

Incentivi in sistemi P2P

Michele Nasti

18 marzo 2011

1 Introduzione

L'anno in cui il P2P è salito alla ribalta è stato sicuramente il 1999, anno in cui reti come Napster e Gnutella hanno ottenuto riconoscimenti in termini di utenti e interesse da parte dei media. Queste reti, dette *peer-to-peer*, permettevano ai singoli utenti di condividere file (legali o meno) in maniera auto-organizzata. Eppure, dopo neanche sei mesi dal lancio di Gnutella, è stato stimato che due terzi degli utenti scaricavano file senza dare nulla in cambio, ossia senza condividere i propri file con gli altri. Questo fenomeno viene detto *free-riding*.

I vantaggi del P2P sono evidenti: reti globali, anonimato, alto movimento di utenti e file; tuttavia una grandissima parte delle transazioni tra gli utenti sono interazioni “one shot” tra due utenti che probabilmente non si incontreranno mai più in futuro. Risulta evidente dunque che bisogna elaborare un metodo per permettere agli utenti di collaborare, magari utilizzando tecniche basate su *incentivi*.

La successiva generazione di reti P2P tentò di inserire un sistema di incentivi basata su *pagamenti* o sulla *reputazione*. Ad esempio, Mojonation remunerava gli utenti che scambiavano propri file con crediti che permettevano di sbloccare altri download; mentre eMule (o KaZaA) assegnano una reputazione maggiore a chi condivide più file in upload. I peer con una reputazione più alta subiscono un trattamento privilegiato nei loro download. Il sistema di file-sharing BitTorrent adotta un altro sistema basato sul *baratto*. Siccome questo protocollo viene usato principalmente per file di grandi dimensioni, come film, programmi o album musicali, questi file vengono partizionati in piccole parti, e i peer del protocollo necessitano di numerose interazioni tra di loro per scaricare un file con successo. Questo sistema permette una maggior cooperazione, basandosi su una reciprocità diretta piuttosto che indiretta; inoltre il sistema diviene più semplice, non c'è bisogno di mantenere liste di “reputazione” dei peer, ed è anche più robusto agli attacchi.

Nella reciprocità diretta, l'utente X decide il suo livello di cooperazione con Y basandosi solo sul servizio ricevuto da Y nel passato. D'altro canto, la reciprocità indiretta tiene conto anche dei servizi forniti da Y ad altri utenti oltre X .

Sperimentalmente è stato dimostrato che con la reciprocità diretta vi sono meno "free-riders", ma la ricerca teorica ha dimostrato che questo tipo di attacchi sono comunque realizzabili da peer egoisti.

Il P2P non è solo file-sharing, come descritto fin'ora; diversi sistemi autonomi utilizzano il peer to peer per i propri scopi, come le reti ad-hoc o le reti wireless mesh, fino allo streaming video o a reti che permettono la comunicazione anonima. Anche il routing inter-dominio di Internet richiede la cooperazione di operatori rivali.

Dal punto di vista dei peer, lo spazio di strategie possibili va oltre alla scelta binaria "condividere/non condividere". I peer possono decidere di agire strategicamente riguardo alla disponibilità di risorse locali, carico di lavoro, etc.; inoltre possono decidere l'ammontare dello sforzo, la loro connessione con la rete di peers, possono decidere in che modo gestire le proprie identità e trattare le identità di altri in maniera differente.

Da queste motivazioni nasce il *P2P Mechanism Design*. I sistemi P2P offrono un esempio concreto di comportamento strategico su sistemi distribuiti su larga scala, ed è un valido laboratorio per testare le soluzioni con una popolazione reale.

La ricerca ha individuato quattro tipi di peer nelle reti P2P. Ad esempio abbiamo i nodi che seguono il protocollo e chiameremo *corretti/obbedienti*, mentre vi sono nodi *difettosi* che possono comportarsi arbitrariamente o fermarsi del tutto. I due tipi di nodi più importanti sono però i nodi *razionali* (che cercando di massimizzare l'utilità) e quelli *irrazionali* (nodi che si comportano strategicamente ma che non seguono il protocollo).

