Reti Peer to Peer

Specifiche cooperative
Gianluca Mazzini

Comunicazioni Peer-to-Peer

Si consideri una rete con un server ed N client tutti interessati al download di un file composto da F bit. Si ipotizzi che la velocità di upload del server sia U e la velocità di download del generico client j sia Dj. Un parametro prestazionale significativo è dato dal tempo necessario per terminare il trasferimento dell'intero file a tutti i client e può essere valutato come T=max[NF/Us, F/Dmin] dove Dmin è la velocità di download minima tra tutti i client e la formulazione considera la necessità di trasferire a tutti i client l'intero file mediante upload alla rete da parte del server e la necessità di effettuare il download da parte del client più lento. All'aumentare di N si ha che T->o(N) per cui la prestazione peggiora con il numero di client interessati. Volendo mantenere costante la prestazione indipendentemente dal numero di client interessati, occorre aumentare la velocità di upload incrementando le caratteristiche della linea fisica o mettendo più linee fisiche in parallelo. Il caso con più linee fisiche poste in parallelo può essere realizzato anche mediante linee geograficamente indipendenti qualora siano disponibili server contenenti gli stessi contenuti e quindi opportunamente sincronizzati.

Si consideri altresì che ogni client è connesso alla rete mediante una linea bidirezionale e che tale linea è tipicamente asimmetrica, con velocità Dj in downlink e velocità Uj in uplink. La asimmetria è dovuta alla razionalizzazione tra il ruolo di fruitore legato al downlink e di fornitore legato al uplink, ove tipicamente e storicamente ogni client vede un ruolo di fruitore molto maggiore rispetto a quello di fornitore. A titolo di esempio parametri caratteristici di asimmetria nelle ADSL vanno da 4:1 sino a 40:1. Nel peer-to-peer (P2P) il client condivide il file ed in generale i flussi che ha a disposizione con altri client, giocando contemporaneamente sia il ruolo di client, rispetto alla sorgente e ad altri client, sia il ruolo di server a favore di altri client. In generale quindi ogni fruitore diventa anche fornitore e quindi ogni nodo diviene pari di un altro, realizzando appunto il peer e consentendo quindi comunicazioni tra i peer, appunto il peer-to-peer. Sebbene l'analisi di una simile tecnica dovrebbe partire dalla strategia architetturale con cui il P2P avviene, ci limitiamo in questa fase a delimitare un ambito ideale, dove tutti i peer collaborano indipendentemente dalla strategia che adottano. Si ha quindi che la sorgente è idealmente impegnata a trasferire alla rete una sola copia del file e che tutti i client presenti mettono a disposizione della ridistribuzione la propria capacità di uplink T=max[F/Us, F/Dmin, NF/(Us+U1+U2+...+UN)].

Risulta interessante osservare che all'aumentare di N prevale la sola terza componente, che la componente di upload della sorgente può essere ritenuta trascurabile e che rispetto ad una velocità media di uplink dei client il denominatore della terza componente risulta crescere linearmente rispetto a N. Da queste considerazioni discende che T->o(1) e questo elemento rappresenta la forza di una rete P2P, cioè l'indipendenza della prestazione dal numero di client presenti, ove la rete può idealmente gestire in modo automatico e assolutamente scalabile la dimensione dei partecipanti.

In una rete P2P i contenuti possono essere file oppure stream. Il caso dello stream è più complicato da gestire in quanto il contenuto non è mai disponibile nella sua interezza e tende ad invecchiare velocemente, nel senso che esiste una finestra di fruibilità da quando il contenuto è stato generato a quando può essere riprodotto. La finestra di fruibilità dipende fortemente dal contenuto ed è bilanciata da meccanismi di buffering i grado di consentire la continuità di visualizzazione rispetto al jitter presente sulla rete piuttosto che alla ritrasmissione di particolari contenuti. La situazione più stringente è relativa ad un live dove tutti i soggetti fruitori devono essere sincronizzati.

Una problematica è data dalla disponibilità dei peer. Mentre in un approccio tradizionale abbiamo un server sempre disponibile ed un client attivo solo quando è intenzionato a fruire del servizio, nel P2P tutti i client dovrebbero sempre essere disponibili per fungere la funzione di server e quindi per mettere a disposizione la propria capacità di uplink e i contenuti già acquisiti. Occorre quindi identificare delle strategie che invoglino il client a rimanere attivo oltre alla avvenuta acquisizione dei contenuti a cui è interessato. Tali strategie possono ad esempio modificare la velocità di download o la pletora di contenuti a disposizione in funzione della continuità di presenza.

Strategie P2P

Approccio alla Napster: directory centralizzata

Questa strategia prevede che i file di interesse siano già residenti nei peer e che vi sia un sistema di directory sul quale ogni peer si registra, ogni peer comunica la lista dei file che pone in condivisione e ogni peer interessato ad un download effettui una richiesta relativa a chi possiede tale file, ottenendo la lista degli IP dei peer dai quali può ottenere il file. Il download avviene da parte dell'interessato che si comporta da client nei confronti di un singolo peer selezionato, contenente il file, che si comporta da server. Il sistema di directory possiede le informazioni di tutti i file presenti nel sistema distribuito. Considerando che mediante il P2P possono essere scambiati contenuti protetti da copyright, storicamente questo approccio ha avuto parecchi problemi di natura legale in quanto identifica in modo chiaro chi metteva a disposizione contenuti, mediante l'IP, e quale tipologia di contenuti venivano messi a disposizione, mediante la relativa descrizione.

