Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Magistrale in Scienze di Internet

LIVE STREAMING IN SISTEMI PEER-TO-PEER: CRITERI E METODI DI VALUTAZIONE

Tesi di Laurea in Microeconomia e Teoria Dei Giochi

Relatore:
Prof. Giovanni Rossi

Presentata da:
Lidia Santi

Sessione I
Anno Accademico 2010/2011

Indice

Elenco delle figure

Introduzione

Questo lavoro di tesi tratta l'argomento del live streaming in sistemi peer-to-peer visto dal punto di vista della teoria dei giochi, si cerca di illustrare i motivi per cui questi particolari sistemi possano essere associati ad un'analisi di tipo economico, ed in che modo gli utenti che partecipano a queste reti di condivisione possano essere considerati dei giocatori che fanno parte di un particolare gioco strategico.

L'aspetto più interessante non riguarda la modellazione dei giocatori all'interno della rete ma la valutazione di questi sistemi in termini di performance e qualità del servizio. Ci si sofferma anche sul fatto che i vari utenti venendo considerati dei giocatori razionali e strategici tendono a comportamenti che portano a massimizzare le proprie necessità, in alcuni casi questi comportamenti altamente egoistici che in letteratura vengono definiti *free-riding* portano ad un deterioramento delle prestazioni del sistema. Per evitare che i giocatori si comportino da free-rider e tendano a lasciare la rete nel momento a loro più opportuno e senza partecipare attivamente alla condivisione all'interno della rete vengono studiati metodi e tecniche per limitare

il verificarsi di tali comportamenti antisociali.

L'argomento è stato scelto in quanto in letteratura negli ultimi anni si è dato particolare peso a questi aspetti di valutazione delle prestazioni dei sistemi di live streaming in reti p2p grazie anche al continuo aumento di società che offrono tale servizio. Gli stessi provider sono infatti interessati a capire se la qualità della trasmissione da loro offerta risulta essere buona, nel caso infatti gli utenti non fossero soddisfatti probabilmente lascerebbero la rete che via via a causa del basso numero di utenti connessi degraderebbe proporzionalmente in base al tasso di abbandono e le basse prestazioni potrebbero portare ad un potenziale collasso delle rete stessa. In letteratura sono presenti anche numerosi metodi che permettono di apportare dei miglioramenti alle performance dei sistemi, grazie a queste tecniche si è in grado di offrire un miglior servizio agli utenti che decidono di partecipare alla rete, così facendo il numero degli utenti aumenta in relazione alla qualità del servizio offerto.

Nel primo capitolo viene introdotta la tematica del live streaming, dapprima illustrandone brevemente la storia per soffermarsi in seguito alle caratteristiche principali e descrivere i modelli secondo i quali le reti p2p possono essere rappresentate.

Nel secondo capitolo si cerca di dimostrare in che modo i vari utenti di una rete p2p possano essere considerati ed analizzati sotto l'aspetto di un congestion game nella teoria dei giochi.

Il terzo capitolo riguarda l'analisi della performance dei sistemi p2p pro-

gettati per il live streaming, vengono introdotte le metriche usate per l'analisi e viene illustrato un metodo progettato per apportare un miglioramento alle prestazioni percepite dagli utenti. All'interno di questo capitolo ci si sofferma anche sulla descrizione dei nodi stabili, ovvero di quei giocatori che decidono di rimanere per un tempo abbastanza prolungato all'interno della rete, permettendo grazie alla loro permanenza all'interno della rete l'assenza di buchi nella trasmissione dei dati causati da repentini abbandoni della rete da parte degli altri utenti non stabili.

L'ultimo capitolo riguarda invece la qualità del servizio e le metriche usate per l'analisi di questa, anche in questo caso viene descritto un metodo che permette il miglioramento della qualità all'interno del sistema.

Capitolo 1

Lo streaming dalle origini ad oggi

In questo primo capitolo verrà introdotta la tematica della tesi, nel primo paragrafo verrà descritta la tecnica dello streaming, le sue caratteristiche e il protocollo che viene utilizzato per la comunicazione tra client e server, nel secondo paragrafo verranno illustrati i modelli per la rappresentazione dei sistemi p2p per lo streaming mentre nell'ultimo verranno introdotti i comportamenti di alcuni peer particolari chiamati free-riders.

1.1 Lo streaming

La teoria dei giochi è quella branca di studi che si pone come punto d'incontro tra la matematica e l'economia e che è possibile definire come un'analisi formale dell'interazione strategica fra unità decisionali , dette "giocatori". In particolare, l'interazione può essere caratterizzata sia da conflittualità sia da

problematiche di coordinamento e/o cooperazione tra i giocatori. E' possibile trovare applicazione di questa teoria nei moderni sistemi p2p (peer-to-peer) con particolare attenzione per quanto riguarda lo streaming.

Lo streaming si concretizza nella ricezione da parte di un client di un flusso di dati audio e video risiedenti su un server, la riproduzione di tale flusso avviene in maniera continuativa man mano che i dati arrivano al ricevitore.

1.1.1 La nascita e lo sviluppo dello streaming

Dagli inizi degli anni '80 fino agli anni '90 i computer si sono enormemente sviluppati e via via sono diventati sempre più potenti per rendere possibile la visualizzazione di una gran quantità di formati audio e video con qualità sempre maggiore. Se in questi primi decenni era ottimale scaricare tutto un file da un server remoto nei successivi grazie ad una bandwidth maggiore e alla creazione di protocolli più specializzati per il traffico internet è stata possibile la diffusione dello streaming.

Il primo evento diffuso tramite la tecnica dello streaming è stata la partita di baseball tra gli Yankees e i Seattle Mariners nel 1995 grazie a Real Networks.

Negli anni intorno al 2000 la maggioranza dei contenuti fruibili via streaming erano in formato Real, il che generò vari problemi agli utenti che si ritrovavano con il Real player installato di default per la visione dei filmati ed inoltre venivano scaricati una grande quantità di altri programmi indesiderati. L'intrusività del player riguardava anche il fatto che tale programma

invitava in maniera costante l'utente a scaricare la versione più aggiornata. Andando avanti con il tempo si notò che il player veniva disinstallato dalla maggioranza degli utenti, per questo motivo i produttori cominciarono a boicottare l'installazione di default del programma all'interno degli elaboratori.

1.1.2 Le caratteristiche

Al giorno d'oggi possiamo evidenziare l'esistenza di due tipologie di streaming:

- streaming on demand
- live streaming

Nello streaming on demand le richieste dei singoli client vengono fatte direttamente al server nel quale è memorizzato il file da diffondere, quando un utente richiede il documento alla sorgente essa provvederà ad instaurare una comunicazione con il client e comincerà l'invio del file. Il client non è obbligato a scaricare completamente il file prima di poterlo visionare, esso sarà infatti in grado di visualizzare le parti ricevute pochi istanti dopo la loro ricezione. Questo tipo di streaming è detto anche *progressive streaming* o *progressive download*, in quanto è possibile salvare il file su un hard disk e visionarlo in qualunque momento si desideri.

Nella versione live o *true streaming* ogni peer comunica con gli altri, questa tipologia di streaming assomiglia alla normale trasmissione di radio e

televisioni in broadcast, la comunicazione avviene al fine di condividere parti del file in maniera tale da aumentare il througput dell'applicazione, ma a differenza del progressive streaming il file non viene salvato su un hard disk e il contenuto multimediale è fruibile un'unica volta: nel momento in cui il programma è trasmesso.

Il fattore discriminante tra le due tipologie di streaming risulta quindi essere il tempo, se nel primo caso non è importante che due o più utenti stiano accedendo ai contenuti multimediali nello stesso periodo di tempo, nel live streaming questo risulta essere fondamentale. Nel live streaming, come già il termine live lo indica, la cooperazione dei peer può avvenire a partire dall'istante in cui inizia la trasmissione dal server o in un momento successivo considerando il fatto che non è possibile accedervi quando la trasmissione è finita.

Le due tipologie applicano un ritardo nella riproduzione delle parti di video ricevute per permettere che un decadimento momentaneo della rete non influisca sulla performance degli elaboratori dei client.

I principali e i più utilizzati sistemi per poter accedere alle funzionalità del live streaming sono PPLive, PPStream, CoolStreaming, SopCast, TVAnt, End System Multicast, Joost, GridMedia, Zattoo, Vudu, Octoshape, Tvkoo, Roxbeam e Tribler. La maggior parte di essi è stata creata nell'est asiatico anche se un gran numero di utilizzatori è di nazionalità europea e nordamericana; PPStream e PPLive, hanno ancora oggi il sito in lingua cinese, PPStream ha comunque una versione in italiano mentre PPLive permette

sia la visualizzazione in cinese che in inglese.

Protocollo per lo streaming

Il protocollo che viene utilizzato nella maggior parte dei casi per lo streaming, nel modello ISO/OSI[1] a livello *applicazione* è l'RTSP acronimo di Real Time Streaming Protocol, creato per ottimizzare lo stream dei dati. L'RTSP non si occupa della trasmissione vera e propria che invece spetta al protocollo RTP (Real Time Transport Protocol) o in alcuni casi a protocolli di trasmissione proprietari; a livello *trasporto* viene invece utilizzato UDP[2] o TCP[3].

Il protocollo RTSP è nato grazie alla collaborazione tra RealNetworks, Netscape Communications e la Columbia University di New York e nel 1998 è stato pubblicato come RFC 2326. RTSP risulta essere simile al protocollo HTTP[4], ma sono presenti alcune differenze [3] :

- Un server RTSP ha bisogno di mantenere lo stato di default in quasi tutti i casi, al contrario della natura *stateless* di HTTP

[1]E' uno standard creato nel 1978 dalla ISO, stabilisce che l'architettura logica di una rete di elaboratori sia composta da una pila di protocolli. Il modello è formato da sette strati detti livelli e sono i seguenti: fisico, datalink(collegamento), rete, trasporto, sessione, presentazione, applicazione.

[2]UDP (User Datagram Protocol) è un protocollo orientato alla trasmissione di pacchetti. Non ha bisogno di instaurare una connessione e per questo motivo supporta trasmissioni con molti client. Non si occupa della ritrasmissione dei pacchetti persi o del riordinamento degli stessi a destinazione, pur non essendo affidabile da questo punto di vista risulta essere un protocollo leggero e veloce.

[3]TCP (Transmission Control Protocol) è un protocollo che a differenza di UDP ha necessità di instaurare una connessione tra due elaboratori prima della trasmissione dei dati. Questo protocollo si occupa del controllo sul flusso di byte che ha ricevuto il destinatario che deve ricevere tutti i segmenti di dati una sola volta.

[4]HTTP (Hypertext Transfer Protocol) è il protocollo usato a livello applicazione per la trasmissione di ipertesti all'interno del web.

HTTP è definito stateless in quanto il server non è tenuto a memorizzare lo stato della connessione con un certo client, nel caso in cui sia invece necessario avere delle informazioni di sessione è stata introdotta la tecnica dei cookie.

Invece nel caso dell'RTSP, ricorrendo ai protocolli UDP e TCP per la trasmissione dei dati, risulta necessaria la presenza di uno stato che permetta di identificare la sessione in corso, in alcuni casi infatti non è raro che una stessa trasmissione avvenga durante diverse connessioni TCP, il server quindi dovrà essere in grado di mantenere lo stato di sessione per permettere la corretta esecuzione della richiesta da parte di un client.

- RTSP introduce dei nuovi metodi

  HTTP definisce nove metodi: HEAD, GET, POST, DELETE , TRACE, OPTIONS, CONNECT , PATCH.

  I metodi di RTSP sono invece i seguenti: DESCRIBE, ANNOUNCE, GET_PARAMETER, OPTIONS, PAUSE, PLAY, RECORD, REDIRECT, SETUP, SET_PARAMETER, TEARDOWN.

- I dati sono portati *out-of-band*[5] da un protocollo differente.

  HTTP è un protocollo asimmetrico e considerato *in-band*, il client fa una richiesta al server, ovvero attraverso il browser richiede al server

[5]Sono i dati che appartengono ad uno stream diverso rispetto alla trasmissione principale

una pagina web, il server dopo aver recepito la richiesta risponde al client restituendo la pagina se essa si trova nel server oppure mostrando un messaggio di errore se nel server la pagina non esiste più (error 404 file not found).

RTSP è invece considerato un protocollo out-of-band, i dati che fanno parte dello streaming sono compresi nell'in-band stream, in aggiunta ci sono dei messaggi di controllo che non fanno parte dello stream di dati principale e che quindi sono definiti out-of-band (fuori dalla banda).

- RTSP è definito per usare la ISO 10646 (UTF-8) piuttosto che la ISO 8859-1

  La ISO 10646 rappresenta lo standard universale per la rappresentazione dei caratteri, al contrario la ISO 8859-1 indica la codifica dei caratteri ad 8 bit che comprende il sottogruppo *Latino-1* corrispondente alle lingue europee occidentali e che viene utilizzato per i documenti HTML.

Come appena detto il protocollo RTSP è caratterizzato da uno stato di sessione che permette una correlazione tra le richieste RTSP e lo stream dei dati attraverso connessioni TCP e UDP. Tra gli stati che vengono utilizzati per gestire l'allocazione delle risorse è possibile trovare: SETUP, PLAY, RECORD, PAUSE e TEARDOWN.

- SETUP: vengono allocate risorse per iniziare una sessione RTSP

- PLAY e RECORD: vengono trasmessi i dati della sessione iniziata tramite SETUP

- PAUSE: lo stream viene interrotto temporaneamente senza liberare le risorse

- TEARDOWN: le risorse allocate vengono liberate e la sessione RTSP viene chiusa e cancellata dal server

La figura 1.1 illustra come avviene una comunicazione tra il client e il server. Per prima cosa viene attivata una comunicazione tramite il protocollo HTTP, ovvero il client chiederà di attivare una sessione perchè è interessato a scaricare un qualche file in streaming, nel momento in cui il web server risponderà al client sarà possibile fare il setup della sessione tramite RTSP e comunicare con un media server nel quale sono memorizzati i dati multimediali, appena il client avrà ricevuto una piccola parte del file sarà in grado di visualizzarla sul proprio elaboratore tramite un player video, il client sarà in grado di mettere in pausa la comunicazione senza comunque perdere la possibilità di riattivare il processo di download dal server, quando il client avrà finito il download del file e la sua visualizzazione, sarà possibile chiudere la comunicazione tra client e server e liberare le risorse finora occupate.

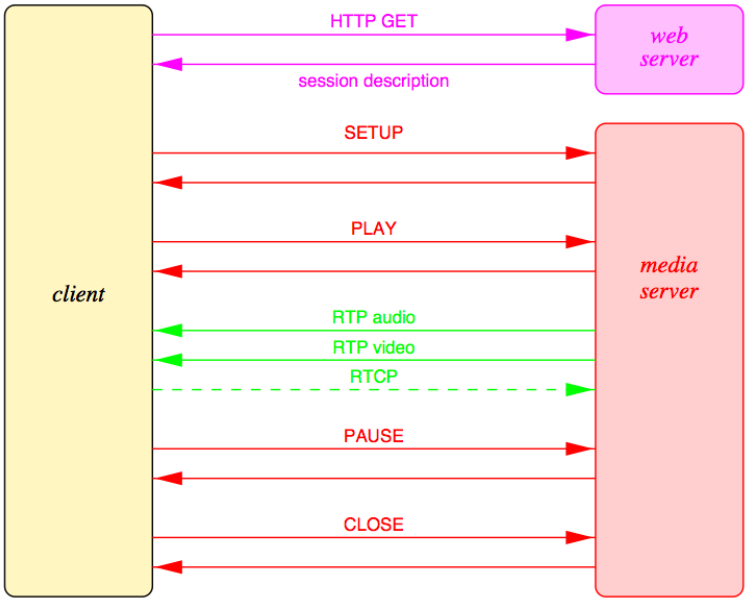

Figura 1.1: Comunicazione tra un client e un media server via RSTP

Modalità di diffusione dei dati

Esistono diverse modalità per la diffusione dei dati in streaming

- Unicast: è il caso più semplice nel quale i dati sono trasmessi alla sorgente della richiesta RTSP, quindi risulta essere una banale trasmissione uno-a-uno, inoltre è il destinatario a scegliere il numero di porta da usare per la comunicazione

- Multicast: i dati vengono trasmessi simultaneamente ad un gruppo di destinatari che si trovano ad avere un indirizzo univoco detto appun-

to indirizzo multicast, che appartiene alla classe D[6] degli indirizzi IP assegnabili. Il vantaggio che porta il multicast è il fatto che la sorgente invierà un'unica copia del file all'indirizzo multicast e saranno poi i router multicast a indirizzare e replicare l'informazione verso ogni host. Esistono due varianti della trasmissione multicast, nella prima è il server a scegliere l'indirizzo e la porta. In questo primo caso rientrano le tipologie della trasmissione live o near-media-on-demand[7]. Nel secondo caso è il client a scegliere l'indirizzo, in particolare se il server partecipa ad una conferenza multicast, l'indirizzo, la porta e la chiave di criptazione sono date dalla conference description.

