RELAZIONE PROGETTO DI ING. DEL WEB

Matteo Bidoli, Andrea Cresta, Jacopo Foglietti

September 13, 2011

Contents

1 Introduzione

1.1 DP2P

Lo scopo del progetto è quello di realizzare un'architettura P2P ibrida per la condivisione di file con directory distribuita. Un sistema peer-to-peer è un sistema distribuito in cui ogni nodo ha identiche capacità e responsabilità e le funzionalità di un peer sono totalmente indipendenti da un server.

1.2 Specifiche

La struttura P2P ha principalmente le seguenti caratteristiche:

- controllo decentralizzato;
- adattabilità;
- capacità autonome di gestione e controllo di risorse e servizi;

Il sistema P2P viene definito ibrido quando:

- la fase di scambio dei file è peer-to-peer;
- la fase di boot utilizza qualche server;
- la fase di ricerca dei file usa un peer particolare, chiamato superpeer;

L'architettura è perciò organizzata in questo modo:

- ogni peer è associato ad un superpeer;
- i superpeer tengono traccia dei file condivisi dai peer ad esso associati e sono collegati tra loro tramite una rete "overlay";
- il server centralizzato di bootstrap tiene traccia dei superpeer presenti.

1.3 Obiettivi

Il software realizzato permette:

- l'ingresso e l'uscita di un nodo peer nella rete;
- la registrazione e l'uscita dei superpeer e la gestione della rete di overlay;
- la visualizzazione sul peer della lista dei file locali condivisi;

- la ricerca di un determinato file posseduto da un altro nodo della rete, tramite interrogazione del superpeer;
- il download di un file da un peer.

1.4 Esecuzione

L'esecuzione di un sistema P2P consiste solitamente di tre fasi distinte:

1. FASE DI BOOT: inizializzazione del sistema, entrata e registrazione di ogni nodo nella rete;
2. FASE DI LOOKUP: ricerca di un file,interazione peer-superpeer e successivo inoltro della richiesta nella rete di overlay;
3. FASE DI DOWNLOAD: trasferimento del file richiesto.

2 Architettura

2.1 Server di Bootstrap

Il server di bootstrap deve permettere l'accesso (e l'uscita) dei peer e dei superpeer alla (e dalla) rete P2P, fornendo loro gli indirizzi di alcuni superpeer attivi nella rete (se presenti). Inoltre è suo compito mantenere le informazioni relative ai superpeer attivi attraverso una lista, nella quale memorizza il loro indirizzo IP, e un campo intero ad esso associato (TTL). Tale campo viene utilizzato per verificare la permanenza del superpeer nella rete.
Il server svolge il suo lavoro in maniera concorrenziale, tramite l'utilizzo del multiplexing, implementato nel codice con la chiamata di sistema select.
I descrittori controllati nella select del server sono illustrati in figura 1:

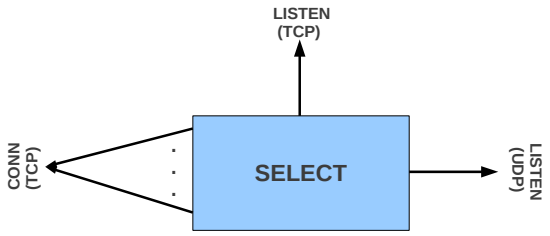


Figure 1: Struttura select del server

Il listen (TCP) è il descrittore della socket in attesa di connessioni entranti. Quando questo descrittore è pronto in lettura vuol dire che è arrivata una richiesta di connesione, la quale deve essere accettata dal server, instaurando così una connessione TCP. Il descrittore della socket relativo a tale connesione viene inserito sia tra i descrittori da controllare nella select, sia in una lista locale che mantiene tutti i descrittori delle connessioni accettate, in attesa di essere servite.
I conn (TCP) sono i descrittori delle connessioni TCP già instaurate e in attesa di servizio. Quando un descrittore di tipo conn è pronto in lettura significa che è stato ricevuto un messaggio dal server che può essere di 3 tipi:

- Join;
- Register;
- Leave;

Una volta letto il messaggio, il server servirà il client e successivamente verrà eliminato dalla select il descrittore della socket relativo alla sua connessione. Se non ci sono altri descrittori pronti in lettura, il server torna in attesa che si sblocchi un qualunque descrittore presente nella select.
Il descrittore della socket listen (UDP) è sempre in attesa di messaggi di tipo "ping" da parte dei superpeer attivi: ogni volta che questo descrittore è pronto in lettura, il server verifica se il messaggio ricevuto è effettivamente un "ping". In caso affermativo, reimposta il TTL del superpeer (al suo valore massimo, ossia 5) dal quale ha ricevuto il messaggio.

2.2 Superpeer

Il superpeer ha un' architettura piuttosto complessa in quanto deve gestire sia la comunicazione con altri superpeer che quella con i peer a lui collegati.

2.2.1 Struttura select superpeer

Per gestire le comunicazioni in modo efficente e simulare la concorrenzialità, il superpeer utilizza il multiplexing, implementato nel codice tramite la select.
La select del superpeer deve gestire principalmente 2 socket di ascolto (una per l'UDP e una per il TCP) e un numero variabile (configurabile da file config) di socket di connessione TCP, relative alle connessioni con altri superpeer.
Si riporta in figura 2 i descrittori controllati nella select.
Il descrittore di listen (TCP) è pronto in lettura quando c'è la richiesta di instaurazione di una connessione TCP da parte di un altro superpeer: se il superpeer che riceve la richiesta non ha raggiunto il numero massimo di connessioni disponibili per collegarsi con altri superpeer, accetterà la connessione inviando un pacchetto con comando uguale ad "ack". I due superpeer così connessi si scambieranno informazioni sulle rispettive statistiche di occupazione a livello di rete overlay (numero superpeer connessi) e su quanti peer possiede ciascuno di loro. Quando la listen (TCP) è pronta in lettura, se il superpeer ha esaurito le connessioni

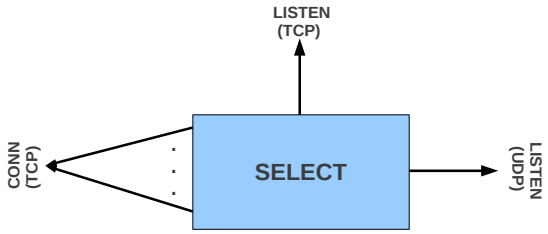


Figure 2: Select del superpeer

disponibili per la rete overlay, accetterà comunque la connessione TCP e cercherà nella sua lista di superpeer vicini se c'è qualcuno di loro che non sia pieno; se presenti, creerà un pacchetto con comando "full" e campo dati contenente gli indirizzi dei superpeer liberi. A questo punto, viene chiusa la connessione TCP e il superpeer che ha iviato la richiesta di connessione e che riceve un pacchetto con comando "full", dopo aver verificato che la dimensione del campo dati sia diversa da zero, parserà quel campo e contatterà gli indirizzi dei superpeer restituiti, al fine di instaurare una connessione TCP.
Il descrittore di listen (UDP) è pronto in lettura quando riceve un messaggio da un peer, oppure quando riceve una risposta/richiesta ad/di una whohas da superpeer che non sono suoi vicini (non direttamente collegati in TCP).I messaggi che si possono essere ricevuti su questa socket sono i seguenti:

Dai peer:

- Join;
- Whohas;
- Leave;
- Elezione;
- Riscontri relativi all'elezione "ackE", "nakE";

- Ping;
- Stop;

Dai superpeer:

- Whohas;
- Ack;

Le azioni che il superpeer intraprende quando riceve uno dei suddetti messaggi verranno spiegate nei capitoli appositi.
I descrittori di conn (TCP) sono in attesa di messaggi provenienti dalla rete di overlay, in particolare dai vicini del superpeer.
I possibili messaggi che possono rendere pronto in lettura uno di questi descrittori sono i seguenti:

- Whohas;
- Ack;
- Merge(fusione);
- ackM riscontro per il messaggio di fusione;
- ping;
- lettura di 0 byte (che equivale a dire che il superpeer corrispondente a questa socket ha interrotto la connessione);

Anche in questo caso le azioni intraprese dal superpeer alla ricezione di questi messaggi verranno specificate in seguito.

2.2.2 Protocolli di comunicazione

Il superpeer utilizza come protocollo di trasporto sia UDP che TCP. Per la comunicazione con i peer è stato scelto di utilizzare l'UDP in quanto ogni superpeer ha in generale molti peer a lui connessi e di conseguenza si è optato per un protocollo che richiede meno risorse e lo scambio di un numero minore di messaggi per la comunicazione (rispetto al TCP).
Per le connessioni nella rete overlay si sono utilizzati entrambi i protocolli TCP e UDP: il TCP è usato per instaurare connessioni permanenti con dei superpeer(che chiameremo vicini), mentre l'UDP è utilizzato durante le whohas per inoltrare le richieste (o risposte) ai superpeer che non sono

vicini, i cui indirizzi sono ottenuti nelle risposte alla whohas ricevute dai superpeer vicini (come si spiegherà in dettaglio nel capitolo riguardante la fase di lookup).
La scelta dell'utilizzo congiunto di TCP e UDP per la rete di overlay è stata fatta in base alla necessità di avere la trasmissione affidabile propria del TCP almeno con i superpeer vicini, che sono un numero limitato. D'altra parte,l'UDP è stato scelto per la comunicazione tra superpeer che non sono vicini poichè i messaggi che vengono scambiati tra di loro appartengono esclusivamente alla fase di lookup; questa potrebbe richiedere il bisogno di contattare numerosi superpeer e di conseguenza sarebbe eccessivamente pesante per la rete overlay l'utilizzo del TCP. Inoltre in questa fase si è ritenuto accettabile la perdita di alcuni pacchetti di risposta o di inoltro delle whohas, in quanto in situazione di regime si possono contattare un numero elevato di superpeer.

