

# Separando Joio de Trigo com *Funnel*: Combate à Poluição de Conteúdo em Comunidades BitTorrent

Flávio Roberto Santos, Luciano Paschoal Gaspar, Marinho Pilla Barcellos

PPGC – Programa de Pós-Graduação em Computação  
UFRGS – Universidade Federal do Rio Grande do Sul  
Av. Bento Gonçalves, 9500 – Campus do Vale – Bloco IV – Porto Alegre, RS

{flavio.santos,paschoal}@inf.ufrgs.br, marinho@acm.org

**Abstract.** *BitTorrent is currently the most adopted Peer-to-Peer (P2P) protocol by file-sharing communities. Such communities are exposed to content pollution attacks (i.e., dissemination of corrupted files, viruses or other malware), which require a moderation effort from their administrators. The complexity of such a cumbersome task increases as the publishing rate of the disseminated content grows. In order to tackle this problem, this paper proposes Funnel, a pollution control mechanism designed to BitTorrent communities. Funnel uses a conservative approach, by assuming every published content as polluted and controlling its dissemination based on users' votes. The instantiation of an automatic and specific mechanism to such communities, in addition to the conservative strategy, comprise the main contributions of this work. The evaluation of the proposed solution, realized through experiments, shows that Funnel mitigates the dissemination of polluted content, imposing a low overhead in the distribution of authentic ones.*

**Resumo.** *BitTorrent é atualmente o protocolo Peer-to-Peer (P2P) mais difundido em comunidades de compartilhamento de arquivos. Tais comunidades estão sujeitas a ataques de poluição de conteúdo (cópias corrompidas, vírus ou outros malwares), exigindo uma tarefa de moderação por parte dos seus administradores. Essa tarefa torna-se crítica, muitas vezes proibitiva, à medida que a taxa de adição de novos conteúdos aumenta. Para lidar com o referido problema, neste artigo propõe-se Funnel, um mecanismo para contenção de poluição em comunidades BitTorrent. O Funnel age de forma conservadora, assumindo que todo conteúdo publicado é poluído e controlando sua disseminação com base nos votos dos usuários. A instanciação de um mecanismo automático e específico para tais comunidades, juntamente com a estratégia conservadora, constituem as principais contribuições do trabalho. A solução resultante foi avaliada por meio de experimentação e os resultados demonstram que a utilização do Funnel reduz a proliferação de cópias poluídas, apresentando uma baixa sobrecarga na distribuição de cópias autênticas.*

## 1. Introdução

Sistemas de compartilhamento de arquivos têm sido amplamente utilizados para troca de conteúdo arbitrário na Internet. O Napster ficou conhecido como o primeiro desses sistemas a alcançar milhões de usuários. Desde então, estudos mostram um constante crescimento do tráfego na Internet gerado por essa classe de aplicações [Kumar et al. 2006], que atualmente inclui KaZaA, Gnutella e BitTorrent.

Em alguns desses sistemas, como é o caso do BitTorrent, tornou-se comum o agrupamento de usuários em comunidades para o compartilhamento de arquivos. Algumas dessas comunidades seguem características específicas, por exemplo fornecendo apenas um determinado tipo de conteúdo (músicas, filmes etc). Observando-se as comunidades por outra perspectiva, existem aquelas em que qualquer usuário pode publicar conteúdos, sendo denominadas de “comunidades abertas”, enquanto outras exigem algum tipo de autenticação, sendo denominadas de “comunidades fechadas”. Apesar de as comunidades fechadas oferecerem barreiras para publicação de conteúdo, se as identidades não forem controladas e de posse de usuários confiáveis, conteúdos impróprios ou considerados poluídos podem ser disponibilizados. Esse problema é ainda mais grave se pensarmos em comunidades abertas.

Poluição de conteúdo pode se manifestar de três formas diferentes. A primeira consiste na inserção de cópias corrompidas de um conteúdo mantendo seus metadados originais [Liang et al. 2005]. Uma segunda técnica consiste em manipular o gerador de identificadores de objetos e inserir um conteúdo poluído com o mesmo identificador do conteúdo original [Benevenuto et al. 2006]. Por fim, a terceira forma de poluição consiste em um ataque de negação de serviço ocasionado pela inserção de identificadores inválidos para conteúdos, induzindo o sistema a falhas no processamento da busca [Liang et al. 2006]. No contexto deste trabalho, foi realizado um levantamento sobre índices de poluição junto a administradores de comunidades BitTorrent, que estimaram que 25% do conteúdo diário publicado necessita de algum tipo de moderação. Uma entrevista recente apresenta relatos sobre a presença de conteúdo poluído em tais comunidades [TorrentFreak 2009].

Algumas abordagens foram criadas para solucionar o problema de poluição em sistemas *Peer-to-Peer* (P2P). Essas abordagens podem ser classificadas em duas categorias. A primeira delas compreende um conjunto de soluções para poluição em sistemas P2P em geral [Kamvar et al. 2003, Walsh and Sirer 2006, Costa et al. 2007, Costa and Almeida 2007, da Silva et al. 2007]. Apesar de serem propostas de estratégias interessantes para o combate à poluição, apenas o Credence [Walsh and Sirer 2006] foi implementado e avaliado experimentalmente. A segunda categoria apresenta soluções com um maior caráter prático e são diretamente voltadas ao combate de poluição em comunidades BitTorrent. Essa categoria compreende abordagens simples em que usuários enviam comentários sobre os conteúdos, fazem denúncias aos administradores das comunidades ou, ainda, emitem votos para banir o conteúdo automaticamente, sendo necessária uma intervenção dos administradores para reintegrá-lo ao sistema. As abordagens dessa categoria exigem um exercício de moderação através da inspeção manual dos conteúdos publicados. Além disso, não são fornecidos mecanismos para incentivar a participação dos usuários (envio de comentários, denúncias ou votos) no combate à poluição.

