

**Valutazione di Tecniche di Ricerca e di Replicazione
in reti P2P *non strutturate***

Antonio Quartulli - Matr. 304338

Università degli studi di Pisa
Corso di PeerToPeer - CdL in Informatica
A.A. 2007/2008
Prof.ssa Laura Ricci

Indice:

- 1 Introduzione
- 2 La simulazione
 - 2.1 L'ambiente
 - 2.2 Le topologie di rete
 - 2.3 I metodi di ricerca
 - 2.4 La distribuzione dei dati
 - 2.5 Le strategie di replicazione
 - 2.6 Il rilevamento dei dati statistici
- 3 I risultati delle misurazioni
 - 3.1 Random Walk
 - 3.2 Random Walk (fixed number: 32)
 - 3.3 Random Walk Degree Biased
 - 3.4 Expanding Ring
- 4 Il codice
 - 4.1 Descrizione generale
 - 4.2 Protocollo Application: Overlay
 - 4.3 Protocollo Linkable: OverlayLink
 - 4.4 I controllori
 - 4.4.1 InitDisk
 - 4.4.2 DegreeInitializer
 - 4.4.3 QueryControl
 - 4.4.4 Stat

1 Introduzione

Questa relazione si prefigge l'obiettivo di descrivere in modo esaustivo il progetto conclusivo del corso di "PeerToPeer", esponendo l'ambiente di test utilizzato, il codice implementato, le topologie di rete ricreate ed i risultati ottenuti. Quest'ultimi verranno esposti mediante l'utilizzo di alcuni grafici che renderanno maggiormente confrontabili i risultati ottenuti con le varie configurazioni utilizzate.

Il progetto consiste nell'applicazione di alcune modifiche all'algoritmo di ricerca utilizzato da Gnutella 0.4, il flooding limitato, che come noto è poco scalabile.

Le modifiche applicate, come descritto nelle specifiche del progetto, sono tratte da due pubblicazioni, "Search and Replication in Unstructured Peer-To-Peer Networks"[1] e "Making Gnutella-like P2P Systems Scalable"[2], in cui viene studiato il protocollo Gnutella e vengono presentate alcune alternative con il fine di renderlo maggiormente scalabile.

2 La simulazione

2.1 L'ambiente

La simulazione è avvenuta utilizzando l'ambiente PeerSim (<http://peersim.sourceforge.net>), un simulatore OpenSource, sviluppato in Java presso l'Università di Bologna.

PeerSim offre la possibilità di simulare una rete con un elevato numero di nodi (fino a $\sim 10^6$) su un singolo host, rendendo così possibile, anche a sviluppatori che non posseggono una particolare infrastruttura, testare i vari protocolli PeerToPeer.

PeerSim offre inoltre la possibilità di eseguire i test in due diverse modalità: CycleDriven ed EventDriven. Il progetto è stato implementato affinché i test venissero eseguiti in modalità EventDriven. (Per maggiori dettagli a riguardo, rimando alla documentazione di PeerSim).

2.2 Le topologie

Le simulazioni utili al progetto sono state effettuate su due particolari topologie di rete:

- Random Graph Network
- Free Scale Network

La prima consiste in un grafo random, creata grazie alla classe *WireKOut* fornita dall'ambiente PeerSim. Ogni nodo ha mediamente un grado $k = 20$.

La seconda consiste in una Free Scale Network, ovvero una rete in cui il grado dei nodi segue una legge di tipo power-law. Questa topologia, in particolare, viene costruita seguendo il modello Barabasi-Albert che produce una distribuzione di probabilità dei gradi di tipo power-law con $\gamma = 3$:

$$P(k) = k^{-3}$$

Anche in questo caso la rete viene generata grazie all'utilizzo di una classe fornita da PeerSim: *WireScaleFreeBA*.

2.3 I metodi di ricerca

Le tecniche alternative di ricerca suggerite da [1] e [2] per favorire un aumento di scalabilità del protocollo Gnutella 0.4, di cui il progetto si occupa di studiarne l'applicazione, sono:

- Expanding Ring
- Random Walk
- Degree Biased Random Walk

Expanding ring:

Consiste in un flooding limitato con TTL dinamico.