2.3 Peer

L'architettura del peer è piuttosto articolata, poichè deve gestire in modo concorrenziale le seguenti attività:

- Funzionalità principali del peer: menu e gestione dei comandi;
- Upload;
- Controllo

Ci sono quindi tre processi per gestire in modo separato questi compiti (figure 3).

2.3.1 Processo principale del peer

È il processo più importante del peer, in quanto permette all'utente di interagire con la rete.
A parte i processi di upload e di controllo, il peer svolge il suo lavoro in maniera concorrenziale, utilizzando il multiplexing, implementato nel codice con la chiamata di sistema select.
La select gestirà due descrittori:

- Il descrittore della socket UDP per poter inviare/ricevere messaggi al/dal superpeer
- Il descrittore dello stdin per poter gestire le richieste immesse dall'utente da tastiera

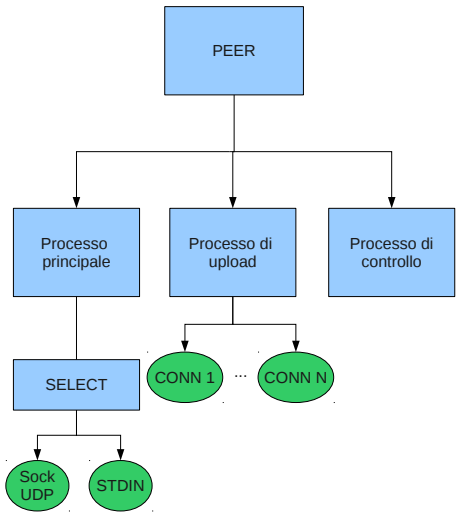


Figure 3: Architettura del peer

Il descrittore della socket UDP sarà attivato quando verrà ricevuto un messaggio UDP dal superpeer. I messaggi che il peer può ricevere dal superpeer sono:

- Stop
- Leave
- Nwsp
- Ack

Il messaggio di stop e di ack sono messaggi inerenti ai risultati di una ricerca precedentemente fatta dal peer. Il primo identifica il fatto che il superpeer ha bloccato la ricerca e lo comunica al peer. Il messaggio di ack invece identifica una lista di ip, ricevuti in risposta ad una ricerca, dai quali scaricare il file cercato.
Il messaggio di nwsp da parte del superpeer implica che il peer che lo riceve è stato eletto superpeer, quindi tale peer avvierà il processo di superpeer e si connetterà ad esso.

Se un superpeer invia un messaggio di leave al peer vuol dire che si sta sconnettendo dalla rete; il peer avrà quindi la necessità di connettersi ad un nuovo superpeer e sarà proprio il superpeer che si sconnette a consigliare al peer a chi connettersi.

Il descrittore dello stdin sarà attivato quando verrà ricevuto un input da tastiera. Il menu fornisce le seguenti scelte:

- Leave
- Update
- Whohas
- Stop

L'utente può avviare ricerce (whohas) e fermarle (stop) quando ha ricevuto qualche risultato, aggiornare i suoi dati sul superpeer (update), e, quando decide di uscire dalla rete, effettuare il comando apposito (leave)

2.3.2 Processo di upload

Questo processo si occupa di servire le richieste di download provenienti da altri peer e di conseguenza c'è una porta TCP perennemente in ascolto. Quando arriva una richiesta di download, il processo sul quale è attiva la socket in ascolto, se non si stanno già servendo il numero massimo di richieste, accetterà la connessione e avvierà un processo specifico per il download del file, il quale, una volta terminato, manderà un segnale al processo padre, così da aggiornare il numero di connessioni attive.
I dettagli delle procedure di download e upload verranno illustrati nel capitolo "Fase di download".

2.3.3 Processo di controllo

Il processo di controllo effettua periodicamente due operazioni: controlla se il processo main si è chiuso (in caso affermativo chiude tutti i processi del peer che non è più in esecuzione) e si occupa di inviare in modo ciclico un segnale di tipo SIGUSR1 al processo principale, al fine di temporizzare l'operazione di ping. Ogni volta che il processo principale riceverà questo segnale, effettuerà la specifica procedura di ping_handler.
La funzione di ping_handler è molto importante perchè notifica al superpeer che il peer è ancora attivo e gli permette anche di comunicare il suo rating a

quel dato istante.
La necessità di comunicare il rating periodicamente al superpeer è dovuta dalla natura dinamica della variabile rating, la quale è molto importante per la gestione dinamica della rete, poichè attraverso di essa si cerca di eleggere sempre superpeer migliori e più stabili.
La variabile rating è somma di due fattori: caratteristiche del peer e quantità di tempo trascorsa da quando esso è connesso alla rete.
Il rating dato dalle caratteristiche del peer è una grandezza fissa e viene calcolata all'inizio della sua esecuzione. In particolare dipende da RAM e CPU, e il suo valore viene ricavato dalla formula:

$$Rcar. = min(\frac{RAM}{4GB} + \frac{CPU}{3MHz}, 2)$$

Il secondo fattore nel calcolo del rating dipende dall'effettivo tempo del peer passato in rete, e questo indice è aggiornato ogni volta che viene invocato il ping_handler(). Esso è calcolato come l'effettivo tempo in secondi diviso il periodo massimo lungo il quale il rating è considerato massimo (in particolare è stato preso come valore di riferimento un giorno, ossia 86400 sec).

$$Rtempo = min(\frac{T\_rete}{86400sec}, 1)$$

Ovviamente il rating iniziale dipenderà solo dalle caratteristiche del peer in quanto il tempo trascorso in rete sarà nullo.
Il rating massimo raggiungibile da un peer è pari a 3, dove il valore dato delle caratteristiche tecniche incide per i 2/3 del rating totale e invece il rating dovuto al tempo per il restante 1/3.

2.3.4 Bloom Filter

I Bloom Filters sono particolari tipi di strutture dati che consentono di favorire l'efficienza in occupazione di memoria dei dati rispetto ad un certo grado di accuratezza nelle operazioni di lookup (ovvero di ricerca dei file). In particolar modo essi possono essere impiegati nelle reti P2P per compattare le liste dei contenuti da condividere le quali potrebbero raggiungere dimensioni eccessive.
Un filtro di Bloom consente di mappare un insieme di n elementi su un array di bit (posti inizialmente tutti a zero), applicando diverse funzioni hash ai vari elementi; viene così prodotto un array di bit che identificherà ogni singolo dato inserito, ponendo a 1 tutti i bit corrispondenti.
Per ritrovare il file sarà sufficiente che il filtro calcoli nuovamente la

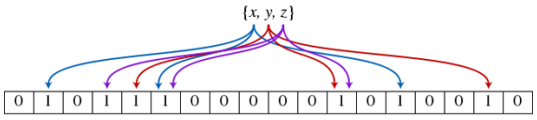


Figure 4: Struttura filtri di bloom

funzione hash per l'elemento cercato e verificare che i bit relativi siano impostati a 1. Se l'esito del matching è negativo, allora l'elemento non appartiene sicuramente all'insieme. Se l'esito del lookup è positivo, è possibile che l'elemento appartenga all'insieme, con una certa probabilità di errore (si parla in questo caso di "falsi positivi").
Nonostante esista questa limitazione, la quantità di memoria occupata è minima rispetto al numero dei dati da condividere nella rete. Se un'operazione di lookup in un Bloom Filter porta ad un falso positivo, il risultato è semplicemente una query ad un nodo remoto per un oggetto che in realtà non possiede.
Accettando una certa probabilità di errore (molto piccola se dimensionati opportunamente) si risparmia molto in occupazione di memoria. Nel progetto infatti il filtro viene creato a partire da un file contenente la lista dei nomi dei file condivisi e la dimensione del filtro è stata fissata a 100 byte. Mantenere intere liste di oggetti associate agli altri nodi della rete sarebbe proibitivo e anche il traffico in rete sarebbe stato maggiore, per questo motivo è stata scelta questa struttura per la rappresentazione dei file condivisi.

2.3.5 Protocolli di comunicazione

Il peer comunica con due entità diverse: il server di bootstrap e il superpeer associato.
Essendo la comunicazione con il server di bootstrap molto importante, in quanto permette al peer di accedere al sistema, il protocollo di comunicazione scelto è TCP. In questo modo il peer è sicuro di ricevere le informazioni necessarie per iniziare la sua corretta esecuzione.
La comunicazione con il superpeer avviene invece in UDP, non perchè meno importante tale comunicazione, ma perchè il superpeer deve poter gestire un numero grande di peer e quindi è importante che la comunicazione sia leggera e che non ci siano troppe connessioni ad appesantire il lavoro svolto dal supernodo.
I messaggi scambiati in udp sono strutturati in tre campi:

- Comand (4 byte)
- Size (4 byte)
- Data (dimensione variabile, indicata nel campo size)

L'unica eccezione a questa struttura sono i pacchetti UDP di whohas, i quali avranno un campo aggiuntivo per l'id della query.
Non essendoci connessione tra peer e superpeer, non c'è garanzia sull'arrivo dei messaggi scambiati. Per offrire un minimo di sicurezza, ogni funzione che invia pacchetti UDP sul peer utilizza un timer per bloccare l'attesa della risposta da parte del superpeer. Il timer viene attivato chiamando una alarm e indicando il tempo dopo il quale sarà inviato il segnale di tipo SIGALRM; quando il peer riceve questo segnale attiverà una specifica funzione (alarm_handler) che modificherà il valore della variabile var_ack bloccando il ciclo in cui c'è la chiamata di ricezione.
Se dopo lo scadere del timer non è stato ricevuto alcun messaggio, la funzione richiamerà se stessa per un numero finito di volte. In particolare è impostato ad un massimo 5 volte il numero di tentativi possibili, e il valore di timeout dipenderà dal numero della chiamata ricorsiva in corso. Nel primo tentativo di invio il timeout è di 1 secondo e nelle chiamate successive aumenterà di un secondo ad ogni chiamata ricorsiva. In questo modo si fornisce un piccolo controllo di congestione, in quanto si evita la ritrasmissione ravvicinata di più pacchetti, dando più tempo al superpeer per rispondere in caso di congestione della rete.

