Métodos para Contenção de Poluição de Conteúdo em Redes P2P

Juliano F. Silva[1] **Marinho P. Barcellos**[1]
Marlom Konrath[1] **Luciano P. Gaspary**[2] **Rodolfo S. Antunes**[1]

[1] PIPCA – Programa de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
Av. Unisinos, 950 – 93.022-000 – São Leopoldo-RS – Brasil

{julianofs,marlomk,rsantunes}@unisinos.br, marinho@acm.org

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500 – 91.501-970 – Porto Alegre-RS – Brasil

paschoal@inf.ufrgs.br

Abstract. *Despite being currently one of the main Internet applications, P2P file sharing has been hampered by content pollution attacks. This paper proposes and analyzes a class of contention methods to reduce the dissemination of polluted content whose basic principle is to limit the amount of instantaneous downloads according to its reputation. The method is first proposed and then evaluated in terms of an idealized environment. The evaluation shows the efficiency of the contention method and the low overhead induced when the content is not polluted. Then, inspired by classic P2P designs, we propose and compare distributed contention methods.*

Resumo. *Apesar de ser uma das principais aplicações da Internet na atualidade, o compartilhamento de arquivos P2P tem sido fortemente prejudicado por ataques de poluição de conteúdo. Este artigo propõe e analisa uma classe de métodos de contenção de disseminação de conteúdo poluído cujo princípio básico é a limitação do número instantâneo de downloads de um conteúdo de acordo com a sua reputação. O método é primeiro proposto e então avaliado em termos de um ambiente idealizado, mostrando-se sua eficiência na contenção de poluição e baixa sobrecarga induzida quando o título não é poluído. A seguir, valendo-se de modelos clássicos para projeto de redes P2P, são propostos métodos de contenção distribuída, que são então comparados.*

1. Introdução

O compartilhamento de arquivos Peer-to-Peer (P2P) é uma das principais aplicações, senão a principal, da Internet na atualidade. Segundo [Kumar et al. 2006], estima-se que mais de 8 milhões de usuários estejam conectados simultaneamente às redes P2P FastTrack/Kazaa, Gnutella, eDonkey e eMule. Em 2005, 80% do tráfego em backbones IP referia-se a aplicações P2P [Barbera et al. 2005]. Em uma rede P2P, conteúdo é disponibilizado na forma de "títulos", que podem ser músicas, vídeos, documentos ou programas. Podem existir diferentes "versões" de um mesmo título, sendo elas diferenciadas por um identificador único, normalmente gerado com base no *hash* do conteúdo. Em função do processo natural de download, são disseminadas "cópias" de uma mesma versão em diversos pares da rede P2P.

Poluição de conteúdo (também denominada "envenenamento" [Christin et al. 2005]) é atualmente uma das maiores ameaças a sistemas P2P de compartilhamento de arquivos. O ataque tem como objetivo impedir a obtenção de determinado título, a partir da disponibilização massiva de versões incorretas na rede P2P. Usuários, incapazes de diferenciar as versões íntegras das comprometidas, efetuam download das versões poluídas. Conforme demonstrado por [Liang et al. 2005a], o ataque tem sido realizado em larga escala: mais de 50% dos arquivos populares de música na rede FastTrack/Kazaa eram poluídos, atingindo 80% em determinados títulos. Dentro da perspectiva de que usuários tentarão obter o mesmo título via repetidos downloads até encontrar a cópia íntegra, poluição causa um impacto negativo a usuários e aumenta desnecessariamente a carga na Internet.

Ataques de poluição somente são possíveis devido a características intrínsecas das redes P2P atuais, como descentralização, autonomia de pares e identidades fracas. Criar mecanismos que reduzam os níveis de poluição é um problema desafiador, que tem atraído grande atenção da comunidade científica recentemente.

Este artigo propõe e analisa um método de contenção de poluição baseado na determinação de limites ao número instantâneo de downloads. O limite é estabelecido de acordo com a reputação da versão, dinamicamente obtida a partir de votos na rede P2P. A taxa permitida de download a uma versão é diretamente proporcional à sua reputação. Inicialmente, o método é proposto em um ambiente ideal, centralizado, e então estendido para um ambiente distribuído, onde diferentes alternativas são descritas e analisadas. Avalia-se a penalidade inserida na disseminação de uma versão correta, bem como a sua eficiência ao reduzir a disseminação de uma versão poluída. O trabalho representa um passo na busca de contramedidas eficazes para o problema de poluição de conteúdo, e sua contribuição é delinear um esboço de solução e avaliar alternativas.

O restante do artigo está organizado como segue. A Seção 2 discute trabalhos relacionados à poluição de conteúdo em redes P2P. A Seção 3 apresenta a modelagem do método de contenção proposto em um ambiente ideal, enquanto a Seção 4 oferece uma avaliação do mesmo. A Seção 5 trata do problema em ambientes distribuídos, identificando aspectos relevantes e apresentando modelos de contenção distribuída. A Seção 6 compara os métodos propostos. A Seção 7 finaliza o artigo apresentando as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Poluição de conteúdo é um assunto que recentemente tem despertado grande interesse da comunidade científica. Diversos trabalhos [Liang et al. 2005a, Liang et al. 2005b, Christin et al. 2005, Lee et al. 2006, Kumar et al. 2006, Costa et al. 2006, Liang et al. 2006, Thommes and Coates 2006] têm se concentrado na avaliação de poluição, com metodologia analítica ou experimental, visando aumentar o entendimento sobre o problema. A seguir, discute-se os trabalhos mais relacionados a este.

[Liang et al. 2005b] propõe a criação de uma "lista-negra" (*blacklist*) a partir da avaliação de metadados em redes P2P, com coleta de informações sobre um título. A lista é composta de faixas de IP rotuladas como poluidoras, sendo construída com base na densidade de arquivos disponibilizados em cada IP, para um dado título. O sucesso da abordagem depende de freqüente download de metadados, gerando significativa sobrecarga. [Liang et al. 2006] dá continuidade ao trabalho anterior, utilizando uma lista-negra gerada por um sistema global de reputação de sub-redes. Em ambos os trabalhos, há limitações

ao caracterizar determinada rede como poluidora com base na alta densidade de versões. Muitos participantes de redes P2P estão conectados à Internet via NAT (*Network Address Translation*) em provedores de acesso; além disso, forjar endereços IP é uma técnica de ataque bastante conhecida e difundida, que poderia ser utilizada para burlar o mecanismo.