Approccio alla Gnutella: directory distribuita

Questa strategia implementa un sistema di directory distribuito e non centralizzato come avviene in Napster. L'idea è di realizzare una rete di peer per la funzione di ricerca, in cui la ricerca di un contenuto viene inviata dall'interessato ai peer vicini, cioè ai peer noti, ed ogni peer propaga a sua volta la richiesta e le relative risposte, attraverso una rete di ricerca che risulta virtuale. La rete virtuale si forma a partire da alcuni peer noti che vengono interrogati in fase di inizializzazione e ogni peer può scoprire, mediante la rete virtuale, altri peer ed eventualmente allargare la lista dei propri vicini. Lo scambio del file, una volta effettuata la ricerca, avviene in modo analogo a quello del meccanismo a directory centralizzata.

Approccio alla Kazaa: directory gerarchica

Questa strategia implementa contemporaneamente sia la parte di directory centralizzata che di directory distribuita. L'idea è organizzare la rete in modo gerarchico su due livelli, livello dei peer e livello dei supernodi dove i peer utilizzano i supernodi come elementi di directory centralizzata e le ricerche vengono effettuate secondo il meccanismo della directory distribuita nella sola rete dei supernodi. Le ricerche risultano più veloci in quanto la rete dei supernodi è più contenuta rispetto alla rete dei peer. Un supernodo è un peer eletto a tale funzione. Lo scambio del file, una volta effettuata la ricerca, avviene in modo analogo a quello del meccanismo a directory centralizzata, sebbene con Kazaa sia stato implementato un meccanismo di download multiplo da più peer, che non viene analizzato in questo contesto.

Approccio alla Bittorrent: parti parallele

Questa strategia è dedicata alla ottimizzazione del trasferimento di file verso un peer richiedente partendo da tanti peer che contengono parti del file. Ogni file, infatti, viene infatti suddiviso in un insieme di parti, tipicamente uguali con eccezione dell'ultima, presenti in differente misura nei vari peer che a loro volta sono interessati quel particolare file. Lo stato di disponibilità di ciascuna parte di un file per ogni peer viene gestito da un tracker, un nodo infrastrutturale su cui i peer si registrano, sul quale aggiornano le informazioni relative alle parti possedute per ogni file, e sul quale effettuano ricerche ottenendo informazioni su chi possiede le varie parti di un file. Ogni peer può richiedere e ricevere in parallelo diverse parti da diversi peer. Quando il sistema inizia la condivisione di un file le parti disponibili sono poche, poi queste aumentano e la loro scelta e messa a disposizione avviene secondo meccanismi che possono cercare di rendere omogenea la disponibilità delle parti nella rete, ad esempio selezionando una maggiore distribuzione di quelle rare e comunque utilizzando meccanismi casuali per rendere il più possibile equa ed omogenea la loro disponibilità all'interno del sistema.

Protocolli semplificati di esempio

Elementi generali

Le finalità sono di natura didattica, per cui tutte le scelte sono orientate alla massima semplicità e leggibilità, piuttosto che alla efficienza. Le comunicazioni sono realizzate mediante comunicazione TCP. Nel seguito B indica Byte. Ogni pacchetto inizia con un identificativo alfanumerico di dimensione 4B. Ogni peer e comunque ogni elemento di rete è identificato dalla coppia indirizzo internet IP e porta P dove l'indirizzo IP è sia versione 6 (v6) che versione 4 (v4) riportato in formato %03d.%03d.%03d.%03dl%04x:%04x:%04x:%04x:%04x:%04x:%04x:%04x [55B] e porta P in formato %05d [5B]. Si suppone che l'indirizzamento sia privato e raggiungibile direttamente da tutti i peer senza vincoli di firewalling o problematiche di traslazione di indirizzi. Per v4 utilizziamo 172.16.A.B/16 e per v6 FC00::A:B/64, dove A è il numero del gruppo di lavoro e B è il numero del componente del gruppo, secondo un ordinamento deciso dal gruppo stesso.

Ogni file è identificato univocamente, o quasi, da una funzione di checksum di tipo md5, cioè rispondente alla RFC 1321, che partendo da una stringa di lunghezza arbitraria, il file appunto, genera un indicativo di 32B con rappresentazione esadecimale, ad esempio MD5("The quick brown fox jumps over the lazy dog")=9e107d9d372bb6826bd81d3542a419d6. La probabilità che due file abbiano lo stesso indicativo md5 è sufficientemente bassa da essere trascurata in questo ambito.

Ogni pacchetto del protocollo è illustrato in due fasi: invio e risposta. Per ogni fase viene indicata la coppia indirizzo IP e porta da cui la comunicazione parte e indirizzo IP e porta a cui la comunicazione è indirizzata. Qualora la porta sia scelta casualmente viene riportata la dicitura RND. I campi dei pacchetti sono riportati indicando il loro nome, mettendolo tra virgolette nel caso si tratti di una stringa e separando i campi con il simbolo punto solo per mera rappresentazione.

Il simbolo cancelletto viene utilizzato come acronimo di numero di evenienze e il formato di rappresentazione è sempre %0xd dove x è specificato nella dimensione del campo [xB]. Nel caso una parte del pacchetto venga ad essere iterata mediante un indice viene utilizzata la parentesi graffa come iteratore, indicando alla sua chiusura gli estremi di interazione sull'indice in formato (indice = partenza .. arrivo). Gli indici eventuali sono indicati mediante un underline e possono essere gerarchici.

Ogni comunicazione può essere effettuata sia in v4 che in v6 e la scelta sulla tipologia di comunicazione viene effettuata in modo casuale con equa allocazione su v4 e su v6 (50%).

