Progetto di L.P.R.-Corso A
Laurea Triennale in Informatica
Anno Accademico 2007/2008

Mini-Gnutella:
Un Sistema P2P per lo Scambio di File

1 Scopo del Progetto

Il progetto richiede la realizzazione di *Mini-Gnutella*, un sistema per lo scambio di file su WAN, ispirato alla rete P2P *Gnutella*.

In *Gnutella* ogni host riveste sia funzionalità di *server*, in quanto condivide un insieme di file e serve le richieste di download provenienti dagli altri hosts, che di *client* in quanto può a sua volta ricercare file memorizzati nella rete ed effettuarne il download. Per indicare il duplice ruolo rivestito da un host sulla rete, in seguito il generico host verrà riferito come *servlet*.

Mini-Gnutella definisce un sottoinsieme delle funzionalità di *Gnutella*.

Mini-Gnutella utilizza una *overlay network* definita da un insieme di connessioni che ogni servlet stabilisce inizialmente in fase di bootstrap e che vengono successivamente modificate dinamicamente. Infatti, poichè ogni servlet può decidere autonomamente di disconnettersi dalla rete e nuovi servlet possono connettersi alla rete, la struttura della overlay network puó essere dinamicamente modificata.

Le principali funzionalità di un servlet *Mini-Gnutella* sono le seguenti:

- *Bootstrap.* Un servlet si unisce a una rete *Mini-Gnutella* individuando un insieme di servlet presenti sulla rete, ed apre una connessione con ognuno di essi.

- *Condivisione di Files.* L'utente può indicare quali file intende condividere ed associare ad ogni file un insieme di *parole chiave* che permettono di individuarlo.

- *Ricerca di Files.* L'utente può sottomette una *query* per ricercare un file F specificando un insieme di *parole chiave*. Il servlet propaga la query agli altri servlet utilizzando l'overlay network. Una volta individuato un insieme di servlet S che posseggono F, il download avviene mediante una *connessione diretta* con uno dei servlet in S.

- *Esplorazione della Rete. Mini-Gnutella* utilizza un meccanismo di *ping-pong* per esplorare periodicamente la rete e scoprire gli host che si sono disconnessi e quelli che sono entrati a far parte dell'overlay. Quindi la struttura della overlay network *può cambiare dinamicamente* sulla base delle informazioni acquisite in questa fase.

2 Mini-Gnutella: Specifica del Protocollo

Ogni servlet conosce l'indirizzo di uno o più *Bootstrap Server*, sempre attivi sulla rete, il cui unico scopo è quello di memorizzare riferimenti ad un insieme di servlet 'affidabili' presenti sulla rete *Mini-Gnutella*. Un servlet viene definito 'affidabile' se il suo tempo di permanenza sulla rete supera una soglia SP predefinita. Ogni servlet è dotato inoltre di *una cache* in cui memorizza riferimenti ad altri servlet. La cache è inizialmente vuota e viene dinamicamente aggiornata man mano che il servlet acquisisce conoscenza della rete. Il contenuto della cache viene salvato al momento della disconnessione del servlet dalla rete e può essere ripristinato ed utilizzato quando il servet si riconnette. Al momento della sua connessione alla rete *Mini-Gnutella*, un servlet S può utilizzare la cache per individuare alcuni servlet verso cui aprire una connessione. Nel caso in cui la cache sia vuota o nessuno di tali servlet sia attivo, S contatta un bootstrap server. Dopo essere rimasto attivo per più di SP secondi, S notifica la sua presenza ai bootstrap server. Un servlet può inoltre registrarsi presso uno o più bootstrap server per richiedere la notifica, da parte di essi, dei riferimenti a nuovi servlet che si connettono dinamicamente all'overlay *Mini-Gnutella*. Il Boorstrap server può quindi notificare *in modo asincrono* la presenza di nuovi servlet ai servlet registrati.

Ogni servlet può definire al massimo MC *connessioni* con servlet vicini. Il valore di MC viene stabilito staticamente in base alle caratteristiche del servlet (banda, potenza computazionale,...). In fase di bootstrap il servlet cerca di definire connessioni con almeno BC servlet, con $BC < MC$. Le rimanenti $MC - BC$ connessioni possono venir definiti dinamicamente durante la permanenza del servlet sulla rete.

Il protocollo definito da *Mini-Gnutella* definisce i messaggi di *Query* e di *Query Hit*, utilizzati per la ricerca di file ed i messaggi di *Ping* e di *Pong* utilizzati per l'esplorazione della rete. Ad ogni messaggio di *Query*, rs. *Ping* corrispondono uno o più messaggi di *Query Hit*, rs. *Pong*. Ad ogni messaggio spedito sull'overlay viene associato un *identificatore* generato dal servlet mittente che deve identificare univocamente il messaggio all'interno della rete.

I messaggi di *Query* e di *Ping* vengono propagati sulla overlay network utilizzando un meccanismo di *flooding limitato* che utilizza un TTL *(Time to Live)* per limitarne la diffusione sulla rete. I messaggi di *Query Hit* e di *Pong* vengono instradati utilizzando un meccanismo di *Backward Routing*, cioé seguono a ritroso lo stesso percorso del rispettivo messaggio di *Query*, rs. *Ping*. Gli identificatori unici vengono utilizzati per implementare il backward routing e per evitare di spedire più volte sull'overlay lo stesso messaggio.