Credence [Walsh and Sirer 2005] é um sistema de reputação de objetos (versões) que possibilita ao usuário obter informações sobre a integridade de conteúdo. Credence não é adequado a ambientes com alta taxa de poluição, como usualmente encontrado em redes P2P atuais. Até a detecção das versões íntegras pelo mecanismo de reputação, grande quantidade de conteúdo poluído pode ser disseminada na rede.

Na proposta de contenção apresentada neste artigo, a taxa de disseminação de novas versões é ajustada de acordo com a confiança na integridade de seu conteúdo. Isso torna possível conter poluição enquanto o conteúdo ainda não foi ou está sendo obtido por muitos participantes. Limitar a taxa de disseminação, além de propiciar segurança aos usuários, reduz a eficiência do ataque, na medida em que conteúdo incorreto obtido da rede não é rapidamente analisado e apagado por usuários, tornando-o disponível a outros pares [Kumar et al. 2006, Costa et al. 2006]. Acredita-se, assim, que o método de contenção proposto neste trabalho é mais efetivo e robusto do que o controle imposto por sistemas de reputação tradicionais.

3. Contenção de Poluição em Ambiente Ideal

Esta seção apresenta uma proposta de contenção de poluição em um ambiente idealizado: memória compartilhada, infinitos processadores e pares que respeitam o protocolo estipulado. Sem perda de generalidade, a modelagem é feita com base em uma única versão de um título. Pares disponibilizam cópias de uma versão, que exceto no momento inicial, são obtidas por download de outros pares. Há um número infinito de pares desejando participar do compartilhamento de uma versão, efetuando seu download e então disponibilizando sua cópia. O crescimento da rede é limitado pela capacidade dos pares (número máximo de uploads simultâneos que um par está disposto a realizar), denotado como "grau de disseminação" ou δ. Um "semeador" (*seeder*) é um par que possui uma cópia da versão e a disponibiliza a outros pares, os "sugadores" (*leechers*).

O funcionamento básico da rede resume-se ao processo natural de download: um sugador contacta um semeador e solicita o download de uma versão. Ao final do download, caso a cópia recebida seja íntegra, o sugador transforma-se em semeador e passa a disponibilizar até δ uploads de cada vez (caso contrário, para efeito de modelo o sugador não se torna semeador e deixa de existir).

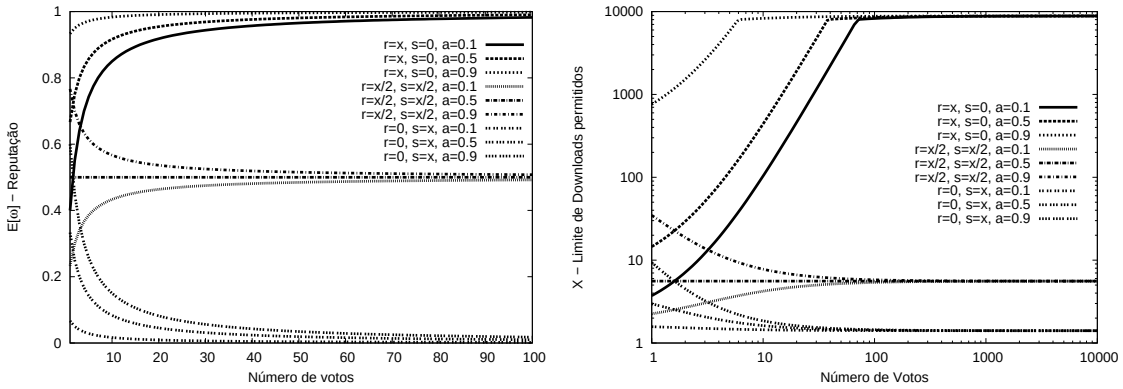
Com o método de contenção, ao final do download o par avalia a integridade da cópia e emite um voto, positivo ou negativo, sobre a mesma. Um voto positivo atesta a integridade da versão. Os totais de votos positivos e negativos são mantidos em r e s, respectivamente, e usados para derivar um escore de reputação da versão. O método proposto neste artigo baseia-se na Lógica Subjetiva [Josang et al. 2006] por ser um modelo de reputação maduro e amplamente aceito. São obtidos escores de crença e descrença de que a versão é de fato correta.

A Lógica Subjetiva baseia-se na avaliação de uma proposição $\omega = (b, d, u, a)$, onde b, d, u e a indicam crença (*belief*), descrença (*disbelief*), incerteza (*uncertainty*) e fator moderador (*base rate*), respectivamente, e $b, d, u, a \in [0, 1]$ e $b + d + u = 1$. O fator a define o peso da incerteza no cálculo do escore unificado de reputação. O escore é obtido por $E[\omega] = b + au$. Conforme demonstrado em [Josang et al. 2006], em um

sistema binário de reputação o escore é expresso por $E[\omega] = \frac{r+2a}{r+s+2}$. No presente caso, a reputação é usada para refletir a expectativa de um conteúdo não ser poluído; mais precisamente, ω é "a versão possui conteúdo íntegro".

A Figura 1(a) ilustra o valor de $E[\omega]$ em função do número de votos (no eixo *x*), para três casos: reputação de uma versão correta ($r = x$,$s = 0$); de uma versão indefinida ($r = \frac{x}{2}$, $s = \frac{x}{2}$) e de uma versão poluída ($r = 0$, $s = x$). Consideram-se ainda fatores moderadores $a = 0,9$, $a = 0,5$ e $a = 0,1$, gerando nove curvas. Conforme pode ser visto na figura, o fator moderador a influi principalmente na reputação inicial, onde a incerteza é muito alta. Conforme aumenta o número de votos, as três curvas convergem para um valor próximo de 1, próximo de 0,5 ou próximo de 0, em velocidade influenciada por a.

O método de contenção proposto se baseia em $E[\omega]$ para obter a reputação de uma versão e limitar a sua taxa de downloads simultâneos. Para tal, duas variáveis são centrais: X e Y, definidas e explicadas a seguir. X representa o número máximo de downloads permitido no momento, enquanto Y representa o número atual de downloads. O valor de X é computado de acordo com a Equação 1, em função de votos positivos e negativos ($E[\omega]$). O valor de Y é incrementado quando um download inicia, e decrementado quando termina (completado ou abortado). Por fim, os valores X e Y são empregados pelo método de contenção ao garantir que $Y \leq X$.