Nel seguito viene usata una rappresentazione dei formati dei pacchetti secondo la seguente notazione. In testa vengono riportati i soggetti tra i quali si svolge la comunicazione, indicando IP e porta. Viene poi riportata la direzione della comunicazione, dove > implica client a sinistra e < client a destra. Le stringhe esplicite sono poste tra due simboli ". La lunghezza di un campo è riportata tra parentesi quadre [] in

Byte. La successione di stringhe da concatenare è riportata mediante il simbolo punto . che ha solo significato di concatenazione ma che non deve comparire nella stringa effettiva. La parentesi graffa è utilizzata per indicare una iterazione secondo un indice che viene riportato con il suo range alla chiusura della iterazione con la chiusura della parentesi graffa, tra parentesi graffa, ad esempio {X_i}(i=1..3)

Directory centralizzata

L'approccio a directory centralizzato vede una serie di rapporti diretti tra il peer e la directory, tipicamente per poter indicare e ricercare i file. Ogni comunicazione ha una azione di richiesta ed una corrispondente risposta, possibile in quanto l'interlocutore del peer è sempre unico e certo, nella fattispecie è il sistema di directory o un altro peer.

Login

Ogni peer con indirizzo IPP2P, abile a fornire contenuti sulla porta PP2P, deve registrarsi nel sistema di directory mediante un processo di login, indicando il proprio indirizzo e la relativa porta di comunicazione, che viene posta in ascolto. Il sistema di directory è in ascolto all'indirizzo IPD sulla porta 3000. Il processo di login ritorna un codice di sessione SessionID di 16B, composto da una stringa di caratteri random generato da numeri e lettere maiuscole scelti casualmente. Qualora il peer sia già registrato viene ritornato il SessionID, qualora altresì vi siano problematiche per cui la registrazione risulti non possibile il codice di ritorno è "0000000000000000", codice che si ritiene a probabilità quasi nulla come SessionID. SessionID non dipende dal tipo di comunicazione v4 o v6 utilizzato.

```
IPP2P:RND <> IPD:3000
      > "LOGI"[4B].IPP2P[55B].PP2P[5B]
      < "ALGI"[4B].SessionID[16B]
```

Aggiunta

Ogni peer registrato può aggiungere un file in un qualsiasi momento, mettendolo a disposizione di tutta la rete P2P. Il file è identificato da una stringa di 100B che lo descrive e sul quale è possibile effettuare una ricerca e da un identificativo md5 unico. La stringa è completata nella sua lunghezza con spazi. In caso di descrizioni differenti relative allo stesso identificativo md5, l'ultima indicata sostituisce le precedenti, indipendentemente dal peer che la ha indicata. Peer differenti possono ospitare lo stesso file. Il pacchetto di risposta riporta quante versioni del file con lo stesso md5 sono presenti nella directory #copy, dopo l'aggiunta. Nel caso venga aggiunto dallo stesso peer un file con identificativo già inserito dallo stesso peer, viene solo aggiornato il nome.

```
IPP2P:RND <> IPD:3000
      > "ADDF"[4B].SessionID[16B].Filemd5[32B].Filename[100B]
      < "AADD"[4B].#copy[3B]
```

Rimozione

Ogni peer registrato può rimuovere un file che ha messo a disposizione dalla directory. La rimozione avviene specificando l'identificativo md5 del file da eliminare. Il pacchetto di risposta riporta quante versioni del file con lo stesso md5 #copy sono presenti nella directory, dopo la rimozione. Nel caso venga eliminato un file non presente non viene effettuata alcuna operazione sulla directory e #copy è posto a 999 per indicare al peer la assenza.

```
IPP2P:RND <> IPD:3000
      > "DELF"[4B].SessionID[16B].Filemd5[32B]
      < "ADEL"[4B].#copy[3B]
```

Ricerca

La ricerca di un file avviene indicando una stringa di ricerca di 20B. Tale stringa viene utilizzata per effettuare una ricerca case insensitive su tutti i titoli presenti, trovando tutti gli md5 relativi ad ogni occorrenza della stringa stessa. Sono possibili più riscontri di differenti titoli, con differente identificativo relativo alla stessa stringa di ricerca e per ogni identificativo sono possibili più peer che lo mettano a disposizione. La risposta è quindi articolata nel numero complessivo di identificativi md5 #idmd5 dove per ognuno di essi viene riportato l'identificativo md5, il nome del file e il numero di copie presenti #copy, mentre per ogni copia viene riportato l'IP e la porta del peer. Nel caso in cui non vi sia alcun riscontro si ha #idmd5=0. Una stringa di ricerca composta dal solo carattere * rappresenta il match con tutte le stringhe e permette di avere informazioni sull'intero contenuto della directory.

```
IPP2P:RND <> IPD:3000
      > "FIND"[4B].SessionID[16B].Ricerca[20B]
      < "AFIN"[4B].#idmd5[3B].{Filemd5_i[32B].Filename_i[100B].#copy_i[3B].
      {IPP2P_i_j[55B].PP2P_i_j[5B]}}(j=1..#copy_i)}(i=1..#idmd5)
```

Download

Il peer interessato ad effettuare il download di un file ha effettuato la ricerca ed ha a disposizione un ampio potenziale di peer dai quali procedere con il download, ma procede a selezionare un solo corrispondente rispetto al quale effettuare il download. In questo contesto supponiamo che la scelta venga effettuata dall'utente. Il peer interessato al download si mette in diretta comunicazione con il peer selezionato grazie alla conoscenza di IPP2P e di PP2P. Ogni peer che partecipa alla rete P2P funge da server su tali indirizzi a partire dal login e sino al logout. Il download è organizzato in #chunk e per ognuno di essi viene specificata la lunghezza ed inviato i relativi dati in formato nativo. Infine per mantenere traccia delle attività di download viene segnalata l'azione al sistema di directory, indicando il file mediante l'identificativo md5. La directory risponde riportando il numero di download complessivi avvenuti per quel contenuto #download.

```
IPP2P:RND <> IPP2P:PP2P
      > "RETR"[4B].Filemd5[32B]
      < "ARET"[4B].#chunk[6B].{Lenchunk_i[5B].data[LB]}(i=1..#chunk)

IPP2P:RND <> IPD:3000
      > "DREG"[4B].SessionID[16B].Filemd5[32B]
      < "ADRE"[4B].#download[5B]
```

