Progetto di L.P.R.-Corso A
Laurea Triennale in Informatica
Anno Accademico 2006/2007

GOSSIP: Un servizio per lo scambio istantaneo/differito di messaggi su WAN

1 Scopo del Progetto

Il progetto richiede la realizzazione di *GOSSIP*, un servizio di supporto per lo scambio di messaggi tra utenti collegati ad una rete *WAN*. Prima di usufruire del servizio, ogni utente deve effettuare *una registrazione* in cui fornisce al sistema il suo *nickname* e l'*indirizzo IP*. Il sistema verifica che il nickname non sia stato già scelto da altri utenti ed, in caso affermativo, registra i dati dell'utente in un proprio database. Successivamente, un utente può usufruire del servizio connettendosi al sistema (*login*) e quindi disconnettersi 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 definisce una propria *Lista di Contatti di Ingresso*, in cui inserisce i nicknames di tutti i mittenti dai quali è disposto a ricevere messaggi ed una lista di *Lista di Contatti di Uscita* che contiene i nicknames di tutti gli utenti a cui intende inviare i messaggi. Prima di inserire un nuovo contatto C nelle proprie liste, l' utente invia il nickname di C a *GOSSIP*, che verifica l'esistenza di un utente con quel nickname. Un utente U_1 può inserire un utente U_2 nella lista dei Contatti di Uscita solo se U_2 lo ha preventivamente inserito nella propria Lista di Contatti di Ingresso.

Ogni utente in linea conosce lo stato di presenza degli utenti presenti nella sua *Lista di Contatti di Uscita*. Nel caso in cui un utente U modifichi il proprio stato, il sistema deve notificare tale cambiamento a tutti gli utenti interessati.

Dopo essersi connesso al sistema, un utente U_1 può inviare messaggi ad un qualsiasi altro utente U_2 contenuto nella sua *Lista dei Contatti di Uscita*. Nel caso in cui il destinatario del messaggio sia *online* il messaggio gli viene recapitato *in tempo reale*, altrimenti il sistema memorizza il messaggio e lo recapita al destinatario al momento della sua successiva connessione al sistema.

GOSSIP realizza quindi un servizio di *Instant Messaging* per gli utenti in linea ed un servizio simile alla *Posta Elettronica* per gli utenti fuori linea.

2 Architettura del Sistema

Il progetto richiede la definizione di un'architettura che implementi i servizi offerti da *GOSSIP* sfruttando sia il modello client/server che quello P2P.

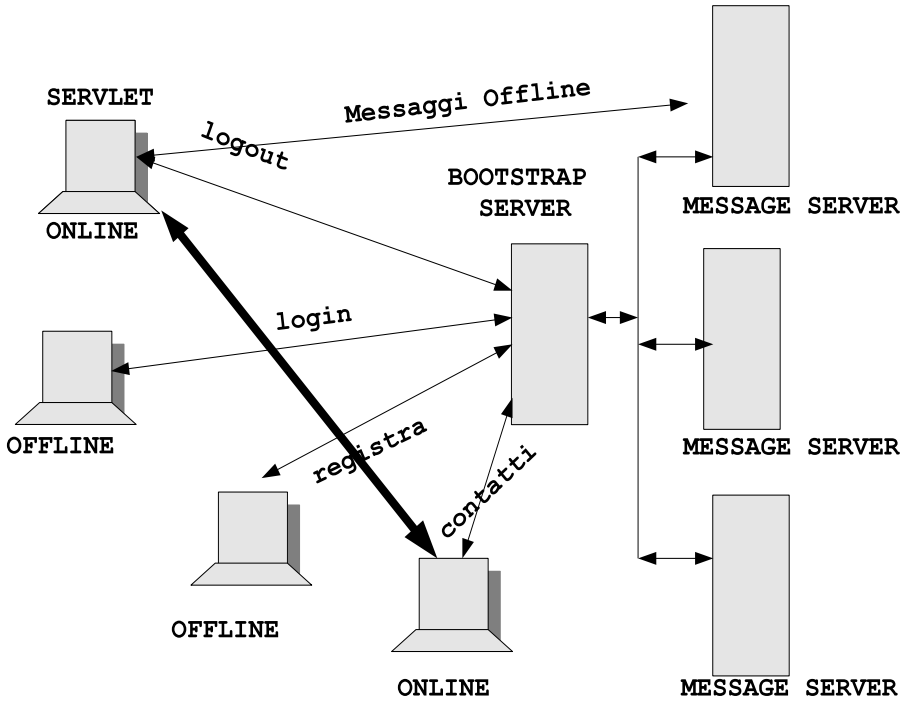

Figure 1: GOSSIP: l'Architettura

L'architettura comprende un *BootStrap Server*, un insieme di *Message Servers* ed un insieme di *Servlets* associati agli utenti del sistema (in ambito 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