2 il P2P file-sharing game

In un sistema P2P ogni utente gioca solitamente due ruoli: per alcune interazioni interpreta il ruolo di client che richiede un file, ed ottiene un beneficio ogni qual volta riesce a scaricare; per altre iterazioni invece interpreta il ruolo di server che deve inviare un file ad altri peer, e se ciò avviene, egli può avere dei costi in termini di banda o di CPU. In tali situazioni, il "free-riding" è la strategia dominante: un giocatore vorrà scaricare quando è client, e si rifiuterà di caricare file quando è server. Come risultato si ottiene che tutti gli individui ragionano in questo modo e attuano una strategia di free-ride sulle spalle di utenti altruisti, portando ad una degradazione delle prestazioni del

sistema e rendendo tutti perdenti, una strategia spesso chiamata *tragedia dei beni comuni digitali*.

Vi sono delle similitudini con il gioco del Dilemma del Prigioniero: nella versione “one-shot” del gioco, i giocatori hanno una sola strategia che porterà ad un equilibrio poco desiderabile. Invece, nella versione iterata del Dilemma del Prigioniero, la cooperazione può essere ottenuta grazie alla reciprocità diretta, ad esempio usando la strategia *Tit-For-Tat*.

Un agente che usa la strategia Tit-For-Tat inizialmente coopererà, successivamente si comporterà allo stesso modo del giocatore avversario. Se nell’ultima iterazione l’avversario è stato cooperativo, anche l’agente si comporterà allo stesso modo, e questo vale anche nel caso in cui l’avversario si comporta in maniera non cooperativa.

Usando questa strategia, un giocatore che dovesse comportarsi da free-rider potrebbe portare l’altro giocatore a rivalersi in un round futuro. In questo modo si può provare a sostenere una cooperazione, in cui un peer decide di inviare un file ad un altro peer con la speranza di scaricare un file da quest’ultimo in futuro.

Chiaramente non c’è garanzia che i due peer si incontreranno per altre transazioni nel corso della loro vita. Se la popolazione è molto grande, la probabilità che due individui si incontrino è così bassa che la cooperazione potrebbe non riuscire a concretizzarsi, e il free-riding potrebbe prevalere.

La reciprocità diretta, dunque, non funziona se la popolazione è molto grande. Un modo per superare questo limite è di favorire le interazioni ripetute tra un piccolo numero di peer, e questo è proprio fatto da BitTorrent, in cui gli utenti condividono grandi file molto popolari, e dato che vi è un alto numero di peer sono contemporaneamente interessati allo stesso file e vogliono avere interazioni ripetute scambiandosi segmenti di file per poterlo ottenere intero.

Per poter supportare la cooperazione su file e su scale temporali più grandi, dev’essere implementata una qualche forma di informazione sui peers. Questo ci porta alla *reciprocità indiretta*, ossia a sistemi basati sulla reputazione e sul comportamento passato dei giocatori. In questo modo, potrei preferire di collaborare con un utente che in passato ha condiviso molti file, piuttosto che con uno che non ha avuto scambi recentemente.

In un sistema p2p spesso i peers contattano altri utenti che sono sconosciuti e di cui non esiste una storia o una reputazione. E’ quindi importante pensare al modo in cui comportarsi con gli sconosciuti. Si potrebbe pensare di usare una Tit-For-Tat che coopera sempre con gli sconosciuti, ma questo comportamento potrebbe essere sfruttato da persone che lasciano e si ricollegano al sistema con nuove identità. Evitando la cooperazione con i nuovi utenti, invece, è un metodo robusto contro coloro che generano continuamen-

te nuove identità, ma scoraggia nuovi arrivi e potrebbe portare a defezioni. E' stato inoltre dimostrato che comportarsi in maniera probabilistica, ossia collaborare con probabilità $0 < p < 1$, non è affidabile contro chi genera nuove identità. Invece, adattare la probabilità di cooperazione con gli sconosciuti basandosi sulle cooperazioni con gli sconosciuti passate sembra essere un buon metodo, almeno nel breve periodo.

3 Sistemi basati sulla reputazione

La reputazione è la prima caratteristica utilizzata da sistemi per aiutare la cooperazione; il suo utilizzo è diffuso in molti ambiti, dalla biologia agli acquisti on-line (si pensi ad eBay, che classifica gli utenti non fraudolenti con punteggi alti).

Il P2P ha preso spunto da questo trend, ed infatti molte piattaforme di file-sharing hanno un sistema basato sulla reputazione dei propri utenti per poter incentivare comportamenti corretti e punire comportamenti scorretti. In generale, chi contribuisce al sistema ottiene una reputazione più alta (sotto forma di punteggio, o crediti); questo avviene ad esempio in reti come KaZaA, in cui gli utenti che caricano i file ad altri vengono "pagati" con una priorità più alta quando scaricano file da altri.