Logout

Ogni peer registrato può uscire dalla condivisione comunicando tale intenzione alla directory. La directory rimuove tutti i file legati a tale peer e restituisce il numero di file eliminati #delete che deve corrispondere al numero dei file che il peer aveva attivato nella condivisione. Con il logout il peer smette di ascoltare sull'IP e la porta specificata per le azioni di P2P.

```
IPP2P:RND <> IPD:3000
      > "LOGO"[4B].SessionID[16B]
      < "ALGO"[4B].#delete[3B]
```

Directory distribuita

L'approccio a directory distribuita vede i peer capaci di ricercare direttamente dai peer stessi, mediante un meccanismo di query flooding che prevede la ritrasmissione della query di ricerca da ogni peer a tutti i propri vicini, con eccezione del vicino da cui arriva query. Le query si propagano per la rete P2P e si comportano come un messaggio broadcast su una rete virtuale.

Per evitare loop sono possibili vari meccanismi, tra questi due sono i più semplici: stato nel pacchetto, in cui il loop è evitato controllando l'IP dei peer attraverso i quali la query è già transitata e appendendo tale IP al pacchetto; stato nel peer, in cui il loop è evitato controllando l'identificativo del pacchetto nel peer e ricordandolo per un certo tempo. Tra questi due approcci si procede con lo stato nel peer, che sebbene lievemente più complesso da gestire ha il vantaggio di operare con pacchetti a struttura, e quindi dimensione, prefissata.

Ogni peer conosce un certo numero di vicini, al minimo, all'accensione conosce un definito numero di vicini che supponiamo presenti e funzionanti. Questi peer, a volte chiamati root, costituiscono una tipologia di nodi atti alla creazione della rete virtuale e solitamente non mettono a disposizione file ma altresì garantiscono in modo continuativo la propria presenza. Consideriamo in seguito, per semplicità, che i nodi root siano nodi uguali a tutti gli altri peer.

Le funzioni previste in un meccanismo a directory distribuita sono due: la ricerca di contenuti e la ricerca di vicini. Essendo entrambe queste funzioni elementi di ricerca, si procede cercando di sfruttare la ricerca di contenuti per implementare la ricerca di vicini.

Ricerca

Ogni pacchetto di query è identificato da Pktid di 16B, composto da una stringa di caratteri random generato da numeri e lettere maiuscole, costruito dal peer che necessita di ricercare un contenuto, ed inviato a tutti i peer vicini. Il Pktid di una query ricevuta viene confrontato con i Pktid delle query già arrivate e ritrasmesse ad un peer, la ritrasmissione della query avviene a tutti i vicini del peer, con eccezione del vicino che la ha inviata. Il Pktid viene mantenuto per un tempo pari a 300s, tempo entro il quale si suppone che la query abbia attraversato l'intera rete e non venga quindi più propagata.

La fase di invio è qui disgiunta dalla fase di risposta, in quanto la query parte indirizzata a tanti vicini mentre la risposta può essere generata da un qualsiasi peer che ha ricevuto la query ed ha verificato la corrispondenza, per cui la query di ricerca contiene in esplicito anche la coppia indirizzo IP e porta sulla quale il peer che ha effettuato la richiesta rimane in ascolto.

Per evitare che una query broadcast si propaghi in modo troppo diffuso in una rete virtuale di grosse dimensioni, si inserisce un time to live (TTL), inizializzato ad un valore che rappresenta il numero massimo di peer attraverso i quali la query può passare e decrementato di una unità ad ogni passaggio, sino ad essere eliminato dal peer che lo riceve con TTL=1. TTL [2B] è rappresentato in decimale ASCII da 00 a 99.

La ricerca di un file avviene indicando una stringa di ricerca di 20B. Tale stringa viene utilizzata per effettuare una ricerca case insensitive su tutti i titoli presenti, trovando ogni occorrenza della stringa stessa. In ogni peer sono possibili più riscontri di differenti titoli con differente identificativo md5 relativi alla stessa stringa di ricerca. Ogni peer che identifica un match provvede a mandare al peer che ha effettuto la ricerca, all'indirizzo e alla porta specificata nella richiesta, l'identificativo md5 del file ed il nome del file.

Il peer che ha generato la ricerca rimane in attesa, colleziona e mostra tutti i pacchetti ricevuti per consentire all'utente di effettuare, in un qualsiasi momento, la selezione di un peer dal quale effettuare il download. Considerando che lo stato relativo al Pktid viene mantenuto per 300s dalla ricezione da parte di ciascun nodo, assumiamo 300s come finestra di ascolto entro la quale sono attese risposte alla query di ricerca. Qualora vi siano più riscontri ad una ricerca in un singolo peer vengono prodotte più risposte indipendenti esattamente come avviene se i riscontri avvengono in peer differenti.

```
IPP2P:RND > IPP2P_i:PP2P_i
      > "QUER"[4B].Pktid[16B].IPP2P[55B].PP2P[5B].TTL[2B].Ricerca[20B]

IPP2P_j:RND > IPP2P:PP2P
      > "AQUE"[4B].Pktid[16B].IPP2P[55B].PP2P[5B].Filemd5[32B].Filename[100B]
```