Il *BootStrap Server* fornisce il servizio di registrazione iniziale degli utenti, di connessione e di sconnessione dal sistema. In particolare le funzioni del *BootStrap Server* sono le seguenti:

- *Registrazione*: controllo dell'unicità del nickname fornito dal client ed eventuale registrazione

- *Gestione Liste dei Contatti.* Ogni utente notifica le sue Liste dei Contatti al *BootStrap Server* che verifica l'esistenza dei nicknames dei partners e, per le *Liste dei Contatti di Uscita*, la disponibilità dei partners a comunicare.

- *Connessione* : il server riceve le richieste di connessione degli utenti. Dopo avere verificato l'identità dell'utente U, sceglie un *Message Server* per la gestione dei messaggi offline ricevuti/inviati dall'utente. E' neces-

sario definire una semplice politica di assegnazione degli utenti ai *Message Servers* che tenda a bilanciare il carico.

Ogni *Message Server* gestisce un insieme di utenti e svolge le seguenti funzionalità:

- nel momento in cui riceve dal *BootStrap Server* la richiesta di gestire un nuovo utente U, gli trasmette tutti i messaggi che gli hanno eventualmente inviato gli altri utenti nel periodo di tempo in cui U è rimasto offline. Per svolgere questa funzione il server può interagire con gli altri *Message Servers* e/o con il *BootStrap Server*.
- durante il periodo di tempo in cui U rimane *on line*, riceve da U tutti i messaggi destinati ad utenti fuori linea e provvede alla loro memorizzazione, eventualmente interagendo con altri servers del sistema.

Il *servlet* interagisce con l'utente mediante una semplice interfaccia grafica (a questo proposito si possono utilizzare le classi introdotte a lezione), con il *BootStrap Server*, con il proprio *Message Server* e con gli altri *Servlets*. L'utente può inviare messaggi ad un qualsiasi altro utente registrato. GOSSIP provvede ad inviare direttamente il messaggio al destinatario, se questo é online, oppure al proprio *Message Server* in caso sontrario.

3 Implementazione dal Sistema

La soluzione implementata deve soddisfare inoltre i seguenti vincoli:

- utilizzare almeno una volta *il protocollo UDP*, *il protocollo TCP* ed *RMI*.
- strutturare sia i servlets che i servers mediante più threads di controllo, eventualmente utilizzando tecniche di thread pooling, se necessario.
- OPZIONALE: è possibile utilizzare sockets sicuri per implementare comunicazioni ritenute critiche.

4 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:

- La stampa di tutto il codice dell'applicazione e di eventuale programmi utilizzati per il test delle funzionalità dell'applicazione stessa.
- Una stampa della relazione *in formato pdf* che descriva tutte le scelte effettuate. La relazione deve contenere

- una descrizione generale dell'architettura del sistema. Motivare le scelte di progetto.
- uno schema generale dei threads attivati da ogni *Servlet*, dal *Bootstrap Server* e da ogni *Message Servers* e delle strutture dati utilizzate.
- una descrizione delle classi definite

L'organizzazione e la chiarezza 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.

Relazione e codice devono essere consegnati sia in formato cartaceo, presso la portineria del Dipartimento, sia in formato elettronico, via e-mail ad entrambe i docenti del corso.

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. *Al momento della prova orale sarà richiesto di effettuare una piccola modifica al progetto e di eseguire il progetto modificato in laboratorio.* La prova del corretto funzionamento del programma verrà effettuta utilizzando la rete locale del laboratorio del Polo Didattico Fibonacci. E' necessario verificare la correttezza del codice in tale ambiente *prima di sostenere l'orale*, in modo che la prova della correttezza del progetto possa essere svolta in breve tempo presso il laboratorio.