Tendo em vista as limitações associadas às soluções mencionadas, este trabalho propõe um mecanismo, denominado Funnel, para controlar a disseminação de conteúdo poluído em comunidades BitTorrent fechadas. O mecanismo se baseia no modelo anteriormente proposto por Silva *et al.* [da Silva et al. 2007], que considera votos positivos e negativos atribuídos aos conteúdos para classificá-los como autênticos ou poluídos. O princípio consiste em controlar de forma automática a distribuição das cópias de acordo com os votos emitidos: quanto maior a proporção de votos negativos, menos *downloads*

concorrentes são autorizados. Além disso, o Funnel dispõe de um mecanismo de incentivo para que os usuários emitam seus testemunhos sobre os conteúdos recuperados. O modelo proposto foi instanciado em um sistema para gerência e compartilhamento de arquivos, dispensando qualquer modificação no funcionamento dos agentes BitTorrent dos usuários. O protótipo foi avaliado por meio de experimentos, cujos resultados permitiram observar que o Funnel reduz satisfatoriamente a proliferação de cópias poluídas e introduz uma baixa sobrecarga na distribuição de cópias autênticas.

As demais seções estão organizadas conforme descrição a seguir. A Seção 2 discute algumas estratégias existentes na literatura para combater a poluição de conteúdo em sistemas P2P e em comunidades BitTorrent. A Seção 3 revisa os fundamentos da arquitetura BitTorrent e ressalta os pontos-chave para o entendimento do Funnel. O mecanismo proposto é apresentado na Seção 4 e, em seguida, na Seção 5, são discutidos os resultados obtidos com a avaliação experimental. Por fim, a Seção 6 apresenta as considerações finais e perspectivas de trabalhos futuros.

## 2. Trabalhos Relacionados

Nesta seção, são analisadas as principais propostas para contenção de poluição em sistemas P2P de compartilhamento de arquivos. O primeiro trabalho apresenta o algoritmo Eigentrust [Kamvar et al. 2003], utilizado no cálculo de valores de confiança globais para os usuários do sistema, empregando para isso apenas métricas de confiança locais. Essa abordagem apresenta uma validação matemática sólida, mas possui limitações importantes, em particular a escalabilidade limitada e a necessidade de um conjunto prévio de usuários confiáveis no sistema para funcionar corretamente.

Um segundo trabalho propõe o modelo de reputação Credence [Walsh and Siner 2006]. Esse modelo apresentou resultados satisfatórios ao ser avaliado sobre o protocolo Gnutella. Todavia, o Credence se baseia na reputação atribuída aos objetos para marginalizá-los do sistema, viabilizando uma ampla proliferação dos mesmos nos instantes iniciais em que sua reputação agrega pouca informação. Esse problema é agravado quando existem altas taxas de adição de novos conteúdos poluídos.

O Hybrid [Costa and Almeida 2007], assim como o Credence, utiliza um mecanismo de reputação para combater poluição. Desenvolvido como uma evolução do Scrubber [Costa et al. 2007], esse mecanismo combina a reputação dos usuários e dos conteúdos para punir os poluidores. Por utilizar uma estratégia semelhante ao Credence, o Hybrid também exige um tempo considerável para classificar um conteúdo como poluído.

Apesar de as abordagens citadas terem sido propostas em um contexto de sistemas P2P, nenhuma delas foi projetada levando em consideração as especificidades das comunidades BitTorrent. A seguir, são discutidas abordagens empregadas especificamente nessas comunidades. Tratam-se de mecanismos que foram desenvolvidos de maneira *ad hoc*, e não possuem um estudo científico que os suporte. A abordagem mais simples permite que usuários enviem comentários gerais sobre um determinado conteúdo publicado. Apesar de os usuários utilizarem tais comentários para enviar relatos sobre poluição, eles também são utilizados de forma arbitrária e não apresentam um propósito bem definido. Uma outra proposta utiliza um mecanismo em que usuários denunciam algum conteúdo da comunidade; e a partir dessas denúncias, um grupo de administradores realiza um trabalho manual de inspeção e validação. Por fim, outra abordagem utiliza um conjunto

de votos emitidos pelos usuários para banir um conteúdo suspeito, sendo necessária uma intervenção do administrador para validá-lo e, se for o caso, reintegrá-lo ao sistema.

Em síntese, todas as abordagens empregadas em comunidades BitTorrent exigem um trabalho de inspeção manual dos administradores, ocasionando um alto custo para a manutenção dos conteúdos publicados. Esse problema é ainda maior em comunidades que possuem altas taxas de adição de novos conteúdos. Além disso, dependem que usuários sejam altruístas em reportar conteúdos poluídos, haja vista que essas abordagens não apresentam mecanismos para incentivar a participação dos mesmos. Nas seções subsequentes, será apresentada uma proposta de solução para os problemas supracitados.

### 3. Arquitetura Básica do BitTorrent

Esta seção revisa características importantes para o entendimento do protocolo BitTorrent. Apesar das mesmas serem fundamentais para a compreensão do Funnel, sua leitura é dispensável ao leitor familiar com o assunto. Dessa forma, considerando a importância do protocolo BitTorrent neste trabalho, serão tratados aspectos como formas de comunicação entre entidades do protocolo, criação da rede de sobreposição (*overlay*) e terminologia comumente utilizada.

Desde a sua criação em 2003, o BitTorrent vem se tornando o protocolo mais utilizado para compartilhar arquivos na Internet através de uma rede P2P. Ele propõe a divisão de arquivos em pedaços menores, denominados peças, para serem trocados entre os participantes do sistema (pares). O conteúdo é disponibilizado, por meio de um arquivo de metadados (*torrent*), em portais como Mininova.org, isoHunt.com e OneBigTorrent.org. Esses *torrents* contêm, entre outras informações, a lista com os nomes dos arquivos distribuídos, o código *hash* de cada peça para verificação de integridade e o endereço do serviço que viabiliza a comunicação entre os pares (rastreador).

O rastreador é a única entidade centralizadora do sistema e atua como um serviço de descoberta, fazendo com que pares interessados em recuperar o mesmo conteúdo se encontrem e troquem peças entre si. O conjunto de pares que compõem a rede P2P de compartilhamento é conhecido como enxame (*swarm*) ou, em alguns casos, *torrent*. Para participar de um enxame, um usuário precisa buscar o *torrent* referente ao conteúdo desejado e utilizar um agente BitTorrent (por exemplo, Vuze, BitTornado e  $\mu$ Torrent) para iniciar o *download*.