- Peer-to-peer: in quest'ultima modalità i vari client si trovano a collaborare tra di loro, la banda di ciascun peer viene utilizzata per trasmettere agli altri utenti. In questa particolare configurazione le prestazioni dei server non sono molto critiche dato che essi sono costretti a comunicare con un numero limitato di utenti, poi saranno questi ultimi ad avere una sorta di funzione da server e comunicare con gli altri peer per la diffusione del file. D'altro canto saranno proprio i peer a causare o meno dei *bottlenecks* (colli di bottiglia), tali situazioni critiche saranno evitate nel momento in cui i peer si troveranno ad avere un'elevata velocità sia in ricezione che in trasmissione.

[6] Gli indirizzi di classe D sono del tipo [224-239].x.x.x. e cominciano con la sequenza di bit 1110. Tutti i 32 bit dell'indirizzo servono per indicare un gruppo di host e per questo motivo non hanno una maschera di rete.

[7] Il media richiesto viene trasmesso su più canali distinti ma ad intervalli regolari che permettono quindi all'utente di scegliere il momento migliore per accedervi.

1.2 Lo streaming p2p e i suoi modelli

E' possibile rapportare i sistemi p2p alla teoria dei giochi, ogni peer viene inteso come un giocatore, nella migliore tradizione dello streaming ogni host era solito ricevere il flusso dati direttamente dalla sorgente, invece nell'approccio proposto dai sistemi p2p ogni giocatore collabora con gli altri per lo scambio dei vari slot del file totale.

Aspetti da tenere in considerazione sono il fatto che nei sistemi p2p ogni nodo possa entrare nella rete e lasciarla quando vuole, questo può causare delle difficoltà nel momento in cui un peer sta scaricando una parte di un file da un altro user e quest'ultimo decide di lasciare la rete per un qualsiasi motivo.

Nei sistemi p2p gli utenti si trovano sia ad usare altri peer sia vengono a loro volta usati per ottenere i vari *chunks*, le parti di file, in questo senso è possibile considerare tale situazione un *multistage congestion game* dove le risorse sono i peer stessi.

Il modello secondo il quale i peer comunicano tra di loro risulta essere il migliore perchè in tal modo si evita il bottleneck causato dai modelli multicast (anche in presenza di server secondari). Il modello multicast infatti presenta un problema rilevante riguardante i bottleneck creati dall'unico server che è proprietario del file, man mano che il numero degli utenti aumenta la situazione di una congestione critica diventa un problema sempre maggiore, dato che ogni peer si mette in comunicazione con la sorgente la quale

riceve un numero sempre maggiore di richieste. Per poter eliminare questa problematica è stato pensato l'utilizzo di server multipli, grazie ai quali è stato possibile aumentare la bandwidth, la scalabilità e la fault tolerance. Il punto focale risulta essere l'esistenza di un protocollo di comunicazione efficace che permetta l'interazione tra i diversi peer in modo ottimale.

Per quanto riguarda l'architettura della rete dei peer essa può essere organizzata in modo strutturato oppure non strutturato, nel primo caso i peer per comunicare tra di loro hanno bisogno di una necessaria strategia, e la struttura della rete può essere un albero, una rete mesh o per esempio una tabella di hash distribuita, nel secondo caso invece si utilizza un protocollo *gossip* che permette una comunicazione libera tra i diversi peer, ovvero ogni peer può comunicare con il nodo che vuole nel momento da lui preferito.

1.2.1 Rete strutturata

Comunicazione basata sull'albero

La comunicazione che avviene attraverso una struttura ad albero è piuttosto semplice: la radice dell'albero è la sorgente del file da diffondere e man mano che si scende negli altri livelli della struttura si trovano i nodi degli elaboratori ai quali arriverà il file da visualizzare. Ogni chunk viene messo in circolazione a partire dai nodi padre per poi essere passato ai nodi figli.

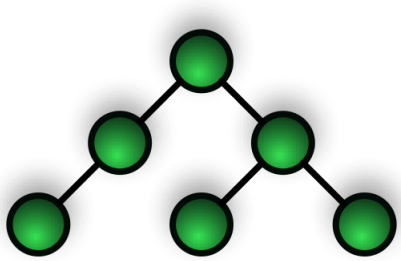

Figura 1.2: Topologia ad albero

La questione importante che riguarda la tecnica basata sull'albero è la generazione di un albero in grado di gestire i nodi che lasciano la rete in modo tale da non avere dei buchi, è possibile infatti che tutto un livello dell'albero decida di lasciare la rete, ciò provocherebbe un danno consistente perchè l'albero si spezzerebbe in due ed alcuni nodi non riceverebbero il flusso di dati. Questa tipologia di organizzazione dei nodi è soggetta al *node churn* [5], ovvero ai problemi causati dal comportamento dinamico dei nodi che formano la rete, non esiste alcun tipo di tecnica in grado di gestire l'abbandono di un nodo dall'albero tramite una qualche notifica, similmente succede anche nel caso in cui un nodo voglia accedere al network. L'unica azione attuabile per evitare che i nodi più esterni vengano penalizzati da un notevole abbandono dei peer più interni è rappresentata dall'aggiornamento delle struttura, che comporta comunque una latenza in termini di tempo e all'inserimento all'interno della rete di informazioni di puro controllo per mantenere coerente la struttura.

Comunicazione basata su multi-alberi

Una soluzione alla questione della dinamicità dei nodi è la creazione di multi alberi, invece di avere degli alberi singoli, è possibile che nodi interni di un albero siano dei nodi foglia di un altro albero, in questo modo viene garantita una route dei dati in modo efficace e una migliore distribuzione delle informazioni. In questa architettura i nodi interni si occupano di passare le informazioni in loro possesso a tutti i loro nodi figli. Ogni peer decide di quanti alberi far parte in base alle caratteristiche della propria connessione, in modo tale da poter connettersi con il maggior numero di peer ma allo stesso tempo deve trovare un equilibrio e non creare delle situazioni critiche di upload delle informazioni.

Per evitare il fenomeno del node churn presente nella comunicazione basata su alberi singoli si ricorre alla costruzione di più multi-alberi distinti.

Comunicazione basata su reti mesh

L'utilizzo di reti mesh si rifà all'organizzazione tipica del protocollo BitTorrent usato nelle reti p2p per lo scambio di file tramite programmi di file sharing. Questo tipo di rete garantisce una gestione migliore dei nodi che decidono di accedere alla rete o di lasciarla. Ogni nodo nella rete ha un gruppo ristretto di nodi-partner con i quali comunica, è perciò facile modificare la topologia del network, nel momento in cui un host decide di lasciare il sistema semplicemente i nodi con cui esso comunicava eliminano il collegamento tenendo comunque sempre attivi i collegamenti con gli altri nodi-partner,

quando invece un host decidere di far parte della rete si crea un collegamento con uno o più nodi che da quel momento diventeranno i nodi-partner del nuovo elemento del network.

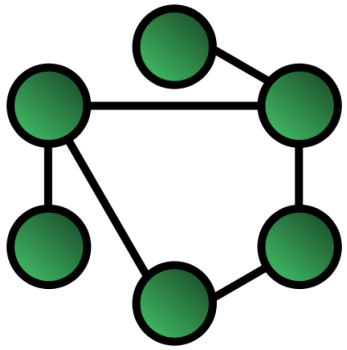

Figura 1.3: Topologia a rete mesh

Un problema evidenziabile nell'architettura a rete mesh è la mancanza di una chiara gerarchia padre-figlio, le informazioni vengono infatti estratte dai nodi vicini, si deve quindi stabilire un compromesso tra l'efficienza e la latenza accettate.

1.2.2 Rete non strutturata

Protocollo Gossip

Il protocollo gossip è un protocollo di comunicazione host-to-host, deve il suo nome proprio al fenomeno nel gossip studiato nelle reti sociali. Un tipico esempio per spiegare il funzionamento di una comunicazione che usa il protocollo gossip è il riferimento a come gli impiegati di un ufficio reagiscano ai

rumors. Si dice che spesso i lavoratori si incontrino vicino alla macchinetta del caffè o al distributore di acqua, e proprio in questi ambienti è facile che un impiegato si scambi con un altro l'ultimo gossip di cui è venuto a conoscenza. Nelle pause successive il gossip si espanderà poichè ogni impiegato ripeterà ciò di cui è venuto a conoscenza interagendo con gli altri colleghi; è chiaro che a fine giornata tutti o quasi i lavoratori saranno a conoscenza del "rumor of the day". Tornando alle reti di computer la comunicazione tramite gossip consiste nella trasmissione computer-to-computer di una certa informazione. Uno degli aspetti interessanti di questo protocollo risiede nel fatto che un peer possa ricevere dei dati (il gossip) non esatti per una perdita di qualche bit durante la comunicazione, ma in ogni caso potrà ricevere nuovamente tale informazione esatta poichè essa arriverà sicuramente da un altro peer. Nella maggior parte dei casi ogni peer riceverà la stessa informazione da diversi altri nodi della rete in modo da effettuare anche un controllo incrociato per verificare la correttezza delle informazioni ricevute.

1.3 I free-riders

La questione dei *free-riders* è una problematica che si presenta in vari ambiti a partire dall'economia passando per la psicologia fino ad arrivare al live streaming, in generale di questa categoria di persone fanno parte coloro che accedono ad un bene comune ma che ne beneficiano senza pagare un costo o pagandone meno del dovuto o meno rispetto ad altre persone.

Nella particolare accezione riguardante il live streaming i free-rider sono quei peer che non contribuiscono con l'upload dei file se non hanno una qualche sorta di incentivo a farlo.

Di default solitamente i peer sono tenuti a condividere i propri file, nonostante ciò è possibile disabilitare la funzione di share per avere una maggiore bandwidth personale. Per limitare l'attività di questi peer i sistemi p2p sono soliti adottare delle politiche particolari nei loro confronti, le più usate sono la Block and Drop (BD) e la Block and Wait (BW).

La prima politica prevede il blocco di questi peer se la free upload capacity risulta essere minore dello streaming rate. Se un peer riesce ad accedere al sistema durante una sessione di streaming può comunque essere estromesso se la capacità di copertura si trova a diminuire a causa del continuo abbandono dei peer contribuenti. In questa particolare politica il free-rider può quindi sia essere bloccato mentre cerca di accedere alla sessione oppure può riuscire ad accedere alla sessione ma poi ne viene estromesso, in modo tale che esso non possa più scaricare altri dati dalla rete p2p.

Nella politica BW invece i free-rider vengono bloccati se la copertura non ha capacità sufficiente per tutti i peer, in questo caso i peer vengono disconnessi dalla rete e sono tenuti ad aspettare un certo periodo di tempo prima di potersi riconnettere.

Un modo per scoraggiare i peer a comportarsi da free-rider consiste nel proporre a questi peer una bassa qualità video, ovviamente a nessuno interessa partecipare ad uno stream video che non gli permette di godere appieno di

tale servizio, più la qualità risulta essere bassa più i peer decidono di lasciare la rete.

Oltre all'attuazione di politiche contro i free-rider sono state adottate alcune tecniche per incentivare i peer a non comportarsi in questo modo, i meccanismi adottati si basano sulla generosità dello share degli altri peer, su micropagamenti in base a quanto i peer contribuiscono alla diffusione dei file e sulla reciprocità delle condivisioni.

Nel caso della generosità ogni utente decide come comportarsi, può scegliere di contribuire in base a come la sua generosità è calcolata riguardo al resto della popolazione o può al contrario scegliere di essere un peer egoista e comportarsi da free-rider non contribuendo in nessun modo alla diffusione dei dati. E' possibile analizzare il sistema e calcolare la percentuale dei free-rider presenti all'interno della rete basandosi sulla distribuzione della generosità dell'intera popolazione dello stesso. Se la generosità dei peer facenti parte della rete si trova al di sotto di un certo valore allora i peer che si comporteranno da free-rider aumenteranno e il sistema si troverà in una situazione critica e andrà verso il collasso in quanto sempre più peer vorranno accedere ai contenuti senza però contribuire in maniera attiva alla diffusione dei file.

Il secondo meccanismo è incentrato su tecniche di remunerazione, coloro che accedono al servizio dovrebbero pagare i fornitori in base a come usano le risorse di questi provider. Questa tecnica non risulta essere molto semplice in quanto sono necessarie delle infrastrutture che permettano delle transazioni tra il destinatario dei dati e coloro che forniscono il servizio.

L'ultima tecnica si basa sul concetto di reciprocità, ogni peer terrà in memoria uno storico riguardante i comportamenti degli altri peer e potrà usare tali informazioni per poter decidere il proprio comportamento. Esistono due versioni del metodo di reciprocità: diretto ed indiretto. Nella reciprocità diretta un utente deciderà come servire un altro utente solamente in base a come quest'ultimo si è comportato con il primo nelle precedenti connessioni. Invece nel caso della reciprocità indiretta un utente che vuole condividere qualcosa con un altro utente non guarderà solamente al comportamento del secondo peer verso il primo ma anche come questo peer si è comportato con il resto della comunità. La reciprocità diretta risulta essere utile nel caso in cui la durata delle sessioni sia piuttosto lunga, in tal modo la reciprocità di ogni coppia di utenti può aumentare in maniera sostanziale. La reciprocità indiretta risulta essere invece più scalabile soprattutto nei casi in cui la popolazione soggetta alla reciprocità risulti essere piuttosto ampia e con un numero di peer altamente dinamico che tende ad accedere e lasciare il sistema in maniera continuativa.

Capitolo 2

Peer come giocatori

Il secondo capitolo illustrerà la relazione che intercorre tra i congestion games e i sistemi p2p, soffermandosi sulla questione della funzione di utilità nel secondo paragrafo e analizzando l'equilibrio di Nash in sistemi omogenei ed eterogenei di peer.

2.1 Congestion games e peer-to-peer

E' stato identificato che molte volte le sessioni degli utenti risultano essere piuttosto corte, ovvero un peer resta connesso alla rete p2p per un tempo compreso tra pochi secondi e un'ora, ciò implica che una gran quantità dei dati potrebbe risultare non disponibile per una buona parte di tempo. La brevità di queste sessioni quindi danneggia la performance del sistema perchè ci si trova nel caso in cui solo pochi peer fungono da server per poter eseguire

il download da essi. Allo stesso tempo con l'aumento del numero degli utenti che diventano free-rider il sistema comincia a perdere il suo spirito di p2p per finire ad essere un banale sistema client-server. In un sistema autonomo ma con partecipanti razionali, nel caso appunto dei peer, risulta corretta l'assunzione che questi client siano incentivati ad un comportamento guidato da principi economici. Nel passato sono state individuate due forme di incentivi: incentivi monetari (un peer paga per consumare risorse ed è pagato per il suo contributo nella diffusione di esse) e servizio differenziato (peer che contribuiscono di più ottengono una migliore qualità del servizio).