E' prevista una "sessione" composta da una sequenza di ricerche basate su flooding limitato, con TTL variante secondo una successione aritmetica di ragione fissata per ogni ricerca. La sessione termina con il successo della ricerca od al raggiungimento di un valore massimo (prefissato) del TTL.

Random Walk:

Fissato un valore $k > 0$, vengono generati k messaggi ed instradati verso diversi vicini scelti a caso fra i possibili. Il messaggio viene spesso chiamato *walker*, da qui il nome *Random Walk*. Ogni nodo instraderà il walker ricevuto verso un altro vicino scelto nuovamente a caso.

La sessione di ricerca può terminare o all'esaurimento del TTL del Walker (scelto adeguatamente alto) oppure al successo della query. Quest'ultimo caso prevede un overhead in ogni hop, in quanto prima del forwarding del walker, l'hop dovrà comunicare con il nodo che ha generato la query per avere informazioni sullo stato della sessione di ricerca. In questo progetto è stata implementata la prima strategia (esaurimento TTL).

L'implementazione della tecnica Random Walk in questo progetto, prevede anche che i nodi siano statefull; in questo modo, se due walker di una stessa sessione di ricerca dovessero passare dal medesimo nodo, è garantito l'instradamento verso vicini differenti (suggerimento tratto da [1]).

Degree Biased Random Walk:

Variante della tecnica *Random Walk*, secondo la quale i vicini non vengono scelti casualmente, ma bensì selezionati in ordine decrescente di grado. In questo modo l'instradamento avviene privilegiando i nodi di grado più alto.

2.4 La distribuzione dei dati

Per poter studiare al meglio l'efficienza delle tecniche di ricerca sopra descritte, è necessario che la distribuzione dei dati ricercati vari in modo ben determinato in modo da studiarne le conseguenze.

Per far ciò, i dati vengono piazzati su *un numero preciso* di nodi scelti casualmente nella rete; in questo modo sarà possibile associare ad ogni esperimento un "fattore di replicazione" dei dati. Questo valore si rivelerà utile al fine di estrapolare dati statistici dagli esperimenti eseguiti.

2.5 Le strategie di replicazione

Per strategia di replicazione intendiamo la tecnica adottata da un protocollo per replicare i contenuti presenti sulla rete su diversi nodi.

Un network PeerToPeer può generalmente adottare diverse strategie di replicazione. La scelta viene effettuata in base all'ambiente in cui dovrà operare la rete ed in base alle varie caratteristiche che dovrà avere.

In questo progetto vengono esaminate due strategie: *owner-replication* e *path-replication*.

L'*owner-replication* è quella classicamente utilizzata dal maggior numero di protocolli di file-sharing: l'oggetto ricercato viene copiato solo sul nodo che ha generato la query.

La path-replication è invece utilizzata su data-storage distribuiti, come per esempio FreeNet; la caratteristica di questa strategia consiste nel replicare l'oggetto ricercato su tutti i nodi presenti nel cammino (path) fra il nodo che ha generato la query e quello ha risposto con successo. Questa strategia fa sì che gli oggetti presenti nella rete si distribuiscano su un elevato numero di hop, riducendo così i tempi di latenza delle ricerche successive ma, ovviamente, aumentando lo spazio di memorizzazione richiesto per ogni nodo.

2.6 Il rilevamento dei dati statistici

Tutti i test atti al rilevamento dei dati statistici sono stati effettuati secondo particolari criteri e con alcuni parametri prefissati, in particolare è utile menzionare:

- numero di nodi della rete: 10000;
- numero di ripetizioni di ogni test: 20;
- numero di query generate in un test: 100 (1% dei nodi);
- scelta del nodo generatore della query: casuale per ogni query;
- TTL massimo per Random Walk: 25;
- numero di walker nei test a valore fisso: 32 (valore ottimale tratto da [1]);

Tutti i test sono stati effettuati sia su topologia Random Graph, con grado medio dei nodi pari a 20, sia su topologia Free Scale (modello BA) in modo da confrontarne parallelamente i risultati.

3 I risultati delle misurazioni

Dai test effettuati sono stati realizzati i seguenti grafici per mettere in evidenza le differenze fra le varie tecniche di ricerca.