(a) (b)

Figura 1. (a) Reputação Teórica $E[\omega]$ e (b) Limiar de contenção de poluição (X), com α=1,3 e β=0,025

Assim como no gráfico da Figura 1(a), o valor de X dependerá do total de votos r e s. Analisa-se a seguir três casos básicos:

- $r \gg s$: X converge para um limiar superior, denominado X_{free};
- $r \simeq s$: X converge para um valor médio, denominado X_{med}, quando há uma divisão equânime entre votos positivos e negativos;
- $r \ll s$: converge para um limiar mínimo, denominado X_{min}.

A Equação 1 é empregada para obter X em função de $E[\omega]$.

$$X = \left(\frac{\alpha}{\max(\beta, (1 - E[\omega]))} \right)^{\alpha + E[\omega]} \quad (1)$$

A Figura 1(b) mostra o comportamento da Equação 1 para α=1,3 e β=0,025 (antes de efetuar um necessário arredondamento, que é superior). Verifica-se que a velocidade de

distribuição de conteúdo íntegro aumenta exponencialmente, convergindo, em diferentes velocidades e conforme a. Os valores de X_{min}, X_{med} e X_{free} dependem das escolhas de α e β; no caso em questão, $X_{min} = 8848$, $X_{med} = 7$ e $X_{min} = 2$. Ressalta-se que X_{free} determina o limiar a partir do qual os downloads solicitados serão liberados, ou seja, não há contenção.

O esquema de reputação proposto até o momento é, entretanto, sujeito a "ataques de traição". Tal ataque ocorre quando uma entidade acumula grande reputação e depois a explora de forma maliciosa [Marti and Garcia-Molina 2006]. Uma versão poluída pode acumular artificialmente votos positivos devido a um esforço inicial promovido por um conjunto de pares maliciosos atuando em conluio. Similarmente, uma cópia íntegra poderia ser prejudicada por um conjunto de votos negativos incorretos. A solução adotada segue a literatura recente em sistemas de reputação: atribuir maior peso a votos mais recentes, aplicando um fator λ ($0 \leq \lambda \leq 1$). A cada unidade de tempo (correspondente a um download), votos positivos e negativos sofrem um decréscimo ($r \leftarrow r \times \lambda$ e $s \leftarrow s \times \lambda$), antes de receberem os novos votos positivos e negativos. O valor de λ determina o peso do histórico de votos na reputação final, tal como sugerido por [Josang et al. 2006]. Analisou-se a influência de λ em X (por questões de espaço, é omitida). Em suma, quanto menor o valor de λ, maior será o crescimento e de decrescimento agregados, na medida em que os votos anteriores serão cada vez mais desvalorizados. Doravante, será considerado um valor de $\lambda = 0,85$.

Note-se que a Figura 1 mostra $E[\omega]$ e X em função do número de votos, e não em função do tempo; como um voto é originado de um download, e múltiplos downloads ocorrem simultaneamente, o número de votos não é um bom indicador de evolução temporal. No restante do artigo, a análise é feita em função do tempo e do número corrente de downloads.

Além disso, o valor de X não corresponde necessariamente ao número de downloads sendo realizados, Y, mesmo no caso mais agudo em que há infinitas requisições de download aos semeadores. O caso $Y < X$ pode ocorrer devido a uma limitação de recursos: cada semeador pode realizar até no máximo δ uploads, ou seja, disseminar δ cópias por unidade de tempo. O número de semeadores na rede, denotado como S, se comporta de forma monotônica crescente. O número de downloads possíveis é portanto $S \times \delta$. Exemplificando, em um sistema sem contenção com $\delta = 1$, cada par irá efetuar upload a um novo par que, ao final do download, irá transformar-se em semeador. Com isso, uma rede com 1 semeador terá 2 semeadores no momento seguinte, então 4, depois 8, e assim por diante. Com $\delta = 4$, a progressão de semeadores é 1, 5, 25, 125, 625... após apenas quatro downloads consecutivos, já existem 625 semeadores e 2500 uploads disponíveis.

4. Avaliação de Contenção de Poluição em Ambiente Ideal

O método de contenção possui dois objetivos conflitantes: minimizar a disseminação de versões poluídas e ao mesmo tempo prejudicar minimamente a disseminação de versões íntegras. Esta seção avalia o método proposto em função desses aspectos. Assume-se que o tempo de download de uma versão, denotado como T_d, é fixo (e por simplicidade, igual a 1). Por se tratar de um ambiente ideal, todos os demais tempos são negligíveis. A avaliação foi realizada de forma analítica, com um algoritmo de avanço discreto de tempo: a cada rodada, um par solicita autorização e, caso liberado, efetua o download, verifica a integridade da cópia e emite um voto. Caso negado o download, um par volta a tentar apenas na rodada seguinte. Atente-se para o fato de que todos os gráficos apresentados nesta seção possuem o eixo *y* em escala logarítmica.

Efetividade do método na contenção de versões poluídas. Neste experimento, considera-se no tempo inicial um ambiente com 20 semeadores de uma mesma versão, no qual todos já emitiram voto positivo (de forma incorreta) sobre essa versão. Neste contexto, é analisada a contenção ocasionada por votos subseqüentes, negativos e divididos, em um período equivalente a 20 unidades de tempo (o que pode corresponder a inúmeros downloads, devido ao paralelismo). A disseminação é efetuada com $\delta = 2$, ou seja, um semeador faz até dois uploads concorrentes de cada vez.

A Figura 2(a) mostra, primeiramente, que a curva sem contenção sobe agressivamente, levando a um rápido espalhamento da cópia poluída. Segundo, mostra que na curva com contenção e apenas votos negativos a taxa permitida de downloads decresce rapidamente, fazendo Y (por causa de X) convergir a 2. Com votos divididos, há igualmente uma convergência para um valor pequeno, em torno de 10, refletindo um tratamento "conservador" em relação à suspeita de poluição. Observa-se, portanto, a efetividade do método: a taxa de disseminação de conteúdo poluído é bastante reduzida se comparada com o download sem contenção.