Vicini

Per trovare i vicini si utilizza un approccio analogo a quello della ricerca di un contenuto, nel quale non si specifica alcuna stringa di ricerca. Per limitare il numero di vicini si utilizza un valore di TTL basso e si limita ad un numero prefissato il massimo numero di vicini da considerare. Valori di esempio, ma comunque configurabili, sono TTL=2 e max 3 vicini per peer. E' importante osservare che per evitare che vi siano aree della rete non connesse il numero di nodi noti alla partenza per ogni peer deve essere maggiore di uno.

```
IPP2P:RND > IPP2P_i:PP2P_i
      > "NEAR"[4B].Pktid[16B].IPP2P[55B].PP2P[5B].TTL[2B]

IPP2P_j:RND > IPP2P:PP2P
      > "ANEA"[4B].Pktid[16B].IPP2P_j[55B].PP2P_j[5B]
```

Download

Il peer interessato ad effettuare il download di un file ha effettuato la ricerca ed ha a disposizione un ampio potenziale di peer dai quali procedere con il download, ma procede a selezionare un solo corrispondente rispetto al quale effettuare il download. In questo contesto supponiamo che la scelta venga effettuata dall'utente. Il peer interessato al download si mette in diretta comunicazione con il peer selezionato grazie alla conoscenza di IPP2P e di PP2P. Ogni peer che partecipa alla rete P2P funge da server su tali indirizzi. Il download è organizzato in #chunk e per ognuno di essi viene specificata la lunghezza ed inviati i relativi dati in formato nativo.

```
IPP2P:RND <> IPP2P:PP2P
      > "RETR"[4B].Filemd5[32B]
      < "ARET"[4B].#chunk[6B].{Lenchunk_i[5B].data[LB]}(i=1..#chunk)
```

Directory gerarchica

L'approccio a directory gerarchica vede la combinazione dell'approccio directory distribuita su un sotto insieme dei peer denominati supernodi per effettuare la ricerca e dell'approccio directory centralizzata per popolare la conoscenza di ogni supernodo.

Ogni peer è associato ad un supernodo, selezionato mediante una azione di ricerca di vicini, a cui rispondono solo i supernodi e su cui la selezione è effettuata a caso. La procedura di ricerca dei supernodi avviene come nel caso del sistema a directory distribuita dei peer, con l'eccezione che rispondono solo i supernodi mentre effettuano la ritrasmissione tutti i peer.

Il peer procede poi ad utilizzare il supernodo come elemento di directory centralizzata, richiedendo al supernodo quali peer hanno la disponibilità di file relativi al parametro di ricerca. Il supernodo di riferimento del peer che effettua la ricerca utilizza un approccio di query a directory distribuita sulla rete dei soli supernodi, cercando quindi tra tutte le directory centralizzate tutte le occorrenze relative al parametro di ricerca, aggregando tutte le risposte pervenute dai vari supernodi e restituendo infine al peer interessato un singolo pacchetto con tutte le risposte.

Sebbene ogni peer possa essere eletto a supernodo, in questo contesto si ipotizza che all'atto della accensione ogni nodo sia già definito come peer o come supernodo. In ogni caso un supernodo si comporta anche da peer, utilizzando se stesso come supernodo.

Supernodo

Per identificare il supernodo a cui fare riferimento si utilizza una procedura analoga a quella di identificazione dei vicini per il sistema a directory distribuita con il vincolo che rispondono solo i supernodi ma che tutti i nodi, sia peer che supernodi, ritrasmettono ai propri vicini la query. Se arrivano risposte da più supernodi all'interno di una finestra di ascolto di risposta di 20s, il peer seleziona il supernodo in modo casuale tra tutti i candidati pervenuti. Il TTL di partenza dipende dalla dimensione della rete, si suggerisce TTL=4. La stessa procedura viene anche utilizzata da un supernodo per identificare gli altri supernodi che compongono la rete virtuale dei supernodi. Ogni realizzazione può selezionare un numero di supernodi vicini differente.

```
IPP2P:RND > IPP2P_i:PP2P_i
      > "SUPE"[4B].Pktid[16B].IPP2P[55B].PP2P[5B].TTL[2B]

IPP2P_j:RND > IPP2P:PP2P
      > "ASUP"[4B].Pktid[16B].IPP2P[55B].PP2P[5B]
```

Login

Ogni peer con indirizzo IPP2P, abile a fornire contenuti sulla porta PP2P, deve registrarsi in un supernodo del sistema mediante un processo di login, indicando il proprio indirizzo e la relativa porta di comunicazione, che viene posta in ascolto. Il supernodo è in ascolto all'indirizzo IPS sulla porta 3000. Il processo di login ritorna un codice di sessione SessionID di 16B, composto da una stringa di caratteri random generato da numeri e lettere maiuscole scelti casualmente. Qualora il peer sia già registrato viene ritornato il SessionID, qualora altresì vi siano problematiche per cui la registrazione risulti non possibile il codice di ritorno è "0000000000000000", codice che si ritiene a probabilità quasi nulla come SessionID.

```
IPP2P:RND <> IPS:3000
      > "LOGI"[4B].IPP2P[55B].PP2P[5B]
      < "ALGI"[4B].SessionID[16B]
```

Aggiunta

Ogni peer registrato su un supernodo con indirizzo IPS può aggiungere un file in un qualsiasi momento, mettendolo a disposizione di tutta la rete P2P. Il file è identificato da una stringa di 100B che lo descrive e sul quale è possibile effettuare una ricerca e da un identificativo md5 unico. In caso di descrizioni differenti relative allo stesso identificativo md5, l'ultima indicata sostituisce le precedenti, indipendentemente dal peer che la ha indicata. Peer differenti possono ospitare lo stesso file. Non viene inserita alcuna risposta in quanto, a differenza dell'approccio a directory centralizzata, il singolo supernodo non ha informazioni complessive sullo stato del numero di istanza dello stesso file presenti nel sistema.

```
IPP2P:RND > IPS:3000
      > "ADFF"[4B].SessionID[16B].Filemd5[32B].Filename[100B]
```