3 Gestione dinamica peer/superpeer

La necessità di rendere dinamica l'interazione che esiste tra un supernodo e i suoi nodi associati nasce dal fatto che, in un 'architettura P2P del tipo parzialamente ditribuita (ovvero ibrida), il sistema complessivo è progettato per trarre vantaggio dalla potenza di calcolo e di memorizzazione di un'ampia rete di calcolatori. E' di fondamentale importanza che sia garantita la capacità della struttura di trattare l'instabilità e la connettività in modo variabile e a seconda della situazione della rete di connessione, con la conseguenza di essere auto-adattabile e tollerante ai guasti. In generale, i requisiti base che il sistema deve rispettare per la selezione dei SuperPeer sono:

- presenza di una quantità di superpeer proporzionale al numero totale di nodi;
- numero massimo di peer connessi limitato per ogni superpeer;
- selezione di peer che possono diventare superpeer in base alle caratteristiche tecniche che hanno. Non tutti i peer infatti possiedono le capacità necessarie a svolgere il ruolo di supernodo, per questo motivo la rete è periodicamente aggiornata (utilizzando il valore di RATING) al fine di garantire le migliori prestazioni offerte dal sistema;

3.1 Connessione ad un superpeer

Le specifiche descritte sopra, servono innanzitutto per gestire il fatto che potenzialmente potrebbero esserci dei guasti o degli errori nella rete che causerebbero il crollo dei superpeer. In questo caso, dopo l'evento di timeout in cui il superpeer dimostra di non essere piu attivo, tutti i suoi peer associati rimarrebbero esclusi dal sistema e avranno bisogno di effettuare una nuova connessione.Il peer chiama quidni la funzione change_SP, con la quale, utilizzando un ulteriore funzione detta connect_to_SP, il peer può cambiare il suo superpeer tramite una nuova join al server e in seguito con la creazione di una nuova connessione con il superpeer che gli è piu vicino (potrebbe verificarsi che il peer nel frattempo venga eletto a sua volta supernodo e che esegua il codice relativo).

3.2 Elezione di un nuovo superpeer

Un'altra situazione problematica potrebbe essere quella in cui il peer che vuole accedere alla rete non trovi un superpeer disponibile, poichè tutti hanno già raggiunto il limite massimo di connessioni disponibili da accettare (o anche perchè la join_UDP non è andata a buon fine dopo tutti i tentativi disponibili). Serve perciò un nuovo superpeer che accetti nuove connessioni, e questo messaggio viene inviato dal peer ad un superpeer, preso dalla lista di indirizzi restituiti dal server dopo la join.
Il messaggio viene inviato utilizzando la funzione eleggi_UDP: tale procedura informerà il superpeer più vicino (comando di eleggi_UDP()) della necessità di un nuovo superpeer; il peer quindi invierà un messaggio strutturato di "elez" al superpeer, il quale proverà ad eleggere il suo best peer a superpeer in modo tale da far connettere il peer al nuovo superpeer (che è perciò anche il miglior nodo presente nella sottorete in quel momento a livello di rating). Il superpeer invierà al best peer un messaggio di tipo "nwsp" e il peer che riceverà questo messaggio abbandonerà il superpeer corrente, diventando a sua volta superpeer, e farà la join a se stesso. Se non ci sono problemi nell'avvio del processo di superpeer, il best peer invierà al suo superpeer precedente un messaggio di "ackE" per indicare che l'elezione è andata a buon fine.
Il superpeer che riceve ackE invierà un messaggio di "crsp" al peer indicando l'ip del nuovo superpeer (se l'elezione è andata bene). A questo punto, dopo aver ricevuto la conferma tramite un messaggio di "crsp" (crea sp), il peer esegue una nuova join all'indirizzo ricevuto, e la funzione restituirà 1 se la connessione verrà accettata e stabilita tra peer e nuovo superpeer.

3.3 Leave del superpeer

L'operazione di leave consiste nell'uscita di un superpeer dal sistema. A differenza dell'uscita di un semplice peer, in cui è sufficiente che il supernodo cancelli l'indirizzo ip e il filtro di bloom associato, la leave del superpeer deve gestire piu aspetti. Per prima cosa il superpeer informa il server di bootstrap, il quale cancella le informazioni relative al supernodo da cui ha ricevuto il messaggio (indirizzo IP piu filtro dei file condivisi dal lato peer); successivamente cerca e registra l'indirizzo del suo best peer (il miglior nodo a livello di rating tra tutti gli altri connessi) e gli invia un messaggio di "nwsp" (anche nella fase di elezione veniva mandato lo stesso messaggio, ma a quest'ultimo ora viene impostato ad 1 un flag che segnala

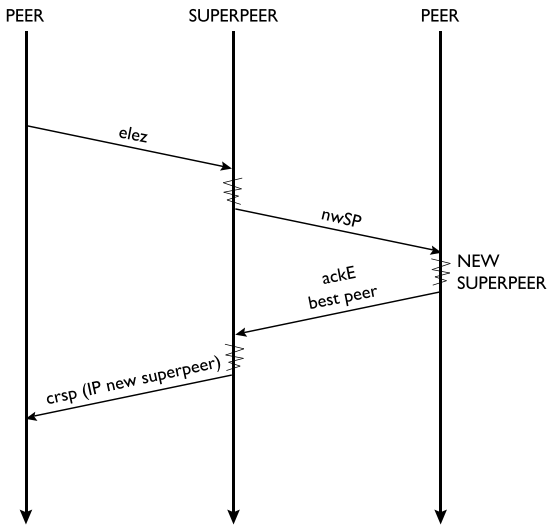


Figure 5: elezione di un peer a superpeer

che il nuovo superpeer prenderà il posto di quello vecchio, anzichè diventarne uno nuovo per accettare nuove connessioni); infine ogni peer viene avvisato del cambio del supernodo, ricevendo un messaggio di leave e contemporaneamente l'indirizzo IP del best peer che nel frattempo è diventato il nuovo superpeer e a cui potranno connettersi. Il processo del superpeer che ha effettuato la leave termina, mentre il processo peer associato proseguirà instaurando la connessione a lnuovo indirizzo ricevuto. Per come è stata strutturata l'architettura del sistema, il comando leave potrebbe essere effettuato da un peer che allo stesso tempo è attivo anche come superpeer. In questo caso le operazioni rimangono le stesse per entrambi: il peer invia una leave al superpeer (cioè a sè stesso), il superperr riconosce il suo indirizzo, cancella le informazioni associate al nodo che ha inviato il comando e allo stesso tempo effettua la leave nello stesso modo in cui è stata descritta sopra, facendo terminare di fatto sia il processo peer che quello superpeer.

3.4 Switch tra peer e superpeer

Svolge un ruolo importante anche l'aggiornamento periodico del rating all'interno sia della sottorete tra peer e superpeer sia della rete di overlay tra i vari superpeer. In questo modo infatti è possibile garantire buone prestazioni del sistema eleggendo al ruolo di supernodo (che dal punto di vista dell'utilizzo delle risorse è il piu impegantivo) i nodi migliori che entrano man mano nel sistema.
Ogni volta che un peer "pinga" il suo superpeer (ed anche quando viene effettuata un operazione di update dei file condivisi), invia a quello anche il suo rating aggiornato, e nel caso in cui si colleghi un nuovo peer che abbia rating maggiore (il doppio) rispetto al supernodo a cui si è collegato, viene attivata una fase di switch tra il superpeer e il miglior peer.
Tale fase consiste di poche fasi:

- Leave al bootstrap del superpeer
- Invio messaggio di elezione al peer designato con flag di switch
- Leave forzata dei peer connessi al superpeer

Il nuovo peer diventa superpeer e la rete viene aggiornata di questo scambio, sia con la register al server, sia con gli altri peer che dovranno collegarsi al nuovo superpeer ed infine con l'instaurazione di nuove connessioni TCP con gli altri supernodi della rete di overlay.
Il flag di switch identifica un pacchetto di tipo "nwsp" con campo dati impostato a "s" al contrario del messaggio di nuovo superpeer usato nell'elezione semplice che ha campo dati vuoto. Tale flag permette di bloccare l'inoltro del messaggio "crsp" il quale senza flag sarebbe stato inviato al peer che ha generato la procedura di switch e questo sarebbe inutile.
Da notare come il valore del rating non dipenda solo dalle capacità tecniche di un nodo, ma anche dal suo tempo di permanenza in rete; viene così garantita una stabilità maggiore all'applicazione, che potrà fare affidamento su calcolatori potenti e allo stesso tempo affidabili per quanto riguarda la continuità del funzionamento.