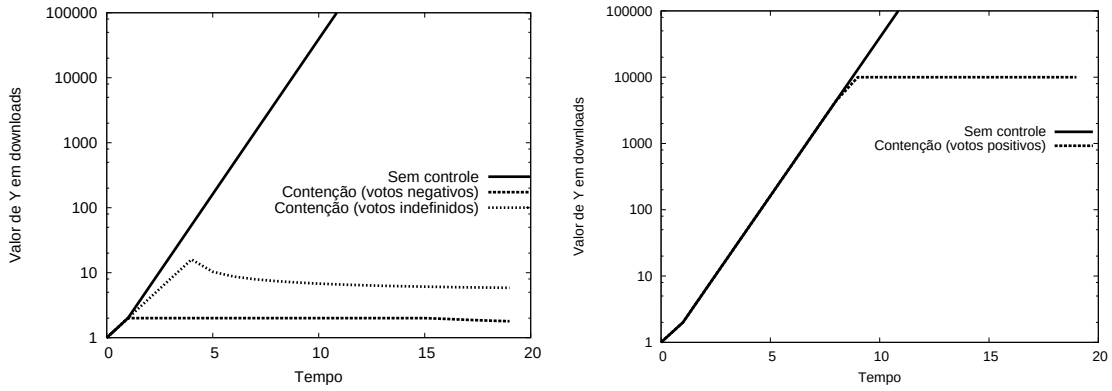


Figura 2. Disseminação de versão (a) poluída e (b) correta, considerando α=1,3 e β=0,025

Penalidade ocasionada ao download de versões íntegras. A Figura 2(b) ilustra essa análise, sob os mesmos parâmetros discutidos no parágrafo anterior, com exceção do número inicial de semeadores (1). A Figura 2(b) mostra que não há contenção indesejável até o momento em que o número de downloads permitidos atinge um patamar derivado de α e β: os esquemas com e sem contenção apresentam resultados idênticos até o tempo 8, quando X atinge X_{free}, passando a limitar em 8848 downloads concorrentes. Este último valor é resultado da escolha dos parâmetros $\alpha = 1,3$ e $\beta = 0,025$.

Método de contenção aplicado em ambientes dinâmicos. O terceiro experimento avalia o método de contenção em cenários onde a reputação tem mudanças de tendência. O sistema necessita ser capaz readequar-se pois, por exemplo, precisa ser resiliente a ataques de traição. A avaliação foi realizada em 4 períodos distintos, T_1, T_2, T_3, T_4, cada um com 50 unidades de tempo. Um período T_n pode exibir um entre três tendências: votos positivos (**P**), negativos (**N**) ou divididos (**D**). Objetivando entender o comportamento do método, foi realizada uma extensa análise com diferentes alternativas aos parâmetros (α, β, δ, λ) para cada um dos cenários possíveis. Por falta de espaço, apenas uma pequena parte dos resultados é apresentada. A Figura 3 mostra duas combinações possíveis: N-P-N-P e P-D-D-D. Em N-P-N-P, é ilustrada a capacidade do

método elevar a taxa permitida de downloads (T_2) com o recebimento de votos positivos, mesmo após um período de mesmo tamanho com votos negativos (T_1). Já quando o sistema passa a receber grande quantidade de votos negativos (T_3), há uma queda brusca na taxa de downloads permitidos. Ressalte-se que em T_3 a taxa de decrescimento inicial é bastante acentuada devido ao grande número de votos negativos recebidos logo no início de T_3 (quando o número votantes é X_{free}). Em P-D-D-D, verifica-se a convergência gradual a $X_{med} = 6$, gerada pela emissão de votos divididos em três períodos (T_2, T_3 e T_4), mesmo após grande quantidade de votos positivos ter sido inserida na rede (T_1).

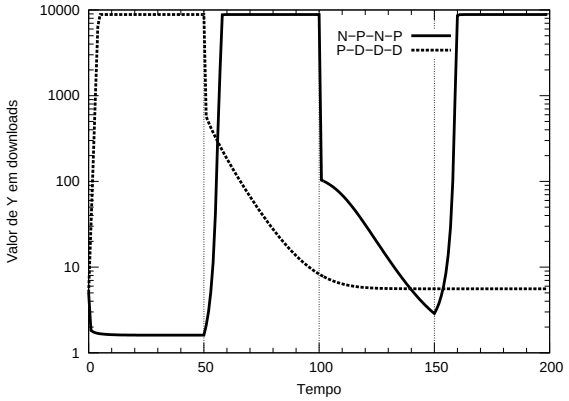


Figura 3. Reputação em cenários dinâmicos

5. Contenção de Poluição em Ambiente Distribuído

Esta seção trata do problema da contenção distribuída, apresentando alternativas que estendem o método apresentado na Seção 3. Ao contrário do ambiente ideal, é necessário lidar com os desafios de sistemas distribuídos, intrínsecos a redes P2P, como baixo acoplamento entre os pares e atrasos imprevisíveis de comunicação e execução. Assim, pela impossibilidade de trabalhar com os valores precisos de X e Y em um ambiente descentralizado, seus valores estimados, representados por $E[X]$ e $E[Y]$, devem ser determinados através de um protocolo distribuído. Naturalmente, quanto mais precisa a estimativa, melhor.

Existem diferentes abordagens para controlar de forma distribuída o valor de Y em relação a X. Aquela a ser adotada é influenciada por premissas ambientais (por exemplo, a estruturação da rede) e operacionais (expectativa dos usuários em relação à proteção contra versões corrompidas). As principais questões de projeto que influenciam os métodos de contenção distribuída são:

- **segmentação** (Global, Segmentada): relativo à divisão da rede P2P em segmentos menores, que podem ser autônomos ou cooperar (segmentos podem ser ainda subdivididos, formando uma hierarquia, mas esta abordagem não é explorada neste trabalho);
- **centralização** (Centralizada, Descentralizada): relativo ao grau de centralização, há um *trade-off* entre simplicidade/controle e tolerância a falhas/escalabilidade;
- **estruturação** (Estruturadas, Não-estruturadas): se o método de contenção depende de uma rede P2P estruturada para funcionar eficientemente;
- **autonomia** (Autônomas, Dependentes): concerne a autonomia de pares em cumprir ou não as decisões do método de contenção – em soluções autônomas, o método de contenção é apenas uma recomendação.