Il modello basto sul pagamento coinvolge una moneta fittizia e richiede un'infrastruttura per permettere il tracciamento delle varie transazioni. Anche se lo schema monetario fornisce un chiaro modello economico, appare essere altamente impraticabile. In generale i nodi in un sistema p2p essendo dei giocatori di strategie sono in grado di manipolare qualsiasi modello di incentivazione. Come risultato è possibile gestire questi comportamenti tramite la teoria dei giochi introducendo un modello formale di incentivi attraverso diversi servizi in sistemi p2p e con l'uso della nozione di equilibrio di Nash per analizzare le scelte strategiche dei diversi peer. Ogni peer viene trattato nel sistema come un giocatore strategico e razionale, che vuole massimizzare la sua utilità partecipando a questo sistema p2p. L'utilità di un peer dipende dal suo profitto (le risorse del sistema che lui può utilizzare) e il suo costo (il suo contributo). Il modello di servizi differenziati collega il profitto di ogni peer che può ottenere dal sistema alla sua contribuzione , il

profitto è una funzione monotona crescente del contributo del peer. Questo è un gioco non cooperativo tra i peer: ognuno vuole massimizzare la propria utilità.

Nella teoria dei giochi relativa ai congestion games, ovvero nel caso dei giochi strategici o non cooperativi, si ha una situazione nella quale i giocatori si trovano a scegliere una qualche risorsa in base alla congestione nel momento della scelta che tipicamente dipende dal numero dei giocatori che stanno usando quella stessa risorsa.

Un gioco strategico viene definito dalla tripla Γ = (N, S, u) dove N indica il numero dei giocatori, $u : X_{i \in N} S_i \rightarrow \mathbb{R}^n$ è una funzione che associa ad ogni profilo o n-upla $S = (S_1, \ldots, Sn) \in X_{i \in N} S_i$ di strategie un payoff $u_i(s)$ per ogni $i \in N$.

Un tradizionale sistema distribuito assume che tutti i partecipanti al sistema lavorino insieme cooperando, essi condividono un goal comune, non competono quindi con gli altri o cercano di sovvertire il sistema. Un sistema p2p, d'altro canto, consiste di componenti autonomi: gli utenti competono per le risorse condivise ma limitate e allo stesso tempo, possono restringere il download dai propri server non permettendo l'accesso o non contribuendo con alcuna risorsa. L'interazione tra i vari peer in un sistema p2p è meglio modellato come un gioco non cooperativo tra giocatori razionali e strategici. I giocatori sono razionali perchè vorrebbero massimizzare il proprio guadagno, e sono strategici perchè possono scegliere le proprie azioni che influenzano il sistema. Il giocatore deriva il suo profitto dalla sua interazione con gli altri

giocatori che è definita come il payoff o l'utilità. Il comportamento economico si verifica quando l'utilità di un giocatore dipende non solo dalla sua strategia ma anche dalla strategia giocata dagli altri giocatori. Dal momento che l'utilità di un giocatore dipende dalla sua strategia, egli potrebbe decidere di cambiare la sua strategia per mettere alla prova la sua utilità. Questo cambio di strategia avrà un risultato sugli altri giocatori cambiando la loro utilità e potrebbero decidere anche loro di cambiare la propria strategia.

In questo tipo di giochi si assume che non ci sia alcun meccanismo incentrato su patti od accordi tra i diversi giocatori verificabili da terzi, in quanto è proprio questo il caso in cui è possibile ricercare un equilibrio all'interno del gioco detto equilibrio di Nash. L'equilibrio di Nash viene definito nel modo seguente in [1]:

un profilo di strategie $s = (s_1, s_2, ..., s_I)$ costituisce un equilibrio di Nash del gioco $\Gamma_N = [I, \{S_i\}, \{u_i(\cdot)\}]$ se per ogni $i = 1, ..., I$ $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_i)$ per tutti gli $s'_i \in S_i$.

Se si è in grado di trovare almeno un equilibrio di Nash, ogni giocatore avrà a disposizione almeno una strategia dalla quale non ha alcun interesse ad allontanarsi, se allo stesso tempo gli altri giocatori hanno giocato la propria strategia, nel caso in cui invece ci si allontani dalla propria strategia si peggiora o al massimo si lascia invariato il proprio guadagno. Nel momento in cui ci si trova ad avere un equilibrio di Nash non sarà possibile avere un guadagno migliore in quanto esso è vincolato dalla strategia che viene giocata da ogni singolo partecipante, se l'equilibrio è unico esso sarà la soluzione

del gioco dato che nessuno dei giocatori è incentivato a cambiare la propria strategia avendo in tale momento il massimo guadagno.

Nel momento in cui ci si trova ad avere un equilibrio di Nash in strategie pure, Rosenthal [20] nel 1973 dimostrò che tutti i congestion games hanno un equilibrio di Nash in strategie pure, la soluzione trovata sarà la migliore possibile in quanto questo equilibrio è vincolato alla strategia che viene giocata da ogni singolo partecipante. Se e solo se l'equilibrio è unico esso sarà la soluzione al gioco dato che nessuno dei giocatori è incentivato a cambiare la propria strategia avendo in tale momento il massimo guadagno. Se invece esistono più equilibri di Nash la strada da intraprendere si spinge verso l'equilibrio pareto ottimo. Prendendo in considerazione un congestion game G e un profilo di strategie A in G, A viene detto strettamente pareto ottimo se non esiste nessun profilo di strategie B tale che $\pi^i(B) \geq \pi^i(A)$ per tutti gli $i = 1, ..., n$ ($\pi^i(A)$ indica la funzione del payoff del giocatore i per il profilo di strategie A) e almeno una di queste disuguaglianze è stretta. A invece è debolmente pareto ottimo se non c'è nessun profilo B per il quale tutte le disuguaglianze sono strette [20].

Il concetto classico di equilibrio di Nash si sofferma sull'uscita dal circolo vizioso tra speculazione e contro speculazione in base a quali strategie useranno gli altri peer. Un punto di equilibrio è un insieme di strategie a livello locale ottimale, dove nessun peer può migliorare la sua utilità dissociandosi dalla strategia. Mentre l'equilibrio di Nash è un potente concetto, il calcolo di questi equilibri non è banale. Di fatto, non è noto nessun algoritmo in

tempo polinomiale per trovare l'equilibrio di Nash di un generale gioco con N giocatori.

Tra i vari tipi di gioco presenti in letteratura, una distinzione importante è tra i simultaneous-move games o multistage games. Nel primo caso tutti i giocatori, simultaneamente, scelgono una mossa o azione. Quindi le strategie equivalgono alle azioni dove queste ultime sono elementi di un insieme finito. Nei multistage games invece viene introdotto il tempo $t = 0, 1, \ldots, T$ e $t = 0$ indica l'inizio del gioco o *root* nella *game tree rapresentation* del gioco. Ad ogni $t > 0$ corrispondono diversi nodi del game tree. I nodi terminali o *leaves* corrispondono agli esiti possibili del gioco (sui quali sono definite le preferenze dei giocatori). In tutti gli altri nodi (compresa quindi anche la root) almeno un giocatore deve scegliere una mossa. Quindi ogni possibile evoluzione del gioco o *game course* è individuata da un *path* che parte dalla root e giunge fino ad una leaf. Trattare i peer in sistemi p2p come giocatori è un approccio molto diffuso in letteratura, in particolare alcuni contributi sono incentrati sul problema del free-riding[8][9][14], che emerge quando i peer vengono considerati quali fruitori di quel bene comune che è la rete stessa, potendo quindi tenere comportamenti cooperativi oppure no.

2.2 La funzione di utilità in un sistema p2p

Per analizzare la funzione di utilità dei sistemi p2p è possibile prendere in considerazione un classico sistema di file sharing che poi potrà essere adattato

al caso del live streaming come situazione particolare. Si assume che ci siano N utenti (peer) nel sistema $(P_1, P_2, ..., P_n)$; indichiamo la funzione di utilità per l'i-esimo peer come U_i. Questa utilità dipende da alcuni parametri, tra cui la misurazione del contributo: il singolo numero D_i indica il contributo del peer P_i. Per concretezza D_i viene preso come uno spazio cumulativo sul disco: lo spazio del disco come contributo durante un certo periodo di tempo, per esempio una settimana. Si possono usare anche altre metriche come il numero di download serviti da un peer verso altri client. Per ogni unità di risorsa contribuita, il peer incorre in un costo c_i misurato in dollari, così che il costo totale di P_i per partecipare al sistema sia $c_i * D_i$. Per l'analisi risulta conveniente definire un contributo adimensionale $d_i \equiv D_i/D_0$ dove D_0 è una misura assoluta di contributo (20 MB/week). D_0 è una costante che è possibile fissare da chi gestisce il sistema p2p, il sistema di incentivi dovrà garantire che tutti i peer contribuiscano con almeno D_0 . Ogni contributo dei peer verso il sistema beneficia potenzialmente tutti gli altri peer ma potrebbe variare di grado. Questo beneficio viene codificato usando una matrice B di dimensione $N \times N$, dove B_{ij} indica quanto del contributo offerto da P_j vale per P_i(in dollari). Per esempio se P_i non è interessato al contributo di P_j, allora $B_{ij} = 0$. In generale $B_{ij} \geqslant 0$ e si assume $B_{ii} = 0$. Un insieme di parametri adimensionali corrispondenti a B_{ij} viene definito da

$$b_{ij} = \frac{B_{ij}}{c_i},$$

$$b = \sum_j b_{ij},$$

$$b_{av} = \frac{1}{N} \sum_i b_i$$

dove b_i è il beneficio totale che P_i può derivare dal sistema se tutti gli utenti danno un contributo di una unità. Il parametro b_i sarà importante per determinare se per P_i vale la pena di partecipare al sistema. Esiste un valore critico b_c di beneficio tale che se $b_i < b_c$ poi per P_i sarà meglio non partecipare al sistema, mentre b_{av} indica semplicemente la media dei b_i per l'intero sistema.

Il servizio differenziato è un gioco di aspettative, un peer premia altri peer in proporzione al loro contributo. Uno schema semplice che implementa questa idea è il seguente: il peer P_j accetta una richiesta per un file dal peer P_i con una probabilità $p(d_i)$ e la rifiuta con probabilità $1 - p(d_i)$. Quindi se il contributo di P_i è piccolo la sua richiesta sarà facilmente rigettata. Un esempio di funzione di probabilità è seguente formula:

$$p(d) = \frac{d^\alpha}{1 + d^\alpha} \quad con\ \alpha > 0$$

Tra le proprietà della funzione di probabilità si ha che $p(0) = 0$, e $p(d) \to 1$ in base a quanto d è grande. Per valori piccoli come $\alpha = 1$ la funzione è piuttosto liscia, ma per valori più grandi come 10 la funzione ha uno scalino

evidente, per contributi sotto lo scalino le richieste hanno un'alta probabilità di essere rifiutate, per un contributo al di sopra dello step viceversa.

Con questi parametri di costi e benefici la totale utilità U_i che il peer P_i otterrà partecipando al sistema sarà:

$$U_i = -c_i D_i + p(d_i) \sum_j B_{ij} D_j,\ B_{ii} \equiv 0$$

Il primo termine è il costo per partecipare al sistema mentre il secondo termine è il beneficio totale atteso dalla partecipazione. In termini di parametri adimensionali

$$u_i = \frac{U_i}{c_i D_0}$$

si può riscrivere l'utilità come

$$u_i = -d_i + p(d_i) \sum_j b_{ij} d_j,\ b_{ii} \equiv 0$$

il termine $-d_i$ è semplicemente il costo di P_i per accedere al sistema e aumenta linearmente rispetto al contributo di P_i in disco/bandwidth. Il beneficio di P_i dipende da quanto gli altri peer contribuiscono al sistema (d_j) e quanto vale per lui (b_{ij}) e quanto sia probabile che sarà in grado di eseguire il download del contenuto ($p(d_i)$). Usando il fatto che $p(0) = 0$ e $p(\infty) = 1$ è possibile trovare i limiti della funzione di utilità

$$\lim_{d_i \to 0} u_i = 0 \qquad \lim_{d_i \to \infty} u_i = -\infty$$

2.3 Modello per il live streaming p2p ereditato dai sistemi di file-sharing

Per analizzare un sistema p2p progettato per il live streaming si può modificare il modello appena illustrato relativo ad un sistema di file sharing. Nel caso del file sharing il contributo D_i rappresenta una quantità fisica di spazio occupato sul disco, quindi si fa riferimento alla quantità di file che un certo utente ha già ottenuto e che può condividere con gli altri utenti, nel caso invece di un sistema di live streaming il contributo viene inteso in termini di banda disponibile. Nel live streaming quindi più un utente mette a disposizione la sua bandwidth più verrà considerato un buon giocatore e avrà un alto valore di condivisione. Ovviamente chi dispone di una banda maggiore condividerà di più, ciò non è solo dettato dalla volontà di un certo peer a condividere più o meno banda ma è una conseguenza del tipo di connessione di cui si dispone e del tipo di elaboratore che si utilizza.

Nel caso del live streaming esistono dei comportamenti particolari che determinano anche i fattori di condivisione dell'intero sistema, la presenza di alcuni nodi particolari che vengono detti nodi stabili, ovvero di quei peer che si trattengono all'interno della rete per una gran quantità del tempo saranno caratterizzati da un alto valore di condivisione, ciò non vuol dire semplice-

mente che accettano di buon grado la condivisione ma che sono interessati a vedere tutto il video in streaming o almeno una buona parte di esso, quindi una volta che hanno acceduto al sistema rimangono all'interno della rete evitando comportamenti da free rider.

Sempre restando nell'ambito del live streaming esiste una concezione di seed un po' diversa rispetto a quella conosciuta nell'ambito del file sharing, essendo il live streaming una trasmissione di un video per un periodo di tempo limitato, esiste un seed principale che è la sorgente dalla quale inizia la trasmissione, gli altri client sono dei peer classici e quando uno di essi si trova ad aver visto tutto lo streaming può essere definito come un seed anche se in questa situazione nessun peer è incentivato a rimanere connesso alla rete in quanto non ne ricaverà nessun ulteriore beneficio.

2.4 Equilibrio di Nash in un sistema omogeneo di peer

Un sistema omogeneo di peer è un sistema nel quale $b_{ij} = b$ per tutti gli $i \neq j$, in altre parole in questo sistema tutti i peer derivano un uguale benficio da tutti gli altri. Questo sistema semplificato permette di studiare il problema in una situazione idealizzata, e ottenere intuizioni che possono essere applicate al più complesso sistema eterogeneo. Nel sistema omogeneo la funzione di

utilità

$$u_i = -d_i + p(d_i) \sum_j b_{ij} d_j,\ b_{ii} \equiv 0$$

si riduce a

$$u = -d + p(d)(N-1)bd.$$

$b_i = b_{av} = b(N-1)$ per tutti i peer P_i.

2.4.1 Gioco a due giocatori

In un gioco a due giocatori la funzione di utilità si riduce a

$$u_1 = -d_1 + b_{12} d_2 p(d_2)$$

$$u_2 = -d_2 + b_{21} d_1 p(d_2)$$

Per semplicità algebrica si assume che $\alpha = 1$ cioè $p(d) = d/(1+d)$. Se b_{12} e b_{21} sono troppo piccoli allora sarà meglio per loro non partecipare al sistema. La questione è di chiedersi a questo punto se esiste un equilibrio di Nash abbastanza grande in termini di valore di beneficio dove entrambi i peer possano derivare un'utilità non nulla dalla loro interazione. Supponendo che P_2 decida di dare un contributo pari a d_2 al sistema, dato questo contributo d_2, naturalmente la miglior cosa per P_1 è di dare il suo contributo d_1 in

modo tale che massimizzi la sua utilità u_1. Massimizzando u_1 rispetto a d_1, immediatamente troviamo la best response d_1 che è data da:

$$r_1(d_2) \equiv d_1 = \sqrt{b_{12} d_2} - 1$$

dove $r_1(d_2)$ è nota come la funzione reazione per P_1, questa è la miglior reazione per P_1 data una strategia fissa per P_2. Da quando P_2 sa che P_1 risponderà in questo modo, la sua funzione reazione alla strategia di P_1 sarà:

$$r_2(d_1) \equiv d_2 = \sqrt{b_{21} d_1} - 1$$

L'equilibrio di Nash esiste se c'è un set di (d_1^*, d_2^*), tale che e questi punti fissi soddisfino

$$d_1^* = \sqrt{b_{12} d_2^*} - 1$$

$$d_2^* = \sqrt{b_{21} d_1^*} - 1$$

Trovando il punto fisso è molto più facile assumendo $b_{12} = b_{21} = b$. In questo caso $d_1^* = d_2^* = d^*$, ovvero:

$$d^* = (b/2 - 1) \pm ((b/2 - 1)^2 - 1)^{1/2}$$

una soluzione a questa equazione esiste se e solo se $b \geq 4 \equiv b_c$. Quindi, $b_c = 4$ è il valore critico del beneficio sotto il quale non c'è profitto per un

peer se accede al sistema. Per scelte diverse di $p(d)$ la costante b_c cambierà, ma sarà sempre una costante indipendente dal numero dei peer nel sistema. Per $b = b_c$, l'unica soluzione è $d_1^* = d_2^* = 1$. Per $b > b_c$ ci sono due soluzioni $d_1^* = d_2^* = d_{lo}^* < 1$ e $d_1^* = d_2^* = d_{hi}^* > 1$.