3.5 Unione di due superpeer

Anche i superpeer scambiano tra di loro messaggi di aggiornamento rating, ma per un motivo diverso: se nella rete di overlay coesistono ad esempio due superpeer con tasso di occupazione minore del 20% (dove per tasso di

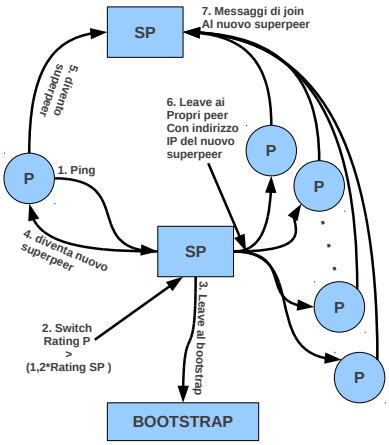


Figure 6: Switch del superpeer con un suo peer con rating migliore almeno del 20 %

occupazione si intende il numero di peer connessi al superpeer rispetto al numero totale che in realtà potrebbero essere collegati contemporaneamente), avviene una fusione tra i due. Quando uno di loro riceve una leave da un suo peer, viene controllata la lista dei nodi che gli sono ancora connessi; se il numero è abbastanza piccolo (minore di 20), il superpeer chiama la funzione getSuperPeerBassaOccupazione(), ottenendo il numero di supernodi che come lui hanno un'occupazione minore del 20%. Se ne esiste almeno uno in questa situazione, invia un comando di merge (fusione appunto) con allegato il valore del suo rating: nel caso in cui il superpeer che manda il messaggio ha rating maggiore, l'altro risponde con un messaggio di conferma e per prima cosa fa collegare i suoi peer al primo (questi fanno una leave al loro superpeer, e, tra i peer che si scollegano, il superpeer riconosce il suo indirizzo associato), manda quindi una leave al server (chiudendo il processo superpeer) e tutte le connessioni vengono aggiornate; se invece ha rating minore, succede il contrario. Viene illustrato in figura un esempio di merge tra due superpeer:

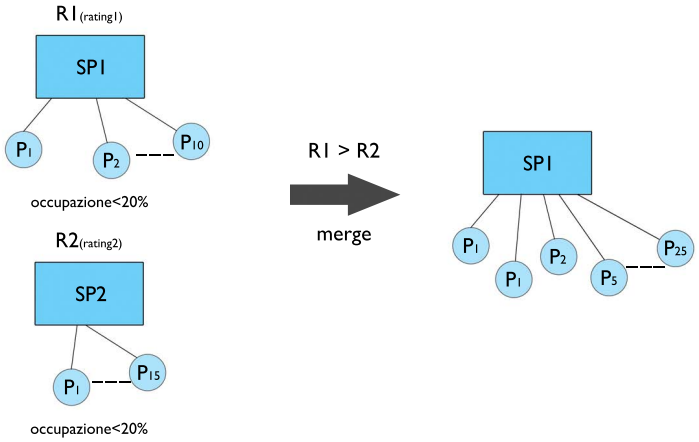


Figure 7: merge tra due superpeer

4 Fase di boot

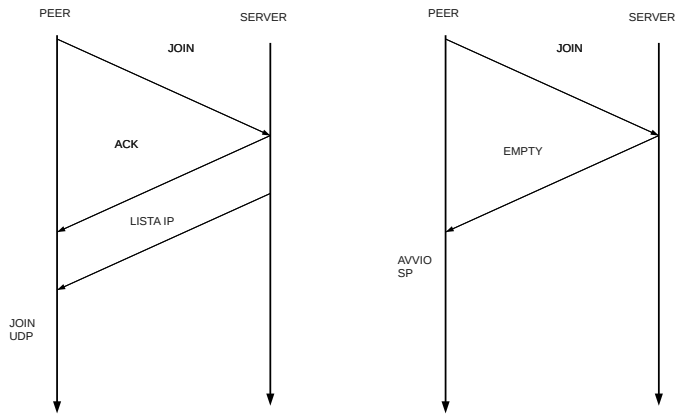
La prima fase dell'esecuzione di un sistema P2P distribuito è quella di boot, ossia di inizializzazione del sistema. I peer che vogliono entrare nella rete comunicano con il server di bootstrap, il quale fornirà loro una lista parziale di IP dei superpeer a cui connettersi.
Allo stesso modo i superpeer avranno una loro fase di boot, nella quale, comunicheranno al server di bootstrap il loro ingresso nella rete.

4.1 Peer

4.1.1 Join al server di bootstrap

Il peer per connettersi alla rete deve contattare il server di boostrap. Questa operazione (detta di *join*) è fondamentale per il corretto funzionamento del peer, il quale deve essere sicuro di ricevere le informazioni per poter essere eseguito correttamente. La *join* al server è molto semplice, viene instaurata una connessione TCP tra le due entità, le cui scambieranno dei messaggi.

Come si può vedere in figura 8, il peer, dopo essersi connesso, invierà un messaggio di "join", il quale verrà letto e processato dal server.
Quest'ultimo scandirà la lista dei superpeer presenti nella rete e risponderà al peer, fornendogli i riferimenti ad essi. Informerà inoltre il peer del fatto



(a) Lista non vuota (b) Lista vuota

Figure 8: Join del peer al server di bootstrap

che esso debba diventare un superpeer o no.
Sono quindi possibili due scenari:

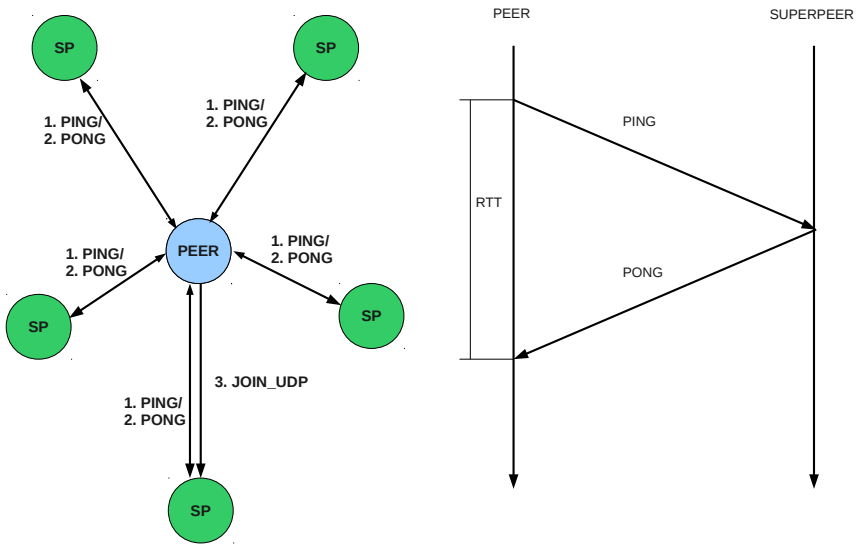
- La lista dei superpeer è vuota
- La lista contiene almeno un superpeer

Nel primo caso il peer che effettua la join verrà eletto automaticamente superpeer dal server, il quale manderà al peer un messaggio di "empty" per comunicare che la lista è vuota. Il peer che è stato eletto superpeer dovrà effettuare una seconda fase di boot riguardante l'inizializzione del superpeer.
Nel secondo caso invece, il peer riceverà un messaggio di "ack", seguito da una lista di IP ai quali può connettersi. Inizierà a questo punto la fase di join al superpeer.
I messaggi scambiati in questo caso non sono strutturati, ma sono semplici stringhe inviate nella rete ("join", "ack", "empty"). Al contrario, gli ip inviati dal server al peer sono degli interi long che identificano l'ip del superpeer nel formato di rete.

4.1.2 Join udp al superpeer

Il peer che vuole connettersi ad uno dei superpeer presenti nella lista ricevuta dal server, cercherà di compiere la scelta localmente migliore,

contattando il superpeer geograficamente più vicino tra quelli disponibili. Per potersi "connettere" al superpeer migliore, il peer effettuerà quindi un ping (utilizzando *ping_UDP*) su tutti gli ip restituiti e li ordinerà in base al miglior tempo di risposta. Dopo aver deciso un ordine di preferenza tra i vari superpeer, proverà a connettersi al miglior superpeer possibile tramite un operazione di *join_UDP*.
Tale operazione è molto importante in quanto permette di creare una "connessione" logica con il superpeer, il quale riceverà dal peer le informazioni sui file condivisi contenute in un filtro di bloom.
Le operazioni di ping_UDP e join_UDP, essendo basate sul protocollo UDP, presentano vantaggi e svantaggi che le operazioni su TCP non hanno.



(a) Ping del peer ai superpeer e successiva join_UDP al superpeer scelto

(b) Ping del peer al superpeer

Figure 9: Fase di boot del peer

PING_UDP
L'operazione di ping_UDP è molto utile, in quanto viene usata per misurare il tempo di risposta di un superpeer (come nel caso della fase di boot di un peer) o per refreshare le informazioni di un peer o un superpeer.
Nel chiamare la funzione ping_UDP, il peer specifica su quale IP effettuare il ping. Questo comando viene inviato al superpeer tramite un unico

segmento udp strutturato in un pacchetto come indicato precedentemente nei dettagli architetturali del peer.
In questa operazione il campo command è ovviamente impostato a "ping", mentre l'unico dato aggiuntivo del pacchetto è il rating del peer che invia il ping. Il rating viene inviato insieme al ping in modo tale da aggiornare le informazioni che il superpeer possiede del peer.
Tale aggiunta al messaggio di ping è dovuta dal fatto che il superpeer così non ha solo l'informazione sui file contenuti nei peer, ma ha anche una panoramica della loro qualità. Queste informazioni possedute dal superpeer sono molto utili per la gestione dinamica della rete, ma per questo aspetto si rimanda allo specifico capitolo.