Considerando uma combinação dessas decisões de projeto, existe uma gama de métodos de contenção distribuída. Neste artigo, são apresentadas as quatro alternativas identificadas como principais: GCED, GDNA, SCED e SCND. Para cada uma delas, define-se como as seguintes operações básicas são realizadas: (a) obtenção da estimativa $E[X]$; (b) obtenção da estimativa $E[Y]$; (c) autorização ou negação de download de acordo com $E[Y] \leq E[X]$; (d) incremento de $E[Y]$ perante autorização de novo download; (e) decremento de $E[Y]$ e recálculo de $E[X]$ ao término de um download.

Adicionalmente, na análise, T_a denota o tempo para a decisão sobre autorização do download; T_d, o tempo de realização do download e T_v, o tempo da verificação da integridade da versão e envio de voto. Por fim, RTT representa *round-trip time*. Note-se que em transferências típicas de sistemas P2P, usualmente $T_d \gg RTT$, sendo T_d também dominante (minutos ou horas) sobre os demais tempos (milisegundos).

GCED – Global, Centralizada, Estruturada, Dependente. Este esquema é o mais simples de todos, e está baseado em um "gerente de versão". Um par solicita autorização para download de uma versão ao gerente via mensagem REQUEST; o gerente verifica o valor corrente de X, de Y, e então imediatamente decide sobre a autorização (com base nos valores correntes, localmente disponíveis). Se $Y \geq X$, a solicitação é negada, e o gerente responde ao par, que deve tentar de novo posteriormente. Caso contrário, o gerente faz $Y \leftarrow Y + 1$ e responde ao par com uma mensagem GRANT, autorizando o mesmo. A seguir, o par realiza o download enviando a mensagem RETR. Ao finalizar, verifica a integridade do arquivo e envia uma mensagem de voto VOTE de positivo ou negativo ao gerente, de acordo com o resultado. O gerente recebe o voto, atualizando r ou s e portanto X, e então $Y \leftarrow Y - 1$. O esquema está ilustrado na Figura 4.

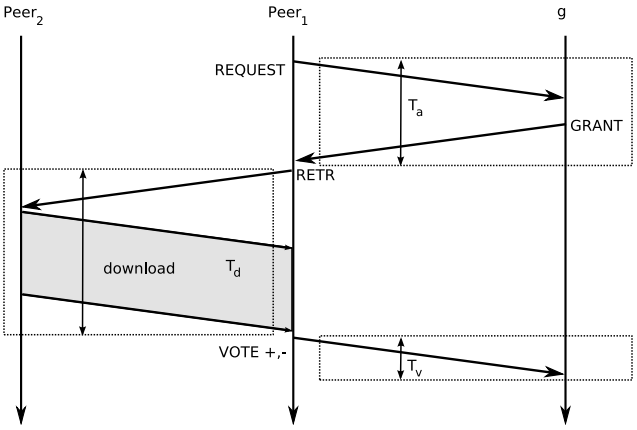


Figura 4. Diagrama de tempo correspondente ao esquema GCED

Embora o método GCED adote uma rede estruturada, ele não depende de tal facilidade. Uma DHT (*Distributed Hash Table*) permite que o gerente de uma dada versão seja encontrado de forma eficiente; do contrário, uma inundação é necessária, de forma a encontrar o gerente responsável pela versão (pragmaticamente, em uma rede não-estruturada poderia ser empregado um único gerente de versões global, de identidade bem conhecida).

GDNA – Global, Descentralizada, Não-estruturada, Autônoma. Ao contrário do anterior, este método é totalmente descentralizado: todos os pares possuem o mesmo papel. Cada par é autônomo ao decidir se pode ou não realizar um download. Sua decisão é baseada na execução *correta* do algoritmo proposto, que inclui a determinação das

estimativas para $E[X]$ e $E[Y]$. Em uma rede P2P usual, não é viável empregar um protocolo distribuído, como por exemplo os de consenso [Veríssimo and Rodrigues 2001]. No esquema proposto, antes de requisitar o download, um par deve consultar os demais pares na rede para obter $E[X]$ e $E[Y]$. Cada par P_i armazena uma tripla de valores binários (y_i, r_i, s_i), que representa, respectivamente, o número de downloads correntes da versão neste par (0 ou 1), se há um voto positivo, e se há um voto negativo em relação à determinada versão. A consulta aos valores em cada par é realizada através de um algoritmo sistólico sobre o *overlay*, descrito a seguir.