Uma vez iniciado, o agente do usuário contacta o rastreador discriminado no arquivo *torrent*, informa alguns dados relevantes para o rastreador (endereço para que outros pares o contactem, progresso do seu *download* e *upload*, entre outros) e obtém uma lista contendo a localização de outros pares do enxame. Esse passo é repetido periodicamente para obtenção de listas atualizadas com novos pares junto ao rastreador. Em seguida, o agente estabelece uma conexão com os pares da lista e eles trocam seus mapas de peças (*bitfield*) entre si. O mapa descreve quais peças cada um deles possui e é utilizado para iniciar o processo de requisição por peças entre os pares. Além disso, ele classifica os pares de duas formas diferentes em enxames BitTorrent: um par que completou todas as peças do arquivo é classificado como “semeador” (*seeder*), enquanto que um outro que ainda não as recuperou completamente é classificado como “sugador” (*leecher*). No entanto, é importante observar que à medida que os sugadores completam as peças de um arquivo, eles passam a enviá-las para os demais sugadores, agindo também como semeadores.

## 4. Funnel

Esta seção apresenta Funnel, um mecanismo para contenção de poluição em comunidades BitTorrent. Primeiramente será feita uma descrição do funcionamento desse mecanismo e, em seguida, será abordado como o mesmo foi instanciado em um rastreador BitTorrent. O Funnel é baseado em uma proposta anterior [da Silva et al. 2007] que atua no controle da disseminação de cópias nos instantes em que estas ainda não foram suficientemente avaliadas pelos usuários (escassez de votos). Essa estratégia foi preferível às demais, que exigem diversas avaliações negativas de usuários até que uma cópia seja classificada como poluída, para evitar que novas cópias sejam inseridas e rapidamente disseminadas na comunidade. Com o objetivo de simplificar o entendimento e não perdendo a generalidade, o mecanismo será descrito sob o contexto da distribuição de uma única versão de um conteúdo. Isso é possível devido ao fato de que a informação agregada a cada versão e o controle de sua disseminação são feitos de forma individual e isolada.

O Funnel utiliza um mecanismo de votação binária para construir a reputação de um objeto na comunidade ( $E[\omega]$ ). Assim que é finalizada a recuperação completa de uma cópia, o usuário emite um voto positivo para classificar a versão como autêntica ou negativo para classificá-la como poluída. Esse cálculo, fundamentado pela Lógica Subjetiva [Josang et al. 2006], representa a confiança que uma versão é autêntica, assumindo valores entre 0 e 1 (mais próximo de 1 quanto maior a confiança). O valor de  $E[\omega]$  representa a fração dos votos positivos em relação ao total de votos atribuídos a uma versão e é calculado conforme Equação 1 a seguir.

$$E[\omega] = \frac{r + 2a}{r + s + 2} \quad (1)$$

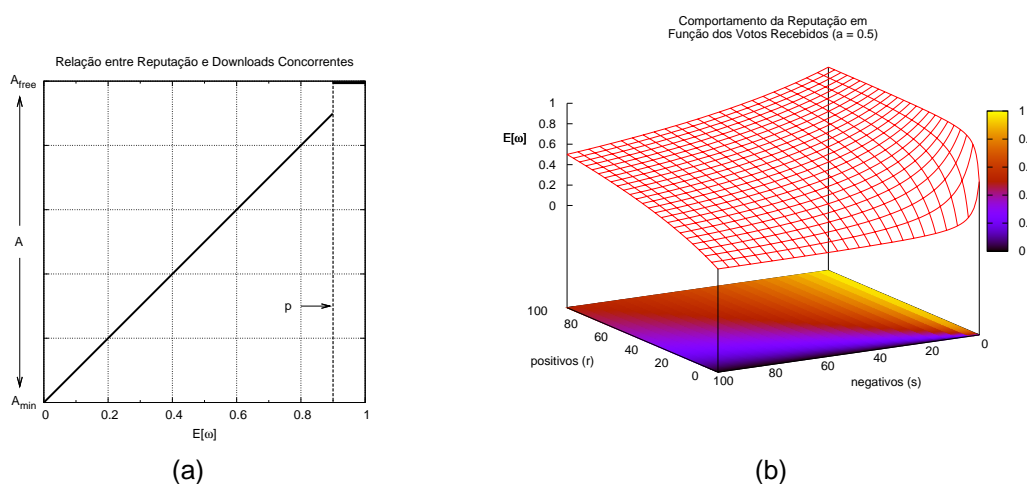
Os valores de  $r$  e  $s$  na equação representam, respectivamente, o número de votos positivos e negativos emitidos pelos usuários. Além do número de votos, o cálculo da reputação de uma versão utiliza uma constante  $a$  para definir o comportamento do mecanismo na presença de poucos votos: se  $r, s \rightarrow 0$ , então  $E[\omega] \rightarrow a$ . Neste ponto, fica evidente a necessidade de um mecanismo que incentive a votação, alimentando o mecanismo, e que previna a duplicidade dos votos. Essas questões serão devidamente abordadas apenas nas Seções 4.4 e 4.5, respectivamente, e por enquanto são dispensáveis ao entendimento do mecanismo.

O cálculo da reputação de uma versão é utilizado para determinar o número de *downloads* concorrentes autorizados pelo mecanismo em um dado instante. Nesse contexto, o Funnel emprega duas variáveis:  $A$  e  $D$ , definidas a seguir.  $A$  representa o número máximo de *downloads* concorrentes autorizados pelo mecanismo, enquanto que  $D$  representa o número de *downloads* atualmente ocorrendo. O valor de  $A$  é calculado de acordo com a Equação 2. O valor de  $D$  é incrementado quando um novo *download* inicia e decrementado quando ele termina.

$$A = E[\omega] \times (A_{free} - A_{min}) + A_{min} \quad (2)$$

A equação acima (2) garante um número mínimo de cópias autorizadas concorrentemente,  $A_{min}$ , sendo alcançado quando  $E[\omega] = 0$ . De forma análoga, o número máximo,  $A_{free}$ , é alcançado quando  $E[\omega] = 1$ . Esse valor deveria ser atingido para que o Funnel

assumisse a cópia como autêntica e autorizasse novos *downloads* sem o controle do mecanismo. Entretanto, desenvolvendo a Equação 1, obtém-se que  $E[\omega] = 1 \leftrightarrow 2a = s + 2$ . A única solução dessa equação, dado que  $a \in [0; 1]$  e  $s \in \mathbb{N}$ , é  $a = 1$  e  $s = 0$  (ausência de votos negativos). Para lidar com esse comportamento, foi definido um valor  $p \in [0; 1]$  que, ao ser alcançado pela reputação  $E[\omega]$ , faz com que o mecanismo assumira a autenticidade da cópia. A Figura 1(a) ilustra o comportamento da variável  $A$  em função da variação da reputação  $E[\omega]$ . Quando  $E[\omega] \geq p = 0,9$ , por exemplo, o Funnel autoriza todos os novos pedidos de *download*, até que a reputação volte novamente a ser menor que 0,9. A Figura 1(b) demonstra como a reputação de uma versão varia, de acordo com a Equação 1, em função dos votos atribuídos a ela.