Uno schema di reputazione sul P2P può essere utilizzato con altre tecniche di sicurezza per identificare e isolare i peer cattivi. Diversi metodi sono stati proposti, ad esempio l'algoritmo Eigentrust che calcola valori di fiducia globali di peer aggregando valori locali di peer, basandosi sulla fiducia transitiva (come il Pagerank per le pagine web). Inoltre è possibile smascherare i peer che introducono file non autentici: questi peer riceveranno una scarsa fiducia e verranno evitati dagli altri. Il sistema Credence estende la nozione di reputazione dai peer agli oggetti condivisi, mantenendo punteggi anche per i singoli oggetti.

I sistemi che si basano sulla reputazione sono comunque suscettibili di vari attacchi: ad esempio più peer possono essere collusi e regalarsi una falsa fiducia a vicenda; se uno stesso peer genera più identità per creare false reputazioni, questo attacco viene detto *Sybil Attack*. Un altro tipo di attacco è detto *whitewashing attack*, e si ha quando un peer commette atti illeciti e può facilmente rigenerare una nuova identità per evitare le conseguenze di una cattiva reputazione.

White washing

I sistemi basati sulla reputazione, ossia quelli che si basano sulla reciprocità indiretta, sono particolarmente vulnerabili a questo tipo di attacchi, dato che un free rider può ripulire la propria cattiva reputazione uscendo e rientrando nel sistema.

Ci sono due modi per affrontare questo attacco. Il primo approccio necessita di pseudonimi gratuiti ma non sostituibili, assegnati da un'autorità centrale fidata, tuttavia questo riduce la natura centralizzata del sistema e introduce un singolo punto di fallimento.

Senza una terza parte fidata, l'unica opzione rimasta è di imporre penalizzazioni a tutti i nuovi arrivati, includendo in questo modo anche i white-washers. E' stato tuttavia dimostrato che questo aumenta il costo sociale del sistema.

Come vedremo successivamente, se il livello di generosità del sistema è basso, un meccanismo che penalizza i free-riders può aumentare notevolmente la performance del sistema e inoltre, se le identità sono gratuite, penalizzare alla cieca tutti i nuovi arrivati può scoraggiare i whitewashers e può portare a un aumento del costo sociale per un numero limitato di casi.

Sybil Attacks

In un attacco Sybil, un attaccante plagia la reputazione e il meccanismo di condivisione di una rete P2P creando un grande numero di identità, e usandole per ottenere un'influenza sproporzionata. E' stato provato che senza un'autorità fidata centrale, questo attacco è sempre possibile eccetto in casi irrealistici quali l'uso di tantissime risorse e di una grande coordinazione tra le entità. questo viene provato con una serie di lemma, che riportiamo qui:

Lemma 3.1 *se ρ è il rapporto tra le risorse di un'entità maliziosa f e le risorse di un'entità di capacità minima, allora f può presentare $g = \lfloor \rho \rfloor$ identità distinte ad una entità locale l .*

Lemma 3.2 *se un'entità l accetta entità che non sono validate simultaneamente, allora una singola entità maliziosa f può presentare un grande numero di identità distinte all'entità l .*

L'idea dietro a questo lemma è che se queste entità vengono validate simultaneamente allora si possono scoprire entità fasulle, se invece le entità non vengono validate simultaneamente, f può validare continuamente in ogni momento un numero di nuove identità e presentarle a l .

Lemma 3.3 *Se un'entità locale l accetta ogni identità garantita da altre q identità accettate, allora un'insieme F di entità maliziose può presentare ad l un numero arbitrariamente grande di identità diverse sia se $F \geq q$ oppure se le risorse collettive disponibili da F eguagliano le $q + |F|$ entità.*

Tutte le identità di F garantiscono per un numero arbitrariamente grande di identità fasulle, che saranno accettate da l .

Lemma 3.4 *Se in un insieme C le entità corrette non si coordinano in intervalli di tempo durante i quali accettano altre identità, e se l'entità locale l accetta ogni identità garantita da altre q identità accettate, allora anche una entità maligna f di capacità minima può presentare $g = \lfloor |C|/q \rfloor$ identità distinte ad l .*

Anche se questi risultati sono fortemente negativi, sono stati proposti dei protocolli come *SybilGuard* che limitano l'influenza di un attacco Sybil, basandosi su reti sociali tra le identità degli utenti, in cui un arco tra due identità indica una relazione fidata. Dato che un nodo può creare molte identità (archi nel grafo) ma poche relazioni di fiducia (archi nel grafo), si possono sfruttare alcune proprietà dei grafi per limitare il numero di identità che un nodo maligno può creare.

