Corrupção, Mentiras e Isolamento: avaliação de impacto de ataques a BitTorrent

Rodrigo B. Mansilha, Marlom A. Konrath, Marinho P. Barcellos

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

rmansilha@gmail.com, marlomk@unisinos.br, marinho@acm.org

Resumo. *BitTorrent se tornou uma das aplicações mais populares da Internet, dado o número de usuários e a fração do tráfego de Internet que ela consome. Sua ampla adoção tem exposto potenciais problemas, como comportamento egoísta de pares e vulnerabilidades de segurança. Esforços de pesquisa relacionados a isso têm se focado particularmente na dinâmica de enxames e em mecanismos de incentivo para melhorar a justiça sem sacrificar a eficiência. Em um trabalho recente, investigamos vulnerabilidades de segurança em BitTorrent. Este artigo expande o referido trabalho em diversos aspectos. Particularmente, há melhorias na modelagem de enxames, tornando resultados mais realistas, e um novo ataque é definido e avaliado:* corrupção de peças*. Nossa análise demonstra a gravidade do problema e a necessidade de mecanismos de contra-medida para tais ataques.*

Abstract. *BitTorrent has become one of the most popular Internet applications, given the number of users and the fraction of the Internet traffic it consumes. Its wide adoption has exposed potential problems, like selfish peer behavior and security vulnerabilities. Related research efforts so far have focused on modeling the dynamics of swarms and on incentive mechanisms to improve fairness without sacrificing efficiency. In a previous paper, we investigated security vulnerabilities in BitTorrent. The present paper expands the aforementioned one in several aspects. In particular, it improves the modelling of swarms, leading to more realistic results, and defines and evaluates a new attack,* piece corruption*. Our analysis demonstrates the seriousness of such attacks and shows the need for security countermeasure mechanisms.*

1. Introdução

Compartilhamento de arquivos em redes P2P tornou-se uma das mais relevantes aplicações da Internet, permitindo uma rápida disseminação de conteúdo entre usuários. Nesta categoria, BitTorrent [Bit 2007b] é um dos protocolos mais populares, sendo utilizado por milhões de pessoas para compartilhar conteúdo tanto legal como protegido por *copyright* [Das et al. 2006]. O protocolo já consome fração substancial do tráfego da Internet [Barbera et al. 2005]. Sistemas de distribuição segura de conteúdo proprietário baseados em BitTorrent estão começando a entrar em operação (ex: Azureus Vuze [Azu 2007]). Grandes empresas como Warner Brothers, 20^{th} Century Fox e BBC já estão oferecendo conteúdo através desses sistemas.

Contudo, com o crescimento da popularidade do BitTorrent, cresce também o risco e o impacto de um ataque malicioso que explore suas vulnerabilidades potenciais. Em [Konrath et al. 2007], analisamos a questão de segurança em BitTorrent, mostrando os ataques de Mentira de Peças e de Isolamento (Eclipse). O presente trabalho expande o anterior, melhorando-o em diversos aspectos importantes, incluindo a modelagem do enxame e os ataques avaliados. Entender as vulnerabilidades existentes no protocolo e seus impactos permite que contramedidas ou modificações na arquitetura sejam propostas de maneira a aperfeiçoar a mesma.

O restante do artigo está estruturado como segue. A Seção 2 aborda os componentes e o funcionamento da arquitetura BitTorrent. Na Seção 3, são identificados ataques visando prejudicar o espalhamento de conteúdo digital. A Seção 4 descreve a simulação do BitTorrent, sendo a mesma usada para avaliar o impacto de ataques, conforme apresentado na Seção 5. A Seção 6 discute os trabalhos relacionados, enquanto comentários finais e perspectivas de trabalhos futuros são indicados na Seção 7.

2. Arquitetura BitTorrent

O tratamento de aspectos de segurança do BitTorrent pressupõe um entendimento sobre o mesmo. Devido à limitação de espaço, não é possível descrever o protocolo em detalhes. A literatura é farta em referências (como [Bit 2007a, Konrath et al. 2007, Liogkas et al. 2006]). A seguir, são descritos resumidamente os principais elementos da arquitetura BitTorrent.

O "conteúdo" (um conjunto de arquivos) a ser disseminado em uma rede BitTorrent é descrito através de um arquivo de metadados e de terminação ".torrent". Tal arquivo é gerado pelo usuário que publica o conteúdo, e contém informações como o nome e tamanho dos arquivos, *hashes* para os dados, e um ou mais endereços IP de "rastreadores" (*trackers*). Um rastreador é um elemento centralizado que coordena um "enxame" (*swarm*) e auxilia pares a encontrar outros pares no mesmo enxame. Um enxame é um conjunto de pares que compartilham o mesmo arquivo .torrent. Pares podem atuar como "semeadores" (*seeders*) ou "sugadores" (*leechers*), dependendo do download estar completo ou não, respectivamente. O conteúdo publicado e descrito no .torrent é organizado em "peças", e peças são subdivididas em "blocos". Blocos são a unidade mínima de troca de dados entre pares. Pares estabelecem conexões TCP entre si e trocam mapas de bits (*bitfields*) contendo informações sobre disponibilidade de peças. Um par solicita a um ou mais pares o download de blocos de uma determinada peça em que está interessado. Solicitações são atendidas através de um mecanismo de incentivo baseado em reciprocidade, conforme explicado a seguir.

O mecanismo de incentivo (denominado *tit-for-tat*) é utilizado para estimular a cooperação entre os pares. A cada rodada, tipicamente com 10 segundos de duração, o par avalia quais os pares remotos que mais contribuíram com ele e retribui na rodada seguinte, atendendo suas requisições. Os pares remotos para os quais um par local está realizando upload são marcados como "não sufocado" (*unchoked*), enquanto os demais são marcados como "sufocado" (*choked*). O cliente sempre sabe quais pares estão sufocando-o e quais não, pois cada vez que um par local muda o estado de um par remoto, uma mensagem é enviada a este último informando-o. Tipicamente, três pares são marcados como não sufocados em função de sua contribuição, e um quarto par, escolhido de forma aleatória entre os conectados, é eleito para receber conteúdo. A escolha de um par desconsiderando sua contribuição é chamada de *optimistic unchoke* e sua função é encontrar pares com melhores taxas de download e upload que as atuais, e permitir que pares iniciantes (sem peças) possam ter a chance de inicialmente obter conteúdo para troca.

