema registra i dati dell'utente in un proprio database dopo aver verificato che il suo indirizzo di posta non sia già presente. Dopo la registrazione, un utente può usufruire dei servizi offerti da *ConnectionBook* connettendosi al sistema (*login*) e disconnettendosi dal sistema (*logout*). Ogni utente è caratterizzato da uno *stato di presenza* sulla rete che indica, in ogni istante, se esso è *online*, cioè ha effettuato il login, oppure *offline*. Ogni utente può eseguire una query per ricercare un amico mediante il suo nome: il sistema gli restituisce una lista di utenti caratterizzati da quel nome. L'utente può scegliere un utente u dalla lista restituita ed inviargli una *richiesta di amicizia*. Se u è online nel momento in cui la richiesta di amicizia è inviata, la richiesta gli viene notificata immediatamente, altrimenti il sistema registra la richiesta e la notifica ad u al momento della sua successiva connessione. u deve quindi confermare la sua disponibilità ad accettare la richiesta di amicizia pervenuta. La richiesta di amicizia viene memorizzata nel sistema fino a che u la accetta oppure la rifiuta.

Ogni utente possiede una propria *bacheca* su cui può pubblicare delle note. Una nota è una insieme di linee di testo. La nota pubblicata da un utente u viene inserita nella bacheca di tutti i suoi amici. Se un utente a amico di u è online al momento della pubblicazione della nota, essa viene immediatamente visualizzate sulla bacheca di a, altrimenti il sistema registra la nota pubblicata da u e la notifica ad a al momento della sua successiva connessione.

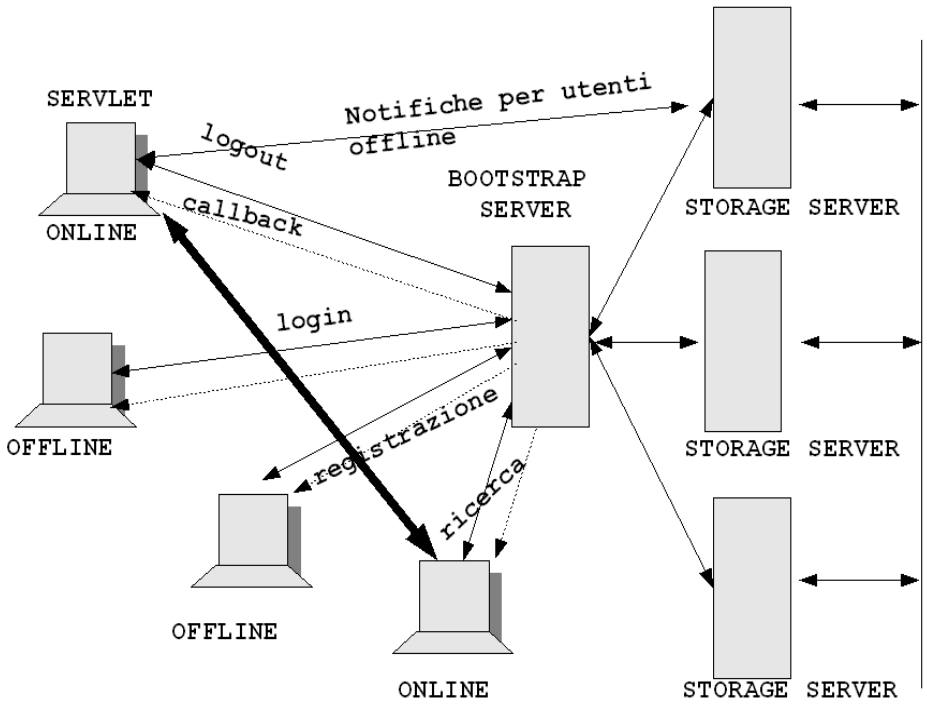

Figure 1: ConnectionBook: l'Architettura

Ogni utente on line conosce in ogni istante lo stato di presenza dei suoi amici. Nel caso in cui un utente modifichi il proprio stato, il sistema deve notificare tale cambiamento a tutti i suoi amici che in quel momento si trovano online.