JOIN_UDP
L'operazione di join_UDP permette al peer di instaurare una "connessione" logica con il superpeer. Il peer, che chiamerà questa funzione, invierà al superpeer un messaggio UDP strutturato di join. Tale messaggio è strutturato, come tutti gli altri messaggi UDP che vengono scambiati tra peer e superpeer, da tre campi: comand, size e dati. Nella join_UDP il campo comand è impostato a "join" (da non confondere con il messaggio NON strutturato tcp di join), e nel campo dati viene salvato il filtro di bloom del peer e il suo rating.
Il superpeer che riceve un messaggio di join_UDP deve innanzi tutto controllare la sua disponibilità ad accettare connessioni, quindi se il numero di peer connessi ad esso non avranno raggiunto il valore massimo di riferimento NUMERO_MAX_PEER allora il superpeer accetterà il nuovo peer. Ovviamente se il peer è già connesso al superpeer la join verrà usata unicamente per aggiornare le informazioni di tale peer (ad esempio il comando update, che non fà altro che chiamare la join_UDP).
Poichè la comunicazione su cui si basa lo scambio di messaggi tra peer e superpeer è di tipo UDP, i messaggi saranno di numero limitato e quindi non appesantiranno la rete. Il problema dell'UDP però è che non si hanno sicurezze sull'arrivo dei messaggi, quindi non si può avere neanche la sicurezza riguardo all'esecuzione di un superpeer.
Il peer che chiama la join_UDP utilizza un semplice sistema di timer interno all'operazione così da evitare che il peer rimanga in attesa per un tempo troppo lungo di una risposta da un superpeer, il quale potrebbe anche non rispondere mai.
Dopo aver mandato il messaggio strutturato, il peer avvierà un timer (utilizzando il segnale di SIGALRM inviato dalla chiamata di sistema alarm(t) dopo un tempo t): se entro un certo tempo non arriverà alcuna risposta, la join_UDP proverà a rifare la join per cinque volte. Nello

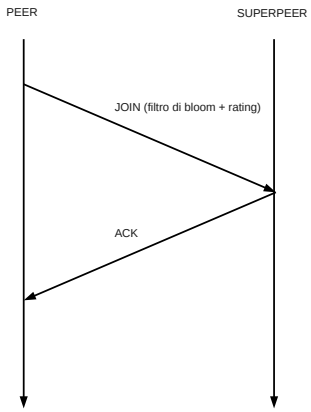


Figure 10: Join del peer al superpeer

scenario in cui non esista il superpeer che si vuole contattare, il peer proverà a contattare per cinque volte il superpeer scelto; tuttavia nei diversi tentativi di contattare il superpeer il timeout non sarà costante, bensì aumenterà ad ogni nuova chiamata della funzione, in modo tale da non occupare la rete con tanti messaggi ravvicinati e per far sì che, anche se la rete è congestionata ma il superpeer è in realtà connesso, quest'ultimo abbia il tempo di rispondere (vedere figura 11).

4.2 Superpeer

4.2.1 Register del superpeer al server di bootstrap

Il peer che viene eletto superpeer dovrà effettuare per prima cosa un operazione detta di *register*, la quale consiste nell'informare il server della sua presenza nella rete e nella disponibilità ad interagire con i peer che si connetteranno successivamente.
Questa operazione, al pari della join del peer al server, è molto semplice: il superpeer invia un messaggio di "register" al quale il server risponderà con "empty" nel caso non ci siano superpeer connessi, o con "ack" seguito da una lista di ip di altri superpeer.
Il server che riceve una register, oltre a rispondere con "ack" o "empty" (come nel caso della join del peer), inserirà nella sua lista di superpeer l'ip di quello che sta effettuando la register.

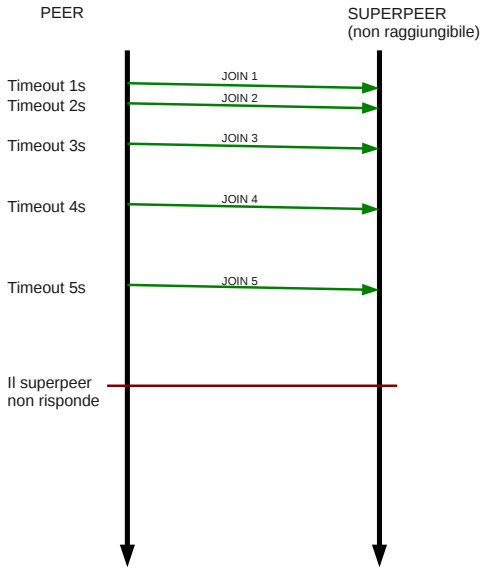


Figure 11: Join del peer a un superpeer non raggiungibile, gestione timeout

Un altra differenza rispetto alla join del peer è che la lista di ip restituita viene usata dal superpeer per creare una rete di overlay con altri superpeer, con i quali poter comunicare per gestire la rete ed effettuare ricerche. Nonostante lo scambio di messaggi tra superpeer e server è totalmente analogo alla join del peer, le due operazioni vengono usate per scopi molto diversi: la join serve al peer solo per poter trovare un superpeer al quale collegarsi, la register invece serve per inizializzare il superpeer il quale, con le informazioni ricevute dal server, creerà la rete sulla quale avverranno tutte le operazioni.

4.2.2 Entrata del superpeer nella rete d'overlay

Dopo aver effettuato la register al server di bootstrap il superpeer inizia a provare ad aprire al più 3(il numero massimo di vicini è stato impostato a 6) connessioni TCP con gli indirizzi dei superpeer attivi ottenuti dal bootstrap i quali possono accettare la connessione rispondendo con un ack e

instaurando di fatto la connessione TCP permanente presente tra vicini.
In caso il superpeer che riceve la join ha esaurito le connessioni TCP disponibili(è già connesso a 6 superpeer), risponderà al messaggio di join con un messaggio di "full" il quale inoltre conterrà gli indirizzi IP dei propri vicini che non hanno esaurito le connessioni (se presenti), dopo di chè chiuderà la connessione TCP, il superpeer che riceve in risposta un messaggio di "full" proverà a contattare i superpeer ricevuti in risposta insieme al messaggio di full (se presenti) inoltrando a loro la join, questo finchè non riesce ad instaurare almeno 3 connessioni TCP oppure non ci sono più superpeer a cui effettuare la join.
La scelta di instaurare al più 3 connessioni TCP e non direttamente tutte e 6 all'inizio è stata fatta per evitare una situazione di stallo in cui tutti i superpeer hanno tutti il numero massimo di connessioni TCP già attive e di conseguenza un nuovo superpeer che arriva rimarrebbe al di fuori della rete di overlay e mano a mano che arrivano nuovi superpeer (se non si dovessero scollegare superpeer) si verrebbe a creare una rete di overlay completamente indipendente da quella che ha tutte le connessioni TCP già attive, e questo potenzialmente si potrebbe verificare più volte, causando la creazione di più reti overlay separate.

5 Fase di lookup

5.1 Introduzione

Una volta terminata la fase di Boot, il peer può iniziare a inoltrare al proprio superpeer richieste di ricerca. La fase di lookup (ricerca) riveste un ruolo fondamentale nell' applicazione, in quanto l'abilità di trovare almeno un istanza di un oggetto è molto importante, altrimenti non sarebbe possibile procedere alla successiva fase di download del file da uno dei peer ottenuti come risposta da questa fase.

5.2 Ricerca

La ricerca da parte di altri peer possessori di un determinato file è svolta in maniera indiretta dal peer, in quanto il lavoro piu grande viene svolto dal suo superpeer.
Le ricerche sono effettuate in maniera sequenziale, durante questa fase le operazioni che il peer può effettuare sono solamente l'update, lo stop della ricerca e l'invio del ping al proprio superpeer. Non è possibile effettuare ricerche in parallelo.
Per lo scambio di messaggi nella fase di lookup è stato creato un pacchetto specifico (pacchetto whohas), con la seguente struttura:

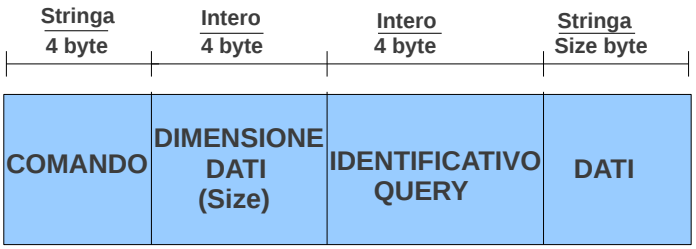


Figure 12: struttura pacchetto per la whohas

La fase di lookup può essere suddivisa in 3 sottofasi principali:

- Invio richiesta di ricerca da parte del peer al proprio superpeer, e attesa delle relative risposte;
- Ricerca da parte del superpeer dei peer a lui collegati che possiedono il file richiesto, inoltro della richiesta sull'overlay e attesa risposte dall'overlay, gestione risposte ricevute e inoltro dei risultati al peer che stà effettuando la ricerca;
- Fine della ricerca; in seguito, il peer che ha effettuato la ricerca, se ci sono dei peer che possiedono il file, effettuerà il download del file cercato da uno dei peer restituiti.