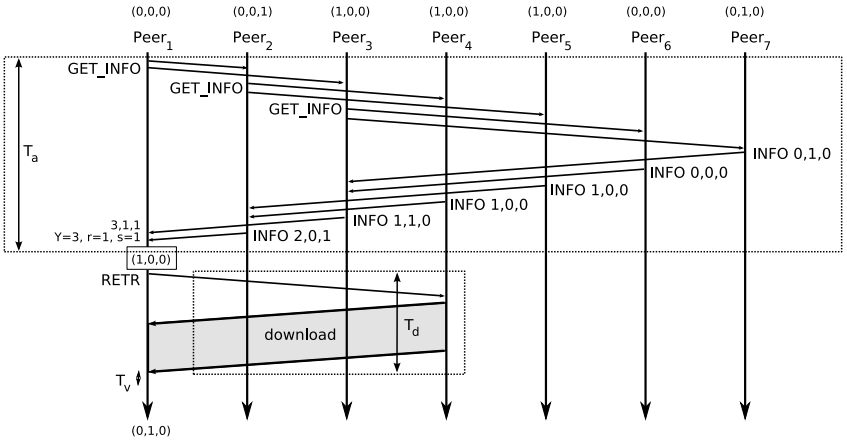


Figura 5. Diagrama de tempo correspondente ao esquema GDNA

Além da tripla, pares mantém uma variável booleana que indica se estão marcados ou não. Inicialmente, todos os (demais) pares estão desmarcados. A rede é inundada com mensagens de coleta de informações, processo que inicia com o envio da mensagem GET_INFO pelo par (que deseja o download) aos seus vizinhos; cada vizinho que ainda não recebeu a marca, é marcado e encaminha a mensagem aos seus vizinhos a não ser para a ligação pela qual a mensagem chegou. Pares que recebem uma mensagem GET_INFO mais cedo ou mais tarde retornam uma mensagem INFO (y, r, s) à origem da mensagem sua tripla (y, r, s). Nestes dois últimos casos, assume-se que o par tenha completado o download e verificado a cópia. Pares marcados que recebem uma mensagem GET_INFO retornam imediatamente uma mensagem INFO com $(0, 0, 0)$; os não marcados esperam respostas de cada um de seus filhos e as consolidam através de uma *soma* dos valores individuais da tripla: para um par com n vizinhos, $(y, r, s) = (\sum_{i=0}^{n} y_i, \sum_{i=0}^{n} r_i, \sum_{i=0}^{n} s_i,)$, onde y_i, r_i, s_i representam os valores recebidos do vizinho i. A mensagem com os valores consolidados é enviada ao par que encaminhou a primeira mensagem GET_INFO. De maneira recursiva, mensagens INFO convergem para o par responsável pela solicitação, permitindo ao mesmo obter ao final uma tripla (y, r, s) que representa estimativas do somatório de downloads em execução, o número de votos positivos e o número de votos negativos. Com base nos votos, o par então computa localmente $E[X]$ – com base na reputação da versão – através da fórmula definida na Equação 1. A Figura 5 ilustra essa abordagem para uma rede com 7 pares.

Considerando que as operações de download ocorrem em um arcabouço temporal de ordens de magnitude superior ao RTT, assume-se que a abordagem proposta produza estimativas relativamente atualizadas. Para ilustrar esse ponto, considere-se o caso de uma consulta a uma árvore binária. A altura da árvore corresponde ao número de saltos que devem ser dados até encontrar o par mais distante, e o dobro para o caminho de ida-e-

volta. O tempo para que tal consulta seja completada, definindo o valor de T_a, é da ordem de $\log_2(N+1) \times RTT$, onde N representa o número de pares na rede. Exemplificando, uma consulta em uma rede com 500.000 pares teria T_a=18,2 s, para um RTT de 200 ms e um tempo de download superior a 10 min. Apesar desse atraso ser comparativamente aceitável (apenas 2.9%), sabe-se que inundações apresentam baixa escalabilidade, sendo proibitivos o tráfego e o processamento gerados.

A solução usualmente empregada para aumentar a escalabilidade em tais sistemas, tal como [Gnutella 2007], é limitar o escopo das mensagens (seu tempo de vida ou *horizonte*) e o número máximo de vizinhos para os quais um par encaminha uma mensagem. No entanto, por não consultar todos os pares, limitar o escopo da inundação pode *subestimar* os valores de Y, r e s, potencialmente diminuindo a precisão das estimativas.

SCED – Segmentada, Centralizada, Estruturada, Dependente. Essa abordagem, para controle de X e Y, baseia-se na divisão de uma rede P2P estruturada (como Chord [Stoica et al. 2003]) em segmentos. A rede, com 2^τ identificadores de pares, é particionada em 2^ϕ segmentos, cada um com $2^{\tau-\phi}$ identificadores. O valor de X é adaptado de acordo com o tamanho do segmento, para representar uma fração do global: $X_i = \lceil X/2^\phi \rceil$. Refletindo isso, o cômputo de Y ($E[Y_i]$) é obtido a partir do somatório dos downloads ocorrendo no segmento. Portanto, $X = \sum_{i=1}^{2^\phi} X_i$ e $Y = \sum_{i=1}^{2^\phi} Y_i$, onde 2^ϕ é o número de segmentos.

Tal como a primeira abordagem proposta, a decisão de autorização de downloads é tomada por um gerente de versão[1]. Entretanto, diferentemente, existe um gerente por segmento. Em cada segmento S_i, um gerente g_i é autônomo para controlar o valor de X correspondente ao segmento, X_i, e o número corrente de downloads no mesmo, Y_i. O gerente do segmento, g_i, é escolhido como o último par do segmento. Baseado nos valores de X e Y locais, autoriza ou não os downloads.

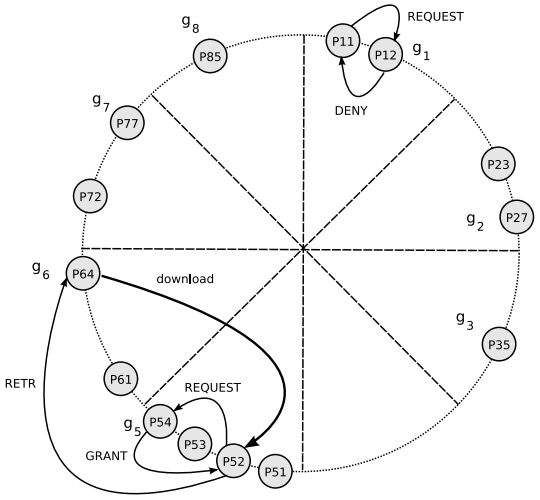


Figura 6. Exemplo ilustrando arquitetura do esquema SCED

Assume-se a título de exemplo que $\tau = 128$ e $\phi = 3$. A Figura 6 ilustra o funcionamento do esquema: par $P11$ requisita autorização de download ao seu gerente em $P12$, que o nega (as mensagens são equivalentes às abordagens anteriores). Já o par

[1]gerentes podem atuar tanto de forma isolada como cooperativa; neste trabalho, como um primeiro passo, é considerada apenas a alternativa mais simples, sem cooperação entre gerentes.

$P52$ solicita autorização ao seu gerente, g_5 (em $P54$), e recebe uma resposta positiva. O par então solicita o download a $P64$, que detém uma cópia da versão em questão. Note-se que o exemplo ilustra o caso em que o recurso está fora do segmento em que se encontra o par origem.

Os valores de τ e ϕ são fundamentais para o bom funcionamento do esquema. A medida que ϕ se aproxima de τ, o número de segmentos aumenta e seu tamanho diminui. Em um extremo, se $\phi = \tau$, existem 2^τ segmentos com um (ou zero) pares cada um. Neste caso, $X_i = 1$, ou seja, $X = 2^\tau$; isto é o equivalente à inexistência de controle, pois todo par está autorizado a fazer um download. No outro, se $\phi = 0$, existe apenas um segmento abrigando todos os pares, caso específico em que SCED é idêntico ao GCED.