2.4.2 Gioco con N giocatori

Nel sistema omogeneo di peer si ha

$$d^* = \sqrt{b(N-1)d^*} - 1$$

quindi con la sostituzione di b con b(N-1) i risultati dei due sistemi di peer sono esattamente applicabili per un sistema con N peer. Sebbene il sistema omogeneo dei peer non sia realistico, le proprietà medie degli equilibri di Nash per un sistema eterogeneo seguono molto da vicino il caso omogeneo.

2.5 Equilibrio di Nash in un sistema eterogeneo di peer

In un sistema eterogeneo, è necessario rivedere la complessità del modello. Le equazioni con il punto fisso per $\alpha = 1$ possono immediatamente essere derivate in analogia con il gioco a due giocatori come:

$$d_i^* = \left[\sum_{j \neq i} b_{ij} d_j^* \right]^{1/2} - 1.$$

Dal momento che non è possibile risolvere questo insieme di equazioni analiticamente, si usa un modello iterativo per risolvere il sistema.

Considerando l'interazione degli utenti in un sistema p2p reale ogni particolare peer P_i interagisce solo con un insieme limitato di possibili peer, che sono i peer che forniscono P_i dei file a cui è interessato. Tramite l'interazione con questi peer P_i apprende i contributi versati da essi e per massimizzare la propria utilità "aggiusta" la propria contribuzione. Ovviamente questo contributo che dà P_i non è globalmente ottimale perchè è basato solo sulle informazioni ottenute da un limitato set di peer. Dopo che P_i ha messo a disposizione il suo insieme di contributi, questa informazione sarà propagata ai peer con i quali lui interagisce e questi a loro volta aggiusteranno la propria contribuzione. In questo modo le azioni di ogni peer P_i raggiungeranno ogni peer possibile. La reazione dei peer al contributo di P_i influenzerà P_i stesso e troverà che forse sarebbe meglio aggiustare il suo contributo un'altra volta. In questo modo ogni peer andrà avanti in un processo iterativo per sistemare il proprio contributo. Se e solo quando questo processo convergerà, le risultanti dei contributi costituiranno un equilibrio di Nash.

Capitolo 3

La Performance

Questo terzo capitolo si soffermerà sull'analisi della performance dei sistemi p2p, il primo paragrafo sarà relativo alle performance in architetture basate sui multi-tree e sulle reti mesh, il secondo si occuperà dell'illustrazione delle metriche utilizzate per l'analisi della performance, il terzo riguarderà l'identificazione dei nodi stabili utili per rendere il servizio più affidabile mentre l'ultimo paragrafo verrà riservato per la presentazione di un sistema chiamato AnySee che risulta essere una buona soluzione per il miglioramento della performance.

3.1 L'importanza della performance

La performance che viene garantita da un sistema p2p risulta essere un punto importante in quanto grazie ad essa è possibile convincere i diversi utenti a

rimanere all'interno della rete condividendo file, evitando quindi il fenomeno del free-riding, in modo tale da aumentare sempre di più la mole dei dati scambiabili e l'incremento sempre maggiore della performance stessa. Per quanto riguarda il live streaming infatti con l'aumento del numero di peer connessi alla rete il livello di performance cresce in proporzione a tale incremento garantendo appunto una performance maggiore nella trasmissione del file in streaming. I peer vengono quindi incoraggiati a giocare il duplice ruolo di client/server, una volta che un peer ha scaricato tutto il file esso viene convinto a rimanere connesso e a diventare quindi un server in modo tale da aumentare il throughput dell'applicazione. Un utente viene incoraggiato alla partecipazione nel ruolo di server grazie all'incentivo proposto dalla performance, ovvero più un peer contribuisce alla diffusione di un file più la sua performance a livello di velocità di download aumenta, viene sfruttata l'idea che sta alla base dei comuni programmi di file sharing.

Come è stato descritto nel primo capitolo sono due i modelli più utilizzati nella gestione dei sistemi p2p, la prima architettura è basta sui multi alberi e la seconda invece è basta sulle reti mesh.

3.1.1 Performance in multiple-tree-based

Il primo modello analizzato è quello sui multi-alberi, è possibile costruire alberi caratterizzati o da una larghezza minima o da una profondità minima; in entrambi i casi è possibile riscontrare dei pro e dei contro. Nel caso di minima larghezza, la gestione degli alberi è abbastanza semplice, ma in questa

particolare conformazione i cammini tra i diversi peer risultano essere molto lunghi. Prendendo invece in considerazione gli alberi con una profondità minima, è possibile rendere il problema dell'abbandono dei nodi interni relativamente di scarsa importanza, dato il grande numero di peer presente su ogni livello la corretta ricezione da parte dei nodi più esterni dovrebbe essere garantita; adottando questa tipologia di albero si riduce notevolmente anche la latenza causata dai collegamenti presente su alberi di larghezza minima.

3.1.2 Performance in mesh-based

Applicando la topologia basata sulle reti mesh e messa in collaborazione con il metodo di *file-swarming* utilizzato da BitTorrent, ovvero lo sfaldamento del file da trasmettere in pezzi che verranno ricevuti da peer diversi, la performance di tutto il sistema migliora rispetto all'architettura basta su multiple-tree.

E' possibile generare una sorta di gerarchia padre-figlio anche nell'architettura a rete mesh, ovviamente essa non è semplice come nello schema ad albero. La radice della rete viene indicata al livello *0*, mentre gli altri peer avranno un valore pari al numero di hop che li separa dalla root, questo valore serve per identificare a che livello dell'albero si trova il nodo; così i peer distanti un salto dalla root saranno al livello *1*, quelli distanti due salti si troveranno al livello *2* e così fino all'ultimo livello n[12].

3.2 Le metriche della performance

Tra le metriche che possono essere utilizzate per analizzare la performance di un sistema p2p è possibile identificare la *service capacity* e la *fairness* (equità).

3.2.1 Service capacity

Il concetto che sta alla base della service capacity è piuttosto complesso e può essere analizzato da due punti di vista [6]: quello del sistema e quello dell'utente. Nel momento in cui i peer si trovano a giocare i diversi ruoli di client e server la service capacity e la performance stessa di tutto il sistema dipendono dall'*offered load* ovvero dal numero dei peer che fanno parte della rete.

Dal punto di vista del sistema è possibile fare riferimento alla metrica "aggregate upload service capacity", essa rappresenta il throughput maggiore che il sistema può offrire ai peer che vogliono scaricare un documento. Questa metrica calcola l'upload effettivo della banda sia nel caso che i peer siano dei *seed*, ovvero siano in possesso del file completo, sia nel caso che essi siano dei peer di tipo client. L'effettività della banda disponibile fa riferimento al fatto che i peer che effettuano l'upload utilizzano la loro banda disponibile mentre quelli che effettuano unicamente il download possono attingere solo ad una parte della banda per effettuare l'upload dato che essi non sono in

possesso del file completo.

Per quanto riguarda invece il punto di vista degli utenti è possibile introdurre una nuova metrica per il calcolo della service capacity ovvero il "per peer download throughput", che in particolare rappresenta il throughput medio di download ottenuto per ogni peer, che può comunque essere generalizzato facendo riferimento alla metrica precedente normalizzandola al numero di peer che si trovano nello stato di download.

La service capacity è una metrica dinamica perchè è in continua variazione, essendovi continuamente una frazione di peer che abbandona il sistema ed un'altra che vi accede, è possibile individuare due fasi nelle quali vengono valutate le performance, che prendono il nome di regime transitorio e regime stazionario.

La prima fase corrispondente al regime transitorio si sofferma sulla performance del sistema nel caso in cui i client facciano richiesta di un file molto popolare che è appena stato messo all'interno della rete. E' chiaro che nei primi tempi di questa fase solo alcuni dei peer si trovano ad avere una copia del file che possono mettere a disposizione degli altri utenti; man mano che si va avanti nel tempo, il file diviene più popolare e la service capacity del sistema cresce esponenzialmente. Nel momento in cui il numero delle richieste del file si stabilizzano si passa alla seconda fase, infatti il sistema è passato da una fase di richiesta non costante ad un regime stazionario, nel quale il throughput dei singoli peer è stabile.

Analisi transitoria della service capacity

L'analisi transitoria della service capacity si focalizza sulla ricerca di quanto velocemente essa riesca a gestire un elevato numero di richieste contemporanee. Infatti risulta molto importante la gestione della diffusione di un file che è appena stato messo a disposizione della rete p2p, tipicamente il file viene introdotto nel network da un peer che lo detiene completo, se ci si trova nel caso in cui un grande numero di peer voglia ottenere tale file, la service capacity risulta non essere sufficiente; per ovviare a questo problema si procede nel modo seguente: è necessario che il file venga diffuso nel modo più veloce possibile prima che si passi alla fase di regime stazionario nella quale la service capacity è messa in relazione con il numero delle richieste.

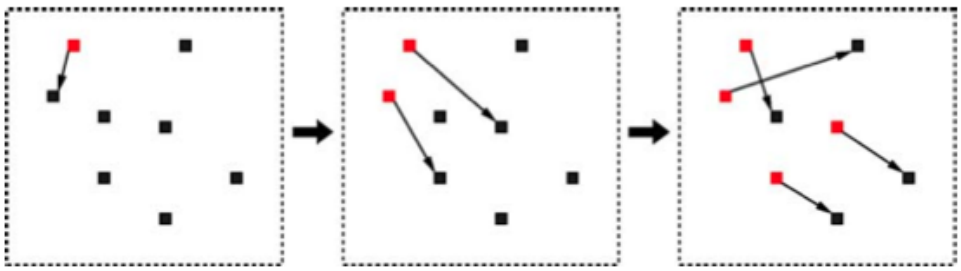

Figura 3.1: Funzionamento di un sistema p2p

Il modello standard secondo cui avviene la trasmissione standard dei file in un sistema di file sharing è illustrato dalla figura 3.1: il peer che è in possesso di tutto il file completo, che viene indicato come il server di partenza, trasferirà tale file ad uno dei client e dopo che quest'ultimo avrà ricevuto tutti

dati sarà possibile effettuare due connessioni contemporanee per la continuare la diffusione delle informazioni, il numero di trasmissioni possibili crescerà quindi in maniera logaritmica.

Il modello deterministico attraverso il quale viene analizzato il regime transitorio è caratterizzato da un numero pari a uno per quanto riguarda il numero dei peer iniziali ed n richieste del file posseduto dal primo peer. Ogni peer ha un limite di *upload capacity* pari a b bps, mentre la *download capacity* non ha un limite preimpostato e il documento posseduto dal primo peer ha una grandezza pari a s bit. Perciò per servire tutte le n richieste è necessario che vengano scambiati ns bit all'interno della rete. Il metodo ottimale per la diffusione del file agli n peer è quello di servire prima un peer ad un rate di trasmissione di b bps, dopodichè è chiaro che la trasmissione potrà avvenire a $2b$ bps, andando avanti in questo modo fino a quando tutti gli n peer saranno serviti. Tutti gli n peer avranno completato il download del file in un tempo pari a $\frac{s}{b} log_2 n$ secondi. E' possibile calcolare il ritardo di download medio nel seguente modo:

$$\bar{r} = \frac{1}{n}\sum_{j=1}^{n} r_j = \sum_{i=0}^{k-1} 2^{i-k}\tau(i+1) = k\tau - \frac{n-1}{n}\tau = \tau(log_2 n - \frac{n-1}{n}) \approx \tau log_2 n$$

Con $\tau = \frac{s}{b}$, r_j indica il ritardo del j-esimo peer per completare il download.

E' comunque possibile aumentare la performance utilizzando la tecnica

del *multi-part downloads*, l'idea è quella di dividere ogni file in m chunks di uguale dimensione. Quindi con l'introduzione di tale tecnica non è più necessario attendere che l'intero file sia stato scaricato ma appena un chunk viene ottenuto è possibile passare al download di un chunk diverso. Tipicamente il server manderà ogni chunk del file iniziale ad un peer diverso, in modo tale che al termine del download di ogni parte del file i peer possano scambiarsi tra di loro il chunk appena ottenuto. Questo procedimento continua fino a quando ogni peer non è in possesso di un chunk del file, al tempo di slot *k,* in questo momento è possibile dividere gli n peer in k insiemi $A_i, i = 1, \ldots, k$ con $|A_i| = 2^{k-i}$, A_i corrisponde all'insieme dei peer che hanno ricevuto l'i-esimo chunk. Ovviamente in questo modo, è possibile effettuare download e upload contemporanei di chunk diversi, e quindi anche la velocità per ottenere tutto il file sarà aumentata. In questo caso è possibile completare la trasmissione del file in $\frac{s}{bm} = \frac{\tau}{m}$ secondi.

$$\bar{r^{(m)}} = \frac{1}{n}\sum_{j=1}^{n} r_j^{(m)} = \frac{1}{2}((k+m-1)+(k+m))\frac{\tau}{m} = \frac{\tau}{m}(log_2 n + \frac{2m-1}{2}) \approx \frac{\tau}{m} log_2 n$$

Analisi stazionaria della service capacity

Dopo aver iniziato il processo di scambio di dati in regime transitorio si accede alla fase di regime stazionario, si prende come ipotesi che i diversi peer facenti parte della rete abbiano un interesse a cooperare perchè interessati al

download di un particolare documento, inoltre si ritiene necessario che esista almeno un peer che funga da server in modo tale da garantire la sopravvivenza del file da trasmettere. Le nuove richieste dei peer seguono una distribuzione di Poisson $P(\lambda)$, il sistema è caratterizzano da una coppia $(x, y) \in \mathbb{N} \times \mathbb{Z}^+$, dove x indica il numero delle richieste in corso, ovvero il numero dei peer che stanno scaricando al momento, mentre y indica il numero dei peer che hanno concluso il download e sono rimasti all'interno della rete fungendo da server e contribuendo alla service capacity. Per ottimizzare il download dei peer il file da scaricare si assume che sia diviso in chunk in modo tale da abilitare il multi-part download. In realtà anche i peer che non hanno completato tutto il download contribuiscono alla service capacity, il loro contributo non è totale ma parziale e viene indicato con la variabile η. La metrica aggregate upload service capacity è perciò proporzionale al numero dei peer all'interno della rete che viene indicata da $\mu(\eta x + y)$, dove μ indica il service rate per una richiesta singola. Come nei classici sistemi p2p ogni qualvolta un peer completa il download di un documento diventa un seed, ma può in ogni momento lasciare il sistema ad un rate γ.

Lo stato del sistema può quindi essere descritto come una catena di Markov, tramite una matrice Q nello spazio designato da $\mathbb{N} \times \mathbb{Z}^+$ nel quale si possono evidenziare tre cambiamenti di stato:

- $q((x, y), (x + 1, y)) = \lambda$, rappresenta la probabilità che un nuovo peer acceda del sistema p2p, il numero dei peer quindi passa da x ad $x + 1$ mentre il numero dei seed resta invariato, tale probabilità assume il

valore λ.