**Figura 1. No instante em que a reputação da versão ultrapassa o valor de  $p$ , esta é considerada autêntica (a), o que ocorre para uma certa faixa de votos (b).**

Para autorizar ou banir o início de um *download*, é feita uma comparação entre as variáveis  $A$  e  $D$ , além dos valores de  $E[\omega]$  e  $p$ . Caso  $D < A$  ou  $E[\omega] \geq p$ , o novo pedido de *download* é aceito e tem autorização para prosseguir. Caso contrário, o pedido é negado e o usuário aguarda um instante até retornar com um novo pedido. Nas subseções seguintes, será descrito como o Funnel foi projetado e instanciado em comunidades BitTorrent de compartilhamento de arquivos. Para isso, será apresentado, na Subseção 4.1, o cenário que relaciona essas comunidades e o rastreador BitTorrent. O método aplicado para estimar o número atual de *downloads* concorrentes ( $D$ ) em um enxame BitTorrent é descrito na Subseção 4.2. A técnica utilizada pelo rastreador para controlar o número de *downloads* concorrentes no enxame é apresentada na Subseção 4.3. Na Subseção 4.4 será discutida a estratégia para permitir apenas um único voto por usuário da comunidade. A apresentação do mecanismo é finalizada na Subseção 4.5, descrevendo o mecanismo de incentivo à votação empregado pelo Funnel.

#### 4.1. Relação entre as comunidades BitTorrent e o rastreador

Em sua essência, uma comunidade BitTorrent é composta por uma entidade centralizadora (portal) que agrega e distribui arquivos *torrent*. Esses portais empregam um conjunto de regras para definir as operações permitidas aos seus usuários. Por exemplo, o portal pode exigir que usuários sejam cadastrados para publicar ou recuperar *torrents*. Comunidades

que utilizam essas regras são denominadas comunidades fechadas. Adicionalmente ao serviço de gerenciamento de identidades, é possível que os portais disponibilizem também um rastreador BitTorrent. Caso o portal não disponha desse serviço, ele pode apenas armazenar *torrents* que referenciam outros rastreadores. Nesse contexto, fica evidente que o rastreador é a entidade-chave responsável por viabilizar o compartilhamento de arquivos em comunidades BitTorrent. Este trabalho é empregado no âmbito de comunidades fechadas e parte do pressuposto que os portais dispõem de barreiras contra ataques de múltiplas identidades (*sybil*) [Douceur 2002]. Entretanto, considerações sobre ataques dessa natureza são revisadas oportunamente no decorrer do trabalho.

Para facilitar o entendimento, o termo “usuário” é utilizado para referenciar o operador do agente BitTorrent que interage e emite votos junto a um portal. Além disso, o rastreador é capaz de identificar as interações com os pares através de um procedimento de autenticação prévio; ou seja, pedidos por listas de pares e emissões de votos são devidamente autorizadas mediante autenticação. Essa estratégia é amplamente difundida em comunidades fechadas e é necessária para que o mecanismo aqui proposto seja capaz de estimar e controlar o número de *downloads* concorrentes, identificar votos duplicados e incentivar a votação. Esses aspectos são discutidos a seguir.

#### 4.2. Estimando o número de *downloads* concorrentes

Para aplicar o mecanismo de controle de disseminação, o rastreador precisa calcular o número de pares recuperando um determinado *torrent* simultaneamente ( $D$ ). Nesse contexto, cada *torrent* pode ser interpretado como uma versão de um conteúdo publicado. Durante o *download* de um *torrent*, os pares agem de forma autônoma e sem a intervenção do rastreador, dificultando o cálculo preciso do valor de  $D$ . Por conta disso, ao adaptar o modelo para o BitTorrent, o valor de  $D$  foi aproximado pelo número de sugadores no enxame. Para realizar esse cálculo, o rastreador mantém uma tabela com registros associados a pares do enxame. A cada pedido por listas de pares, o par requisitante tem seu registro atualizado. Um registro da tabela armazena o progresso do *download* de um par e é removido após algum tempo de inatividade. A estimativa pra o valor de  $D$  é, então, definida como a quantidade de registros com o *download* inacabado na tabela.

Essa estratégia torna-se vulnerável a ataques em que pares maliciosos forjam a informação de progresso de *download* que enviam ao rastreador. Dessa forma, um grupo de pares maliciosos pode fazer com que o rastreador superestime o valor de  $D$ . Entretanto, fazendo uma análise de uma das condições suficientes para autorizar um novo *download* ( $D < A$ ), é possível observar que o aumento do valor de  $D$  poderia contrariar essa condição e causar uma negação do pedido. Esse efeito mostra-se interessante apenas no cenário em que atacantes têm por objetivo provocar uma negação de serviço durante a distribuição de uma versão autêntica. É importante ressaltar que o custo do ataque torna-se consideravelmente alto haja vista que os pares maliciosos precisam se manter no enxame para serem considerados pelo rastreador.

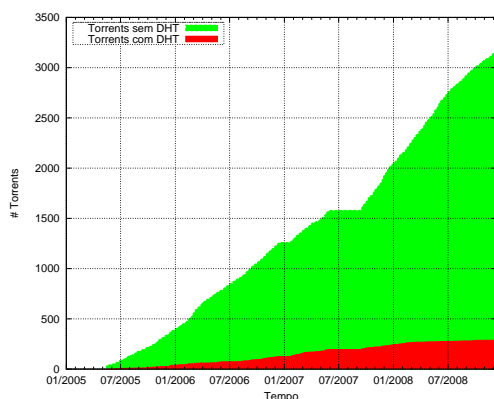
O rastreador poderia utilizar uma técnica probabilística para diminuir os efeitos causado por esse ataque. Por exemplo, mesmo que as condições apontassem para a negação de novos pedidos de *download*, autorizações seriam concedidas seguindo uma determinada probabilidade. Dessa forma, pares honestos teriam a chance de progredir, livrando o mecanismo do controle dos atacantes. Por outro lado, essa técnica pode não ser

muito eficaz na presença de pares maliciosos trabalhando em conluio, pois eles poderiam monopolizar a seleção aleatória do rastreador. A eficiência desse ataque está condicionada à proporção de pares maliciosos aguardando autorização para iniciar o *download*, sendo comprometida à medida que os atacantes “disputam” uma oportunidade com maiores quantidades de pares honestos.