5.2.1 Richiesta da parte del peer al superpeer

Quando un utente decide di avviare una ricerca dovrà selezionare nel menù il comando di whohas. Inserito il comando da terminale, verrà richiesto il nome del file da cercare (il nome non può essere abbreviato ma deve essere esattamente uguale al nome completo del file, eccezione fatta per le maiuscole e le minuscole). Una volta inserite queste informazioni l'applicazione provvederà a creare il pacchetto di tipo whohas descritto precedentemente, con i relativi campi riempiti come segue:

- Comando = "whhs";
- Size = dimensione in byte del nome del file da cercare;
- Identificativo whohas (ID) = 0;
- Dati = nome del file da cercare.

Una volta creato il pacchetto, questo verrà passato alla socket UDP che provvederà a inoltrarlo verso il superpeer di destinazione. Si noti come un pacchetto whohas inviato da un peer verso il proprio superpeer abbia sempre identificativo uguale a 0 (questo per permettere al superpeer di distinguere le whohas ricevute dai propri peer da quelle ricevute dagli altri superpeer nella rete di overlay, le quali hanno sempre identificativo diverso da zero, come vedremo successivamente). Si conclude questa parte illustrando lo scambio di messaggi tra peer e superpeer associato durante una ricerca, trascurando momentaneamente il dettaglio su come poi il superpeer interagisca con gli altri superpeer della rete di overlay e la fine della ricerca.

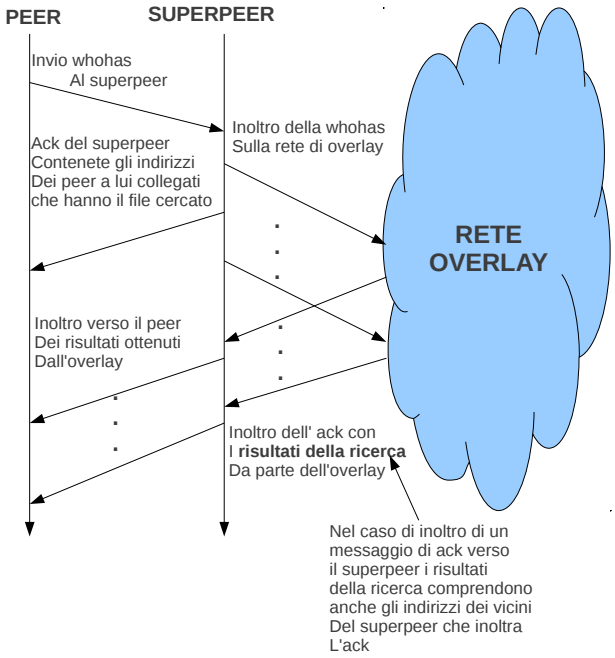


Figure 13: Scambio di messaggi tra peer e superpeer durante una ricerca

5.2.2 Ricerca dei peer possessori del file da parte del superpeer

Quando un superpeer riceve sulla sua socket UDP un messaggio di whohas, si crea il pacchetto dal buffer della socket attraverso apposite funzioni di parsing e memorizzazione. Se il comando ricevuto è uguale a "whhs" e la richiesta è stata inoltrata da un peer che è collegato a questo superpeer, come prima cosa il superpeer controlla il valore del campo ID: se quest'ultimo è pari a zero, si entrerà nella parte di codice relativa alla gestione della whohas ricevuta da un propio peer (nuova ricerca).
Anche se la ricerca a livello di peer è sequenziale, il superpeer deve poter gestire più ricerche attive in parallelo e nasce quindi l'esigenza di distinguere le varie ricerche pendenti. Per far ciò, quando il superpeer riceve una whohas con identificativo pari a 0, assegna alla query

corrispondente un nuovo ID univoco (a livello locale di superpeer), assegnato in maniera incrementale.
Una volta assegnato un nuovo ID alla whohas (di una nuova ricerca), il superpeer ricava il nome del file di cui si vogliono conoscere i possessori dal campo dati del pacchetto e inoltre memorizza anche l'indirizzo IP del peer che ha inviato la richiesta. Per la gestione di query pendenti, il superpeer utilizza una lista (lista_query) che contiene le seguenti informazioni:

- ID della query (assegnata dal superpeer);
- IP del peer che ha inoltrato la query;
- Nome del file da cercare;
- Un contatore per mantenere l'informazione su quanti superpeer sono già stati contattati;
- Un contatore per mantenere l'informazione su quanti IP di peer si sono già ricevuti in risposta;
- Due liste di indirizzi IP (try_list, done_list, il cui scopo verrà illustrato in seguito).

Dopo aver effettuato i controlli descritti precedentemente ed aver inserito la query appena ricevuta nella lista delle query pendenti, il superpeer come prima cosa verifica se tra i suoi peer c'è qualcuno che possiede il file richiesto. In caso affermativo inoltra un pacchetto whohas il cui comando viene settato ad "ack", ID a zero, il campo dati con gli indirizzi IP dei peer che possiedono il file e il campo size con il numero di byte presenti nel campo dati, inoltre viene aggiornato anche il contatore in lista_query relativo al numero di IP di peer che hanno il file cercato. Nel caso in cui nessuno dei suoi peer abbia il file cercato invierà un pacchetto whohas con comando settato ad "ack", ID a zero, campo dati vuoto e size uguale a zero.
Dopo aver effettuato la ricerca in locale tra i suoi peer, il superpeer inoltra la whohas nella rete overlay ai suoi vicini, con il pacchetto whohas riempito come nel caso visto precedentemente per il peer quando vuole inoltrare una richiesta al suo superpeer, fatta eccezione per il campo ID nel quale si mette l'ID assegnato dal superpeer nel momento in cui gli era arrivata la richiesta dal suo peer.
Come descritto nel capitolo riguardante l'architettura, si ricorda che la comunicazione tra un superpeer e i suoi vicini avviene in TCP in maniera tale da avere a disposizione il "trasferimento dati affidabile" propio del

TCP, come ultimo passo di questa fase il superpeer inserisce gli indirizzi dei vicini così contattati nella done_list relativa all'attuale query. La done_list è una per ogni query ed è stata inserita al fine di evitare di inoltrare una whohas più volte allo stesso superpeer.
Una volta finita questa prima fase il superpeer per quanto riguarda la fase di ricerca si mette in attesa di risposte dai suoi vicini.
Le risposte che arrivano al superpeer dalla rete di overlay sono strutturate come segue:

- Comando = "ack";
- Size = dimensione in byte del campo dati;
- Identificativo whohas (ID) = ID;
- Dati = IP dei peer che possiedono il file, poi è stato inserito un carattere di delimitazione uguale a ";" seguito dagli indirizzi dei superpeer vicini del superpeer che mi stà inoltrando la risposta.

Dopo la verifica che il comando ricevuto sia un ack il superpeer verifica che l'ID ricevuto sia presente nella lista_query altrimenti scarta il pacchetto in quanto la risposta è relativa ad una query la cui ricerca è già terminata.
Una volta verficato che il comando sia un "ack" e l'ID ricevuto sia presente nella lista_query il superpeer tramite la funzione parsingWhoHas(), parserà il campo Dati del pacchetto in modo tale da ottenere gli indirizzi IP dei peer possessori del file i quali sono messi nel campo dati di un pacchetto whohas strutturato come quello inviato dal superpeer al suo peer dopo aver fatto la ricerca in locale tra i suoi peer. Gli indirizzi relativi ai vicini del superpeer che ha inviato la risposta, vengono controllati uno ad uno se sono già presenti in try_list o done_list, nel caso non siano presenti in nessuna delle due liste, vengono inseriti in try_list la quale è una lista anch'essa come done_list relativa ad ogni query, il cui scopo però è quello di memorizzare tutti gli indirizzi dei superpeer ottenuti come risposta ad una whohas che non siano stati già contattati.
Alla fine della gestione della ricezione di un "ack" da un superpeer, il superpeer controllerà se la try_list relativa alla query di cui si è ricevuto l'"ack" è piena o vuota.
Se try_list è vuota allora la gestione dell'ack finisce, altrimenti si inoltra la whohas a tutti i superpeer presenti in try_list in UDP, spostandoli poi dalla try_list in done_list. La scelta dell'UDP per l'invio delle whohas ai superpeer che non sono vicini del superpeer che le stà inviando, è stata

effettuata allo scopo di non appesantire la rete di overlay in quanto l'UDP rispetto al TCP richiede uno scambio di messaggi minore non essendo necessario instaurare una connessione, inoltre usando l'UDP c'è una minore richiesta di risorse sui superpeer, questi vantaggi vanno a discapito della "trasmissione dati affidabile" alla quale si è rinunciato in quanto a regime i superpeer che si contattano sono un numero elevato e la perdita di alcuni ack o di alcune whohas sono accetabili.
In figura 14 è illustrato lo scambio di messaggi tra superpeer e come si propaga la whohas sull'overlay.