3.1 Il modello del P2P

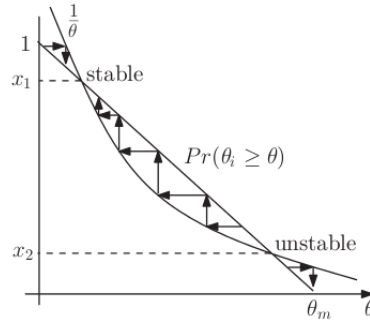
Formuleremo un modello matematico molto semplice per descrivere le decisioni dei peer a fronte di alcuni parametri. Consideriamo una popolazione di giocatori razionali con diverse volontà e capacità di connessione al sistema. Ogni giocatore i ha un tipo privato θ_i , che rappresenta il massimo costo sostenibile dal giocatore per contribuire al sistema. Ogni peer decide in modo autonomo se contribuire al sistema oppure se essere "free-rider", a seconda del costo di contribuzione e il suo tipo.

Sia x la frazione di utenti che contribuiscono al sistema ($x \in [0, 1]$). Poiché i giocatori che contribuiscono devono anche sopportare il carico del sistema, il costo di contribuzione si può modellare come l'inverso della frazione di utenti contribuiscono al sistema (ossia, $Costo(C) = 1/x$). Pertanto la decisione di un agente razionale con tipo privato θ_i sarà:

Se $Costo(C) = 1/x < \theta_i$, contribuisci
Altrimenti, Free-ride.

In questo semplice modello non vi è alcun tipo di incentivo. Vediamo alcune implicazioni e proprietà interessanti.

Sia $x = Pr(\theta_i \geq \theta)$ la retta della funzione di ripartizione dei tipi, assumendo che θ_i sia uniformemente distribuito in $[0, \theta_m]$. Possiamo specificare la curva dei costi di contribuzione come $x = 1/\theta$. I punti di equilibrio del sistema è determinato dall'intersezione di queste due curve: vi sono tre punti di equilibrio nel sistema, che sono x_1 , x_2 , e $x_3 = 0$.



Se il sistema è fuori equilibrio, possono accadere due casi. Il primo si ha quando $x < x_2$ e $x > x_1$: se la curva dei costi è maggiore della curva di distribuzione dei tipi (ricordiamo che i tipi sono la generosità degli utenti), i costi di contribuzione sono più alti rispetto alla frazione di utenti che vogliono contribuire a quel costo, dunque la frazione di contributori decresce. Nel caso in cui ci troviamo sopra x_1 , questo processo si arresta quando si arriva ad x_1 , che è un equilibrio. Lo stesso accade quando siamo sotto x_2 , solo che in questo caso il sistema tenderà verso l'equilibrio $x_3 = 0$, ossia non vi saranno più utenti che useranno il servizio.

Invece, se siamo nell'intervallo $x_1 < x < x_2$, i costi di contribuzione sono più bassi rispetto alla generosità degli utenti, dunque il livello di contribuzione aumenta e si giunge a x_1 .

Dunque, x_1 e x_3 sono due equilibri che sono anche punti fissi; l'equilibrio x_2 è un equilibrio solo se i nodi partono da quest'equilibrio, altrimenti si sposteranno verso uno degli altri due.

Il livello di contribuzione al sistema x si determina risolvendo l'equazione $x = Prob(\theta_i \geq 1/x)$ e, se la generosità è uniformemente distribuita tra 0 e θ_m , allora $Prob(\theta_i \geq 1/x) = 1 - \frac{1}{x\theta_m}$. L'equazione diventa pertanto

$$x = 1 - \frac{1}{\theta_m \cdot x}$$

$$x - 1 + \frac{1}{\theta_m \cdot x} = 0$$

$$\theta_m \cdot x^2 - \theta_m \cdot x + 1 = 0$$

e le soluzioni sono $x_{1,2} = \frac{\theta_m \pm \sqrt{\theta_m^2 - 4\theta_m}}{2\theta_m}$; la x_1 più grande è l'equilibrio stabile.

Dunque possiamo affermare che l'equilibrio stabile diverso da zero aumenta con θ_m , ma diventa uguale a zero se $\theta_m < 4$ (perchè l'argomento della radice nella soluzione dell'equazione precedente diventa negativo).