Escluso il primo caso (Random Walk), in cui sono stati messi a confronto dati provenienti da overlay configurate solo con *owner-replication*, tutti gli altri grafici mettono a confronto le 4 possibili combinazioni di *replication-type* e *topologia della rete*.

3.1 Random Walk

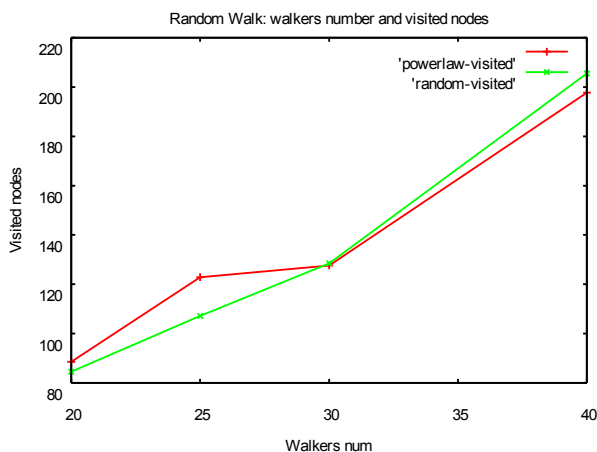


Figura 1

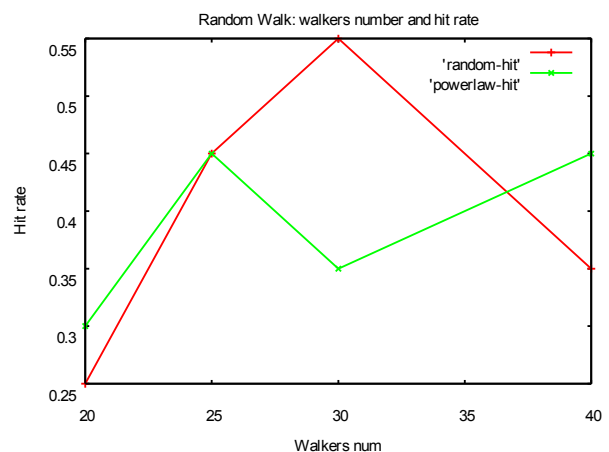


Figura 2

Secondo le statistiche ottenute dallo studio della tecnica di ricerca *Random Walk*, si è dedotto che il

numero ottimale di Walker, in una rete con topologia RandomGraph, è intorno a 30 (Figura 2). Secondo altre fonti, che hanno studiato questa tecnica in maniera maggiormente approfondita, è stato riscontrato un valore ottimale di 32 walker. Per questo motivo tutti i successivi test di questa tecnica sono stati eseguiti con un numero fisso di walker pari a 32. Per quanto riguarda una rete con topologia *Powerlaw*, i dati ottenuti non riscontrano un vero e proprio valore ottimale, ma bensì abbiamo due valori pari a 25 e 40 walker in cui si è registrata la più alta probabilità di successo.