Na ausência de uma descrição formal do protocolo, foram criados diagramas de estado e de tempo, através da análise da descrição do protocolo [Bit 2007a] e de traços de implementações populares do protocolo capturadas em ambiente controlado. Diagramas de estado foram apresentados em [Konrath et al. 2007]. No presente artigo, o comportamento do protocolo é comentado através de um diagrama de tempo. A Figura 1 apresenta o diagrama com um enxame simplificado. Para ser representável, é mostrada apenas a troca de mensagens entre um rastreador e três pares, denominados Semeador, Par1 e Par2.

Inicialmente, o Semeador, que possui uma cópia completa do conteúdo, realiza uma conexão TCP (não ilustrada) com o rastreador e envia uma mensagem Get (através do protocolo HTTP). O rastreador responde com uma lista vazia.

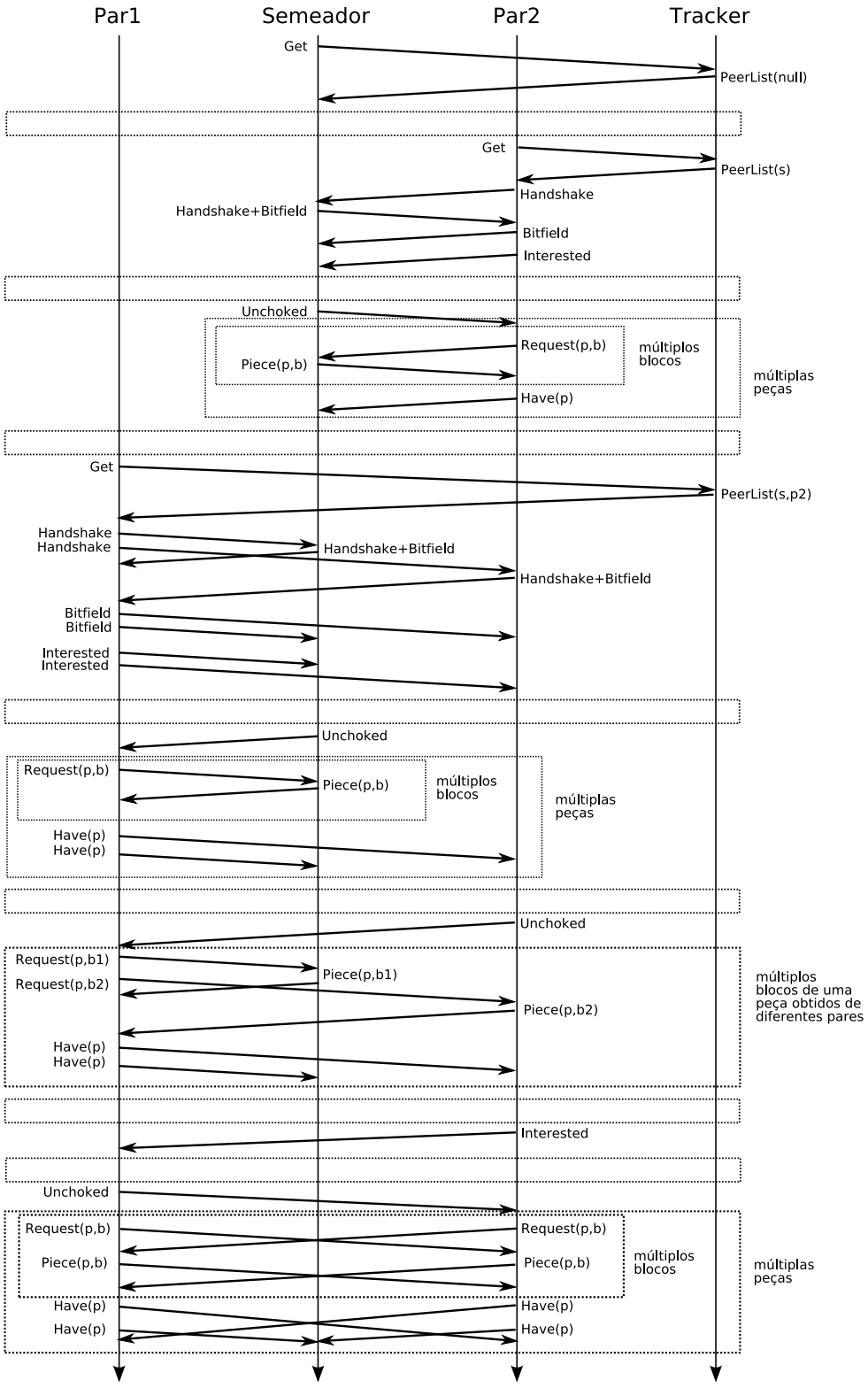


Figura 1. Diagrama de tempo ilustrando o comportamento de um enxame

Após algum tempo, simbolizado pelo retângulo pontilhado, o Par2 estabelece uma conexão com o rastreador, envia uma mensagem Get e obtém o endereço do Semeador, pois este está participando do enxame. O Par2 estabelece uma conexão com o Semeador e envia uma mensagem Handshake, recebe o mapa do par (mensagem Handshake + Bitfield) e envia o seu (mensagem BitField). Como o Semeador possui peças que Par2 não possui, uma mensagem Interested é enviada para o Semeador. Quando receber uma mensagem Unchoked, o par estará liberado para obter conteúdo a partir do Semeador. Ele então passará a solicitar blocos de peças através de mensagens Request(p,b). O Semeador enviará os blocos solicitados através de mensagens Piece(p,b). Múltiplos blocos são solicitados em paralelo. Ao receber todos os blocos de uma peça, há uma verificação de seu hash, a fim de garantir a integridade dos dados recebidos. Caso íntegra a peça, uma mensagem Have é enviada ao Semeador, para que este atualize a informação do mapa referente ao Par2.