Consideriamo ora i benefici ricevuti dai giocatori. Un giocatore riceve un beneficio dal partecipare al sistema, anche se non contribuisce; questo beneficio è proporzionale al livello di contribuzione totale nel sistema, dunque è una funzione del tipo αx per qualche costante $\alpha > 1$. Concentriamoci su α molto grandi. Ora definiamo le performance del sistema W_s come la differenza tra i benefici ricevuti dai giocatori e il costo speso dai giocatori per contribuire:

$$W_s = \alpha x - \left(\frac{1}{x}\right)x = \alpha x - 1$$

Anche se la partecipazione può dare alti benefici ai peer, il sistema potrebbe collassare se la generosità massima è bassa (come dimostrato in precedenza), dato che la performance del sistema è limitata da un basso livello di contribuzione.

3.2 Reputazione e servizi di differenziazione

Consideriamo un sistema di reputazione che riesce a scoprire i free-riders con probabilità p , e una regola di differenziazione del servizio tale che i free-riders sono esclusi dal sistema. In alternativa si può immaginare un meccanismo di reputazione che riesce a scoprire perfettamente i free-riders, e questo meccanismo viene utilizzato con una regola di differenziazione del sistema in cui i free-riders sono penalizzati con un servizio pari a $1 - p$ volte quello dei contributori.

Degradare la performance ha due effetti, entrambi i quali portano a un livello di contribuzione più alto. Innanzitutto, dato che i free-riders prendono solo una frazione $1 - p$ dei benefici, il carico sul sistema decresce a $x + (1 - x)(1 - p)$. Pertanto, il costo necessario per contribuire al sistema è $\frac{x + (1 - x)(1 - p)}{x}$. Inoltre, questa punizione porta con se anche una minaccia agli utenti, dato che i free-riders sanno che riceveranno un servizio più scadente oppure che verranno espulsi.

Sia Q il beneficio individuale, R il costo di contribuzione ridotto, e T la minaccia per ciascun utente. Un utente che contribuisce al sistema realizza una performance di

$$W_{Sc} = Q - R = \alpha x - \frac{x + (1 - x)(1 - p)}{x}$$

Mentre un free-rider realizza una performance di

$$W_{Sf} = Q - T = \alpha x - p\alpha x$$

Con questo meccanismo di reputazione e di differenziazione del servizio, le performance del sistema diventano

$$W_s(p) = x(Q - R) + (1 - x)(Q - T) = (\alpha x - 1)(x + (1 - x)(1 - p))$$

Imporre una penalizzazione per i free-riders, pur aumentando il livello di contribuzione, comporta una qualche perdita. Tuttavia si può impostare p per raggiungere un certo livello di cooperazione. Notiamo che se la penalità è abbastanza alta, la minaccia T supera il costo di contribuzione R , e i peer non avranno più motivo di non contribuire al sistema, e in questo caso non vi saranno neanche sanzioni. Senza free-riders otterremo una performance ottima di $\alpha - 1$.

Claim 3.5 *se usiamo lo schema di penalizzazione, esiste un equilibrio nel quale $x = 1$ se $p \geq 1/\alpha$.*

questo vuol dire che se i benefici di partecipare al sistema sono alti (α è grande), è sufficiente o un servizio di differenziazione che impone una diminuzione delle performance ai freeriders, oppure un meccanismo che trova i free-riders con una bassa probabilità, per indurre ad un alto livello di cooperazione.

3.3 Credence

Credence è un sistema di reputazione distribuito, progettato per combattere l'inquinamento dei contenuti nei sistemi P2P. Credence permette ai peer di valutare l'autenticità di un file o di altri contenuti online basandosi sull'accuratezza della descrizione proposta in contrasto con l'oggetto stesso. I membri della rete Credence votano gli oggetti, e il sistema aggrega i voti e li pesa attraverso delle misure di similarità in modo tale che voti da peer simili siano pesati maggiormente, rispetto ai voti di spammer o altri intrusi. Questa tecnologia permette agli utenti un grande incentivo a votare onestamente. Credence è stato implementato su LimeWire, popolare client per la rete Gnutella, potendo osservare i giudizi degli altri utenti prima ancora di scaricare il file.