3.2 Random Walk (fixed number: 32)

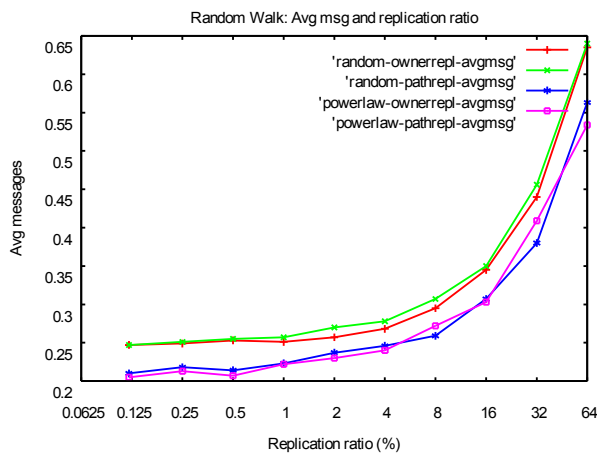


Figura 3

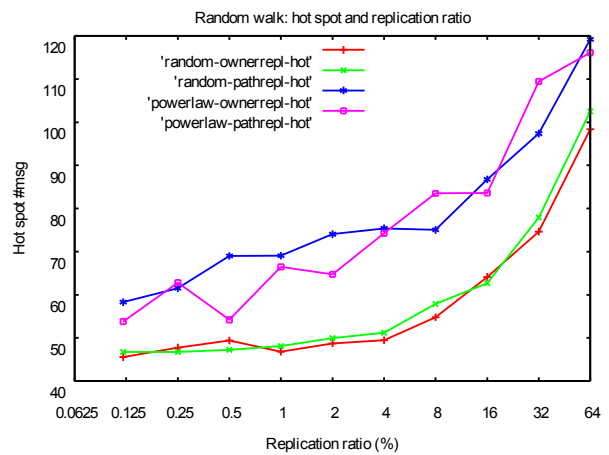


Figura 4

Analizzando le figure 5 e 6 possiamo notare come, utilizzando la tecnica di ricerca *RandomWalk*, con una replicazione superiore al 4% abbiamo un altissima probabilità di successo ed un numero medio di hop, necessari a raggiungere il dato, pari a 2. Inoltre quest'ultimo valore tende a decrescere maggiormente all'aumentare della replicazione.