2 Architettura del Sistema ed Implementazione

L'architettura di *ConnectionBook* sfrutta sia il modello client/server che quello peer to peer e comprende un *BootStrap Server*, un insieme di *Storage Servers* ed un insieme di *Servlets* associati agli utenti del sistema (in ambiente P2P un servlet è un'entità che usufruisce di servizi ed allo stesso tempo offre servizi). La Fig. 1 mostra uno schema generale dell'architettura

Gli *Storage Servers* memorizzano lo stato di ogni utente u offline, cioè la lista dei suoi amici, lo stato della sua bacheca al momento della sua ultima disconnessione, le note pubblicate dagli amici durante il periodo in cui u è offline. Quando un utente si connette a *ConnectionBook* riceve la sua bacheca

aggiornata con l'inserimento delle note pubblicate dai suoi amici durante il periodo in cui si trovava offline e le eventuali richieste di amicizia.

Il *BootStrap Server* fornisce il servizio di registrazione iniziale degli utenti, di ricerca di amici e registra lo stato (offline/online) di ogni utente. In particolare le funzioni del *BootStrap Server* sono le seguenti:

- *Registrazione*: controllo dell'unicità dell'indirizzo mail fornito dall'utente ed eventuale registrazione. La registrazione memorizza i dati dell'utente in un database ed assegna a quell'utente uno *Storage Server* per la gestione del suo stato.

- *Ricerca di Amici*: ricevuto da un servlet il nome di un amico da ricercare, ricerca e restituisce al servlet le informazioni relative a tutti gli utenti registrati con quel nome.

- *Gestione stato degli utenti*: ricevuta una richiesta di connessione o di disconnessione, registra il cambiamento di stato dell'utente.

Un servlet invia sia le richieste di amicizia che le note pubblicate dal proprio utente ai suoi amici, se questi si trovano on line al momento della richiesta/pubblicazione. In caso contrario, il servlet invia una notifica al proprio storage server che contatta lo storage server dell'utente interessato alla richiesta/pubblicazione e gli propaga la notifica. E' necessario che un utente riceva la sua bacheca aggiornata e le eventuali richieste di amicizia che gli sono pervenute al momento della sua connessione al sistema. Ogni servlet notifica inoltre al proprio Storage Server ogni nuova amicizia (amicizia confermata) in modo che esso possa mantenere sempre aggiornata la sua lista degli amici.

Occorre inoltre tenere presente i seguenti vincoli sull'implementazione del sistema:

- il servlet utilizza *RMI* per effettuare la registrazione al Bootstrap Server.

- al momento della connessione al sistema, il servlet registra una *callback*. Il Bootstrap Server utilizza la callback per notificare al servet il cambiamento di stato di uno degli amici dell'utente collegato mediante quel servlet.

- il protocollo TCP viene utilizzato per lo scambio dei messaggi tra servlet e tra servlet e Servers.

- il *multicast* viene utilizzato per lo scambio delle notifiche di richiesta amicizia/modifica bacheca tra storage servers

Si definisca inoltre una semplice politica di assegnazione degli utenti agli *Storage Servers* che tenda a bilanciare il carico e si strutturino sia i servlets che i servers mediante più threads di controllo utilizzando eventualmente, se necessario, tecniche di thread pooling.

Definire infine una semplice interfaccia grafica per il servlet. L'interfaccia deve consentire una semplice interazione dell'utente con il proprio servlet. La complessità dell'interfaccia non ha avrà alcuna influenza sulla valutazione finale del progetto.

3 Modalità di svolgimento del Progetto

Il progetto può essere svolto in gruppo. Ogni gruppo deve essere composto al massimo da *due studenti*. Il materiale consegnato deve comprendere:

- il codice di tutte le componenti di *ConnectionBook*, con le indicazioni per la loro compilazione/esecuzione.
- La stampa di tutto il codice di *ConnectionBook* e di eventuali programmi utilizzati per il test delle sue funzionalità. *Deve essere generato un file pdf contenente il codice di tutte le classi.*
- Una stampa *in formato pdf* di una relazione contenente
  - una descrizione generale delle scelte di progetto effettuate.
  - una descrizione delle strutture dati utilizzate
  - uno schema generale dei threads attivati dai servlet/servers.

L'organizzazione e la chiarezza dell'esposizione della relazione influiranno sul voto finale dell'esame. L'utilizzo di metodologie di documentazione del software quali diagrammi UML (delle classi, di sequenza,...) sarà considerato positivamente ai fini della valutazionde del progetto.
Il progetto deve essere consegnato una settimana prima della data dell'orale. L'orale verterà sia sulla discussione del progetto che sul programma svolto durante il corso. La prova del corretto funzionamento del programma verrà effettuta utilizzando la rete locale del laboratorio del Polo Marconi.
Per qualsiasi problema potete contattarmi via e-mail. Nel caso si questioni complesse possiamo fissare un appuntamento alla Spezia oppure una call via skype.