Gli autori definiscono "inquinamento" quei file il cui contenuto non corrisponde all'etichetta che l'accompagna. L'inquinamento di un file può essere facilmente rivelato da utenti onesti senza tecniche matematiche troppo complesse. Infatti, una volta scaricato un file, l'utente deve soltanto specificare (attraverso una semplice icona col pollice alzato) se il file è corretto. I voti sono firmati crittograficamente per evitare attacchi Sybil. E' il client che,

prima di scaricare un file, fa una ricerca dei voti per quel file e calcola un voto aggregato basato sui voti di altri utenti.

Per evitare peer maliziosi, ad ogni utente è assegnato un coefficiente di correlazione che riflette l'utilità storica dei suoi voti attraverso una reciprocità indiretta. Tuttavia, a causa del tipo di traffico e di utenza delle reti P2P, i peer votano pochi file; Per ovviare a questo problema, gli autori hanno proposto l'uso di una tecnica per disseminare informazioni tra piccoli gruppi. In Creedence un client richiede ripetutamente queste informazioni da peer scelti a caso e poi valuta la relazione tra questi peer e i propri vicini. I dati vengono autenticati dal client e incorporati nel database locale. Questo metodo funziona anche con pochi client che valutano i file, senza l'interazione dell'utente e senza calcoli complessi.

Gli autori di Creedence hanno studiato l'impatto agli attacchi. Ad esempio, i voti di un peer malvagio non vengono presi in considerazione; i voti generati a caso da un attacco Sybil verrebbero evitati, dato che i peer hanno una correlazione che tende a zero. Anche il whitewashing ha un impatto limitato a causa dell'alto numero di voti corretti, e whitewashing indipendenti e consecutivi hanno l'effetto di annullarsi l'un l'altro.

3.4 BitTorrent: un sistema basato sul baratto

BitTorrent è un protocollo di file-sharing P2P con un sistema di incentivi implementato a livello di design. Il meccanismo degli incentivi è basato sulla reciprocità diretta piuttosto che su quella indiretta.

In BitTorrent, il nodo origine divide un grande file in piccoli pezzi di grandezza fissa (*swarming*), e invia diversi pezzi ad altri peer, che si scambiano questi pezzi a loro volta. Una volta ottenuti tutti i pezzi, il peer può ricostruire il file originario. Per indurre i peer a inviare i loro pezzi, la velocità di download è calcolata sul suo tasso di upload attraverso uno schema di baratto o su una reciprocità diretta.

BitTorrent cerca di alleviare il problema del random matching, sofferto dagli altri sistemi p2p: quando un peer cerca di scaricare un file egli riceve una lista di 40 peer che stanno scaricando o caricando pezzi dello stesso file. Il peer sceglie 4 o 5 tra questi peer come vicini, e aggiorna periodicamente la lista di vicini con i peer che offrono le migliori velocità di download. Inoltre un peer sceglie a caso un peer tra quelli che stanno caricando, sempre per cercare di migliorare la propria banda in download.

Grazie a questo design, i peer intrattengono continui scambi con pochi altri utenti per tutta la durata del download; questo è vantaggioso specialmente per file grandi come film o software. Tuttavia, bitTorrent non prevede un sistema di collaborazione che vada oltre il periodo di download del fi-

le: i peer non sono invogliati a mantenere il file in “seed”, ossia a renderlo disponibile agli altri dopo averlo scaricato; per ovviare a questo problema, alcune comunità di torrent si sono impegnate per creare una sorta di schema di reputazione, e di sopperire a questa mancanza del protocollo.

Vi sono alcuni studi che hanno rivelato delle falle nel suo schema di incentivi. Attraverso la formalizzazione della *fedeltà alle specifiche*, è stato dimostrato che BitTorrent è vulnerabile a un certo tipo di manipolazioni da parte di peer egoisti, ad esempio (1) dichiarando di avere una banda in upload più bassa mantenendo comunque l'ordine relativo agli altri peer, in modo da diminuire il traffico in upload senza compromettere la velocità di download; (2) moltiplicando le proprie identità in modo da avere più chance di essere selezionato casualmente da altri nodi; (3) resettando la propria identità quando è conveniente (attacco whitewashing); (4) caricando dati spazzatura per aumentare il proprio tasso di upload. A causa di tutti questi problemi, vi sono ancora molti studi su come rendere BitTorrent un protocollo robusto alle manipolazioni.

Per i sistemi che utilizzano il file-swarming (come BitTorrent) è stato proposto un protocollo detto Fair, Optimal eXchange (FOX) per risolvere il problema del free-riding. Dato che tutti i peer cercano di minimizzare il proprio tempo di download, FOX usa un protocollo distribuito e sincronizzato basato su un albero k-ario per schedulare lo scambio di blocchi di file tra peers. Se tutti i peer usano questo protocollo, il tempo di completamento di un download è ottimo.