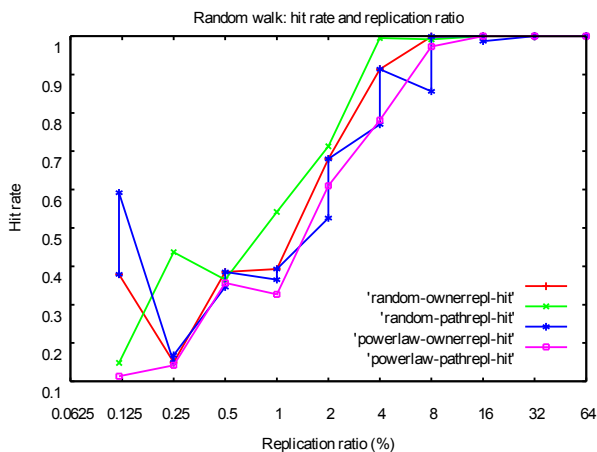


Figura 5

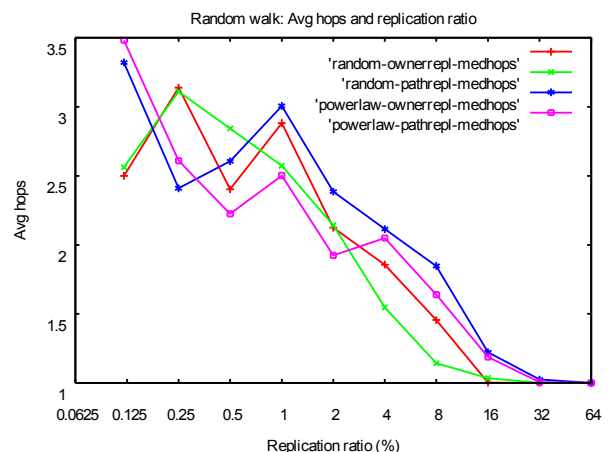


Figura 6

3.3 Random Walk, degree biased

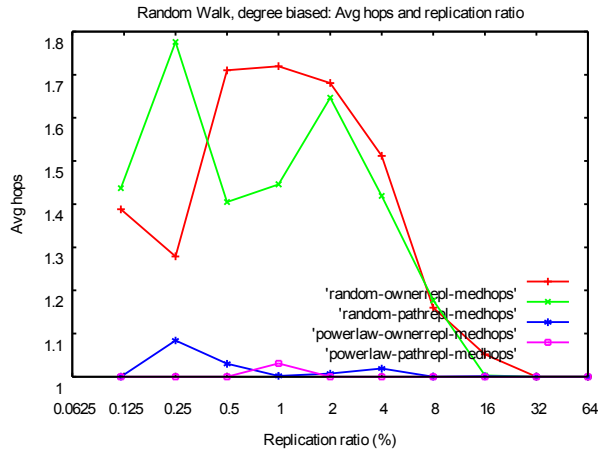


Figura 7

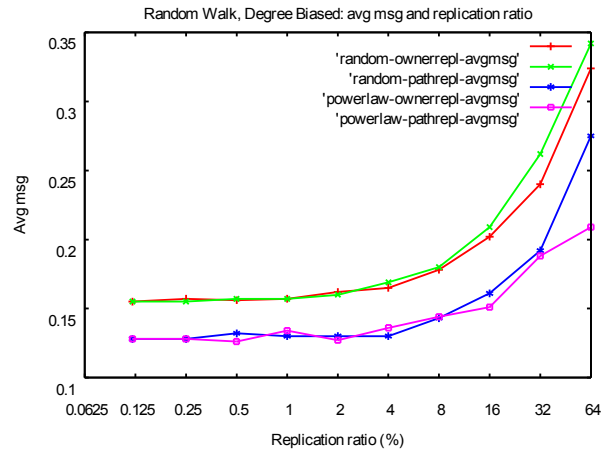


Figura 8

Utilizzando una variante della tecnica *RandomWalk*, ovvero la *Random Walk Degree Biased*, è possibile osservare che le overlay network con topologia *powerlaw* godono di eccellenti proprietà: numero di hop medi necessari ad una ricerca per andare a buon fine, sempre contenuti nell'intervallo [1; 1,1] e con un numero medio di messaggi per nodo sempre inferiore alla topologia RandomGraph, per qualsiasi fattore di replicazione del dato cercato (Figura 7, 8).

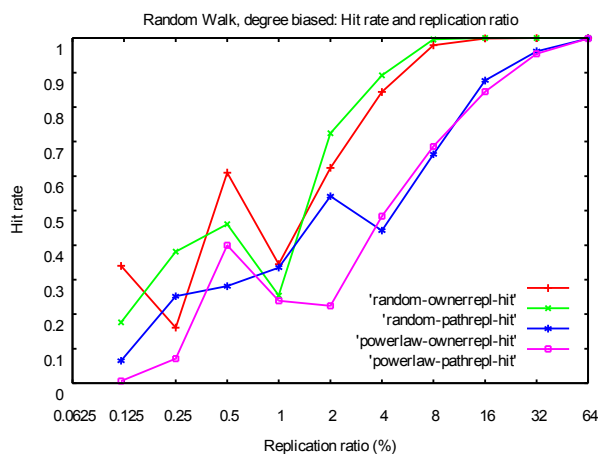


Figura 9

3.4 Expanding Ring

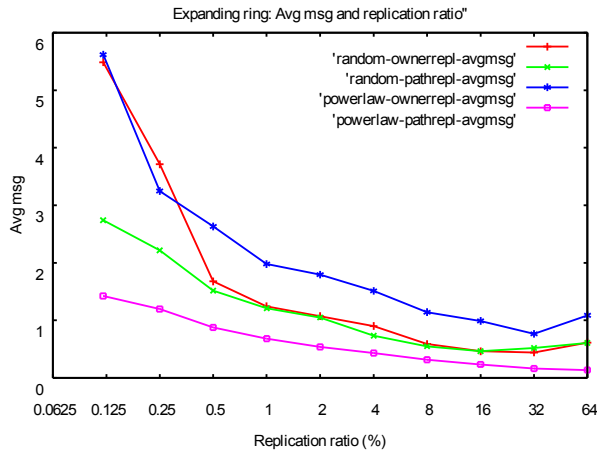


Figura 10

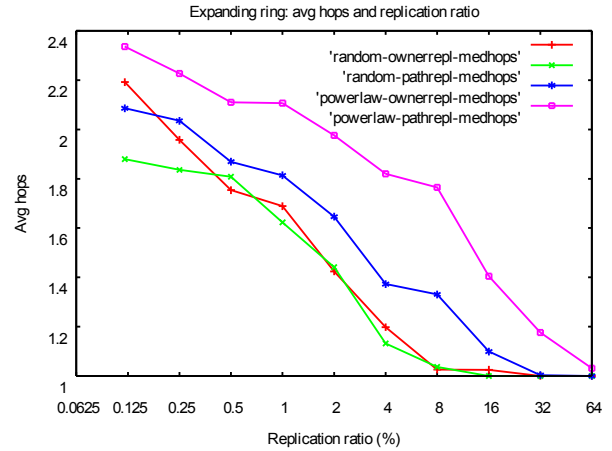


Figura 11

L'Expanding Ring e' una tecnica che si dimostra efficace nel caso di alta replicazione del dato da cercare, ma dimostra una notevole inferiorità rispetto al *RandomWalk*, in caso di bassa replicazione. Anche nel caso migliore pero', il risultato del confronto con la tecnica precedente risulta essere a favore di questa. La conclusione che si può trarre è quindi che, paragonandola con le altre, l'*ExpandingRing* non risulta essere una tecnica abbastanza efficace.