- $q((x,y),(x-1,y+1)) = \mu(\eta x + y)$, questa seconda probabilità descrive il cambiamento dello stato del sistema nel momento in cui un peer ha scaricato tutti i chunk e quindi passa dallo stato di peer allo stato di seed. In questo caso la probabilità sarà data dal rate per la richiesta singola (μ) moltiplicato per il rate parziale dei peer sommato al numero dei seed.

- $q((x,y),(x,y-1)) = \gamma y$, l'ultimo caso rappresenta la situazione nella quale un seed, ovvero un peer che ha ottenuto tutto il file in download decide di lasciare il sistema, il numero dei peer cha stanno scaricando non cambia ma il numero dei seed passa da y ad $y-1$. Come detto prima la probabilità che un seed decida di abbandonare la rete è rappresentata da un rate pari a γ.

In [6] vengono stabiliti dei valori per μ e η, per quanto riguarda il parametro η ad esso viene assegnato 0.5, in questo modo viene associato lo stesso valore a tutti i peer che stanno scaricando il file in streaming, tutti i peer vengono quindi messi sullo stesso piano sia nel caso che abbiano appena acceduto alla rete sia che essi abbiano quasi finito di scaricare i chunk. Un suggerimento che potrebbe rendere il valore di η più equo sarebbe quello di assegnare ad ogni peer un valore di η in base al numero di chunk ottenuti, per capire meglio considerando per esempio un file che viene diviso in 10 chunk (viene preso un numero basso di chunk per semplificare la comprensione), un peer

che è entrato nella rete da poco e che ha ottenuto solo un chunk avrà un valore di η pari a 0.1, man mano che i peer ottengono altri parti del file tale valore aumenterà, in questo modo un peer che ha ottenuto metà dei chunk avrà un valore pari a 0.5 mentre per finire un peer che avrà tutti i chunk tranne l'ultimo avrà un valore pari a 0.9.

3.2.2 Equità o *fairness*

La nozione di equità deriva dalla concezione che più un peer si trova a condividere con gli altri le informazioni, più il suo rate di download sarà maggiore, gli incentivi proposti dalla rete quindi sono puramente legati al througput che potrà ottenere ogni singolo peer.

Quando ci si trova ad avere un sistema nel quale il numero dei peer che accedono alla rete è molto grande risulta difficoltoso assegnare ad ogni peer un equo limite di download. Nel momento in cui un peer accede alla rete non è possibile sapere se ha già partecipato alla condivisione di file in passato, una soluzione alla questione di come il sistema si debba comportare è l'attuazione di una politica secondo la quale appena un peer entra nella rete, quindi in regime transitorio, gli si dà la possibilità di eseguire download ad un alto rate in modo tale che esso possa partecipare alla condivisione delle informazioni nel minor tempo possibile aumentando in tal modo anche la service capacity di tutto il sistema. Dopodichè si passa al regime stazionario nel quale l'equità sarà assegnata in base a criteri proporzionali a quanto il peer ha partecipato alla condivisione.

L'analisi dell'equità in regime stazionario viene fatta tenendo in considerazione i seguenti parametri: N indica l'insieme dei peer il cui numero totale è $|N| = n$ e $\mathbf{x} = (x_{ij}|i \neq j, i, j \in N)$ indica il servizio fornito da ogni peer, x_{ij} rappresenta la misura dell'allocazione delle risorse da i a j. Si suppone che ogni peer abbia un vincolo di upload dato da $\mathbf{b} = (b_i|i \in N)$ dove b_i è proprio il vincolo in upload per il peer i. Il set composto dai vicini del peer i viene indicato con $N_i = N \setminus \{i\}$.

Equità globalmente proporzionale

Un'allocazione $\mathbf{x}$ viene detta *globalmente proporzionalmente equa* se per ogni $i, j \in N$, con $i \neq j$ viene soddisfatta la seguente formula:

$$x_{ij} = \frac{u_j}{\sum u_k} b_i \qquad \text{dove } u_j = \sum_{i \in N_j} x_{ij} \ .$$

L'idea è quella di allocare la bandwidth uscente dal peer i verso il peer j in proporzione a quanto il peer j contribuisce al sistema, che corrisponde ad u_j.

$$x_{ij} = \frac{b_j}{\sum b_k} b_i \qquad \text{e } d_j = \sum_{i \in N_j} x_{ij} = b_j \sum_{i \in N_j} \left[\frac{b_i}{\sum b_k} \right] \ .$$

Il valore d_j risulta essere quindi l'aggregate download bandwidth ottenuta dal peer j. E' chiaro che l'equità globalmente proporzionale porta dei vantaggi, gli incentivi che offre ai peer sono proporzionali alla loro cooperazione all'interno del sistema.

Equità peerwise proporzionale

Esiste un'alternativa al concetto precedente, ovvero l'equità peerwise proporzionale, i peer in questo caso allocano la propria capacità in upload basandosi sul servizio che ricevono dagli altri peer. Per fare ciò è sufficiente tenere traccia dei download effettuati dai peer che hanno concesso il download.

Un'allocazione $\mathbf{x}$ viene detta *peerwise proportional fair* se per ogni $i, j \in N$, con $i \neq j$ viene soddisfatta

$$x_{ij} = \frac{x_{ij}}{\sum x_{ki}} b_i$$

sebbene ciò sembri chiaramente equo, è possibile che non esista un'unicità nelle allocazioni (a differenza dell'equità globalmente proporzionale che risulta unica), portando quindi ad una non equità generale.

Equità ε-peerwise proporzionale

Mentre le due equità precedenti prendevano in considerazione un sistema p2p completamente connesso in quest'ultimo caso si fa riferimento ad un sistema che rispecchia di più la realtà e il quale è caratterizzato da un comportamento dinamico dei peer che si trovano ad accedere o lasciare il sistema in maniera continuativa. Per eliminare il problema relativo all'equità peerwise proporzionale, può essere preferibile garantire che esista un'unica soluzione positiva e distribuire meccanismi per la ricerca di essa. Una semplice idea è quella di

assicurare che un peer continui a scambiarsi chunk con un altro in modo tale da evitare la disconnessione. Questa strategia è quella comunemente adottata dai sistemi BitTorrent, nei quali i peer sono tenuti a scambiarsi parti di file in continuazione e non esistono delle code di peer come invece accade in altri tipi di sistemi di file sharing. L'idea può essere modellata garantendo che ci sia uno scambio minimo tra tutti i peer facenti parte del sistema, viene anche indicata una allocazione persistente di risorse che fa un uso medio della bandwidth a lungo termine ε. Detto ciò è possibile definire una allocazione di bandwidth $\mathbf{x}$ come *ε-peerwise proportional fair* per tutti gli $i, j \in N$ con $i \neq j$ se soddisfa:

$$x_{i,j} = \frac{x_{i,j} + \epsilon}{\sum(x_{k,i} + \varepsilon)} b_i$$

Queste allocazioni sono uniche anche se i b_i non sono uguali. Supponendo che le allocazioni siano uniche si possono mostrare le seguenti proprietà dell'equità ε-peerwise proporzionale: in primo luogo, per ε sufficientemente piccolo, ci si aspetterebbe che l'allocazione risultante fosse vicino al valore uno che è strettamente positivo e soddisfa i requisiti peerwise proportional fair, se invece ε è grande un peer dovrebbe ripartire la sua banda in upload in parti uguali tra i suoi vicini; in secondo luogo, il throughput di download d_i di un peer aumenta con la sua bandwidth di upload b_i.

Chiaramente il sistema non deve essere esattamente "equo" per migliorare gli incentivi, vale a dire, il concetto di bisogno di 'equità' richiede solo

vagamente che 'più upload fai più avrai migliori prestazioni di download', infatti gli utenti sono dei giocatori razionali e il loro comportamento non è controllato solo dalla performance di download percepita rispetto alla sua velocità di upload.

3.2.3 Fairness come problema di massimizzazione

Per analizzare un classico sistema p2p si può indicare con P il set di peer che fanno parte della rete e con S un set di servizi; in particolare un servizio viene definito come un'allocazione di risorse. Ogni peer $p \in P$ è interessato ad usufruire di uno o più servizi, e offre uno o più servizi allo stesso tempo.

Si assume che alcune risorse siano scarse ovvero che la loro distribuzione sia un processo abbastanza critico e che ci sia competizione tra i peer. Per semplicità si assume che ci sia solo una risorsa in tutto il network che sia poco popolare e la sua upload capacity da parte del peer p viene indicata come Cp. Per differenziare tra servizio offerto e servizio richiesto nel modello matematico vengono introdotti il set di *service providers* SP e il set di *service customers* SC. Un peer è membro di SP o SC se, rispettivamente, offre o richiede ameno un servizio, un peer può anche far parte di tutti e due gli insiemi ed è il caso più comune che si può trovare nei sistemi p2p. Dal momento che ogni peer ha una visuale parziale dell'intero sistema p2p, un peer che richiede un servizio non è consapevole di tutti i peer che provvedono a quel particolare servizio e viceversa.

Viene inoltre definito il set di peer che offrono almeno un servizio al peer c, il service customer, e sono noti a c come il set di service providers $SP(c)$ del peer c. Analogamente $SC(p)$ è l'insieme dei client che ricevono un qualche servizio dal peer p.

Supponendo che l'utilità di un service customer c sia definita da una funzione di utilità U concava e strettamente crescente che dipende dal rate totale del servizio y_c è possibile modellare l'allocazione di risorse come un problema di ottimizzazione:

$$massimizzare \sum_{c \in SC} U_c(y_c)$$

$$tenendo\ conto\ che \sum_{p \in SP(c)} x_{pc} = y_c, \forall c \in SC$$

$$\sum_{c \in SC(p)} x_{pc} \leq C_p, \forall p \in SP$$

$$con\ x_{pc} \geq 0.$$

Massimizzare l'aggregate utility del servizio al rate y_c sul servizio di richiesta su tutti i peer è il goal di tutto il sistema. In tal modo y_c è il totale dei rate x_{pc} di tutti i servizi offerti dal peer p al peer c. Questo problema è limitato dalla capacità della risorsa ad offrire il servizio ai peer, cioè la somma dei service rate di un peer che offre il servizio deve essere più piccola od

uguale alla capacità di tale peer. Con una funzione di utilità concava questo problema di ottimizzazione ha un unico ottimo che rispetta y_c, sebbene molti rate possibili di allocazioni possano esistere rispetto ad x_{pc}.

Il lagrangiano è il seguente:

$$L(x, y; \lambda, \mu; s) = \sum_{c \in SC} \left[U_c(y_c) - \lambda_c \left(y_c - \sum_{p \in SP(c)} x_{pc} \right) \right] + \sum_{p \in SP} \mu_p \left(C_p - \sum_{c \in SC(p)} x_{pc} - s_p^2 \right)$$

dove λ_c e μ_p sono i moltiplicatori di lagrange e s_p è una variabile slack. I moltiplicatori di lagrange possono essere interpretati come prezzi per l'uso delle risorse; si può quindi dire che λ_c e μ_p sono i prezzi per unità offerti dal customer c e addebitati al provider p rispettivamente. Questo modello riflette un mercato per service capacity dove i prezzi controllano il rate delle allocazioni, dal momento che la service capacity è un bene non immagazzinabile il mercato è di tipo a pronti dove la domanda corrente influenza i prezzi futuri.

Una carenza delle reti p2p è il problema della mancanza di incentivi a fornire risorse agli altri peer della rete. La maggior parte degli utenti sono free-riders, che risultano essere beneficiari di un servizio e non condividono le loro risorse disponibili. Per controllare questo comportamento di condivisione dei file le reti p2p implementano un sistema basato su crediti, in cui vengono premiati i peer con un tasso di servizio più rapido in base al loro contributo delle risorse alla rete.

Ci si trova di fronte il problema di free-riding, garantendo una sorta di fairness nella rete impostando la funzione di utilità in in modo adeguato, in particolare la funzione di utilità di tutti i peer sarà pari a $U_c(y_c) = w_c ln(y_c)$, dove w_c indica la disponibilità a pagare. Questa funzione di utilità assicura un'allocazione delle risorse proporzionale, un'allocazione $y = (y_c, c \in SC)$ è *weighted proportional fair* se è fattibile e se per ogni altra allocazione fattibile y' risulta che

$$\sum_{c \in SC} w_c \frac{y'_c - y_c}{y_c} \leq 0.$$

Poiché l'obiettivo è quello di massimizzare l'utilità aggregata del tasso di servizio totale, la funzione di utilità assicura la correttezza proporzionale rispetto a y_c. Questa equità si realizza su una prospettiva globale, dove i peer sono serviti indipendentemente dal loro numero di connessioni. Questo è in molti casi diverso da una equità locale in cui ogni service provider divide equamente la sua capacità tra sè stesso e i client a lui connessi. All'ottimo il tasso di servizio totale dipende solo da un prezzo unico e quindi la fairness proporzionale è equivalente alla max-min fairness in questo contesto. Un tasso di allocazione è detta max-min fair se è fattibile e, per ogni $c \in SC$, y_c non può essere aumentata senza diminuire il tasso $y_{c'}$ di un servizio clienti c' con $y_{c'} \leq y_c$. Questo concetto ottimizza la velocità del peer con la ripartizione percentuale minima.

Un altro parametro che studia l'equità è l'indice di equità di Jain. In

questo contesto per un set di rate di servizi $(y_1, y_2, ..., y_n)$, l'indice di equità è calcolato con

$$f(y_1, y_2, \ldots, y_n) = \frac{(\sum_{i=1}^{n} y_i)^2}{n \sum_{i=1}^{n} y_i^2}.$$

Questa metrica ha un valore che si trova tra 0 ed 1, valori alti indicano un'equità migliore. Se tutti i peer ricevono un'uguale service rate , l'indice sarà pari ad 1. Dall'altra parte l'indice di equità di Jain sarà $1/n$ se tutte le risorse sono allocate ad un unico utente e gli altri $n - 1$ utenti non ricevono nulla.

3.3 Nodi stabili

All'interno di un sistema p2p è possibile identificare un sottoinsieme dei peer che vengono detti nodi stabili, come suggerisce il nome questi peer vengono in aiuto alla sorgente, sono i nodi che rimangono sempre connessi e si occupano della distribuzione del file in streaming verso gli altri peer che invece lasciano la rete quando vogliono.

Prendendo in considerazione T come il tempo corrente, $X(t)$ è il numero di nodi arrivati al tempo t e $F(x)$ è la funzione di ripartizione della vita del nodo. Quindi il numero totale dei nodi al tempo T è

$$\sum_{i=0}^{T-1} X(i) \cdot (1 - F(T-1)).$$

Ponendo l come il minimo valore della vita di un nodo per essere considerato stabile il numero totale dei nodi stabili al tempo T è :

$$\sum_{i=0}^{T-1} \left\{ X(i) \cdot (1 - F(T-i)) \cdot I_{[T-i \geq l]} + X(i) \cdot (1 - F(l)) \cdot I_{[T-i<l]} \right\}$$

$$= \sum_{i=0}^{T-1} X(i) \cdot (1 - F(t-i) + \sum_{i=T-l+1}^{T-1} X(i) \cdot (1 - F(l))$$

dove con I viene indicata la funzione identità.

In un sistema p2p è noto che la vita di un nodo può essere approssimata con la distribuzione di Pareto [5] , essa è definita in questo modo:

Se X è una variabile aleatoria con una distribuzione di Pareto allora la probabilità che X sia più grande di un qualche numero x è data da

$$P(X > x) \begin{cases} \left(\frac{x_m}{x}\right)^k & per\ x \geq x_m \\ 1 & per\ x < x_m \end{cases}$$

dove x_m è una quantità positiva corrispondente al minimo valore possibile di X e k è un parametro positivo.