Rimozione

Ogni peer registrato su un supernodo può rimuove un file che ha messo a disposizione dell'intero sistema. La rimozione avviene specificando l'identificativo md5 del file da eliminare. Nel caso si tenti di eliminare un file non presente per quel peer non viene effettuata alcuna operazione sul supernodo.

```
IPP2P:RND > IPS:3000
      > "DEFF"[4B].SessionID[16B].Filemd5[32B]
```

Logout

Ogni peer registrato può uscire dalla condivisione comunicando tale intenzione al supernodo che provvede a rimuovere tutti i file legati a tale peer e restituisce il numero di file eliminati #delete, che devono corrispondere al numero dei file che il peer aveva attivato nella condivisione. Con il logout il peer smette di ascoltare sull'IP e la porta specificata per le azioni di P2P.

```
IPP2P:RND <> IPS:3000
      > "LOGO"[4B].SessionID[16B]
      < "ALGO"[4B].#delete[3B]
```

Ricerca

La ricerca di un file avviene indicando una stringa di ricerca di 20B. Tale stringa viene utilizzata per effettuare una ricerca case insensitive su tutti i titoli presenti nella rete P2P, trovando ogni occorrenza della stringa stessa.

La ricerca è un servizio che viene richiesto al supernodo di afferenza, e viene effettuata sia sul supernodo stesso che su tutti gli altri supernodi attraverso un meccanismo di query analogo a quello

dell'approccio a directory distribuita, propagato sulla rete virtuale dei soli supernodi, dal supernodo di riferimento. Anche in questo caso si considera per il Pktid un tempo di validità di 20s.

Ogni supernodo che identifica nella propria directory un match provvede a mandare al supernodo che ha indetto la ricerca, all'indirizzo e alla porta specificata nella richiesta, l'identificativo md5 del file, il nome del file e la coppia indirizzo e porta del peer che ha messo a disposizione il contenuto. Considerando che lo stato relativo al PktD viene mantenuto per 20s dalla ricezione da parte di ciascun nodo, assumiamo 20s come finestra di ascolto entro la quale sono attese risposte alla query di ricerca.

```
IPP2P:RND > IPP2P_i:PP2P_i
      > "QUER"[4B].Pktid[16B].IPP2P[55B].PP2P[5B].TTL[2B].Ricerca[20B]

IPP2P_j:RND > IPP2P:PP2P
      > "AQUE"[4B].Pktid[16B].IPP2P[55B].PP2P[5B].Filemd5[32B].Filename[100B]
```

Sono possibili più riscontri di differenti titoli con differente identificativo md5 relativi alla stessa stringa di ricerca e per ogni identificativo sono possibili più peer che lo mettano a disposizione. La risposta collezionata dal supernodo di riferimento, in un tempo di 20s, è quindi articolata nel numero complessivo di identificativi come del file e il numero di md5 #idmd5 dove per ognuno di essi viene riportato l'identificativo md5, in copie presenti #copy, mentre per ogni copia viene riportato l'IP e la porta del peer. Nel caso in cui non vi sia alcun riscontro si ha #idmd5=0. Se più file con lo stesso md5 hanno nome di file differente, in quanto provenienti da differenti supernodi, si utilizza un solo nome tra quelli possibili, lasciando tale scelta libera nella implementazione.

```
IPP2P:RND <> IPS:3000
      > "FIND"[4B].SessionID[16B].Ricerca[20B]
      < "AFIN"[4B].#idmd5[3B].{Filemd5_i[32B].Filename_i[100B].#copy_i[3B].
      {IPP2P_i_j[55B].PP2P_i_j[5B]}(j=1..#copy_i)}(i=1..#idmd5)
```

Download

Il peer interessato ad effettuare il download di un file ha effettuato la ricerca ed ha a disposizione un ampio potenziale di peer dai quali procedere con il download, ma procede a selezionare un solo corrispondente rispetto al quale effettuare il download. In questo contesto supponiamo che la scelta venga effettuata dall'utente. Il peer interessato al download si mette in diretta comunicazione con il peer selezionato grazie alla conoscenza di IPP2P e di PP2P. Ogni peer che partecipa alla rete P2P funge da server su tali indirizzi. Il download è organizzato in #chunk e per ognuno di essi viene specificata la lunghezza ed inviati i relativi dati in formato nativo.

```
IPP2P:RND <> IPP2P:PP2P
      > "RETR"[4B].Filemd5[16B]
```

```
      < "ARET"[4B].#chunk[6B].{Lenchunk_i[5B].data[LB]}(i=1..#chunk)
```

Parti parallele

Il sistema a parti parallele è analogo al meccanismo a directory centralizzata in quanto ogni peer ha la necessità di comunicare al tracker le parti che mette a disposizione, aggiornandone periodicamente lo stato. Ogni peer può mettere a disposizione nuovi file ma si vincola a mettere a disposizione tutte le parti dei file che ha richiesto. Il download avviene richiedendo più parti in parallelo da differenti peer. La scelta dei peer da utilizzare per il download è effettuata dal peer ricevente, sulla base della conoscenza aggiornata delle parti possedute da ciascun peer, con un meccanismo che privilegia le parti meno presenti e che comunque sceglie a caso tra i peer a parità di priorità. I peer forniscono al tracker un aggiornamento periodico della situazione delle proprie parti e richiedono la situazione aggiornata delle parti relative ai file di cui stanno effettuando il download.