Posteriormente, o Par1 entra no enxame, conecta com o rastreador, recebe a lista de pares, conecta com Semeador e Par2, e começa a trocar peças com eles. No diagrama, primeiramente Par1 recebe conteúdo apenas do Semeador, mas note-se que ao receber uma mensagem Unchoked de Par2, e não estando sufocado pelo Semeador, Par1 solicitaria blocos diferentes de uma mesma peça para ambos, realizando assim um download em paralelo. Por fim, ao obter novas peças do Semeador e anunciar a posse delas, em algum momento o Par2 fica interessado no conteúdo de Par1 e envia uma mensagem Interested. Quando receber uma mensagem Unchoked de Par1, Par2 passa a solicitar blocos de peças para Par1. Neste momento, os dois pares estarão trocando conteúdo ente si, solicitando blocos de peças que não possuem e evoluindo na obtenção de conteúdo. Note-se que na última parte apenas a troca entre Par1 e Par2 são ilustradas, mas ambos poderiam estar realizando downloads também do Semeador.

2.1. Política de Seleção de Peças

O sucesso de um par no seu download, bem como do enxame como um todo, é influenciado pela política de seleção de peças aplicada pelo par. A seleção no BitTorrent segue a política da "Peça Local Mais Rara Primeiro" (*Local Rarest First* - LRF), na qual o par escolhe primeiro as peças que estão menos replicadas entre pares remotos com os quais o par está conectado; caso exista mais de uma peça com grau mínimo de replicação, a escolha é aleatória. Esta política tem por objetivo manter o equilíbrio entre a quantidade de peças disponíveis no enxame, minimizando o risco de alguma peça tornar-se indisponível enquanto existam outras replicadas.

Existem três exceções para essa regra: quando o par está iniciando o download (situação em que possui menos de 4 peças), finalizando-o (já requisitou todos os blocos que faltam) ou atuando como semeador (possui todas as peças).

2.2. Política para Conexões com outros Pares

Pares obtém os endereços IP de outros pares em um enxame através do rastreador. Pares que participam de um enxame periodicamente se conectam ao rastreador e registram assim sua presença no enxame. Na conexão do par com o rastreador, é informado o número de pares que o par local deseja receber (numwant), que é por padrão, 50.

Além da lista de pares, o rastreador retorna uma série de informações, incluindo o campo interval: o tempo em segundos que o par deve esperar antes de realizar outro anúncio ao rastreador. Esse valor também é utilizado para cálculo do tempo que o rastreador manterá o IP do cliente na lista de pares ativos. Se o tempo exceder este limite mais certa margem, o rastreador considera que o par abandonou o enxame sem informá-lo. O tamanho da lista de pares mantida por um rastreador é o tamanho de um enxame: varia tipicamente de dois ou três pares até centenas de milhares. Quando solicitado, o rastreador monta aleatoriamente uma lista com até 50 pares (ou o valor solicitado em numwant) e envia-a ao par solicitante.

Há diversas situações em que um par pode obter a identidade de mais pares. Primeiro, quando um par entra no enxame, ele envia uma mensagem Get ao rastreador, que responde com o PeerList. Segundo, quando a lista decresce abaixo de 20 pares (limite inferior), o par envia novo Get ao rastreador. Terceiro, quando o par é contatado por outro par remoto no enxame, o qual não constava na lista do par local. Existem também extensões ao protocolo que permitem aos pares obterem identidades a partir de outros pares sem envolver o rastreador. O número máximo de conexões que um par pode manter é normalmente limitada a 55. Tipicamente, o par inicia até 30 conexões com outros pares, sendo as restantes deixadas para conexões iniciadas por pares remotos.

2.3. Detecção de peças corrompidas

A arquitetura BitTorrent possui proteção contra tentativas de adulteração do conteúdo. Uma peça que contém um ou mais blocos incorretos não será aceita pelo par realizando o download, levando ao descarte e novo download.

A proteção contra modificação se dá através de uma verificação de integridade, realizada calculando-se o hash SHA-1 da peça e comparando-o com aquele presente no arquivo .torrent. Uma das propriedades da função SHA-1, assim como das funções hash em geral, é ser uma função unidirecional. Isso significa dizer que é relativamente fácil a partir de um conteúdo calcular o seu hash, mas pragmaticamente inviável determinar, a partir do hash, qual o conteúdo que o produziu. Isso se deve ao fato de um conteúdo ser cifrado em blocos, num processo conhecido como *feedforward*, e no qual a modificação de um único bit no conteúdo provoca um "efeito avalanche" em que há uma inversão média de aproximadamente 50% dos bits resultantes da função [Wang 2006].

3. Estratégias de Subversão/Ataque

Esta seção trata de ataques a enxames BitTorrent. Primeiro discute-se como o ataque *Sybil* em BitTorrent pode servir de base para gerar dois ataques a enxames: de Mentira de Peças (Subseção 3.1) e de Isolamento de Pares (Subseção 3.2). Após, é apresentado o ataque de *corrupção de peças*, em que um par malicioso ardilosamente envia um ou mais blocos corrompidos no sentido de comprometer a integridade de uma peça recebida por um par correto, levando a seu descarte (Subseção 3.3).

O ataque Sybil [Douceur 2002] em redes P2P consiste em um mesmo par apresentar-se com múltiplas identidades virtuais. Com esse ataque, um par malicioso pode comprometer um sistema P2P ao conseguir o controle de uma fração substancial do mesmo. Os autores sugerem o uso de autoridades certificadoras de identidade como a melhor forma de proteção contra este tipo de ataque. No BitTorrent, identidades são geradas de forma pseudo-aleatória. Um atacante, portanto, pode valer-se desta vulnerabilidade e obter múltiplas identidades. Pares podem banir um par que se recusa a cooperar, mas nem sempre é fácil distinguir ente um par que não está cooperando e um par que enfrenta problemas de conectividade, por exemplo.