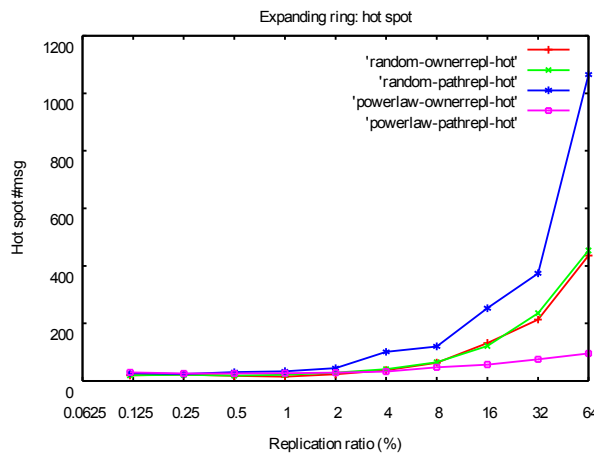


Figura 12

4 Il codice

4.1 Descrizione generale

Per la realizzazione di questo progetto sono stati implementati due "protocolli PeerSim", uno del livello Application ed uno del livello Linkable. I rispettivi nomi delle due classi, che verranno maggiormente approfondite nei prossimi paragrafi, sono *Overlay* ed *OverlayLink*.

4.2 Protocollo Application: Overlay

La classe Overlay è il vero e proprio cuore del progetto. E' l'implementazione del protocollo P2P (appunto a livello applicazione) che regola le comunicazioni sull'overlay network.

Nel file di configurazione, utilizzato per comunicare a PeerSim i vari parametri, e' possibile, tramite diverse opzioni, scegliere quale tecnica di ricerca utilizzare nella rete.

Le opzioni della classe Overlay, selezionabili nel file di configurazione, sono:

- *ttl X*: indica il valore del TTL con cui i messaggi vengono inizializzati. Valore di default 7;
- *expanding_ring*: abilita l'utilizzo della tecnica *ExpandingRing*; in questo caso il valore di default del TTL diventa automaticamente 1. Le sotto-opzioni sono:
 - *expanding_ring.ttl_end X*: indica il valore massimo che il TTL potrà assumere durante le sessioni di ricerca. Valore di default 10;
 - *expanding_ring.ttl_step X*: indica di quanto dovrà aumentare il valore del TTL ad ogni round di ricerca. Valore di default 2;
- *random_walk*: abilita l'utilizzo della tecnica *RandomWalk*. Le sotto-opzioni sono:
 - *random_walk.k X*: indica il numero di walker da istanziare ad ogni ricerca. Valore di default 1;
 - *random_walk.degree_biased*: abilita la variante *Random Walk Degree Biased*;
- *path-replication*: abilita la strategia *Path-Replication*, in caso di omissione sarà adottata quella standard: *Owner-Replication*.

Nel caso in cui nessuna delle tecniche di ricerca sopraelencate dovesse essere selezionata, verrà adottato il comportamento standard del protocollo Gnutella 0.4: *LimitedFlooding* con TTL pari a 7.

4.3 Protocollo Linkable: OverlayLink

Il protocollo Linkable, nello stack dei protocolli di PeerSim, prende posto subito sotto il livello Application. Ha lo scopo di mantenere le informazioni topologiche "locali" dell'overlay network riguardanti il nodo proprietario del protocollo.

La necessità di reimplementare il protocollo Linkable, invece di utilizzare una implementazione già presente in PeerSim, si è resa necessaria per consentire il funzionamento della tecnica di ricerca *Random Walk Degree Biased*. L'operazione necessaria al funzionamento di tale tecnica consiste nella generazione di un elenco dei vicini ordinati per grado, semplificando così di gran lunga la fase di inoltro. Per far ciò, l'elenco dei nodi all'interno del protocollo è implementato come un vettore di strutture ordinate, contenenti l'oggetto nodo vero e proprio ed il suo grado, valore utilizzato per l'ordinamento.