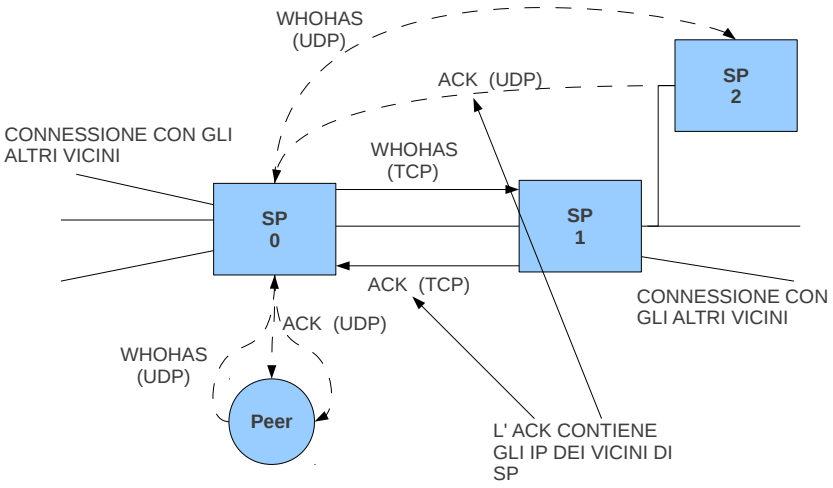


Figure 14: Propagazione della whohas sull'overlay

Come si può vedere un superpeer che riceve la whohas dal suo peer la inoltra ai suoi vicini in TCP, in figura si è evidenziato lo scambio con uno solo di questi il quale quando inoltra l'ack, comunica anche gli indirizzi IP dei superpeer adiacenti (sempre utilizzando TCP), ai quali verrà successivamente inoltrata la whohas questa volta però in UDP.
I superpeer, che hanno ricevuto la whohas, a loro volta inoltreranno l'ack in modo simile ai vicini con la sola differenza di utilizzare UDP come protocollo di trasporto, e la ricerca proseguirà nel modo descritto fino a quando non si verificheranno determinate condizioni discusse successivamente.

5.2.3 Fine ricerca

Il controllo sulla fine della ricerca è implementato allo scopo di non congestionare la rete di overlay, in quanto senza questo controllo i pacchetti di whohas e di ack circolerebbero all'infinito nella rete. Nel metodo implementato, il superpeer che origina la query contatta iterativamente alcuni superpeer della rete, finché non si verifichi almeno una delle seguenti condizioni:

- Il superpeer che ha originato la query ha contattato un numero massimo di superpeer(configurabile tramite file config);
- Il superpeer che ha originato la query ha ottenuto in risposta un numero massimo di peer che contengono il file cercato (configurabile tramite file config);
- Il peer che ha effettuato la whohas al superpeer decide di fermare la ricerca(Stop UDP).

STOP_UDP
La stop_UDP viene richiamata dal peer quando l'utente sceglie il comando di STOP durante una ricerca, dopo di chè il peer invierà un messaggio di STOP in UDP con un meccanismo di ritrasmissione come tutti gli altri messaggi UDP. Quando il superpeer riceve il messaggio di STOP della ricerca dal peer cancellerà dalla sua lista di query la query corrispondente al peer(che riconosce nella lista tramite l'indirizzo IP da cui gli arriva il messaggio di STOP), in questo modo anche se gli dovessero arrivare altre risposte alla query appena cancellata(se quando gli arriva lo STOP aveva già inoltrato qualche whohas di cui attende risposta), cercando nella lista delle query pendenti l'id e non trovandolo scarterà il pacchetto evitando così di ritrasmetterlo ad altri superpeer, ed inoltre non invierà le risposte ricevute neanche al peer in quanto per lui ormai la ricerca si è conclusa.

6 Fase di Download

6.1 Descrizione

Il trasferimento dati da un peer ad un altro è il cuore dell'applicazione. Dopo infatti la fase di registrazione e quella di ricerca di un determinato file, un peer può finalmente contattare l'indirizzo specifico di chi lo possiede (sempre nel caso in cui almeno un peer nella rete abbia il file da condividere).

6.2 Struttura

Ogni peer mantiene una socket TCP attiva in ascolto, pronto a ricevere una richiesta di trasferimento dati e avviare l'upload del file richiesto.
Tramite la configurazione del file config, è possibile impostare il numero massimo di upload che ogni nodo puo avviare contemporaneamente. Per quanto riguarda il download invece, è possibile effetuarne solo uno alla volta ed anche la ricezione del file avviene da un unico indirizzo IP specifico.

I dati da inviare vengono suddivisi in pacchetti di dimensione prefissata (512 byte) detti CHUNK: questa struttura consente sia di avere una sicurezza ulteriore nel download (un file molto grande inviato per intero avrebbe piu possibilità di essere ricevuto corrotto) sia di poter riprendere il download nel caso in cui il trasferimento sia stato interrotto per un qualche motivo. Il peer infatti potrebbe richiedere il file dall'ultimo CHUNK ricevuto senza ricominciare il download dall'inizio.
La fase di download è "bloccante": l'utente non puo effettuare altre operazioni una volta che ha digitato il comando per avviare il trasferimento dei dati; l'unico segnale che continua a rimanere attivo è il ping inviato periodicamente al superpeer.

L'applicazione richiede che ogni peer metta a disposizione due cartelle diverse, una per i file da condividere (cartella_condivisa; impostabile nel file di configurazione) ed una per i file scaricati (Scaricati; non impostabile). Questa distinzione nasce dal fatto di voler evitare che un peer cominci a condividere un oggetto che lui stesso sta scaricando da un'altra fonte, ma che non è ancora completo (la verifica infatti che un file sia disponibile ad essere inviato viene fatta sulla presenza del nome del file in cartella_condivisa).

6.3 Esempio di esecuzione

Si consideri il caso in cui ci siano due peer, A e B, e il primo voglia scaricare un file dal secondo. La funzione di download implementata, che prende come parametro il nome del file cercato e l'indirizzo IP del peer che possiede tale file, lavora secondo il seguente schema:

1. il peer A fa una richiesta ad un peer B di tipo get, specificando il nome del file.
2. il peer B risponde con la dimensione (come informazione e come ack al peer A).
3. il peer A quindi dà la conferma dell'esecuzione del download specificando da quale chunk partire.
4. il peer B invierà i chunk dal punto richiesto.
5. quando il peer A leggerà un chunk lo salverà sul file in modo sequenziale.

Si può vedere piu in dettaglio tale esecuzione nel diagramma in figura 15 che rappresenta appunto lo stesso scenario.

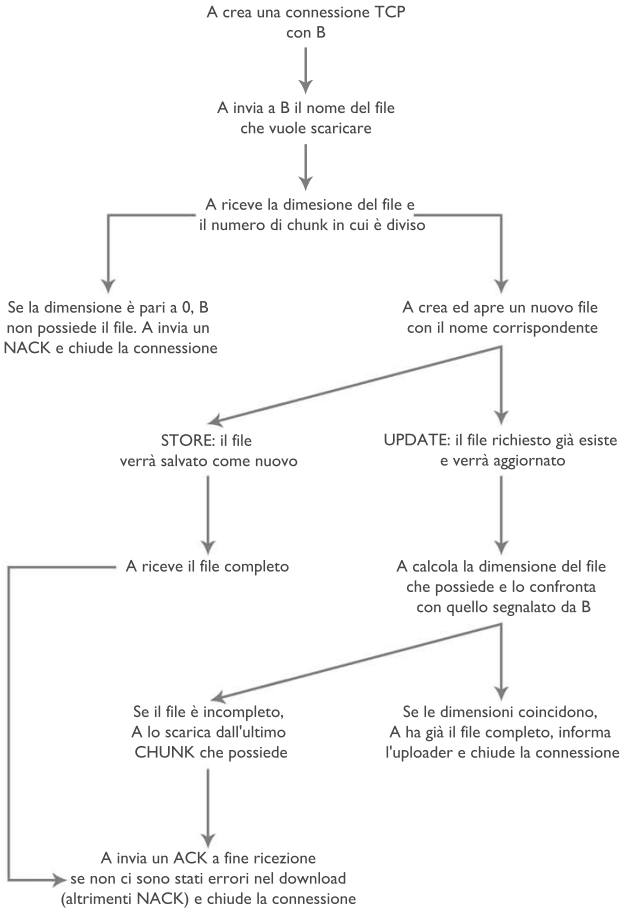


Figure 15: Esecuzione del download tra due peer

7 Esempi di funzionamento e limitazioni riscontrate

7.1 Test download

7.1.1 Output del test

PEER

```
rating ram = 0.735992
rating cpu = 0.756667
CODICE PEER

JOIN (5) IP=160.80.146.107
comando ricevuto=ack
size=0
cont=5 RESULT=> ack
join effettuato a 160.80.146.107
----------------------------------

#COMANDI
1: LEAVE
2: UPDATE
3: WHOHAS
=>inserire il numero della scelta:
----------------------------------
3

quale file si vuole cercare?
roxanne.mp3

WHOHAS:roxanne.mp3 - cont=5
comando ricevuto=ack
size=4
ID=0
numero di ip 1
LISTA IP RESTITUITI
0 = 160.80.148.15

cont=5 RESULT OF THE OPERATION=> ack
esito della whohas di 'roxanne.mp3'  : 1
```

```
---------------------------------
#COMANDI
2: UPDATE
4: STOP (della ricerca)
=>inserire il numero della scelta:
---------------------------------
4

STOP (5)
comando ricevuto=ack
size=0
cont=5 RESULT=> ack
FINE RICERCA , ip trovati: 1
STAMPA LISTA
ELEM[0] IP= 160.80.148.15
digitare 1 se si vuole scaricare il file
1

AVVIO DEL DOWNLOAD DI roxanne.mp3 dal peer 160.80.148.15

INIZIO DOWNLOAD
nome file : roxanne.mp3
dimensione file: 4613617
numero chunk file: 9010
invio ack riuscito
-----------------
posizione in cui verrà salvato il file : Scaricati/roxanne.mp3
store
numero chunk già ricevuti di questo file: 0
file incompleto
**********
DOWNLOAD IN CORSO
**********
fine lettura da socket
transazione completata, file ricevuto senza errori
result inviato: ack

FINE DOWNLOAD
TEMPO IMPIEGATO PER IL DOWNLOAD = 12.035000
```