3.1. Mentindo a posse de peças

Conforme discutido na Seção 2, pares empregam mensagens Bitfield e Have para informar seus pares remotos sobre posse de peças. Seguindo a política LRF, pares tendem a uniformizar a quantidade de cópias de cada peça disponíveis no enxame. O ataque *Mentira de peças* procura destruir este equilíbrio: um par malicioso não segue o protocolo e anuncia que possui peças que não possui realmente (daí a denominação mentiroso). Um número suficiente de pares maliciosos mentindo sobre determinada peça induz os corretos a escolherem outras peças primeiro. Em outras palavras, o ataque consiste em tornar peças gradualmente raras ao ponto de desaparecer do sistema, causando a falha global do enxame. Como o par malicioso não quer entregar a peça anunciada, ele mantém os pares conectados permanentemente no estado sufocado.

A efetividade do ataque será afetada pela força do ataque e pelo estado do enxame quando este começar. Aspectos relevantes incluem o número de peças mentidas pelos pares maliciosos (e o número de peças do conteúdo distribuído), número de pares maliciosos, número de pares corretos sugadores e semeadores, o grau de disseminação das peças, e se os pares maliciosos estão atuando em conluio.

Intuitivamente, mentir a posse de um pequeno conjunto de peças ao invés de apenas uma aumenta a chance de uma delas constituir um problema. No entanto, existe uma relação entre o tamanho desse conjunto e a efetividade do ataque; no extremo, mentir todas as peças seria inefetivo no sentido de tornar uma peça mais rara.

O impacto de ataques a um enxame é geralmente mais efetivo quando executado por múltiplos pares maliciosos atuando em conluio. No caso específico de tornar uma peça mais rara, a expectativa é que o número de pares maliciosos influencie diretamente no ataque, porque quanto mais pares mentirem ter uma peça específica, menos rara ela *parecerá* se tornar. O impacto do número de mentirosos no ataque será discutido na Seção 5.

3.2. Isolando pares corretos (Eclipse)

Se um atacante possui suficientes recursos físicos ou cria um grande número de Sybils, ele pode atacar um enxame mais efetivamente. O ataque é facilitado pelo fato que um único conjunto de pares maliciosos pode ser usado para atacar centenas ou milhares de enxames, desde que apenas mensagens de controles sejam trocadas entre os pares.

No ataque de isolamento a redes P2P ou Eclipse [Singh et al. 2006], um conjunto de pares maliciosos atua em conluio para que um par correto ocupe todas suas conexões com pares da coalisão. Em caso de sucesso, o atacante pode intermediar a maior parte ou toda a comunicação da vítima. No contexto do BitTorrent, o ataque de isolamento consiste em utilizar um número suficientemente alto de pares maliciosos de forma que aqueles corretos conectem em maioria (ou apenas) com pares maliciosos. Um par correto mantém, por padrão, 55 conexões. Um par malicioso, por sua vez, pode manter um número muito maior de conexões. Com isso, 55 pares maliciosos poderiam ser suficientes para isolar qualquer número de pares corretos. Este caso ocorre apenas quando pares corretos ocupam todas suas conexões com pares maliciosos, e os pares maliciosos possuam uma capacidade de conexão igual ou maior ao número de pares corretos.

Na prática, contudo, durante um ataque a lista que um par correto receberá do rastreador será mista, composta por pares corretos e maliciosos. A lista é gerada aleatoriamente toda vez que o rastreador é consultado. Assim, quanto maior for a proporção de pares maliciosos em relação aos pares corretos na lista global do rastreador, maior será a probabilidade de uma lista gerada aleatoriamente conter apenas pares maliciosos.

3.3. Enviando peças corrompidas

Os ataques discutidos anteriormente utilizam a estratégia de consumir o mínimo de recursos (processamento e principalmente largura de banda) possíveis para maximizar o número de atacantes inseridos no enxame. Quando o atacante possui recursos, outra estratégia, baseada em envio de blocos corrompidos, pode ser adotada.

Devido à sistemática de verificação de integridade de peças no BitTorrent (vide Subseção 2.3), não é possível a um par correto descobrir o conjunto de blocos incorretos em uma peça não íntegra. Como resultado, o par tem que descartar a peça inteira e realizar seu download novamente. Ilustrando, se o tamanho da peça é 4MB, o envio de um único bloco errado (16KB) provocará o desperdício dos outros 255 blocos (4080KB) recebidos de forma correta, pois não há como o par saber qual dos 256 blocos está inconsistente com o conteúdo da peça.

Algumas implementações de BitTorrent possuem um mecanismo de punição a pares que enviam peças erradas, mas esta punição só pode ser realizada (a) para peças, isto é, caso um mesmo par realize o envio de todos os blocos de uma peça, (b) bloqueando-se todos os pares que contribuíram com blocos para uma determinada peça ou (c) de forma manual. Aliado a isso, o fato de não enviar todos os blocos de uma peça faz com que um atacante possa ampliar o ataque para mais pares e peças. Estes são os motivos principais pelos quais um atacante enviaria blocos forjados e não peças inteiras.

Para efetivar um ataque desta natureza, o atacante terá que informar a um determinado par que ele não está mais sufocado, aguardar a solicitação de um bloco de determinada peça e enviar uma *stream* de conteúdo aleatório em seu lugar. Logo após, o par malicioso informará ao par sendo atacado que este está sufocado novamente. Isso obrigará o par correto a solicitar a outros pares os blocos restantes da peça desejada. Periodicamente, este processo deverá ser repetido; a escolha do intervalo entre repetições deverá ser tal que minimize o tráfego de upload e maximize as chances de o bloco anterior já ter sido descartado com a respectiva peça, e seu novo download ainda não ter sido completado.