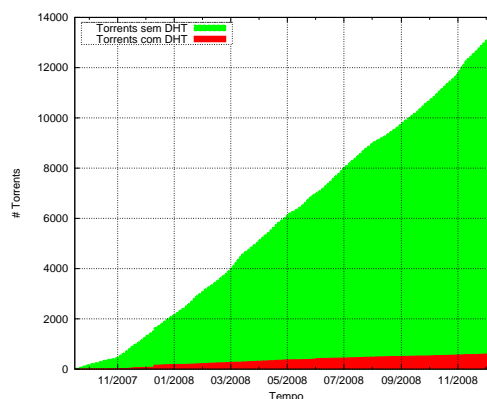
### 4.3. Controlando o número de *downloads* concorrentes

Para controlar a disseminação das cópias na comunidade, o rastreador dispõe de apenas um recurso: a lista de pares retornada aos agentes BitTorrent dos usuários. A idéia proposta pelo Funnel consiste em ajustar o tamanho dessa lista dependendo se um novo pedido de *download* deve ou não prosseguir. Em caso positivo, o rastreador age normalmente e retorna uma lista para o par. Em caso negativo, uma lista vazia é retornada. A partir do momento que um par  $P$  recebe autorização do rastreador para prosseguir, diz-se que uma sessão foi iniciada. Nessa situação,  $P$  continua recebendo listas, ainda que o valor de  $A$  seja decrementado e a condição  $D \geq A$  passe a ser verdadeira. Entretanto, caso a sessão de  $P$  expire, o pedido de  $P$  é reavaliado pelo rastreador.

Deve-se considerar a situação em que um par  $P$  tem seu pedido ao rastreador negado e utiliza algum outro recurso para participar do *torrent*. Dois mecanismos que estendem o protocolo original são empregados na descoberta de pares em enxames: *PeerExchange* (PEX) e tabelas *hash* distribuídas (DHTs). O primeiro deles não afeta o Funnel, pois PEX é um protocolo baseado em *gossip* e exige que pares se conheçam para descobrirem outros pares, fato este que não ocorre pois  $P$  está ingressando no enxame e ainda é totalmente desconhecido. Em contraste, a utilização de DHTs viabiliza a descoberta de pares sem a presença de um rastreador. Com base nisso, foi feito um estudo para medir a popularidade das DHTs em comunidades BitTorrent e verificar o potencial impacto desse recurso no Funnel. Para tal, milhares de arquivos *torrent* foram coletados de quatro comunidades (OneBigTorrent.org, etree.org, PirateBay.org e isoHunt.com) e analisados. A Figura 2 ilustra um resumo dessa análise. A proporção dos *torrents* obtidos da comunidade etree.org, desde a sua criação, com suporte a DHT atingiu apenas 0,12% até o instante da coleta e seu gráfico foi omitido. O comportamento obtido na análise da comunidade PirateBay.org foi semelhante ao do isoHunt.com e também foi removido.



(a) OneBigTorrent.org



(b) isoHunt.com

Figura 2. Evolução da popularidade do uso de DHTs em comunidades BitTorrent

Observando os gráficos, fica evidente o lento crescimento dos *torrents* com suporte a DHT em relação ao total de *torrents* adicionado nas comunidades. Em síntese, essa pequena fração corresponde aos conteúdos que podem ser recuperados pelos pares do enxame sem a existência de um rastreador e, conseqüentemente, podem comprometer a eficácia do Funnel. Para esses casos, se for desejável um controle mais rígido na comunidade, uma política para impedir a distribuição desses *torrents* pode ser criada.

#### 4.4. Mantendo votos únicos por usuários

Dado que o Funnel é uma solução para controle de poluição em comunidades fechadas de compartilhamento, cada interação de um par com o rastreador é associada a sua identidade no portal. Nesse cenário, o portal agrega os votos de cada usuário a cada *torrent* publicado, armazenando tuplas no formato  $(Par, Torrent, Voto)$  para evitar elementos  $(Par, Torrent)$  duplicados. O portal também utiliza essa informação para contabilizar o número de votos positivos e negativos atribuídos a cada *torrent*. É importante notar que um usuário pode votar sem ter sequer recuperado completamente o conteúdo. Isso é necessário para não obrigar um usuário a recuperar completamente uma cópia, mesmo tendo sido capaz de julgá-la poluída apenas fazendo uma avaliação parcial da mesma.

Um conjunto de atacantes pode tirar proveito dessa característica para realizar um ataque em conluio e tentar manipular a reputação de um *torrent*, votando negativamente em *torrents* autênticos e positivamente em poluídos. Para diminuir esse impacto, o portal pode manter um registro indicando em quais *torrents* cada usuário participou, mesmo que temporariamente, e aceitar votos apenas desses usuários. Além disso, recursos como desafios computacionais (ex.: CAPTCHAs) podem ser utilizados para conter esse tipo de ataque [Awerbuch and Scheideler 2004].

#### 4.5. Incentivando os usuários a votar

O mecanismo aqui proposto utiliza os votos emitidos pelos usuários para construir a reputação dos *torrents* na comunidade. Caso esses usuários recuperem uma versão e não emitam seus votos, o mecanismo perde eficácia. Para resolver essa questão, é necessário que um mecanismo de incentivo à votação seja incorporado ao Funnel. Duas abordagens são possíveis: recompensar os usuários que votam ou punir os que não votam. Recompensar os usuários que votam pode incentivar a emissão de votos indiscriminados na comunidade com o objetivo de obter benefícios do rastreador. Por outro lado, punir os que não votam não é um mecanismo perfeito, pois, mesmo com o custo associado à criação de uma nova identidade, ainda é possível que usuários providenciem uma nova identidade. Esse comportamento é denominado *whitewashing*.