Login

Ogni peer con indirizzo IPP2P, abile a fornire contenuti sulla porta PP2P, deve registrarsi sul tracker mediante un processo di login, indicando il proprio indirizzo e la relativa porta di comunicazione, che viene posta in ascolto. Il tracker è in ascolto all'indirizzo IPD sulla porta 3000. Il processo di login ritorna un codice di sessione SessionID di 16B, composto da una stringa di caratteri random generato da numeri e lettere maiuscole scelti casualmente. Qualora il peer sia già registrato viene ritornato il SessionID, qualora altresì vi siano problematiche per cui la registrazione risulti non possibile il codice di ritorno è "0000000000000000", codice che si ritiene a probabilità quasi nulla come SessionID. Con il login il peer si pone in ascolto all'indirizzo IPP2P e sulla posta PP2P.

```
IPP2P:RND <> IPT:3000
      > "LOGI"[4B].IPP2P[55B].PP2P[5B]
      < "ALGI"[4B].SessionID[16B]
```

Aggiunta

Ogni peer registrato può mettere a disposizione un file in un qualsiasi momento comunicandolo al tracker. Il file è identificato da una stringa di 100B che lo descrive e sul quale è possibile effettuare una ricerca che dal relativo md5 ottenuto dal file aggiungendo in coda il proprio indirizzo IPv4+IPv6 in modo da generare un identificativo unico per ogni sorgente, correlato alle dimensioni delle parti. Ogni peer che mette a disposizione un contenuto comunica sia la dimensione complessiva del file che la dimensione della singola parte. La dimensione tipica, ma non vincolante, di una parte è 256KB=262144B. Il numero delle parti è tipicamente #part=supint[LenFile/LenPart].

```
IPP2P:RND <> IPT:3000
      > "ADDR"[4B].SessionID[16B].LenFile[10B].LenPart[6B].
      Filename[100B].Filemd5_i[32B]
      < "AADR"[4B].#part[8B]
```

Ricerca

La ricerca di un file è un processo che avviene in due fasi. La prima fase serve per identificare il file a cui il peer è effettivamente interessato mentre la seconda fase serve per avere la situazione delle parti su ogni peer e viene ripetuta periodicamente durante il download.

La prima fase avviene indicando una stringa di ricerca di 20B. Tale stringa viene utilizzata per effettuare una ricerca case insensive su tutti i titoli presenti, trovando ogni occorrenza della stringa stessa. Sono possibili più riscontri di differenti titoli con differente md5 relativi alla stessa stringa di ricerca. La risposta è quindi articolata nel numero complessivo di identificativi md5 #idmd5 dove per ognuno viene riportato l'identificativo md5 e il nome del file.

```
IPP2P:RND <> IPT:3000
      > "LOOK"[4B].SessionID[16B].Ricerca[20B]
      < "ALOO"[4B].#idmd5[3B].{Filemd5_i[32B].Filename_i[100B].LenFile[10B].
      LenPart[6B]}(i=1..#idmd5)
```

La seconda fase avviene periodicamente ed ha lo scopo di indicare l'attuale situazione di ogni parte per un dato file identificato dal proprio md5 su ogni peer che ne ha effettuato il download o che lo ha immesso nel sistema. Complessivamente il numero dei peer che hanno interesse al file in questione sono #hitpeer. Sebbene le ottimizzazioni non siano un elemento affrontato in questo contesto, la lista delle parti disponibili può risultare particolarmente lunga ed onerosa da trasferire periodicamente, per cui PartList viene rappresentata direttamente in binario, dove 0 implica assenza di parte e 1 invece presenza. Considerando una rappresentazione a 8 bit, il vettore di byte che rappresenta la PartList ha dimensione #part8=supint[#part/8] e le informazioni di presenza o assenza sono rappresentate al Byte B=infint[part/8] e al bit b=part mod 8 (dove il b=0 è LSB cioè quello più a sinistra e b=7 è MSB cioè quello più a destra). La prima parte ha label 0. Il peer interessato ad un download deve aggiornare periodicamente la propria conoscenza dello stato, mediante una esplicita richiesta al tracker, e assumiamo che tale aggiornamento avvenga ogni 60s.

```
IPP2P:RND <> IPT:3000
      > "FCHU"[4B].SessionID[16B].Filemd5_i[32B]
      <
"AFCH"[4B].#hitpeer[3B].{IPP2P_i[55B].PP2P_i[5B].PartList_i[8bit][#part8]}
      (i=1..#hitpeer)
```

Download

Il peer interessato ad effettuare il download di un file ha identificato il file di interesse e conosce la situazione delle parti in ogni peer presente nella rete, identificato opportunamente dalla conoscenza di

IPP2P e di PP2P ove i peer sono all'ascolto dall'atto del login. Sulla base della seconda fase della ricerca il peer è in grado di compilare una tabella in cui sono presenti le occorrenze delle parti. Il peer procede a: identificare le occorrenze di ogni parte, scartare le occorrenze delle parti che già possiede, ordinare le occorrenze in modo crescente, selezionare le parti con minore occorrenza, tra le parti selezionate con uguale occorrenza selezionare una parte casualmente, effettuare il download della parte selezionata da uno dei peer che la rende disponibile scelto a caso, aggiornare il tracker. Il sistema deve consentire il download parallelo di più parti in parallelo, specificando esplicitamente il numero delle parti parallelle trattabili.