O desafio para o atacante, neste caso, é definir a freqüência com que este conjunto de ações deve acontecer. Uma aproximação possível pode ser realizada monitorando-se a freqüência de mensagens Have recebidas do par remoto. Se um par remoto envia uma mensagem Have a cada 5 segundos, pode-se inferir que este é o tempo, em média, que uma peça leva para ser obtida.

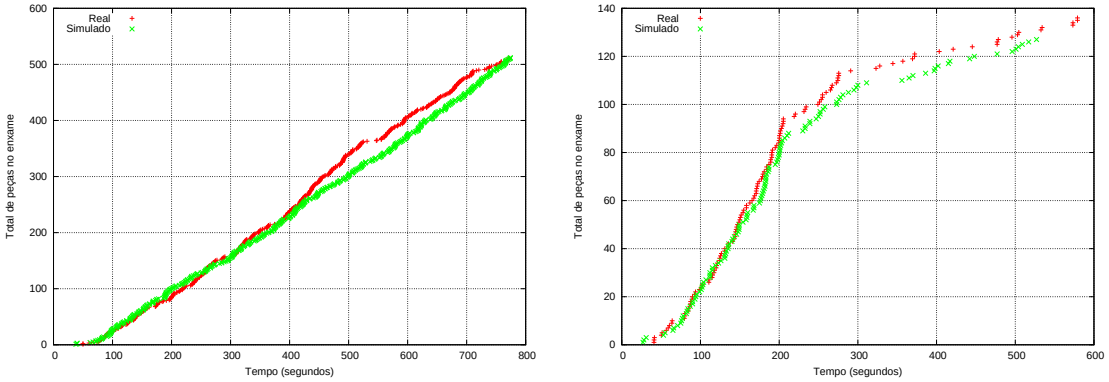
Por último, caso o atacante queira forçar o envio de blocos de uma determinada peça ao par sendo atacado, ele poderá forçar o par a escolher esta peça anunciando a posse apenas dela no momento da conexão e conseqüente troca do Bitfield entre os pares.

4. Simulação do BitTorrent

A fim de avaliar as proposições da Seção 3, criou-se um modelo de simulação do protocolo BitTorrent. O modelo está baseado em uma combinação da especificação do protocolo e da análise de traços de pacotes de duas implementações populares. Ele incorpora as características do protocolo citadas na Seção 2, com a implementação de rastreadores, pares e todas as mensagens e estados mostradas nos diagramas daquela seção. O modelo foi implementado estendendo arcabouço de simulação Simmcast [Barcellos et al. 2006].

Embora o modelo seja bastante realista, diversas premissas simplificatórias foram adotadas. Primeiro, existe apenas um único rastreador, enquanto nos enxames atuais podem haver múltiplos rastreadores e os pares podem alternar entre eles. Segundo, não há troca efetiva de dados entre pares; portanto, as mensagens de dados carregam apenas a informação de controle necessária. Um par envia mensagens Piece(p,b) "contendo" um bloco b de uma peça p. Terceiro, a simulação assume o TCP como protocolo de camada de transporte subjacente, mas não inclui detalhes de envio de segmentos. Na implementação do BitTorrent, um bloco é confiavelmente transmitido através de uma conexão TCP para outro par, potencialmente empregando múltiplos segmentos, requisitando confirmações e retransmissões. Na simulação, a mensagem Piece(p,b) é enviada através de apenas um segmento. Controle de fluxo é realizado em nível de aplicação, embora um par envie apenas uma confirmação por bloco para o outro par.

Para validar o modelo de simulação e implementação, foi rodado um conjunto de experimentos em um ambiente controlado. Enxames com diferentes tamanhos de conteúdo, tamanho de peças, largura de banda, e número de pares foram comparados. Pacotes foram coletados utilizando monitoração passiva. Foi analisado o comportamento e o desempenho das execuções simuladas e reais em configurações de 8 pares com largura de banda homogênea e heterogênea. A Figura 2 ilustra a similaridade; ela apresenta o número total de peças distribuídas no enxame, em dois cenários: homogêneo com 64 peças e heterogêneo com 16 peças.



(a) Cenário homogêneo, 64 peças (b) Cenário heterogêneo, 16 peças

Figura 2. Comparação entre simulação e experimento

5. Avaliação do Impacto

Esta seção procura responder quatro questões: primeiro, qual o impacto do ataque Mentira de Peças sobre a dinâmica do enxame, comparado a um sistema sem ataque; segundo, o que acontece quando o número de mentirosos aumenta; terceiro, no ataque de isolamento, qual o número de pares maliciosos é mais efetivo e porque (o que acontece durante a vida do enxame). Quarto, qual o prejuízo causado por ataques de corrupção e qual o impacto do intervalo entre ataques e o número de atacantes na efetividade do ataque. Executaram-se os experimentos múltiplas vezes utilizando-se sementes diferentes para fins de obter dados estatisticamente corretos.

A simulação segue o modelo descrito na Seção 4. Empregou-se valores mencionados na literatura, como [Legout et al. 2006], ou tipicamente encontrados em configurações reais. A análise focou nas primeiras horas do enxame. Um enxame parte de um semeador inicial. Através do tempo, 100 sugadores ingressam no enxame de acordo com um processo de Poisson, com tempos entre chegadas de pares seguindo uma distribuição Exponencial [Eger and Killat 2006, Guo et al. 2005]. Os pares maliciosos atuam em conluio e iniciam o ataque no momento em que o quinto sugador entra no enxame. A reconexão entre par e o rastreador ocorre periodicamente a cada 30s, usualmente o mínimo permitido. A cada vez, o rastreador envia uma nova lista aleatória de pares.