La funzione di ripartizione nella distribuzione di Pareto con i parametri x_m e k è quindi la seguente:

$$F_X(x) \begin{cases} 1 - \left(\frac{x_m}{x}\right)^k & per\ x \geq x_m \\ 0 & per\ x < x_m \end{cases}$$

Se si considera che un nodo arriva ad un rate μ si può calcolare il rapporto atteso dei nodi stabili al tempo T nel seguente modo:

$$\frac{\sum_{i=0}^{T-l} E(X(i)) \cdot (1 - F(t-i)) + \sum_{i=T-l+1}^{T-1} E(X(i)) \cdot (1 - F(l))}{\sum_{i=0}^{T-1} E(X(i)) \cdot (1 - F(T-i))}$$

$$= \frac{\sum_{i=0}^{T-l} \mu \cdot \left(\frac{T-i}{x_m}\right)^{-k} + \sum_{i=T-l+1}^{T-1} \mu \cdot \left(\frac{l}{x_m}\right)^{-k}}{\sum_{i=0}^{T-x_m} \mu \left(\frac{T-i}{x_m}\right)^{-k} + \mu \cdot (x_m - 1)}$$

$$= \frac{\sum_{i=0}^{T-l} \frac{1}{(T-i)^k} + \frac{l-1}{l^k}}{\sum_{i=0}^{T-x_m} \frac{1}{(T-i)^k} + \frac{x_{m-1}}{x_m^k}}$$

$$= \frac{\sum_{i=l}^{T} \frac{1}{i^k} + \frac{l-1}{l^k}}{\sum_{i=x_m}^{T} \frac{1}{i^k} + \frac{x_m - 1}{x_m^k}}$$

In [5] viene introdotto lo *stability index* (s-Index) per indicare il grado di stabilità di un nodo, un valore pari a 0 indica che il nodo non è stabile mentre al contrario un valore molto vicino ad 1 indica una maggiore stabilità del peer. Lo stability index viene calcolato nel seguente modo:

$$SI = \begin{cases} \frac{2s}{L-t} & se\ s \leq \frac{L-t}{2} \\ 1 & altrimenti \end{cases}$$

s indica la permanenza del nodo all'interno della sessione, t è il tempo di arrivo ovvero il momento in cui il nodo ha acceduto alla rete mentre L rappresenta la lunghezza (durata) della sessione.

Il calcolo dell'indice di stabilità suggerisce che più un nodo rimane stabile all'interno della rete più nel futuro esso continuerà a rimanere stabile.

A questo punto sapendo qual è lo stability index per un nodo sarà facile identificare quale altro nodo che al momento si trova nel gruppo dei nodi non stabili diventerà stabile ponendo una soglia H oltre la quale un nodo passerà dall'*instabilità* (dall'insieme dei nodi non stabili) alla *stabilità* (all'insieme dei nodi stabili).

L'utilizzo della tecnica tramite la soglia per decidere quando un nodo diventa stabile, tuttavia, soffre di due svantaggi: all'inizio della sessione, nessun nodo, è considerato stabile, e ci vuole molto tempo per costruire la sovrapposizione di più livelli per fornire dati efficienti e in secondo luogo una flash crowd ovvero un folla di peer che si connette insieme avrà un impatto doppio, cioè, quando un gran numero di nuovi nodi aderisce contemporaneamente e quando questi nodi sono identificati stabili simultaneamente. Si possono risolvere tali inconvenienti tramite degli algoritmi randomizzati. L'algoritmo si adopera per ottenere una probabilità SI/H per un nodo di essere predetto

stabile. Per realizzare la probabilità linearmente crescente SI/H, ogni nodo che si trova nel livello 2 controlla il suo stato per ogni unità di tempo, per il controllo s-esimo (cioè, al tempo $t+s$), il nodo sarà promosso con probabilità $2/((L-t)\cdot(H-SI)+2)$.

3.3.1 Labeled Tree

Tenendo in considerazione la presenza dei nodi stabili che possono essere inseriti in un primo livello di overlay è necessario riuscire ad organizzare tali peer in modo tale che riescano ad offrire i migliori servizi ai nodi che si trovano al secondo livello, in questo senso le caratteristiche che interessano maggiormente sono l'evitare le perdite continue di dati e un bon metodo per il recupero dei dati persi. E' infatti evidente che una perdita consistente di dati da parte di un peer si potrà estendere molto facilmente ai suoi discendenti. Un'altro appunto riguarda il fatto di non confondere i nodi stabili con nodi che di fatto sono più persistenti come router o proxy, inoltre è possibile che la predizione dei nodi stabili non sia stata effettuata in modo accurato e sia quindi ingannevole.

Per far luce su queste ultime questioni è stato introdotto un nuovo overlay per i nodi che si trovano al livello uno e che è stato definito Labeled Tree (LBTree) pensato appositamente per i nodi stabili, che permette un recupero dei dati persi e una miglioria rispetto alle tradizionali architetture basate sugli alberi.

Un LBTree è caratterizzato da un'etichetta che per ogni nodo indica la sua posizione all'interno del livello 1, le etichette sono del tipo $(R, (a, b))$, dove R indica in quale livello risiede il nodo, mentre (a, b) rappresenta un intervallo calcolato ricorsivamente in questo modo: partendo dall'etichetta del nodo e dal suo numero massimo di figli N, il suo range sarà diviso equamente in N sottoranghi non sovrapposti ognuno di questi assegnato ad uno dei figli. Quindi per un nodo standard con etichetta $(R, (a, b))$ che ha due figli, le etichette dei due figli saranno rispettivamente $(R + 1, (a, (a + b)/2)$ e $(R + 1, ((a + b), b))$. La tecnica delle etichette è utile in quanto grazie ad essa è possibile risalire alla relazione che intercorre tra due nodi.

La costruzione dell'albero avviene nel seguente modo: per prima cosa tutti i nodi tranne la sorgente si trovano nel secondo livello dell'overlay, vengono quindi considerati tutti dei nodi non stabili, l'LBTree è quindi formato unicamente dalla root e man mano che i nodi vengono classificati come nodi stabili entrano a far parte dell'albero tramite l'associazione di etichette. Lo stability index atteso è generalmente più alto per i nodi che si trovano vicino alla sorgente e dato che l's-index aumenta con il tempo questa relazione risulta essere persistente durante tutta la durata della sessione di download.

Nel primo livello dei nodi stabili non solo i nodi sono caratterizzati da una forte stabilità ma anche le connessioni tra di essi rispetto ai nodi considerati non stabili godono di questa proprietà, è possibile quindi intuire che la perdita dei dati non risulta essere così critica come lo è nel secondo livello.

Soluzioni che riguardano il problema della perdita dei dati all'interno

dell'albero di comunicazione è lo stabilire *side link* tra coppie di nodi per il recupero dei dati persi. I side link fanno un uso effettivo della *bandwidth remainder* ovvero del residuo di banda che non può supportare uno streaming completo, per trasmettere una richiesta di recupero e la sua risposta.

Ogni nodo può mantenere multipli side link e prendere per esempio uno di essi per una richiesta di recupero per ogni blocco perso.

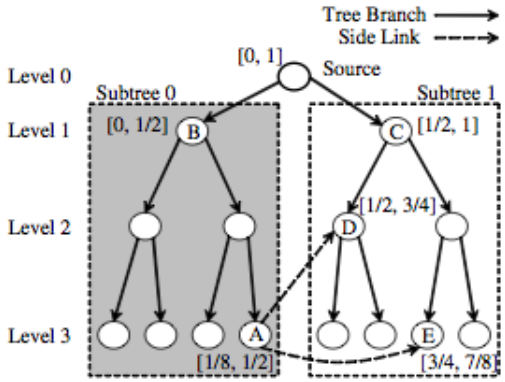

Figura 3.2: Labeled tree con side links

Per il corretto recupero dei dati persi tramite side link vengono seguiti questi criteri:

- un side link che parte da un nodo dovrebbe essere connesso ad un nodo che è molto vicino alla sorgente. In questo senso se due nodi che instaurano un side link tra di essi e che si trovano allo stesso livello dell'LBTree è indifferente da quale nodo parta il side link, invece nel caso in cui i due nodi si trovino a due livelli diversi il side link necessariamente partirà dal nodo che si trova più lontano dalla sorgente e

finirà in quello che invece si trova più vicino ad essa, in questo modo viene infatti garantita una buona probabilità di recupero dei dati.

- i due estremi di un side link non dovrebbero condividere nessun antecedente eccettuata la sorgente che viene detta ortogonale. Infatti se due nodi facessero parte di uno stesso sottoalbero non beneficerebbero dell'utilità del side link perchè il recupero dei dati mancanti potrebbe non portare a nessun risultato.

3.4 AnySee: un miglioramento della performance

Con la richiesta da parte degli utenti di avere dei sistemi che offrano un servizio efficiente, vengono studiati sul campo diversi metodi che permettano di migliorare la performance di questi sistemi, uno tra questi è AnySee [16].

Prendendo in considerazione i sistemi p2p è facile comprendere come esista sia una topologia fisica che una topologia logica della rete, la parte (a) e (b) della figura 3.3 illustra come i peer siano logicamente connessi o alla sorgente S_1 o alla sorgente S_2 dello streaming mentre in (c) viene mostrata la topologia fisica dei client.

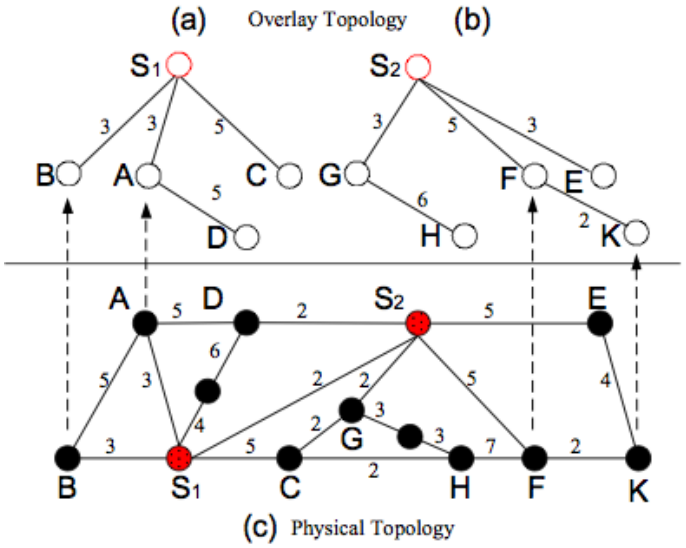

Figura 3.3: Topologia logica e fisica di una rete p2p

Facendo un confronto delle due architetture è possibile verificare che gli alberi non sono una scelta ottimale, nel caso del peer D per esempio dalla parte (a) si vede che esso disti 8 hop da S_1 mentre considerando la topologia fisica e prendendo in considerazione il path $S_1 \rightarrow S_2 \rightarrow D$ il numero di hop viene dimezzato.

AnySee si propone come uno schema attuabile per aumentare le prestazioni dei sistemi p2p, l'idea che sta alla base è quella di permettere ai peer di poter partecipare nello stesso momento a più di un albero di trasmissione dei dati; in questo modo l'utilizzo delle risorse viene distribuito in maniera più omogenea all'interno della rete fisica evitando anche problemi di traffico, le risorse vengono allocate in base al proprio posizionamento e in base al ritardo (numero di hop) dai peer sorgente, permettendo anche la comunicazione con

nodi vicini anche se appartenenti a due alberi distinti.

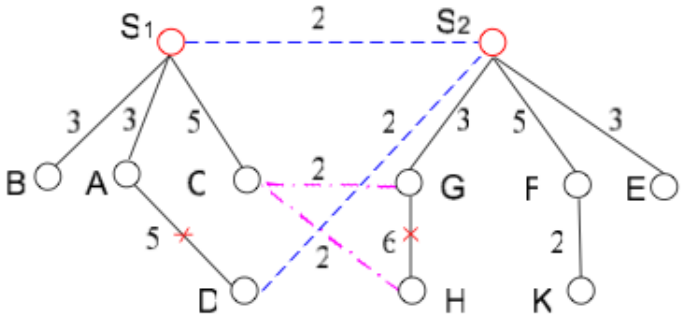

Figura 3.4: Esempio di architettura basata su AnySee

Perchè questo modello funzioni è necessario che alcune regole vangano rispettate, in particolare la scoperta e l'identificazione dei vicini deve avvenire in maniera efficiente come anche l'assegnazione delle risorse e la costruzione degli alberi.

3.4.1 Funzionamento AnySee

Quando un peer accede al sistema per prima cosa si connette con i peer che fanno già parte della rete e ne sceglie alcuni per costruire le proprie connessioni logiche, per fare ciò è necessario che tramite la rete fisica si riesca a realizzare la rete logica. Per permettere ai nuovi peer di riuscire a trovare i vicini ottimali, AnySee utilizza la tecnica LTM (Location-aware Topology Matching) per l'ottimizzazione di questo processo.

Tramite LTM è possibile effettuare due operazioni: l'identificazione dei peer tramite flooding e l'update delle connessioni logiche. Durante la prima

operazione, ogni peer manda un messaggio $dm(id, S, TTL)$ a tutti i peer vicini con la tecnica del flooding. Il messaggio serve per rendere pubbliche le impostazioni di un nodo ed in particolare un peer inizierà il suo messaggio con un identificatore id, e un tempo di hop uguale TTL (Time To Live). La seconda operazione si occupa dell'update dei collegamenti, ogni peer controlla i diversi path che lo collegano alla sorgente e decide di eliminare i collegamenti più lunghi, ovvero quelli che lo distanziano con più hop dalla sorgente.

Un peer P sotto una sorgente S con uno streaming rate $rate(S)$ mantiene attivo uno streaming path con una soglia $\delta_a(P, S)$ e un backup di streaming path con una soglia $\delta_b(P, S)$. Ogni streaming path SP_i che parte dalla sorgente S ed arriva al peer P ha due parametri: $delay(SP_i, S, P)$ che rappresenta il ritardo dalla sorgente S fino al peer di destinazione P; $rate(SP_i, S, P)$ è lo streaming rate nell'ultimo hop del path. Quindi si può scrivere che

$$\sum_{i=1}^{\delta_a(P,S)} rate(SP_i, S, P) \geq rate(S)$$

$$\sum_{i=1}^{\delta_b(P,S)} rate(SP_i, S, P) = p \sum_{i=1}^{\delta_a(P,S)} rate(SP_i, S, P)$$

Con $\mu_D(S)$ viene indicata la soglia per il delay; un path viene considerato non valido se la source-to-end delay è maggiore della soglia stabilita $\mu_D(S)$ o se il padre diretto del peer di destinazione lascia la rete.

Come è facile intuire i peer che si trovano ad un livello alto hanno un maggiore impatto sulla qualità del servizio che viene offerto ai peer che invece

si trovano nei livelli bassi dell'albero di trasmissione. AnySee attua uno schema basato sui livelli per bufferizzare lo streaming. La dimensione del buffer per un certo peer A che si trova al livello m è dato da

$$T_A = f(m) = \varepsilon \times t_A + t_A^{'}$$

dove con t_A viene indicato il ritardo totale sui collegamenti, t'_A è il ritardo totale del trasporto, e ϵ è il tempo medio di disconnessione.

Supponendo che la probabilità di un collegamento o di un nodo mancante sia P_b e che un peer abbia bisogno di un tempo t_b per trovare un nuovo nodo padre, il ritardo limite sarà quindi $t_l = P_b \times t_b$ e il ritardo di link t_A è l'accumulo dei ritardi limite. Supponendo inoltre che il ritardo di trasporto per hop sia μ e il numero totale di hop tra la sorgente e il peer A sia m, il ritardo totale del trasporto del peer A sarà $t'_A = \mu \times m$.

Se il path dalla sorgente è $l_{S \to A} = \left\{ l_{S \to a_1}, l_{a_1 \to a_2}, \ldots, l_{a_{m-1} \to A} \right\}$, il cammino avrà le seguenti proprietà:

- il peer sorgente rimarrà connesso per tutta la durata della trasmissione e il path $l_{S \to a_1}$ non sarà interrotto
- il peer A sarà nel network ed esisterà il path $l_{a_{m-1} \to A}$
- se un peer a_i lascia la rete un nuovo peer a'_i accederà alla rete e si metterà al posto del nodo che ha abbandonato il sistema

Il ritardo totale dei collegamenti sarà calcolato da:

$$t_A = \sum_{i=1}^{i=m-1} t_{l_{a_i \to a_{i+1}}}$$

dove

$$t_{l_{a_i \to a_i+1}} = (1 - P_b)^{i-1} \times t_l$$

poi

$$t_A = t_l + (1 - P_b) \times t_l + \ldots + (1 - P_b)^{m-2} \times t_l$$

infine si ottiene

$$t_a = t_l \times \frac{1 - (1 - P_b)^{m-1}}{P_b} = t_b \times (1 - (1 - P_b)^{m-1})$$

$$T_A = \varepsilon \times t_b \times (1 - (1 - P_b)^{m-1}) + \mu \times m$$

il massimo buffer per il peer A all'm-esimo layer sarà calcolato solamente in relazione al layer m.