Per aumentare l'utilizzo del protocollo, se un nodo trova un vicino che non si comporta secondo il protocollo può avvisare l'intero sistema affinché gli altri nodi possano escludere il protocollo specificato. FOX comunque non è molto utilizzato perché il sistema è vulnerabile ad un singolo utente che “caccia” tutti gli altri peer.

3.5 BitTyrant

BitTyrant è un esempio di client che non aderisce alle specifiche di BitTorrent e tenta di migliorare le performance dell'utente a discapito delle prestazioni globali del sistema.

L'idea base di BitTyrant è che il client sceglie dinamicamente il numero e il tipo di peers a cui inviare dati. L'algoritmo dinamico di gestione del programma utilizza due stime statistiche, d (la velocità a cui i peer forniscono dati) e u (la velocità necessaria per guadagnare reciprocità). Vengono poi scelti i peer con la capacità più alta, basandosi su queste stime, e i dati vengono inviati alla minima velocità possibile che gli assicura la reciprocità. Alla

fine di ogni round, se un peer non scambia dati con lui, BitTyrant aumenta il valore di u mentre se il peer inizia a trasmettere, u viene abbassato.

I risultati hanno confermato che BitTyrant è molto più veloce del client “ufficiale”, anche del 70 %. Gli sviluppatori hanno mostrato che BitTyrant è performante sul lungo periodo, che può identificare il punto di diminuzione del guadagno marginale per i client con alta capacità, e che i peer con basse capacità sono anch’essi avvantaggiati dall’uso di BitTyrant. Tuttavia, queste migliori performance hanno un costo; ad esempio i nuovi utenti potranno notare lunghi tempi di avvio del sistema, e le relazioni con i peer vicini possono diventare instabili nel lungo periodo. Come BitTorrent, BitTyrant è vulnerabile ai whitewashers e agli attacchi Sybil.

4 Pagamenti

Un sistema P2P può utilizzare un sistema di pagamenti, in cui condividendo risorse si ottengono dei crediti da spendere poi per scaricare risorse da altri. Alcuni esempi di questo tipo sono MojoNation e Karma.

La prima analisi degli equilibri di un sistema p2p con i pagamenti è stata effettuata da Golle (2001); in questo modello, ogni peer prende una decisione indipendente dagli altri sul suo ammontare di file in download e in upload. Se ad ogni peer è addebitato un pagamento proporzionale al rapporto tra download e upload, esiste un unico equilibrio Nash in cui tutti i peer massimizzano i loro tassi di upload e download.

In un altro lavoro di Friedman (2006) si studia l’efficienza di un sistema basato sui pagamenti. Egli stabilisce l’esistenza di un equilibrio Nash non banale in cui tutti i peer giocano una strategia “di soglia” per ogni quantità totale fissata di denaro nel sistema. Per strategia di soglia si intende che un peer soddisferà una richiesta (e guadagnerà dei crediti) se il suo saldo è minore di un valore soglia, mentre si rifiuterà di soddisfare una richiesta se il suo saldo è superiore a questa soglia.

Analizzando l’efficienza degli equilibri per ogni quantità totale di credito nel sistema, è possibile determinare la quantità di moneta che massimizza l’efficienza per un sistema di una data grandezza. E’ interessante notare che è possibile controllare l’efficienza del sistema controllandone l’inflazione, ossia iniettando o eliminando denaro dal sistema man mano che questo cresce.

In un sistema P2P basato sui pagamenti è importante difendersi da sybil e whitewashing attacks. Infatti un sistema è vulnerabile se i nuovi arrivati hanno credito positivo, o se è concesso avere credito negativo, anche temporaneamente.

4.1 Karma

Così come accade nel sistema economico tradizionale, i peer in Karma guadagnano monete virtuali contribuendo al sistema con le loro risorse, monete che poi vengono spese per comprare altre risorse.

Karma cerca di eliminare il problema dei free-riders tenendo traccia delle monete disponibili per ogni peer; la valuta di queste monete è detta appunto *karma*. Tuttavia i valori di karma per ogni nodo sono mantenuti da altri nodi, chiamati “bank-set”, che sono responsabili dell’aggiornamento del valore dei karma di ogni utente.