Un messaggio di *Query* viene spedito dal servlet S che intende ricercare un file ai suoi vicini nella overlay network e contiene un insieme di parole chiave che individuano il file. I vicini a loro volta propagano il messaggio ai propri vicini, decrementando il valore del TTL, fino a che tale valore diventa 0. Quando un servlet R riceve il messaggio controlla se possiede il file ricercato ed, in questo

caso, invia un messaggio di *Query Hit* che contiene il suo *indirizzo IP* e la *porta* ed eventuali altre informazioni che possono essere utilizzate da S per scegliere il servlet da cui effettuare il download. Dopo aver ricevuto un numero sufficiente di messaggi di *Query Hit*, S sceglie un servlet da cui effettuare il download, apre una *connessione diretta* con tale servlet e scarica il file. Nel caso in cui R si trovi a monte di un NAT o di un $firewall$ e non possa accettare una richiesta di connessione da S, R stesso apre una connessione verso S.

Un servet S invia inoltre periodicamente dei messaggi di *Ping* per esplorare lo stato della rete. Tali messaggi vengono inviati sulla overlay network ed utilizzano il meccansimo di flooding limitato. Ogni messaggio di *Pong* contine *indirizzo IP* e *porta* del servlet che l'ha prodotto. S memorizza nella propria cache i riferimenti ai nuovi host individuati mediante il meccanismo di ping pong. Il meccanismo di ping-pong viene utilizzato anche per controllare se alcuni vicini si sono disconnessi dalla rete. In questo caso S può stabilire connessioni con altri servlet memorizzati nella propria propria cache.

3 Implementazione dal Sistema

La soluzione implementata deve soddisfare i seguenti vincoli:

- utilizzare il *protocollo RMI* e le callback RMI per l'interazione tra servlet e Bootstrap Server.
- utilizzare il protocollo TCP per implementare le connessioni che definiscono la overlay network. I messaggi di *Query* e di *Query Hit* vengono inviati su queste connessioni.
- utilizzare il protocollo UDP per l'invio dei messaggi di esplorazione della rete. I messaggi di *Ping* possono essere spediti da un servlet S solo a quei servlet con cui ha aperto una connessione TCP.
- *Gnutella* utilizza il protocollo $HTTP$ per il trasferimento diretto di file tra servlet. *Mini-Gnutella* deve definire un proprio protocollo basato su TCP per implementare la connessione diretta tra i servlet per il download dei file. In alternativa, è possibile utilizzare anche in *Mini-Gnutella* il protocollo $HTTP$ per il trasferimento diretto di file, estendendo il semplice server $HTTP$ presentato a lezione.

E' possibile verificare la corretta implementazione del protocollo *Mini-Gnutella* anche su reti di dimensione limitata. A titolo di esempio, la figura 1 mostra la topologia di un overlay *Mini-Gnutella*

Per la definizione degli identificatori unici associati ai messaggi é possibile utilizzare la classe JAVA *Message Digest*, la cui documentazione puó essere reperita alla seguente URL

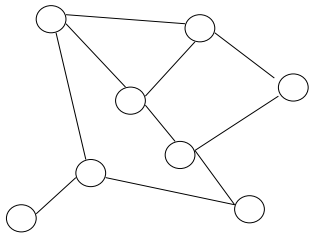

Figure 1: Un'Overlay Mini-Gnutella

`http://java.sun.com/javase/6/docs/api/java/security/MessageDigest.html`

Le seguenti funzionalità sono *opzionali*. L'implementazione di tali funzionalità inciderà ovviamente sulla valutazione finale del progetto.

- realizzare un supporto per query complesse. A questo scopo è possibile utilizzare un package JAVA che consente di manipolare espressioni regolari. La documentazione può essere reperita alla seguente URL:

  `http://java.sun.com/docs/books/tutorial/essential/regex/index.html`

- la specifica del protocollo di *Gnutella* è definita alla seguente *URL*

  `http://rfc-gnutella.sourceforge.net/developer/index.html`

  E' possibile realizzare qualsiasi funzionalità definita in tale documento e non compresa nella specifica di *Mini-Gnutella*

Occorre inoltre prevedere un insieme di meccanismi che consentano di monitorare in modo diretto il funzionamento del servlet *Mini-Gnutella*. Ad esempio, definire una semplice interfaccia che consenta all'utente di monitorare le connessioni stabilite dal proprio servlet, di visualizzare lo stato della sua cache, lo stato di download dei file, una maschera per effettuare le query, etc..

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 dello strumento e di eventuale programmi utilizzati per il test delle funzionalità dello strumento.
- Una stampa della relazione *in formato pdf* che descriva tutte le scelte effettuate. La relazione deve contenere

- una descrizione generale della architettura del sistema e della scelte di progetto effettuate.
- una descrizione delle strutture dati utilizzate
- uno schema generale dei threads attivati da ogni *servlet* e dai *Bootstrap Server* e delle modalità di interazione di tali threads
- un manuale d'uso che indichi chiaramente le modalità di interazione con il servlet *Mini-Gnutella*.

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 valutazione del progetto. Viceversa, non sarà ritenuta importante ai fini della valutazione del progetto la progettazione di un'interfaccia grafica complessa.

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

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 Didattico Fibonacci.