Relativamente alla singola parte, questa viene suddivisa in chunk, e il download dal peer selezionato prevede l'indicazione del file mediante il proprio identificativo random univoco e l'indicazione della parte di interesse. Il numero dei chunk #chunk di cui è composta ogni singola parte è costante, con possibile e probabile eccezione dell'ultima. Al termine del download di una parte il peer comunica al tracker la nuova parte di cui è venuto in possesso e il tracker gli ritorna il numero delle parti di quel file che gli risultano il peer complessivamente possegga.

```
IPP2P:RND <> IPP2P:PP2P
      > "RETP"[4B].Filemd5_i[32B].PartNum[8B]
      < "AREP"[4B].#chunk[6B].{Lenchunk_i[5B].data[LB]}(i=1..#chunk)

IPP2P:RND <> IPT:3000
      > "RPAD"[4B].SessionID[16B].Filemd5_i[32B].PartNum[8B]
      < "APAD"[4B].#Part[8B]
```

Logout

Supponiamo che un peer che mette a disposizione un nuovo file rimanga presente nel sistema almeno fino a quando ogni sua parte sia stata copiata in almeno un altro peer. Con tale definizione non si considera una procedura di rimozione dei file messi in condivisione, che una volta aggiunti nel sistema vengono considerati patrimonio comune non rimovibile. Un peer può effettuare il logout e due sono le possibilità: non viene consentito con una risposta "NLOG" dal tracker in quanto i file messi a disposizione come sorgente non sono ancora divenuti patrimonio del sistema e nella risposta del tracker viene riportato il numero complessivo di parti già scaricate almeno una volta da altri peer della rete relativi ai file di cui il peer in logout è sorgente #partdown; viene consentito con risposta "ALOG" del tracker non rendendo più disponibili tutte le parti possedute dei vari file, ritornando il numero complessivo di parti #partown che sono in possesso del peer in logout su ogni file di cui è sorgente o che ha scaricato. Con il logout il peer smette di ascoltare sull'IP e la porta specificata per le azioni di P2P. Coerentemente con l'update ogni 60s si esce dal sistema dopo 60s e comunque se non vi sono download in corso.

```
IPP2P:RND <> IPT:3000
      > "LOGO"[4B].SessionID[16B]
      1< "NLOG"[4B].#partdown[10B]
```

```
2< "ALOG"[4B].#partown[10B]
```

NAT e P2P

Network Address Translation (NAT) diffuso

La traslazione di indirizzi più utilizzata è quella effettuata dai router casalinghi. Tali dispositivi hanno assegnato un singolo indirizzo IP dinamico dal provider di rete, tipicamente pubblico, e offrono più indirizzi IP privati nell'ambito locale mediante l'integrazione di un server DHCP.

Gli indirizzi IP privati locali devono essere traslati nell'indirizzo dinamico assegnato e quindi si ha un passaggio, che deve essere invertibile, molti ad uno, dove è necessario utilizzare oltre all'indirizzo IP anche un ulteriore indirizzo e l'unico disponibile, solo per comunicazioni UDP e TCP, è la Porta. Si realizza così una traslazione (IP sorgente pre NAT, Porta sorgente pre NAT) in (IP sorgente post NAT, Porta sorgente post NAT), dove IP sorgente post NAT è fisso e IP sorgente pre NAT appartiene alla rete locale distribuita dal router con CIDR /xx.

Complessivamente si ha l'intera mappatura sulla Porta sorgente post NAT, con 16+32-xx=48-xxbit mappati in 16bit; un caso tipico casalingo vede xx=24 per cui si ha una mappatura di 24bit in 16bit, mentre il caso più esteso di utilizzo dell'intera classe A 10.0.0.0/8 vede la mappatura di 40bit in 16bit. In ogni caso la mappatura di traslazione è dinamica ed i bit pre da mappare sono maggiori dei bit post, per cui si parla di mappatura con overload o comunque di port translation. Evidentemente questa possibilità si ha solo con UDP e TCP, mentre non è applicabile su pacchetti IP nativi senza layer di trasporto in quanto non si hanno Porte su cui agire.

Ogni mappatura rimane nelle tabelle dei router per un tempo determinato, legato all'utilizzo, legato alla implementazione e comunque non standardizzato. Si assume che la mappatura risulti indipendente dalla destinazione e quindi che la traslazione non consideri (IP sorgente, Porta sorgente, IP destinatario, Porta destinataria) ma solo (IP sorgente, Porta sorgente). Questo è uno dei casi più diffusi in termini di implementazione e viene denominato full cone NAT.

Il P2P non è nativamente compatibile con il NAT in quanto non si conoscono a priori le decisioni relative alla mappatura effettuata dal NAT ed occorre capire se due generici peer sono sulla stessa rete locale, e quindi possono operare pre NAT, oppure sono separati da uno o più NAT, posti in cascata. Lo scenario sull'utilizzo del NAT, ed anche di più NAT applicati uno dopo l'altro, è potenzialmente complesso.

Hole Punching

Hole Punching, o perforatura, è basato sulla presenza di un elemento terzo, rispetto a due peer che

vogliono comunicare A e B, che rappresenta un punto di rendezvous R utilizzato non per effettuare la comunicazione ma per avere conoscenza della coppia IP e Porta post NAT. Nel seguito gli elementi procedurali:

- A e B hanno una sessione con R, che informa sulla coppia pre NAT e permette di desumere la coppia post NAT;
- A vuole instaurare una connessione con B;
- A chiede a R la coppia pre e post NAT di B;
- R risponde ad A con la coppia pre e post NAT;
- R informa B della coppia pre e post NAT di A, notificando l'interesse di A ad una comunicazione
- A genera due flussi di pacchetti verso B utilizzando sia la coppia pre NAT che quella post NAT e considerando valido il primo flusso da cui ha risposta;
- dato l'interesse segnalato da R, B genera due flussi di pacchetti verso A utilizzando sia la coppia pre NAT che quella post NAT e considerando valido il primo flusso da cui ha risposta.

Questa procedura funziona semplicemente con UDP mentre occorre porre attenzione con il TCP in quanto i flussi sono procedure di connessione e bisogna fare in modo che vi sia una singola porta TCP capace sia di stare in ascolto che di realizzare connessioni multiple verso l'esterno, situazione atipica in comunicazioni client-server ma altresì possibile mediante funzioni di riuso della Porta introdotte come opzioni nella programmazione delle socket TCP.