4.4 I controllori

Per l'interazione con la rete generata da PeerSim, prima, durante ed al termine dell'esecuzione del test, è necessario implementare delle classi *controllore*.

I controllori che operano durante i test sono chiamati ad intervenire alla fine di ogni intervallo di lunghezza prefissata; nel file di configurazione è possibile specificare la lunghezza di questo intervallo oppure se si tratta di controllori da richiamare unicamente in fase di inizializzazione.

Per la realizzazione del progetto sono state implementate tre classi controllore, una per la fase di inizializzazione, *DegreeInitializer*, e due di supporto durante l'esecuzione dei test, *QueryControl* e

Stat.

4.4.1 *InitDisk*

In fase di inizializzazione, quindi prima che qualsiasi tipo di esperimento possa aver luogo, è necessario distribuire i dati, che saranno poi oggetto della ricerca, sui vari nodi. Per questo scopo è stata implementata una classe controllore che si occupa di popolare i database dei nodi.

Per semplificarne l'implementazione, sia di questo controllore che del database dei nodi, è stato scelto di utilizzare delle semplici stringhe come dato elementare.

Tramite la configurazione è possibile specificare quanti nodi dovranno memorizzare dati utili, in questo modo, prima che il progetto venga avviato, sarà possibile calcolare un fattore di replicazione dei dati:

$$R = \text{numUtili} / \text{numNodi}$$

4.4.2 *DegreeInitializer*

Affinchè la tecnica di ricerca *Random Walk Degree Biased* funzioni correttamente, è necessario che ogni nodo sia a conoscenza dello stato degli altri nodi a cui è connesso. In particolare è necessario che ogni host conosca il grado dei suoi vicini; questa informazione sarà utile a costruire l'elenco ordinato da utilizzare durante la fase di inoltro.

La topologia della rete e le connessioni fra i nodi, in PeerSim, vengono stabilite in fase di inizializzazione e restano costanti durante l'intero esperimento, a meno di richieste esplicite di modifica; sfruttando questa caratteristica del simulatore, la classe *DegreeInitializer* esegue la sua procedura subito dopo che la rete è stata instaurata, affiliando ad ogni nodo un valore, indicante il grado, e popolando le opportune liste dei vicini per tutti i nodi presenti.

Questa operazione è eseguita esclusivamente in fase di inizializzazione.

4.4.3 *QueryControl*

Questa classe ha il compito di generare le Query da nodi casuali; tramite la configurazione è possibile impostare il numero di query da eseguire. Il controllore distribuirà equamente nel tempo le query richieste, facendo sì che alla scadenza di ogni intervallo un nodo casuale generi una query.

4.4.4 *Stat*

La classe *Stat* è il controllore più importante ai fini del progetto. Si occupa di raccogliere e stampare valori statistici riguardo le simulazioni, utili nelle successive analisi.

Nel file di configurazione, è stato impostato per questo controllore un intervallo pari all'intera durata della simulazione. In questo modo, verrà richiamato una volta all'inizio ed una volta alla fine dell'esperimento. Ignorando l'esecuzione a tempo 0 e prendendo in considerazione solo l'esecuzione finale, l'oggetto raccoglierà i dati dai vari nodi che hanno partecipato alla simulazione; in particolare, alla fine di ogni esperimento produrrà:

- la probabilità media di successo;
- il numero medio di hop per le query con esito positivo;
- il numero di messaggi medi per nodo;
- il numero massimo di messaggi elaborati da un nodo (HotSpot);
- il numero di nodi visitati.

Il formato dell'output sarà:

```
prSucc|avgHops|avgMsg|hotSpot|visNodes
```

E' stata scelta questa particolare formattazione per far sì che l'output potesse essere successivamente elaborato da *script bash* in modo semplice ed efficiente.

Riferimenti:

- [1] Qin Lv, Pei Cao, Edith Cohen, Kai Li, Scott Shenker, *Search and Replication in Unstructured Peer-To-Peer Networks*
- [2] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, Scott Shenker, *Making Gnutella-like P2P Systems Scalable*