Os experimentos são baseados em uma *rede justa*, onde todos os sugadores contribuem com pelo menos a mesma quantidade de recursos que receberam. Assim, o parâmetro *ratio* (upload/download) a ser atingido por semeadores foi definido como 2 para os semeadores iniciais e 1 para aqueles que se tornam semeadores posteriormente. O nível de contribuição de pares dependerá das características Evidências sugerem que em enxames reais, dependendo do conteúdo, usuários não contribuem nem mesmo com este fator mínimo, circunstância mais suscetível a ataques. Adotar valores mais baixos tornariam os ataques mais efetivos.

O conteúdo compartilhado tem o tamanho de 640MB, dividido em 640 peças de 64 blocos de 16 KB de dados cada. Os sugadores são configurados com um tempo máximo de download, que ao ser extrapolado determina a saída do enxame (representando a desistência do usuário). O valor utilizado é equivalente a dez vezes o tempo médio de download em um enxame sem ataque. Um sugador poderá também abortar o download devido à morte de todo o enxame, que ocorre quando alguma peça alcança disponibilidade no enxame igual a 0 (assumindo-se que os pares não retornam ao enxame). Quando um enxame falha, são considerados todos os presentes e futuros sugadores como *downloads falhos*. Para permitir uma análise, adota-se um limite inferior no número de downloads falhos na ocorrência de uma falha global de enxame: o número de donwloads falhos é representado em complemento de 100 sugadores, enquanto na

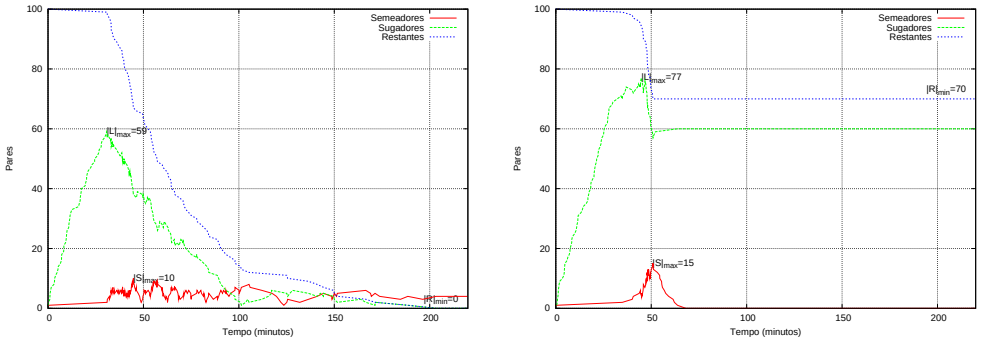
prática o dano poderia ser muito maior.

Adota-se larguras de banda de download e upload distintas, 1Mbps e 256Kbps para download e upload, respectivamente, refletindo-se a assimetria tipicamente encontrada em redes com banda larga. O conjunto de pares de enlaces, contudo, são homogêneos: todos os pares possuem a mesma capacidade. Assume-se também que o rastreador é conectado através de um enlace simétrico mais poderoso, de 4Mbps. A latência entre quaisquer dois pares (e o rastreador) é configurada em 100ms.

Como foi mencionado anteriormente, a efetividade de um ataque é influenciada pelo número de peças sendo mentidas e atacantes envolvidos. Para avaliar a efetividade de um ataque, aplicaram-se duas métricas: o aumento no tempo de download e número de downloads falhos devido a timeout ou falha de enxame.

Os símbolos $|L|_{max}$ e $|S|_{max}$ representam o número máximo de sugadores (*leechers*) e semeadores (*seeders*) alcançados, respectivamente, durante o enxame. Considerando o cenário previamente definido, um enxame apresentará a dinâmica mostrada na Figura3(a). Ela representa a chegada de 100 sugadores, sua gradual transição para semeadores, e saída do enxame aproximadamente aos 200 minutos. São três as curvas: número de sugadores, semeadores e downloads restantes. Esta última representa o número de downloads que precisam ser completados para alcançar o número total de downloads (100). Logo, o experimento termina quando esta curva alcança 0 (ou, como mencionado anteriormente, quando a disponibilidade de peças igualar-se a 0 determinando a morte do enxame). O símbolo $|R|_{min}$ representa o valor mínimo atingido nessa curva; portanto, quando $|R|_{min} = 0$, o enxame teve pleno sucesso e todos os downloads foram completados.

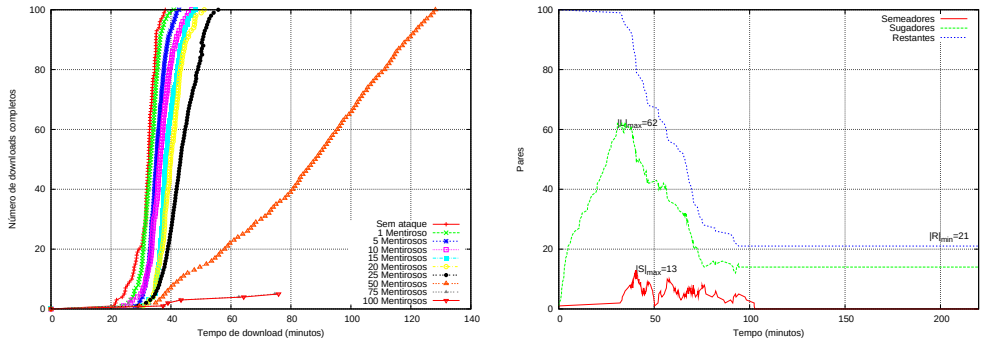
A Figura 3(a) mostra um *flash crowd* de sugadores chegando no começo, até atingir um máximo em $|L|_{max} = 59$. Neste ponto, os sugadores passam a tornar-se semeadores, e a curva decresce. A tendência mantém-se até próximo dos 100 min, quando a multidão de sugadores diminui e uma taxa de entrada estável é alcançada; compensando estes novos semeadores, existe uma taxa de (sugadores completos atuando como) semeadores deixando o enxame assim que alcançam razão de contribuição 1,0.