SCND – Segmentada, Centralizada, Não-estruturada, Dependente. Em uma rede P2P não-estruturada, executar a segmentação de forma adequada não é uma tarefa trivial. A criação de segmentos é dificultada pelo desconhecimento do tamanho da rede pelos pares e a transiência da rede. A alternativa proposta é utilizar a estrutura criada por super-pares para definir segmentos na rede. Assim, pares ligados a um mesmo super-par pertencem ao mesmo segmento.

O papel de gerente de versão é mantido pelo super-par do segmento em questão. A rede P2P, assim, será dividida em U segmentos, onde U é o número de super-pares. A Figura 7 ilustra o funcionamento desse método. Um par P_i que deseja efetuar o download encaminha uma requisição ao super-par g_i (REQUEST). Em g_i são mantidas as estimativas locais $E[X_i]$ e $E[Y_i]$. Ao avaliar $E[X_i] \leq E[Y_i]$, é realizada a decisão de autorizar (GRANT) ou não (DENY) o download. Ao aprovar a requisição, g_i deverá incrementar $E[Y_i]$. Por fim, P_i encaminhará um voto positivo (r) ou negativo (s) ao gerente, que atualizará a sua estimativa $E[X_i]$.

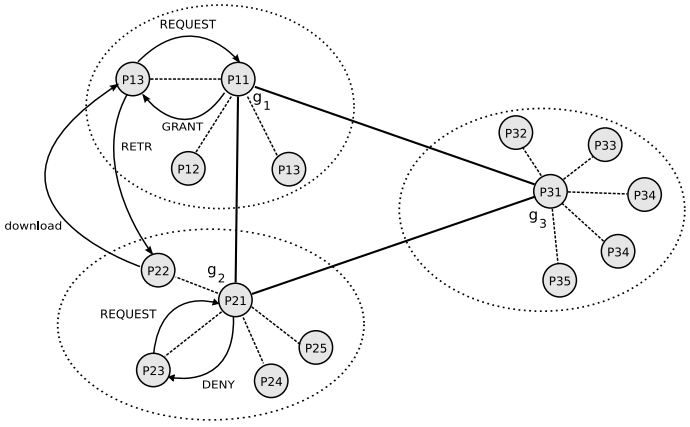


Figura 7. Exemplo ilustrando arquitetura do esquema SCND

Uma possibilidade alternativa é promover a cooperação entre os segmentos através dos super-pares. Como os mesmos estão interligados (em uma rede não-estruturada), os valores pertinentes a cada segmento (números de votos e downloads correntes) poderiam ser trocados entre os super-pares, possibilitando estimativas globais. Como base nas estimativas para r e s globais, um super-par manteria a estimativa de $E[X]$ global, e obteria a estimativa local com $E[X_i] = \frac{E[X]}{U}$, permitindo assim um $E[X_i]$ mais preciso.

6. Avaliação da Contenção de Poluição em Ambiente Distribuído

A análise das abordagens de implementação distribuídas considera os seguintes requisitos: tolerância a falhas, escalabilidade, vulnerabilidades potenciais, precisão e sobrecarga. Considera-se a sobrecarga como a latência induzida por uma abordagem, denotada como l, e definida como $l = \frac{T_a+T_v}{T_a+T_d+T_v}$ (relembrando, T_a, T_d e T_v são respectivamente os tempos de autorização, de download e de verificação e voto).

GCED. O gerente de versão é um ponto central de falhas e de ataques, e um candidato à ocorrência de gargalos na comunicação. Como vulnerabilidade, devido à autonomia, um par pode ignorar a presença do gerente e contactar diretamente o detentor da versão para efetuar o download. Isso, entretanto, agrega risco ao participante, que não possui qualquer indício sobre a validade do conteúdo. Uma alternativa refere-se ao gerente manter a referência à versão, que somente seria enviada ao participante no caso de o download ter sido autorizado. Além disso, para manter estimativas corretas o gerente deve ser capaz de detectar downloads não finalizados ou falhas – o que agrega certa complexidade ao método. Como a gerência é centralizada, as estimativas são tão precisas quanto atuais e corretos os valores de r e s. Com $T_a = RTT$ e $T_v = \frac{RTT}{2}$, pode-se obter l.

GDNA. Abordagem descentralizada, eliminando a existência de um elemento centralizador (e as características negativas associadas ao mesmo). Para obter autorização, um par deve consultar outros pares quanto aos seus votos e se estão executando um download, mas poderia seguir adiante caso não recebesse a resposta de um determinado par. A escalabilidade da abordagem é comprometida pela necessidade de inundar a rede (no algoritmo sistólico), devido à quantidade de mensagens. A sobrecarga l é afetada por $T_a = (\log_h N + 1) \times RTT$, onde h refere-se ao horizonte da consulta, e $T_v = 0$ (local). As estimativas podem não ser exatas, devido à alta transiência de pares, horizonte limitado e mensagens em trânsito. Outro ponto negativo é que os participantes também podem optar por não seguir o protocolo especificado, ignorando a decisão local do sistema de reputação, porém, com risco de obtenção de conteúdo poluído. Além disso, a ação de pares maliciosos tende a afetar diretamente o método, na medida em que podem adulterar as informações consolidadas recebidas na fase de contração do algoritmo (*recebimento das respostas*). Como alternativa, pode-se adaptar o método para encaminhar os votos diretamente ao requisitante, elevando, em contrapartida, a sobrecarga de mensagens na rede.

SCED. Este esquema emprega um gerente por segmento, possuindo portanto um ponto central assim como no GCED. Entretanto, existem múltiplos gerentes, diminuindo a dependência do sistema por uma entidade única, central. Além disso, deve-se considerar que o gerente é responsável por uma versão apenas, e não por todas as versões na rede. Essas características impactam na tolerância a falhas, escalabilidade e segurança. Além disso, redes P2P estruturadas, tradicionalmente, possuem métodos para realocação de identificadores, necessários à acomodação da rede devido à entrada e saída de pares; tal pode ser usado para aumentar a tolerância a falhas do SCED. O fato de o gerente ser malicioso tende a prejudicar o funcionamento do método; por outro lado, trata-se de apenas um segmento, o que agrega robustez ao método frente a abordagem de gerente global. O número de segmentos (2^ϕ) afeta a precisão do método: quanto maior for o valor de τ, menos preciso será o método. O intercâmbio de informações entre os gerentes de cada segmento pode aumentar a precisão na estimativa global de X, $E[X]$. A sobrecarga ocasionada em cada segmento depende de T_a, que no pior caso corresponde a contactar (usando a *finger table*) um gerente mais distante no segmento de tamanho 2^ϕ, e portanto

$T_a = \phi \times \frac{RTT}{2} + \frac{RTT}{2}$. O tempo de verificação e voto é $T_v = \frac{RTT}{2}$.