3.4.2 Comportamento degli utenti

Il comportamento degli utenti influisce in maniera sostanziale sul sistema, overlay nei quali sono trasmessi programmi popolari come per esempio film sono caratterizzati da un alto numero di peer che vi accedono per poterli visualizzare ma questo causa anche degli ampi ritardi medi. Il ritardo co-

munque non risulta essere il motivo principale che causa la dipartita dei peer dal sistema, infatti nel caso preso in considerazione ovvero quando lo streaming riguarda una trasmissione molto popolare anche un ampio ritardo nella ricezione non influisce sul numero di peer che lasciano il sistema. Inoltre il ritardo considerato non influisce molto in quanto i peer sono disposti ad accettare anche un delay che si aggira intorno ai 30 secondi pur di poter usufruire del servizio di streaming.

Quindi AnySee non determina la sua ottimizzazione solo in relazione del ritardo medio ma anche in relazione al numero di peer che abbandonano il sistema, se infatti si presentasse il caso in cui un alto numero di peer abbandonano il sistema risulterebbe chiara una necessaria ottimizzazione dello stesso. AnySee utilizza un indice (ADL) per capire se si presenta la necessità di un'ottimizzazione che è dato da

$$ADL = 100\frac{percentuale\ di\ abbandono}{ritardo\ medio}$$

ottenuto il valore dell'indice di ottimizzazione AnySee stabilirà un valore limite oltre il quale l'ottimizzazione risulta essere necessaria.

Capitolo 4

Qualità del servizio

Il quarto capitolo analizzerà la qualità del servizio, nel primo paragrafo si faranno alcune considerazioni sull'importanza di questo aspetto, nel secondo si illustreranno le metriche utilizzate per determinare il livello di qualità offerto da un sistema di live streaming p2p mentre nell'ultimo paragrafo verrà presentato un modello basato sul calcolo della durata on line dei peer per ottenere un miglioramento della qualità del servizio.

4.1 L'importanza della qualità del servizio

Avere una buona qualità del servizio permette ad un sistema p2p progettato per live streaming di mantenere e aumentare il numero di peer che si connettono per usufruire di tale servizio, è chiaro che se il sistema non è in grado di garantire una qualità del servizio ad un livello soddisfacente per gli utenti

essi tenderanno ad un abbandono generale della rete. Allo stesso tempo è necessario che il sistema sia anche in grado di supportare un gran numero di peer dato che per lo streaming di programmi molto popolari il numero degli utenti si aggira intorno a migliaia o milioni di utenti, e si rende necessaria una buona qualità del servizio per tutti i nodi connessi alla rete. Se la qualità del servizio comincia a degradare e gli utenti decidono di lasciare il canale sarà compito dei service provider identificare queste situazioni critiche e fare in modo di migliorare le prestazioni aggiungendo capacità di upload usando i propri server o in altri casi ricorrendo alla banda concessa dai Content Distribution Networks[1], nel momento in cui si raggiunge un livello di qualità soddisfacente tale risorse possono essere rilasciate per altri scopi.

Grazie allo studio della qualità del servizio offerto dai diversi sitemi p2p gli ISP e i CDN sono in grado di stimare le risorse che sono necessarie ad ogni sistema per il corretto ed efficiente funzionamento dello stesso, fornendo la banda addizionale nel momento in cui è necessaria. Inoltre ai ricercatori risulta utile lo studio approfondito di tale aspetto delle comunicazioni p2p in modo tale da poter migliorare algoritmi per la selezione dei nodi con cui comunicare all'interno della rete e per lo scheduling dei dati.

Per poter analizzare la qualità del servizio sarebbe necessario dotare ogni peer di uno strumento in grado di memorizzare e monitorare tutte le attività di streaming, l'unica possibilità per fare ciò sarebbe l'introduzione di tale

[1]I CDN rappresentano un sistema di computer collegati in rete attraverso Internet che comunicano al fine di distribuire i contenuti agli utenti che ne fanno richiesta

strumento all'interno del software, in assenza di questo l'analisi della qualità del sistema sarebbe invece piuttosto critica.

Il fenomeno del peer churn, ovvero il comportamento dinamico degli utenti che accedono alla rete o che la lasciano, contribuisce al cambiamento dello stato della rete come la bandwidth, ma anche i ritardi risultano essere delle problematiche riguardanti la qualità dei video che vengono visualizzati dallo streaming degli utenti.

Una gran parte dei peer si connette, si disconnette e poi si riconnette di nuovo diverse volte, ed inoltre molti utenti eseguono più download rispetto al numero di upload, ma ciò è in gran parte dettato dalla connessione dei peer che in molti casi limita il limite di upload ad una certa soglia oltre la quale non è possibile andare.

4.2 Le metriche della qualità del servizio

Una metrica che è possibile considerare nell'analisi dei sistemi p2p utilizzati per il live streaming, è la *quality of service* che in italiano è possibile semplicemente tradurre con qualità del servizio. La qualità del servizio viene analizzata da due punti di vista in particolare: la qualità dello stream video e il tempo di start necessario al canale.

4.2.1 Stream quality

La stream quality fa riferimento esattamente come suggerisce il nome alla qualità dello stream video, il sistema p2p che permette la condivisione di un video, si occupa di rendere disponibili ai vari peer dei blocchi di stream. Questo aspetto della qualità del servizio viene calcolato in base al rapporto tra il numero di blocchi che sono stati resi disponibili al client rispetto al numero di blocchi che sono necessari per una codificazione eccellente del video.

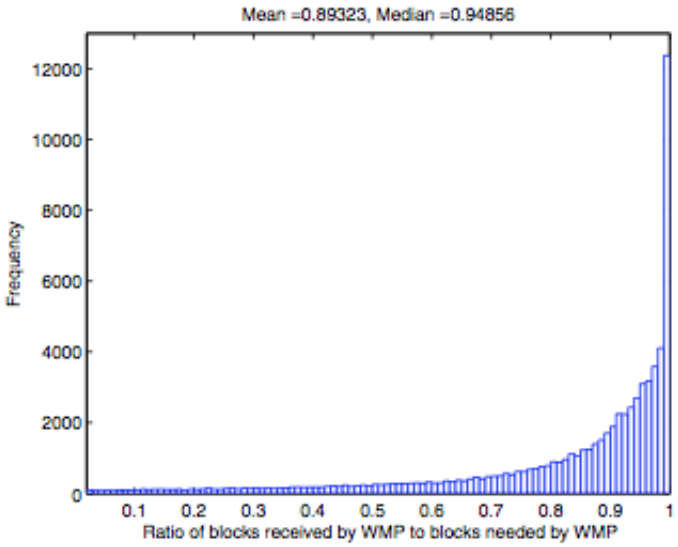

Figura 4.1: Istogramma blocchi ricevuti/blocchi necessari

In molti casi i video decoder usano una tecnica chiamata "copy previous error concealment" grazie alla quale gli errori causati da blocchi mancanti vengono nascosti all'utente finale. Ciò si concretizza nel fatto che nel caso

un frame dello streaming non sia arrivato a destinazione l'utente vedrà visualizzato l'ultimo frame correttamente renderizzato e il video verrà fermato a quell'istante. Questo inconveniente viene facilmente reso non intrusivo nel momento in cui vengono ricevuti dei frame contigui corretti. Da quanto detto è facile capire che invece nel caso ci siano molti frame mancanti essi contribuiscono in maniera consistente al deterioramento dello streaming ricevuto, per questo motivo la misurazione e l'analisi della lunghezza dei frame mancanti risulta essere un'ottimo indice sulla qualità totale offerta dal servizio.

Un numero abbastanza consistente di peer inoltre non partecipa alle statistiche dei blocchi persi in quanto non riceve neanche un blocco di dati, questo succede dal momento che questi peer non sono in grado di connettersi alla rete p2p a causa della presenza di firewall, problemi di connessione e alla bassa persistenza all'interno del network.

La variazione di qualità video dei diversi sistemi autonomi dipende in maniera sostanziale rispetto a quale sistema autonomo[2] il peer viene connesso.

A causa di errori causati da firewall installati nei computer dei client o a problemi di connessione non sono rare perdite di blocchi che rendono la stream quality di basso livello. Per ovviare a tali problematiche i sistemi p2p dovrebbero migliorare i propri algoritmi per permettere ai diversi sistemi autonomi di migliorare la propria qualità del servizio anche per esempio aumentando la bandwidth dei server.

[2]In internet indica un insieme di router e reti sotto il controllo di una singola e ben definita autorità amministrativa.

4.2.2 Channel startup time

Il channel startup time è la misura relativa al tempo necessario perchè il player video sia in grado di visualizzare i dati contenuti nel buffer della memoria. Questo tempo è il ritardo totale nella connessione tra il sistema p2p e i client, come illustra la figura sottostante questo valore è molto variabile, e tale oscillazione dipende dalle differenti condizioni di network e dalle dinamiche degli algoritmi di connessione.

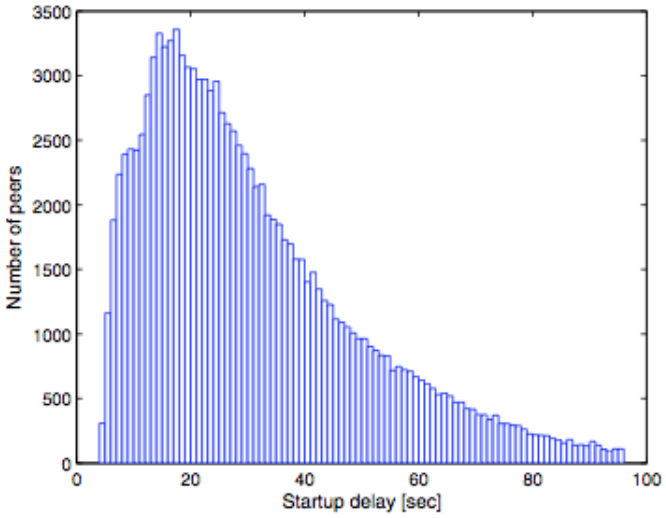

Figura 4.2: Istogramma channel startup time

Inoltre uno degli aspetti caratterizzanti del channel startup time riguarda il fatto che come mostra la figura 4.3 la maggior parte dei client non ha un indirizzo IP diretto ma sta dietro a firewall o router NAT[3].

[3]NAT acronimo di network address translation, è la tecnica che si occupa di mascherare gli indirizzi IP in modo tale che la macchina con la quale è in comunicazione pensi di essere connessa ad un altro indirizzo IP.

Direct connections	8.98
UPNP	6.53
Firewall	6.80
NAT	70.24
Other/Unknown	7.44

Figura 4.3: Statistiche in percentuale del tipo di connessione internet

4.3 Miglioramento della qualità del servizio

Come è stato detto in precedenza il fenomeno del node churn porta degli scompensi alla qualità totale del servizio proposta dai sistemi p2p, per analizzare e progettare un miglioramento della qualità viene usato un modello della durata dei peer on line. Tramite questo modello si è infatti in grado di valutare la correlazione statistica tra il tempo trascorso on line e il tempo che ci si aspetta che un peer rimanga ancora connesso.

4.3.1 Modello della durata on line dei peer

Nel modello della durata online dei peer viene indicata con $p(t)$ la probabilità della durata on line, quindi il tempo atteso dei peer che rimarranno online T viene calcolato da:

$$E(t-T|t \geq T) = \frac{\int_T^\infty (t-T)p(t)dt}{\int_T^\infty p(t)dt}$$

4.3.2 Utilizzo della durata on line per il calcolo della streaming stability

Quasi la maggioranza delle costruzioni di overlay adottano l'architettura ad albero per trasferire i contenuti agli utenti finali. Ovviamente la stabilità di questi alberi determina in maniera diretta la qualità ricevuta da ogni peer. Considerando che l'abbandono della rete da parte di alcuni client o gli errori di upload creano l'interruzione dei dati trasmessi, è possibile sfruttare il modello di durata definito prima per usare i peer con la durata on line più lunga e con più grande capacità di upload che si trovano vicino alla sorgente.

Prendendo come esempio la figura 4.4 è possibile vedere come l'abbandono della rete da parte dei peer C e D sia piuttosto distruttiva per la corretta ricezione dei dati, se infatti questi due peer decidono di abbandonare la rete tutti i peer che si trovano ai livelli sottostanti non saranno in grado in alcun modo di ricevere lo stream.

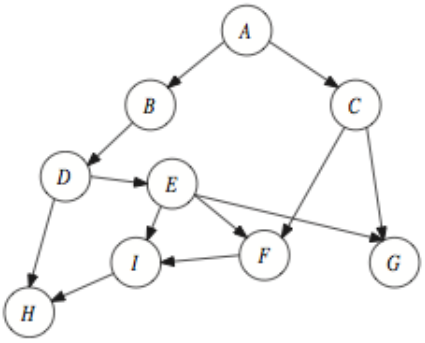

Figura 4.4: Esempio di rete p2p

Pertanto il goal principale è quello di riuscire a coinvolgere le informazioni della durata online nel costruire l'albero di distribuzione in modo da evitare frequenti abbandoni dei peer a posizioni più importanti.

E' possibile creare uno schema di costruzione dell'albero con il tempo trascorso online per ogni nodo per poter capire dove si posizionerà un peer che vuole accedere al sistema.

In figura 4.5 i peer da A ad E hanno formato un albero di cinque nodi, le 2-tuple di ogni peer indicano l'upload capacity e la durata totale online.

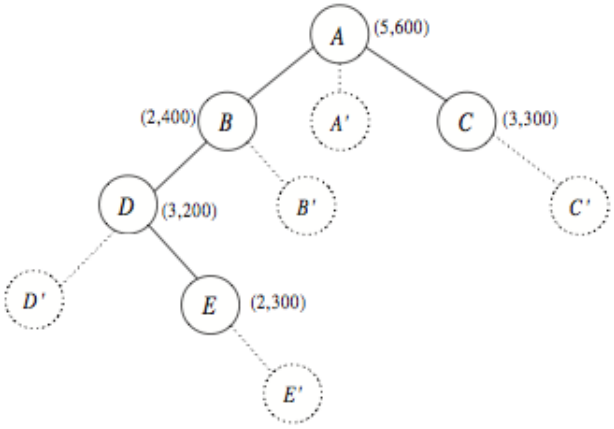

Figura 4.5: Posizionamenti possibili di un nodo che vuole accedere alla rete

Quando un peer F con capacità di upload 3 vuole accedere alla rete esso può essere ipoteticamente posizionato in diversi punti dell'albero, può quindi essere messo in una delle posizioni tra le seguenti $\varphi F = \{A', B', C', D', E'\}$, illustrati nell'immagine come nodi tratteggiati. Il nodo F potrà scegliere in quale punto dell'albero posizionarsi in relazione ai passi seguenti:

1. Calcolo della profondità media dell'albero, in relazione a questo esempio la profondità media è $d = 1.4$

2. Ordinamento delle posizioni. Selezione delle posizioni in φF le cui profondità non sono superiori a d , ordinamento secondo la durata online dei precedenti in modo ascendente. Il punto di accesso è indicato come $\varphi' F = \{A', B', C'\}$. Questi due passaggi servono per trovare le posizioni disponibili più vicine alla sorgente con l'upstream più grande e la maggior durata online.

3. Calcolo dell'impatto dopo essere stato inserito nell'albero. Calcolo del rapporto r del numero dei peer che ricevono una service capacity maggiore di F sul numero totale dei peer che sono attivi all'interno dell'albero in seguito all'accesso di F'. In figura $r \approx 0.143$, lo scopo di questo rapporto è di aiutare la determinazione del punto finale di accesso.