(a) Enxame normal sem ataque (b) 25 mentirosos mentindo 4 peças

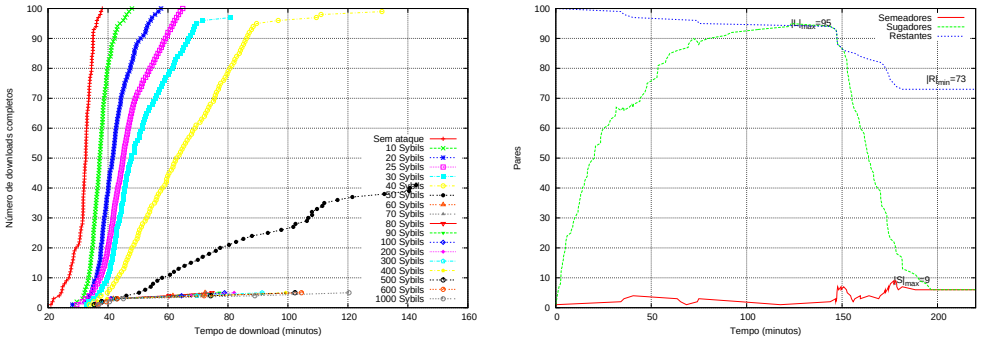
Figura 3. Dinâmica de enxames com e sem ataques de Mentira de peças

A Figura 3(b) mostra o mesmo cenário mas sob ataque de um conluio de 25 mentirosos que falsamente anunciam a posse de 4 peças. Apesar de o ataque começar no início da vida do enxame, seus efeitos são sentidos apenas próximo aos 50 min. A partir deste ponto, nenhum progresso é feito – nenhum download é completado, porque a disponibilidade de uma ou mais peças chega a 0 e o enxame falha. Observe que $|L|_{max} = 77$, maior que o caso sem ataque, o que ocorre devido aos últimos semeadores partirem do enxame, deixando sugadores em espera eterna pelas peças mentidas pelos maliciosos.



(a) Impacto do número de mentirosos (b) 10 mentirosos mentindo 32 peças

Figura 4. Impacto do número de mentirosos



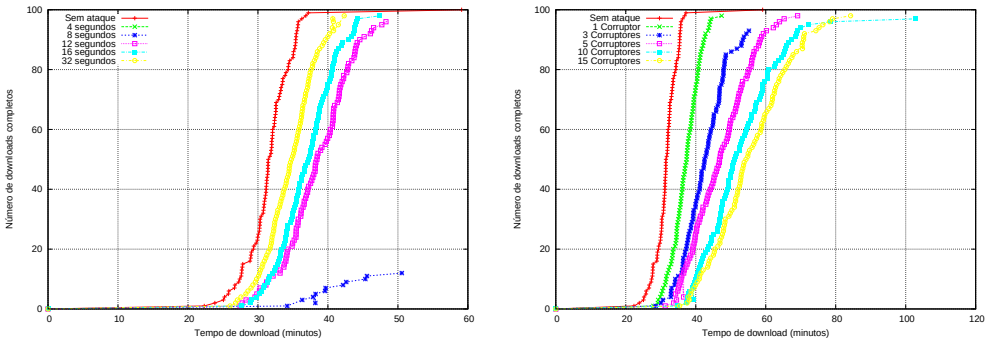
(a) Impacto do número de Sybils (b) Isolamento com 50 Sybils

Figura 5. Impacto do número de Sybils no ataque de Isolamento

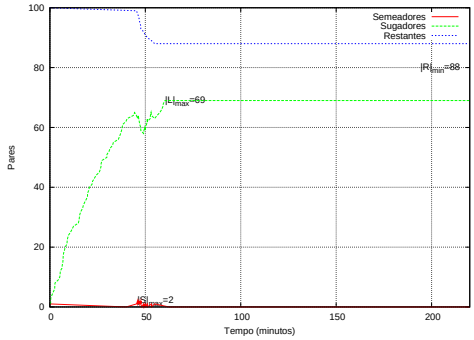
A Figura 4(a) ilustra o impacto de um ataque onde números diferentes de pares maliciosos mentem possuir 32 peças, tornando-as mais raras. Está claro a partir da figura que para conjuntos de até 25 mentirosos, o atraso é negligível. Contudo, alguns dos downloads falharam ou abortaram por falha no enxame, como detalhado a seguir. A curva diagonal correspondente a 50 mentirosos é um claro divisor: a partir de 50, o ataque aumenta substancialmente o atraso (um download pode levar até três vezes mais tempo). Com 75, 100 ou mais pares maliciosos, o ataque de Mentira de Peças é muito efetivo e leva freqüentemente à morte do enxame em estágios iniciais: menos que 10 pares conseguem completar o download com sucesso. No entanto, dada a alta proporcionalidade de número de pares maliciosos envolvidos, não é possível determinar se o efeito é causado devido à mentira de peças ou ao isolamento dos pares corretos por pares maliciosos.

A Figura 4(b) mostra em detalhes um enxame particular onde 10 pares mentem sobre 32 peças. Em 100 pares, 79 completam o download e 21 não. Isto ocorre porque uma ou mais peças mentidas tornam-se indisponíveis, já que os pares corretos escolhem as peças mais raras. Após algum tempo (em torno de 40 min), a multidão de sugadores (que ajuda a distribuir peças) cai e não há mais semeadores. Conseqüentemente, o enxame morre.

No próximo experimento, o impacto de um ataque de isolamento é avaliado de acordo com o número de pares maliciosos (presumidamente Sybils). A Figura 5(a) ilustra o impacto em termos de downloads cumulativos, para conjunto de número de Sybils variando entre 10 e 1000. Os resultados indicam que existe um atraso perceptível em todos os casos, que torna-se maior a partir de 40 Sybils. Como os pares maliciosos atuam em conluio, assim que um par correto estabelece uma conexão com um malicioso, este informará seus comparsas sobre a



(a) Impacto do intervalo entre ataques (b) Impacto do número de corruptores



(c) Enxame com um corruptor e intervalo 8s

Figura 6. Ataque de corrupção de peças

existência daquele. Com 50 Sybils ou mais, a lista de pares retornada pelo rastreador tende a ser dominada por pares maliciosos; além disso, o ataque de isolamento torna-se muito efetivo: menos que 50% dos pares completam seus downloads. A partir de 60 mentirosos, o semeador e a maioria dos outros pares tornam-se isolados, e todos enxames sofrem falha completa (todos downloads restantes sofrem timeout ou falham). Consequentemente, apenas alguns dos pares corretos conseguiram completar seus downloads com sucesso (aproximadamente 5).