SCND. A existência de um gerente por segmento reduz o grau de centralização, não havendo ponto central ao mecanismo de contenção. Tolerância a falhas vale-se de mecanismos de contingência já existentes em redes não-estruturadas semi-centralizadas, mitigando falhas do gerente (super-par). Além disso, ataques ou falhas de um gerente não prejudicam a rede como um todo (dada a redundância de conexões), agregando robustez à abordagem. Entretanto, como segmentos podem estar desbalanceados, o comprometimento de um gerente pode afetar um número arbitrariamente grande de participantes. Quanto à precisão, as estimativas mantidas possuirão erro proporcional ao número de segmentos; quanto maior o número de segmentos, menor será a precisão. A sobrecarga para cada segmento pode ser obtida em função de $T_a = RTT$ e $T_v = \frac{RTT}{2}$.

7. Conclusão

Nos últimos anos, sistemas P2P tem obtido grande notoriedade entre os usuários e no meio científico, sendo o compartilhamento de arquivos a aplicação de maior visibilidade. Esses sistemas tem sido alvo de ataques massivos de poluição de conteúdo, o que pode comprometer fortemente a sua utilização. Boa parte dos trabalhos atuais nessa área concentra-se em mapear e delimitar o ataque, criando o embasamento necessário a propostas de solução. Entretanto, verifica-se ainda a ausência de trabalhos que proponham métodos robustos e eficazes para o tratamento de ataques de poluição.

Este artigo propõe e analisa métodos para contenção de poluição de conteúdo baseados na limitação do número corrente de downloads. A velocidade de disseminação é controlada de acordo com a confiança na integridade de versão. Inicialmente, o método é proposto e avaliado em um ambiente ideal, onde foi possível mostrar a eficiência teórica do método na contenção de poluição e a baixa penalidade à disseminação de versões corretas. Posteriormente, foram identificadas as principais questões de projeto, e então propostos quatro métodos básicos de contenção distribuída. Os métodos são comparados em termos de critérios importantes como escalabilidade e sobrecarga.

A contribuição principal deste trabalho é a proposta e avaliação de uma classe de métodos para contenção de poluição baseada na limitação de downloads simultâneos de acordo com a reputação de versões (usando Lógica Subjetiva). Sabe-se, entretanto, que a solução para ataques de poluição é uma tarefa bastante complexa e desafiadora; este artigo não pretende esgotar o assunto, mas sim apontar mais uma possibilidade no espaço de soluções contra poluição de conteúdo em P2P. Como trabalhos futuros, novos estudos estão sendo conduzidos no sentido de eliminar as premissas simplificatórias adotadas e determinar a resiliência do método à ação de atacantes. Além disso, está sendo realizada uma avaliação da eficácia dos métodos de contenção distribuída e sobrecarga de latência e comunicação para os métodos de contenção distribuída propostos.

Referências

Barbera, M., Lombardo, A., Schembra, G., and Tribastone, M. (2005). A markov model of a freerider in a bittorrent P2P network. In *IEEE Global Telecommunications Conference (GLOBECOM '05)*, volume 2, pages 985–989, St. Louis, MO, USA.

Christin, N., Weigend, A. S., and Chuang, J. (2005). Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *6th ACM conference on Electronic commerce (EC '05)*, pages 68–77, New York, NY, USA. ACM Press.

Costa, C., Soares, V., Benevenuto, F., Vasconcelos, M., Almeida, J., Almeida, V., and Mowbray, M. (2006). Disseminação de conteúdo poluído em redes P2P. In *XXIV Simpósio Brasileiro de Redes de Computadores*, Curitiba, PR, Brasil.

Gnutella (2007). Gnutella website. http://www.gnutella.com/.

Josang, A., Hayward, R., and Pope, S. (2006). Trust network analysis with subjective logic. In *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*, pages 85–94, Darlinghurst, Australia. Australian Computer Society, Inc.

Kumar, R., Yao, D., Bagchi, A., Ross, K. W., and Rubenstein, D. (2006). Fluid modeling of pollution proliferation in P2P networks. In *ACM/IFIP SIGMETRICS/Performance 2006*, volume 34, pages 335–346, St. Malo, France.

Lee, U., Choiz, M., Choy, J., Sanadidiy, M. Y., and Gerla, M. (2006). Understanding pollution dynamics in P2P file sharing. In *5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, Santa Babara, CA, USA.

Liang, J., Kumar, R., Xi, Y., and Ross, K. W. (2005a). Pollution in P2P file sharing systems. In *The 24th Conference on Computer Communications (INFOCOM 2005)*, volume 2, pages 1174–1185, Miami, FL, USA.

Liang, J., Naoumov, N., and Ross, K. W. (2005b). Efficient blacklisting and pollution-level estimation in P2P file-sharing systems. In *ASIAN INTERNET ENGINEERING CONFERENCE (AINTEC)*, pages 1–21, Bangkok, Thailand.

Liang, J., Naoumov, N., and Ross, K. W. (2006). The index poisoning attack in P2P file-sharing systems. In *The 25th Conference on Computer Communications (INFOCOM 2006)*, Barcelona, Spain.

Marti, S. and Garcia-Molina, H. (2006). Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472–484.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, F. M., Dabek, F., and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32.

Thommes, R. and Coates, M. (2006). Epidemiological modelling of peer-to-peer viruses and pollution. In *The 25th Conference on Computer Communications (INFOCOM 2006)*, pages 981–993, Barcelona, Spain. IEEE.

Veríssimo, P. and Rodrigues, L. (2001). *Distributed Systems for System Architects*. Springer, Boston, USA, 1 edition.

Walsh, K. and Sirer, E. G. (2005). Fighting peer-to-peer spam and decoys with object reputation. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 138–143, New York, NY, USA. ACM Press.