4. Calcolo del possibile punto di accesso in $\varphi' F$. Moltiplicazione del rapporto r con la capacità residua in $\varphi' F$ come p. Nel caso in esempio risulta quindi $p = 0.143 \times [cap(A') + cap(B') + cap(C')] = 1$.

5. Scelta del punto finale di accesso. In accodo con $\varphi' F$ e p, seleziona il punto di accesso per la partecipazione al sistema. Dal momento che nella struttura di esempio, $p = 1 < cap(A') = 3$, F si unirà in A'. Se si presentasse il caso in cui fossero presenti più di un peer in $\varphi' F$ con le stesse condizioni, la selezione della posizione di accesso di F avverrebbe in modo casuale.

Il vantaggio maggiore dell'introduzione della durata online nell'albero di costruzione è quello di mantenere i peer alle posizioni più alte nel modo più stabile possibile.

E' possibile confrontare questo modello con altri metodi per l'inserimento di un nodo nell'albero:

(1) Random-select: seleziona casualmente una posizione di accesso nella struttura,

(2) Min-profondità: seleziona la posizione di accesso con profondità minima nella struttura,

(3) Max-online: seleziona la posizione di accesso con la massima durata on line.

Il modello di arrivo degli utenti segue la distribuzione di Poisson ($\lambda = 1$), è possibile adottare la metrica "tempo accumulato di interrupt" per valutare le prestazioni del sistema proposto, di Random-select, di Min-profondità e di Max-online. Qualsiasi cambiamento di peer a monte si tradurrà in una aggregazione di interruzione. Si osserva che il sistema proposto presenta prestazioni migliori rispetto agli altri in quanto ha ritenuto sia la durata online e la capacità di caricamento nella costruzione dell'albero di consegna dei dati.

4.3.3 Utilizzo della durata on line per incentivazione di cooperazione

Oltre a migliorare la stabilità del servizio di streaming, il modello di durata online degli utenti finali potrebbe anche essere utilizzato per facilitare la reciproca cooperazione tra coppie di peer. In questo sottoparagrafo, viene illustrato il modo in cui è possibile sfruttare statisticamente la relazione positiva tra la durata trascorsa on line e il tempo che il peer resterà on line per studiare meccanismo di incentivazione nel live streaming p2p.

Un peer razionale arriverà alla conclusione che la migliore strategia è quella di consumare il più possibile, senza alcun contributo individuale, che a sua volta mina la performance globale del sistema. Molti studi si sono soffermati sulla questione dell'incentivazione ad essere dei peer attivi all'interno della rete, la maggior parte di essi ha generalmente valutato il contributo di ogni peer e la possibilità di fornire servizi differenziati di conseguenza. L'idea chiave dello schema in [19] è abbastanza semplice: peer che hanno più grandi durate on line e contribuiscono di più hanno una priorità maggiore rispetto agli altri peer.

In fig. 4.6 la 3-tupla in ogni peer denota la capacità di caricamento, la durata online e il contributo, rispettivamente. I peer B, C, D, E richiedono un qualche servizio al peer A, mentre F, G richiedono i segmenti video unicamente a C. Le richieste di C ed E saranno accettate da A in virtù della loro durata e del contributo on line. Sebbene la durata on line di B sia maggiore

di quella di D, la sua priorità è inferiore a quella di D, questo dato mostra come il comportamento di B sia di natura free ride in quanto ha un'alta durata on line ma un basso valore di contribuzione.

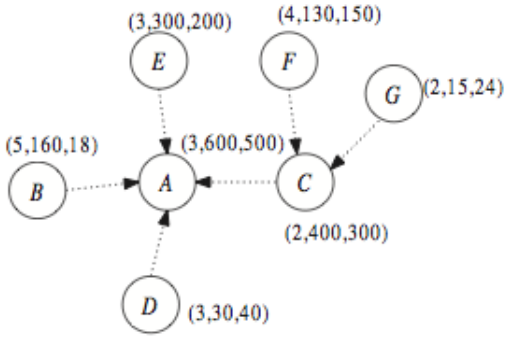

Figura 4.6: Rete p2p nella quale ogni nodo ha un'etichetta comprendente l'upload capacity, il tempo di durata on line e il contributo offerto

Formalmente, la priorità del peer P viene indicata come I_P, I_P può essere calcolata come:

$$I_P = \begin{cases} \frac{\gamma C_P}{dur(P)} & \frac{C_P}{dur(P)} < \delta \\ \gamma C_P + dur(P) & \frac{C_P}{dur(P)} \geq \delta \end{cases}$$

dove C_P denota contributo di P che potrebbe essere calcolato, il fattore γ è normalizzato e δ rappresenta la soglia configurabile delle specifiche del sistema. Con questo servizio on line basato sulla durata differenziata, i peer con più alta priorità sono ricompensati con un buon peer a monte e ottengono

una migliore qualità in cambio, mentre i peer con bassa priorità desiderano contribuire di più per migliorare la loro priorità.

Conclusioni

In questo lavoro di tesi si è cercato di esporre i metodi che vengono utilizzati per l'analisi degli attuali sistemi p2p che offrono il servizio di live streaming, con particolare attenzione per quanto riguarda l'aspetto della performance e della qualità del servizio.

Nella prima parte si è visto come sia possibile adattare un modello per l'analisi di un classico sistema p2p di file sharing ad un sistema di streaming, lo studio tramite questo modello è reso possibile dal fatto che il servizio di live streaming possa essere considerato nell'ottica della teoria dei giochi.

Si è visto come la presenza di alcuni peer che assumono comportamenti da free rider costituisca una situazione nella quale la performance totale del sistema venga danneggiata, questi utenti infatti ritengono opportuno che la massimizzazione della propria utilità sia più importante dell'utilità complessiva di tutta la popolazione che accede al sistema, si tengono quindi il diritto di abbandonare la rete nel momento a loro più congeniale, inoltre questione ancora più rilevante essi non contribuiscono in alcun modo alla diffusione delle informazioni in loro possesso. D'altro canto invece la presenza prolun-

gata di alcuni peer, definiti nodi stabili, permette un miglioramento della performance in quanto grazie ad essi è possibile garantire che i buchi all'interno della rete non abbiano un grande impatto negativo. Grazie a questi nodi infatti la ricostruzione dei dati mancanti o dei dati errati viene garantita dal momento che questi peer si posizionano all'interno della rete vicino alla sorgente della trasmissione. Per quello che riguarda i nodi stabili si è anche visto che tramite l'organizzazione della rete tramite la topologia ad albero e con l'inserimento di side links sia possibile migliorare ulteriormente la performance in quanto tramite questi links è possibile effettuare il recupero dei dati mancanti in maniera più agile e diretta. I nodi che si trovano più lontani dalla root infatti cercheranno di connettersi tramite i side link verso nodi che invece sono in posizione più favorevole e più vicini alla root. In letteratura sono presenti vari studi incentrati sul miglioramento della performance, uno di questi è AnySee, grazie a questo modello si è visto come tramite la topologia logica della rete sia possibile realizzare una rete che offra un servizio migliore tramite la creazione di più collegamenti tra gli alberi di condivisione.

Per quanto riguarda invece il secondo aspetto, quello incentrato sulla qualità del servizio, si è visto che anche in questo caso se il sistema p2p non è in grado di offrire un servizio soddisfacente in termini di qualità, un gran numero di peer abbandonerà il sistema in quanto nessun giocatore razionale è interessato ad accedere ad un sistema che non gli permette di usufruire in maniera ottimale di quello stesso servizio offerto. Sempre in relazione alla qualità del servizio si è studiato il tempo che è necessario al canale per poter

avviare lo streaming, più il tempo si accorcia più gli utenti sono soddisfatti, si è visto che comunque anche se il tempo risulta essere elevato una buona parte dei giocatori tende a rimanere all'interno del sistema a patto che la qualità della trasmissione una volta instaurata sia adeguata alle aspettative di ogni singolo utente. Come per la performance anche per la qualità del servizio è possibile proporre un metodo che permette un miglioramento di questa particolare questione, tramite il modello della durata on line dei peer è infatti possibile considerare tale aspetto per poter inserire un nuovo client nella posizione migliore per poter partecipare alla condivisione all'interno della rete.

Bibliografia

[1] A. Mas-Colell , M. D. Whinston, J. R. Green,"Microeconomic Theory", Oxford University Press, 1995

[2] http://en.wikipedia.org

[3] http://tools.ietf.org/html/rfc2326

[4] http://www.cs.columbia.edu/~hgs/teaching/ais/slides/2003/RTSP.pdf

[5] F. Wang, J. Liu, Y. Xiong, "Stable peers: Existence, Importance, and Application in Peer-to-Peer Live Video Streaming", 2008, International Conference on Computer Communications (INFOCOM 2008), pp. 1364 - 1372

[6] X. Yang, G. de Veciana, "Performance of peer-to-peer networks: Service capacity and role of resource sharing policies", 2006, International Journal of Performance Evaluation, Volume 63, Issue 3, pp. 175 - 194

[7] S. Agarwal, J. P. Singh, A. Mavlankar, P. Baccichet, B. Girod, "Performance and Quality-of-Service analysis of a Live P2P Video Multicast

Session on the Internet", 2008, International Workshop on Quality of Service, pp. 11 - 19

[8] J.J.D. Mol, J.A. Powelse, D.H.J. Epama, H.J. Sips, "Free-riding, Fairness, and Firewalls in P2P File-Sharing", 2008, International Conference on Peer-to-Peer Computing , pp. 301 - 310

[9] R. Krishnan, M.D. Smith, Z. Tang, R. Telang, "The Impact of Free-Riding on Peer-to-Peer Networks", 2004, Annual Hawaii International Conference on System Sciences

[10] V. N. Padmanabhan, H. J. Wang, P. A. Chou, "Distributing Streaming Media Content Using Cooperative Networking", 2002, International Workshop on Network and operating systems support for digital audio and video (NOSSDAV 2002)

[11] G. Dan, V. Fodor, I. Chatzidrossos, "On the Performance of multiple-tree-based peer-to-peer live streaming", 2007, International Conference on Computer Communications (INFOCOM 2007), pp. 2556 - 2560

[12] N, Magharei, R. Rejaie, "Understanding Mesh-based Peer-to-Peer Streaming", 2006, International workshop on Network and operating systems support for digital audio and video (NOSSDAV 2006)

[13] X. Hei, Y. Liu, "Inferring Network-Wide Quality in p2p Live Streaming, Systems", 2007, IEEE Journal on Selected Areas in Communications, Volume 25, Issue 9, pp. 1640 - 1654

[14] I. Chatzidrossos, V. Fodor, “On the effects of Free Riders in p2p streaming systems”, 2008, International Telecommunication Networking Workshop on QoS in Multiservice IP Networks, pp. 8 - 13

[15] M. Feldman, J. Chuang, “Overcoming Free-Riding Behavior in Peer-to-Peer Systems”, 2005, ACM SIGecom Exchanges, Volume 5, Issue 4, pp. 41–50

[16] X. Liao, H. Jin, Y. Liu, L. M. Ni, D. Deng, “AnySee: Peer-to-Peer Live Streaming”, 2006, International Conference on Computer Communications (INFOCOM 2006), pp. 1 - 10

[17] K. Eger, U. Killat, “Fair Resource Allocation in Peer-to-Peer Networks”, 2007, Journal Computer Communications, Volume 30, Issue 16

[18] C. Buragohain, D. Agrawal, S. Suri, “A Game Theoretic Framework for Incentives in P2P Systems” , 2003, International Conference on Peer-to-Peer Computing (P2P 2003), pp 48 - 56

[19] Y. Tang, L. Sun, J. Luo, S. Yang, Y. Zhong, “Improving Quality of Live Streaming Service over P2P Networks with User Behavior Model”, 2007, International MultiMedia Modeling Conference(MMM 2007), pp. 333 - 342

[20] R.W. Rosenthal, “” A Class of Games Possessing Pure-Strategy Nash Equilibria and the Potential Maximizer”, International Journal of Game Theory, Volume 2, Issue 1, pp. 65 - 67

Ringraziamenti

Eccomi qui, questa è davvero la fine di un'Era con la e maiuscola, mi sembra ieri d'aver iniziato l'università e invece sono passati un bel po' di anni e mi sento in dover di ringraziare un sacco di persone che hanno percorso un pezzo di strada insieme a me e con le quali ho condiviso diverse esperienze.

Per iniziare vorrei ringraziare mamma, papà e Totò per avermi supportato e sopportato in quest'avventura.

Venendo alla categoria amici & co. ringrazio gli amici mestrini Angela, Cavaz, Jordy, Pino, Asta e Dal con i quali ho condiviso gli anni del liceo e delle medie e che ogni volta che torno a casa rivedo con piacere.

Nella mia precedente tesi triennale non ho ringraziato nessuno e per questo motivo questa è l'occasione giusta per ricordare tutti i miei compagni del DSI, con i quali ho passato quei primi tre anni, o per meglio dire qualcuno in più, per questo grazie alle liste di puntatori, con i quali ho passato intere giornate nei corridoi a studiare e perdere tempo. Arrivando a tutti quelli che hanno reso più piacevoli quegli anni vorrei ringraziare in particolar modo la Glo e la Cilly che con me hanno affollato di fie il dipartimento, Matteoz

Rosi e Jack i miei supporti tecnici e morali, Enrike che mi porta sempre a sciare, Dex che mi ha seguita a Bologna, Ianez e il rombo di sprisss, Fede, AleTronky, lo Zio Urban, Trevi, Frapple, Basqua, Skara, Dozzo, Favaro e Lorenzo.

Ringrazio Trenitalia che mi ha portato fino a Bologna la prima volta e la mia prima coinquilina Erica con la quale ho condiviso nel primo anno due case, un trasloco e un numero indefinito di altri coinquilini psicopatici e sociopatici. Dato che a me piace cambiare sono arrivata in una terza casa, sicuramente voglio ringraziare Anna per essere sempre così solare e attiva, Giò per avermi fatto ascoltare in anteprima tutta la sua nuova produzione per il Kindergarten, Fabrizio per averci fatto incontrare.

Arrivando ad SDI sono in dovere (e in volere) di ringraziare un po' di persone, cominciando da Roxy che mi ha permesso di scrivere questa tesi. Ringrazio i miei compagni di mille progetti Stefy, Lu, Barbyeee e SteSte. Non posso non ricordare Paolo e Michè, Bibi, Fede, Macsssss, e il mio compagno di laurea Anto. Ringrazio anche chi è arrivato un po' dopo come Dany, Marino, Pao Lov e Maria. No non mi sono dimenticata, per ultima ma non ultima Ali e le sue piade, con lei ho condiviso non solo i vari scleri universitari ma devo ringraziarla particolarmente perchè con lei abbiamo deciso di cominciare anche un'altra avventura, insime siamo approdate alla palestra di via del riccio e alla squadra del cusb. Due anni intensi di cusb non si possono dimenticare, vorrei ringraziare il cusb per essere un pessimo organizzatore di tornei, il torneo di san pietro per averci fatto vincere i pro-

sciutti e Bibione, zanzare-tigre comprese, per averci regalato un'esperienza di aggregazione bellissima e scottature non indifferenti. In particolare vorrei ringraziare lo zoccolo duro del primo anno Chia, Federì, la Buri, Stefanga e il gioco a premi del cusb, Anna H_2O (anche se il campo più grande non l'ha ottenuto) e MC, le new entry di quest'anno Ro, la Fra e Napoleone, Terri, Fifi e il suo korfball. Ringrazio la MISSter ufficiale Tullia e il mio personal coach di quest'anno Anto, la GAP che ci asfalta sempre, Giò e Lucy.

Per finire vorrei ringraziare il mio vecchio pc R.I.P. , il mio nuovo compagno mac, il ventilatore che mi sta aiutando a sopportare l'afa bolognese anche ora che sto scrivendo questi ringraziamenti, il calcetto del De Marchi, la neve, il pasticciotto del Salentino, la musica, Bologna per avermi ospitato questi due anni abbondanti, lo staff di itasa che mi permette di perdere un sacco di tempo guardando i telefilm in lingua originale, a Google che sà sempre tutto e al suo inventore MM.