Esta proposta utiliza a segunda abordagem: o rastreador aplica uma política para “punir” os usuários que não votam. A estratégia consiste em enviar listas de pares com tamanhos proporcionais à fração dos *torrents* em que o usuário participou e votou. Assim, sendo  $N$  o tamanho original da lista de pares que seria retornada pelo rastreador,  $V$  o número de *torrents* em que o usuário votou e  $R$  o número de *torrents* em que ele participou, a lista retornada a um par requisitante tem seu tamanho definido por  $N \times \min\{\frac{V+1}{R}, 1\}$ . Dado que um usuário não vota em um *torrent* sem ter participado do mesmo e que o voto pode ser emitido antes do fim do *download*,  $V \leq R$ . O operador *min* garante que o tamanho da lista retornada não ultrapasse  $N$ , mesmo se algum voto for emitido antes da recuperação completa de uma cópia ( $V = R$ ).

## 5. Avaliação

Esta seção apresenta uma avaliação experimental do Funnel. O objetivo dessa avaliação é responder a três questões fundamentais. Primeiro, qual o impacto negativo causado pelo mecanismo em um cenário em que é distribuído um conteúdo autêntico. Segundo, qual a eficiência do mecanismo na contenção da distribuição de conteúdos poluídos. Terceiro, como o mecanismo se comporta na presença de ataques em conluio. Para simplificar a explicação seguinte, cada cenário foi composto pela distribuição de um único conteúdo. Essa é uma abordagem válida, pois o mecanismo é aplicado individualmente por versão de conteúdo disponível, não sofrendo influência na distribuição de múltiplas versões ou conteúdos. Além disso, uma avaliação da distribuição de múltiplas versões não agregaria resultados práticos significativos ao estudo apresentado. A seguir são apresentados os cenários utilizados e a notação empregada para a descrição dos mesmos.

### 5.1. Descrição dos cenários

Os cenários avaliados consistem em  $C$  pares honestos,  $M$  maliciosos e  $I$  semeadores que iniciam um enxame e permanecem disponíveis durante todos os experimentos. Dois cenários foram avaliados: (A)  $I$  pares honestos semeando um conteúdo autêntico e (B)  $I$  pares maliciosos semeando um conteúdo poluído. Neste estudo, os valores de  $I$  e  $C$  foram definidos como 20 e 500, respectivamente, variando o valor de  $M$  entre 0 e 150 para analisar o efeito do ataque em conluio ao Funnel. Após a criação do *torrent* com os  $I$  semeadores iniciais, os  $M$  pares maliciosos entram no enxame. Esse comportamento foi adotado para avaliar o pior caso, no qual esses pares chegam em conluio antes dos pares honestos. Em contrapartida, os pares honestos chegam seguindo uma distribuição exponencial, conforme medições apresentadas em um trabalho anterior [Guo et al. 2005]. Nos cenários avaliados, os pares abandonam o enxame apenas após o fim do *download* guiados por uma métrica de contribuição ( $\rho$ ). Em síntese,  $\rho$  representa o número de vezes que um par fará *upload*, após recuperar completamente a cópia, em relação à quantidade de *download* que ele fez. Por exemplo,  $\rho = 0$  significa que um par abandonará o enxame assim que finalizar o *download* e  $\rho = 2$  significa que ele fará *upload* de duas vezes a quantidade de conteúdo recuperado.

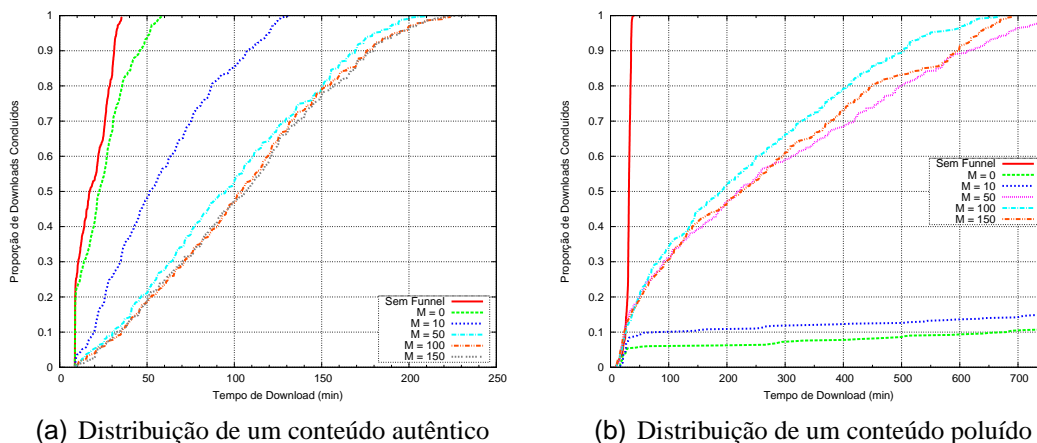
Nesse contexto, definiu-se o comportamento dos pares honestos e maliciosos ao recuperarem uma cópia autêntica e uma poluída, resultando em quatro situações possíveis. Ao recuperar uma cópia poluída, um par honesto deixa o sistema imediatamente ( $\rho = 0$ ), pois ele detecta a poluição. O mesmo ocorre quando um par malicioso recupera uma cópia autêntica, pois ele não deseja semear essa cópia. No entanto, quando um par malicioso recupera uma cópia poluída, ele permanece no enxame contribuindo indeterminadamente com a disseminação ( $\rho = \infty$ ). A situação mais elaborada ocorre quando um par honesto recupera uma cópia autêntica. Seguindo uma metodologia presente em estudo anterior [Anagnostakis et al. 2006], após recuperar completamente a cópia, 25% dos sugadores deixam o enxame sem se tornar semeadores ( $\rho = 0$ ), 41% saem após fazer a mesma quantidade de *upload* que fizeram de *download* ( $\rho = 1$ ) e 34% fazem duas vezes mais *upload* em relação ao que fizeram de *download* ( $\rho = 2$ ).

Foram utilizados *torrents* com tamanhos iguais a 60MB. Os agentes BitTorrent foram distribuídos em 10 máquinas dedicadas interconectadas através de um *switch* de rede e configurados com limite de sete conexões de *upload* e com conexões ilimitadas de

*download*. Além disso, tiveram suas capacidades de *upload* e *download* ajustadas para 256Kbps e 1Mbps, respectivamente. Por fim, as variáveis do mecanismo foram configuradas, baseando-se na experiência adquirida em um trabalho anterior [da Silva et al. 2007], com os valores:  $A_{min} = 1$ ,  $A_{free} = 50$ ,  $a = 0,5$  e  $p = 0,95$ .