SUPERPEER

```
comando ricevuto=join
size=108
ricevuto filtro
RATING:1.492659
STAMPA LISTA PEER
elem [0] ==> IP= 160.80.146.107, RAT= 1.492659

...

aggiornato TTL dell'IP 160.80.146.107
comando ricevuto=whhs
size=11
ID=0
dati=roxanne.mp3
INIZIO CODICE WHOHAS
inserita la query
ricerca di : roxanne.mp3
lunghezza nome file:11
numero di peer connessi : 2
match del file roxanne.mp3 nel peer 160.80.148.15
IP trovati 1
ip 0 = 160.80.148.15

result inviato

aggiornato TTL dell'IP 160.80.146.107
comando ricevuto=stop
size=0
QUERY CANCELLATA DALLA LISTA
result inviato
...
```

7.1.2 Commento

In questo test è mostrato come il peer entra nella rete (calcolando il suo rating iniziale e facendo la join a un superpeer), compie una ricerca e il successivo download del file cercato.
Come si può vedere all'utente è fornito un menù testuale nel quale vengono fornite delle voci selezionabili inserendo da tastiera il numero relativo alla scelta.
Nel caso di una ricerca, dopo aver selezionato la giusta scelta nel menu, il

programma chiede anche il nome del file che si vuole scaricare inviando un messaggio di whohas al suo superpeer specificando il file cercato.
Il superpeer risponderà con i dati relativi ai peer a lui connessi e restituendo la lista dei peer che hanno il file.

```
numero di ip 1
LISTA IP RESTITUITI
0 = 160.80.148.15
```

A questo punto il peer rimane in attesa di altri risultati successivi fino a che l'utente non specifica uno stop (che nel frattempo è apparso nel menù di scelta). Una volta che l'utente ha bloccato la ricerca il peer inizia la fase di download, che nell'output sopra è compreso tra la stampa "INIZIO DOWNLOAD" e "FINE DOWNLOAD".
In questo esempio il file non era presente (neanche in parte) sul peer, e quindi è stato scaricato dal chunk 0. Se il file fosse già stato parzialmente ricevuto si avrebbe avuto una stampa del tipo: *numero chunk già ricevuti di questo file: 135*; e il download sarebbe stato eseguito esattamente uguale solo partendo da tale chunk.
Nell'esecuzione del superpeer si può vedere come esso ricevi e processi i comandi inviati dal peer. Quando riceve una whohas inserisce la query in una lista con i riferimenti delle ricerche in corso, effettua il controllo sui peer a lui connessi e risponde al peer.

7.2 Test switch peer-superpeer

7.2.1 output del test

SUPERPEER

```
ENTRATO IN ISSET UDP
aggiornato TTL dell'IP 160.80.146.107
comando ricevuto=ping
size=8
aggiornato rating (1.493180) dell'IP 160.80.146.107

INIZIO OPERAZIONE SWITCH
inviato messaggio di pong al peer
inviato leave al bootstrap
invio messaggio di leave dal SP a 160.80.148.15
```

```
FINE SUPERPEER
```

PEER

```
[...]

PEER:
ricevuto nwsp da: 160.80.148.15
comando ricevuto=nwsp
size=1

SUPERPEER:
register=1

ENTRATO IN ISSET UDP
comando ricevuto=join
size=108
ricevuto filtro
RATING:0.686023

STAMPA LISTA PEER
elem [0] IP= 160.80.148.15, RAT= 0.686023
result inviato


ENTRATO IN ISSET UDP
comando ricevuto=join
size=108
ricevuto filtro
RATING:1.493180

STAMPA LISTA PEER
elem [0] IP= 160.80.146.107, RAT= 1.493180
elem [1] IP= 160.80.148.15, RAT= 0.686023

result inviato
```

7.2.2 Commento

Il superpeer che riceve il ping da un peer con rating sufficentemente maggiore del suo, avvia l'operazione di switch. Quindi abbandona la rete come superpeer effettuando la leave al bootstrap e informa i peer (escluso quello che ha pingato) a lui connessi che sta abbandonando la rete informandoli dell'ip del nuovo superpeer, in questo caso solo 160.80.148.15. Il peer che deve deve diventare superpeer riceve il comando di "nwsp" e avvia il processo del superpeer, al quale si connetteranno i peer che erano connessi al superpeer che ha abbandonato la rete (160.80.146.107, 160.80.148.15).

7.3 Test ricerca

7.3.1 output del test

```
ENTRATO IN ISSET UDP
aggiornato TTL dell'IP 160.80.146.107
comando ricevuto=whhs
size=7
ID=0
dati=gdl.pdf

INIZIO CODICE WHOHAS
inserita la query
inoltro sulla rete di overlay
ricerca di : gdl.pdf
lunghezza nome file:7
numero di peer connessi : 1
IP trovati 0
result inviato

[...]

NUMERO SUPERPEER CONNESSI : 1
ENTRATO IN ISSET TCP
comando ricevuto=ack
size=5
ID=1
dati=;ipvicini
aggiungere nella try list: 160.80.128.69
INOLTRO WHOHAS
```

```
FINE ISSET TCP

[...]

ENTRATO IN ISSET UDP
comando ricevuto=ack
size= 9
ID= 1
dati= ippeer;ipvicini

[...]

ENTRATO IN ISSET UDP
aggiornato TTL dell'IP 160.80.146.107
comando ricevuto=stop
size=0
QUERY CANCELLATA DALLA LISTA
result inviato
```

7.3.2 commento

In questo test viene illustrato il caso in cui una ricerca viene inoltrata sulla rete d'overlay passando sia sulle connesioni TCP che UDP. Il messaggio di whohas viene quindi mandato al superpeer di riferimento del peer, il quale controlla la presenza del file sui propri peer, memorizza la query associata alla ricerca ed inoltra la ricerca sui suoi vicini con cui comunica in TCP.

```
INIZIO CODICE WHOHAS
inserita la query
inoltro sulla rete di overlay
ricerca di : gdl.pdf
lunghezza nome file:7
numero di peer connessi : 1
IP trovati 0
result inviato
```

Il superpeer dopo aver inoltrato la richiesta riceve le risposte dai vicini che conterranno gli ip in cui è stato matchato il file e gli ip dei vicini del superpeer che ha inviato la risposta. Gli indirizzi dei peer che hanno il file verranno inoltrati al peer che ha effettuato la ricerca mentre gli indirizzi dei superpeer verranno aggiunti alla try-list e successivamente contattati in udp

e memorizzati in done-list.
I vicini dei vicini che ricevono la whohas risponderanno sempre indicando i peer che hanno il file e i superpeer vicini.

```
ENTRATO IN ISSET UDP
comando ricevuto=ack
size= 9
ID= 1
dati= ippeer;ipvicini
```

Queste operazioni continuano finchè non viene ricevuto il comando di stop dal peer o viene raggiunto il numero massimo di peer o superpeer contattati.

7.4 Limitazioni riscontrate

Una delle principali limitazioni riscontrate è che, usando i filtri di bloom per memorizzare le informazioni sui file dei peer, non è possibile cercare un file di cui non si conosca il nome esatto con la propria estensione. Inoltre il superpeer non sà esattamente quali file sono presenti nei peer connessi, però può interrogare i filtri associati ad ogni peer, in modo tale da sapere se il file cercato è presente o no.
Un altro problema nell'uso dei filtri di bloom è quello dei cosiddetti "falsi positivi", ossia che l'interrogazione del filtro potrebbe avere esito positivo nonostante il file non sia presente. Questo problema però è meno importante del primo, in quanto la probabilità di errore risulta essere trascurabile (minore del 10%) finchè gli elementi condivisi sono meno di 900 circa (come si puo vedere in figura 16).

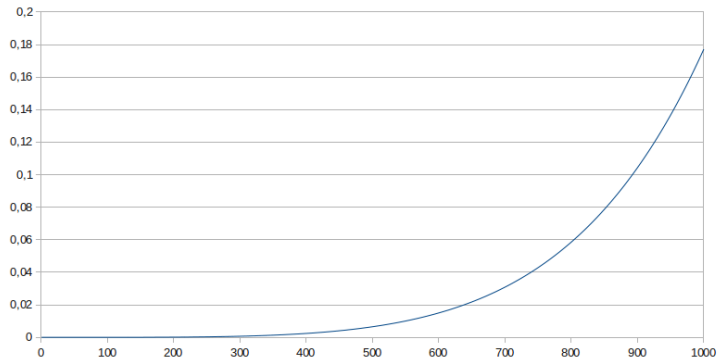


Figure 16: Probabilità di falsi positivi al variare del numero degli elementi condivisi

Nonostante è problematico cercare file di cui non si conosca il nome esatto, l'uso dei filtri di bloom garantisce efficenza e scarsa occupazione in memoria. Inoltre, utilizzando un numero di funzioni hash maggiore la probabilità di falsi positivi diminuirebbe ulteriormente.

Un ulteriore limitazione è che non è possibile per un peer svolgere ricerche e download in parallelo, in quanto sono state implementate come operazioni seriali. Ovviamente i superpeer possono gestire più ricerche in parallelo come specificato nel capitolo relativo alla fase di lookup.

Non è possibile contattare computer dietro a NAT o firewall in quanto non sono raggiungibili a meno di un settaggio specifico di router e firewall.

Nonostante il sistema sia parzialmente distribuito, in caso di caduta del server di bootstrap, è impossibile per nuovi peer l'entrata nella rete anche se quest'ultima potrebbe continuare a svolgere il suo lavoro normalmente.

List of Figures