Inizialmente un utente riceve un piccolo ammontare di karma quando accede per la prima volta (questo incoraggia il whitewashing). L’utente non può “acquistare” risorse se non ha sufficienti karma per procedere al pagamento.

Un elemento chiave dell’architettura di Karma è la mancanza di un’entità centrale, dato che la funzionalità di banca viene svolta da altri membri. Al fine di evitare che un solo elemento del bank-set falsifichi i karma di alcuni utenti, i karma sono memorizzati in più nodi, anche per questioni di fault-tolerance. Infatti Karma assume che ci siano almeno k nodi nel sistema in ogni momento, e che una certa frazione di questi nodi non è maligna.

Le informazioni del bank-set sono mantenute in una Distributed Hash Table (DHT), gestita tramite il protocollo chord. Ogni nodo è mappato al bank-set, in modo tale che i k nodi più vicini al nodo banchiere A sono il bank-set di A . Ogni membro del bank-set mantiene il valore corrente dei karma crittografati con la chiave privata di A , insieme al log delle operazioni e a un numero di epoca. Alla fine di ogni epoca vengono effettuati degli aggiustamenti economici con inflazione e deflazione, per migliorare le prestazioni del sistema quando molti nodi lasciano o accedono al sistema.

Un nodo che vuole scaricare un file acquisisce una lista di uploader potenziali e inizia un’asta al ribasso, o asta di secondo prezzo (asta di Vickrey). Una volta scelto un peer, inizia il protocollo Karma che permette inconsistenze momentanee durante lo scambio.

Il vantaggio principale di Karma è che, gestendo moneta virtuale, forza i peer a mantenere un saldo positivo tra le fasi di download e upload del sistema. Tuttavia c’è un overhead dato che alcuni peer devono anche comportarsi da banchieri per altri nodi, e dal punto di vista della game theory nessun peer ha un incentivo a prendersi questo ulteriore onere.

Karma resiste a diversi tipi di attacchi, tra cui replay attacks, nodi maligni, banchieri corrotti e attacchi denial of service. Karma permette i Sybil attack ma in modo limitato. Non è noto il comportamento del sistema su attacchi di tipo whitewashing o per quanto riguarda le azioni nascoste.

5 Cenni sulle azioni nascoste nei sistemi p2p

Come è stato già accennato nell'introduzione, essere free-rider non è l'unica scelta che un peer può fare in un sistema p2p. Infatti i peer possono decidere tutta una serie di variabili, ad esempio a quali peer connettersi, in quale momento connettersi alla rete, o se dire la verità su informazioni private come ad esempio i costi. In che modo possiamo evitare queste "azioni nascoste" da parte dei peer, utilizzando meccanismi basati sui pagamenti?

Consideriamo ad esempio il caso della condivisione di file su p2p. In reti come Gnutella o KaZaA i peer non devono solo condividere i file, ma devono anche inviare messaggi sullo stato della rete o sulle query ricevute da altri peer. Ad esempio, se un peer invia una query di ricerca, un altro peer deve forwardare il messaggio ai suoi vicini, e magari tentare di rispondere anch'esso alla query. Se un peer dovesse decidere di non rispondere più a queste query, o se volesse farlo in maniera probabilistica, le prestazioni del sistema degraderebbero notevolmente. Un nodo che si comporta in questo modo è inoltre difficilmente individuabile, dato che la rete cambia continuamente e i messaggi vengono spediti con criteri best-effort. Come incentivare i nodi a forwardare questi messaggi?

Questo problema si può generalizzare anche ad altri aspetti del P2P, e non solo: ad esempio nelle MANET i nodi mobili spesso eliminano i pacchetti da reinviare per evitare di consumare le batterie. Analizzeremo dunque l'efficienza di sistemi sotto questo framework, e come progettare soluzioni per invogliare gli utenti a partecipare.

6 Conclusioni

Il comportamento razionale degli utenti può potenzialmente distruggere ogni sistema P2P che si basa sull'assunzione di peer altruisti. Allo stato attuale non esiste un meccanismo compatibile agli incentivi che disinvolgia gli utenti al free-riding; è probabile che strumenti di altre discipline, come la sociologia, la finanza e le scienze comportamentali possano aiutare a comprendere il comportamento in sistemi P2P.

L'obiettivo principale sarà dunque di scoprire un meccanismo di incentivi computazionalmente trattabile che forza gli utenti razionali ad allineare i propri interessi personali con l'obiettivo del sistema, ossia di massimizzare il social welfare.