## 5.2. Eficiência do mecanismo e resistência a ataques em conluio

O mecanismo proposto utiliza um esquema de reputação baseado em votação para classificar os objetos como poluídos ou autênticos. Tais esquemas estão sujeitos a ataques de falso testemunho, em que um usuário malicioso mente para manipular o valor de uma reputação. No contexto deste trabalho, um atacante vota positivamente para um conteúdo poluído e negativamente para um conteúdo autêntico. Com o objetivo de avaliar o comportamento do mecanismo frente a esse tipo de ataque, foi observado o tempo necessário para completar o *download* dos pares honestos. A Figura 3 ilustra a evolução dos *downloads* concluídos em função do tempo de *download* nos cenários A e B. Um ponto  $(x,y)$  no gráfico significa que uma fração  $y$  de *downloads* foi concluída consumindo até  $x$  minutos.



**Figura 3. Eficiência do mecanismo frente a ataques em conluio**

A Figura 3(a) ilustra a reação do mecanismo no cenário A. É possível notar a curva que representa a evolução dos *downloads* na ausência do mecanismo bastante próxima à curva sem atacantes ( $M = 0$ ), indicando que o atraso gerado pela ativação do mecanismo, nesse caso, é de no máximo 10 minutos para 80% dos pares. Essa comparação demonstra a sobrecarga inserida pelo Funnel em um ambiente em que seu uso seria desnecessário (ausência de atacantes). Ao executar um experimento com 10 atacantes, o mecanismo é alimentado com votos negativos (falsos), causando uma contenção na disseminação que provoca um atraso de até 70 minutos em relação ao caso sem atacantes. Para duplicar esse atraso, contudo, é necessário quintuplicar o número de atacantes ( $M = 50$ ). Além disso, há um ganho reduzido na eficiência do ataque ao utilizar 50 ou mais pares maliciosos. Tal fenômeno ocorre porque ao entrar no enxame em conluio, no máximo  $A = 0,5 \times (1 + 50) + 1 = 26,5$  dos  $M$  pares têm autorização para iniciar o *download* ( $E[\omega] = 0,5$  pois no instante inicial não há votos registrados, logo  $E[\omega] = a$ ); ou seja,  $M - A$  restantes devem “disputar” a oportunidade de iniciar o *download* com os demais pares honestos que se juntam ao enxame.

O cenário B, ilustrado na Figura 3(b), mostra que, na ausência do mecanismo de controle, 100% dos pares concluem o *download* do conteúdo poluído em menos de 50 minutos. O ganho é bastante expressivo ao ativar o mecanismo: aproximadamente 90% dos pares consomem mais que 750 minutos (12,5 horas) para recuperar completamente o conteúdo. O mecanismo também atua de forma eficiente, controlando a disseminação do conteúdo poluído, mesmo na presença de 10 atacantes votando positivamente. Nos demais ambientes ( $M \geq 50$ ), os atacantes obtêm sucesso em acelerar a disseminação total do conteúdo; no entanto, para os pares honestos obterem os 60MB são necessários aproximadamente 600 minutos (10 horas), 12 vezes mais que na ausência do Funnel. O fato de que os pares concluem seus *downloads* mais rapidamente no cenário com 100 atacantes (ao invés de 150, como é intuitivamente esperado), é explicado pela característica do mecanismo de conter os pedidos de *download* e autorizando-os aleatoriamente. Esse comportamento pode inserir mais pares honestos ou maliciosos no enxame, provocando uma leve variação nas curvas para os cenários com contenção. Além disso, é importante ressaltar que, nos experimentos, os pares não desistem de obter o conteúdo, fato este que causaria um crescimento menos acentuado das curvas.

### 5.3. Mecanismo de controle de *downloads*

Em uma análise mais detalhada dos cenários avaliados, é possível observar a atuação do mecanismo controlando a disseminação de poluição entre os pares do enxame. A Figura 4 apresenta a evolução do número de pares (honestos e maliciosos) nos enxames observados. O gráfico da esquerda (a) ilustra o ambiente de distribuição de um conteúdo autêntico sendo atacado por um conluio de 10 pares. Nos primeiros 10 minutos, o enxame é tomado pelos 10 atacantes, seguidos de  $A - 10 = 16$  pares honestos. Ao serem emitidos os votos negativos, o valor de  $A$ , calculado pelo rastreador e representado pela linha pontilhada no gráfico, é decrementado. O efeito do ataque faz com que o número de pares recuperando o conteúdo seja reduzido (pares atualmente no enxame concluem o *download* e novos pares são impedidos de entrar devido ao decréscimo de  $A$ ). No entanto, os votos positivos (honestos) causam um acréscimo do valor de  $A$ , que atinge  $A_{free}$  após aproximadamente 1 hora e 45 minutos de experimento. Tal efeito pode ser observado na Figura 4(a) com o fim da linha pontilhada e o aumento abrupto no número de pares honestos.

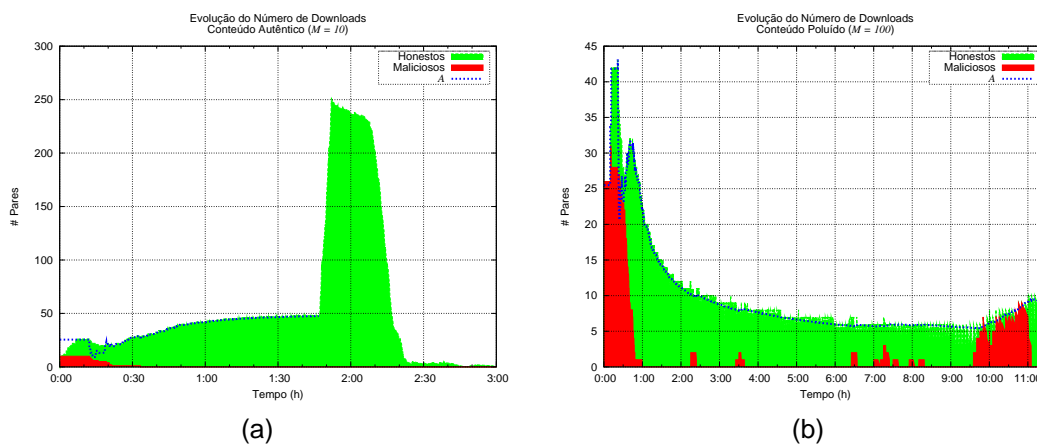


Figura 4. Evolução do número de *downloads* durante a distribuição de um conteúdo autêntico (a) e um poluído (b).

O gráfico da direita (b) ilustra o ambiente de distribuição de um conteúdo poluído sendo atacado por um conluio de 100 pares. No entanto,  $100 - A = 74$  atacantes são impedidos pelo Funnel de iniciar seus *downloads*. Observando a Figura 4(b), é possível notar o efeito causado pelos votos positivos dos atacantes: ocorre um acréscimo no valor de  $A$  durante os primeiros 30 minutos. À medida que novos *downloads* são autorizados, tanto pares honestos quanto pares maliciosos entram no enxame. Os votos honestos provocam um decréscimo no valor de  $A$ , amortizado por uma minoria de votos falsos de atacantes que tiveram oportunidade de iniciar seus *downloads*. Esse procedimento ocorre ao longo do experimento e permite concluir que o sistema se adapta rapidamente, a depender da proporção de atacantes, para uma situação estável em que versões autênticas são livremente distribuídas e as poluídas são disseminadas lentamente entre os pares.

## 6. Conclusões e Trabalhos Futuros

Comunidades BitTorrent têm sido amplamente empregadas para compartilhar conteúdo. Tais comunidades estão sujeitas a ataques de poluição, em que usuários maliciosos publicam cópias corrompidas, comprometendo a utilidade do sistema. Neste trabalho, propusemos um mecanismo, denominado Funnel, para mitigar esse problema através de um controle da disseminação das cópias entre os usuários baseado em reputação.

Ao que sabemos, nenhum outro trabalho na literatura abordou o problema da poluição em comunidades BitTorrent. O presente artigo representa a primeira proposta específica para o contexto em questão. Além disso, demonstramos a viabilidade e eficiência do mecanismo proposto através de uma avaliação experimental conduzida com uma implementação do Funnel. Os experimentos em rede controlada foram fundamentais para confirmar nossa investigação anterior, cujos resultados foram obtidos via simulação.

Na avaliação experimental, o mecanismo proposto mostrou-se eficiente, causando atrasos muito significativos na disseminação de cópias poluídas. Além disso, ao ser avaliado em um cenário de distribuição de cópias autênticas, o Funnel apresentou um atraso máximo, em decorrência do mecanismo de controle, de aproximadamente 200 minutos.

Como trabalhos futuros, pretendemos generalizar o modelo, removendo a premissa de que o Funnel precisa ser instanciado em comunidades fechadas. Para isso, uma série de desafios precisam ser vencidos, destacando entre eles a criação de mecanismos de incentivo à votação e controle de duplicidade de votos na ausência de autenticação.

## Agradecimentos

Os autores agradecem a Luis Otávio Soares pelo suporte no laboratório utilizado nos experimentos, a Weverton Cordeiro pelas discussões gerais sobre o trabalho e a Alan Mezzomo pela ajuda no entendimento das extensões DHT e PEX do protocolo BitTorrent.

## Referências

- Anagnostakis, K. G., Harmantzis, F. C., Ioannidis, S., and Zghaibeh, M. (2006). On the impact of practical p2p incentive mechanisms on user behavior. Technical report, NET Institute.
- Awerbuch, B. and Scheideler, C. (2004). Group spreading: A protocol for provably secure distributed name service. In *31st Annual International Colloquium on Automata, Languages and Programming (ICALP'04)*, pages 183–195. LNCS.

- Benevenuto, F., Costa, C., Vasconcelos, M., Almeida, V., Almeida, J., and Mowbray, M. (2006). Impact of peer incentives on the dissemination of polluted content. In *21st Annual ACM Symposium on Applied Computing (SAC'06)*, pages 1875–1879, New York, NY, USA. ACM.
- Costa, C. and Almeida, J. (2007). Reputation systems for fighting pollution in peer-to-peer file sharing systems. *7th IEEE International Conference on Peer-to-Peer Computing (P2P'07)*, pages 53–60.
- Costa, C., Soares, V., Almeida, J., and Almeida, V. (2007). Fighting pollution dissemination in peer-to-peer networks. In *ACM Symposium on Applied Computing (SAC'07)*, pages 1586–1590, New York, NY, USA. ACM.
- da Silva, J. F., Barcellos, M. P., Konrath, M. A., Gaspary, L. P., and Antunes, R. S. (2007). Métodos para contenção de poluição de conteúdo em redes p2p. In *25<sup>o</sup> Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2007)*, pages 855–868.
- Douceur, J. R. (2002). The sybil attack. In *1st International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, pages 251–260.
- Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X. (2005). Measurements, analysis, and modeling of bittorrent-like systems. In *5th Conference on Internet Measurement (IMC'05)*, pages 4–4, Berkeley, CA, USA. USENIX Association.
- Josang, A., Pope, S., and Hayward, R. (2006). Trust network analysis with subjective logic. In *29th Australasian Computer Science Conference (ACSC'06)*, pages 85–94, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.
- Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *12th International Conference on World Wide Web (WWW'03)*, pages 640–651, New York, NY, USA. ACM.
- Kumar, R., Yao, D. D., Bagchi, A., Ross, K. W., and Rubenstein, D. (2006). Fluid modeling of pollution proliferation in p2p networks. *ACM SIGMetrics Performance Evaluation Review*, 34(1):335–346.
- Liang, J., Kumar, R., Xi, Y., and Ross, K. W. (2005). Pollution in p2p file sharing systems. In *24th IEEE International Conference on Computer Communications (INFOCOM 2005)*, pages 1586–1590, Miami, Florida, USA.
- Liang, J., Naoumov, N., and Ross, K. W. (2006). The index poisoning attack in p2p file-sharing systems. In *25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–12, Barcelona, Catalunya, Spain.
- TorrentFreak (2009). Fake aXXo Torrents Bombard BitTorrent. <http://torrentfreak.com/fake-axxo-torrents-bombard-bittorrent-090313/>.
- Walsh, K. and Sirer, E. G. (2006). Experience with an object reputation system for peer-to-peer filesharing. In *3rd USENIX Symposium on Networked Systems Design & Implementation (NSDI'06)*, page 1, Berkeley, CA, USA. USENIX Association.