A Figura 5(b) mostra a dinâmica do enxame quando ele está sujeito a um ataque de isolamento com 50 sybils. O número máximo de sugadores alcança um limite $|L|_{max} = 95$, onde a massa de sugadores em sua quase totalidade não consegue completar o download. A curva intersecciona com a de downloads restantes em torno de 140 min, quando os sugadores começam a desistir do enxame (timeout). Em 50 min, o número de sugadores cai a 6. Note-se que $|R|_{min} = 73$, o que significa que 73 downloads falharam devido a timeout de par ou a morte do enxame por completo. Em outras palavras, este tipo de ataque é muito efetivo.

O último conjunto de experimentos trata do ataque de corrupção de peças. Neste tipo de ataque, conforme comentado na Subseção 3.3, um dos fatores principais é o intervalo entre os ataques individuais, ou seja, quando o par malicioso envia uma mensagem Unchoke para o par correto, provavelmente recebe uma requisição de bloco, e então responde à mesma enviando um bloco corrompido. A Figura 6(a) mostra o impacto de um ataque efetuado por um par malicioso, considerando diferentes valores de intervalo entre ataques (4s, 8s, 12s, 16s, 32s). Primeiramente, repare-se que a curva relativa ao intervalo 4s não aparece, devido ao fato que todos os ataques foram altamente efetivos e causaram uma falha global no enxame bem cedo. No caso da curva relativa ao intervalo 8s, o ataque é bastante efetivo, posto que 12 pares apenas completaram seus downloads. Para os demais intervalos, 12, 16 e 32s, houve um impacto em termos de atraso e falhas, mas não chegou a ser significativo.

A Figura 6(b) mostra o impacto do número de pares corruptores em um ataque com intervalo de 16s entre ataques (assumindo-se que os pares não são Sybils e sim pares "normais", com larguras de banda iguais aos pares corretos). Primeiramente, observe-se que, em consonância com o gráfico na Figura 6(a), um atacante com intervalo 16s não possui quase impacto. Em seguida, como esperado, nota-se que o impacto do ataque aumenta a medida que cresce o número de corruptores. No entanto, apesar de ser significativo o atraso induzido, não se trata de ataque com grande impacto: o tempo de download apenas dobra quando há 15 corruptores, em relação ao caso sem ataque, e mais de 95 pares completam com sucesso seus downloads em todos os casos.

A Figura 6(c) ilustra a dinâmica de um enxame quando um único corruptor efetua um ataque usando intervalo de 8s entre sucessivos ataques. Em primeiro lugar, nota-se que o ataque é bastante efetivo nessa configuração, pois apenas 12 dos pares conseguiram completar seus downloads ($|R|_{min} = 88$). Este resultado está de acordo com aquele apresentado na Figura 6(a), curva "8s". O ataque faz com que sugadores não progridam, e ao mesmo tempo semeadores permanecem no enxame por um tempo limitado. Pouco após 50min, quando há 69 sugadores no enxame, este sofre uma falha global por ausência de uma ou mais peças.

6. Trabalhos Relacionados

Diversos trabalhos na literatura [Qiu and Srikant 2004, Yang and de Veciana 2004, Veciana and Yant 2003] estudam a capacidade de serviço do sistema BitTorrent com população transeunte. São apresentados modelos matemáticos baseados em cadeias de Markov e na teoria dos fluidos como forma de modelar o comportamento da arquitetura. Estes artigos nos ajudaram a entender a dinâmica do BitTorrent e construir um modelo de simulação consistente com suas descobertas.

Os trabalhos mais próximos do aqui apresentado são aqueles relacionados ao comportamento egoísta de pares. Diversos trabalhos na literatura demonstram primeiro que o mecanismo de incentivo utilizado pelo BitTorrent nem sempre garante justiça no compartilhamento, para então propor novos mecanismos de interação ([Purandare and Guha 2006, Bharambe et al. 2005a, Bharambe et al. 2005b, Guo et al. 2005, Jun and Ahamad 2005, Legout et al. 2006]).

Entre esses, o trabalho mais fortemente relacionado é [Liogkas et al. 2006], que apresenta três técnicas nas quais um par não respeita o protocolo e "mente" sobre a disponibilidade de peças ou se recusa a cooperar. Em contraste, no presente artigo o anúncio falso de peças é uma das técnicas usadas apenas para prejudicar um enxame BitTorrent, ao invés de se beneficiar do mesmo. A idéia é usar anúncio falso de peças para atrasar ou, idealmente, evitar que um conteúdo seja distribuído. Nós analisamos também cenários onde pares maliciosos enviam blocos corrompidos para os pares corretos (conforme sugerido em [Shneidman et al. 2004]), ou que exploram a falta de um esquema de identidade forte para atacar em *conluio* e isolar pares corretos.

O presente artigo representa uma evolução do trabalho em [Konrath et al. 2007], melhorando-o em diversos aspectos. A parte que descreve BitTorrent foi enriquecida com um diagrama de tempo, que ajuda a mostrar a dinâmica de um enxame. A modelagem de enxames foi aperfeiçoada, no sentido de representar mais adequadamente a dinâmica real do enxame – foram removidas diversas premissas simplificatórias da simulação, como por exemplo o fato de que todos os pares pertencentes ao enxame iniciavam no mesmo instante 0, e que quando pares semeavam, o faziam eternamente. Além disso, o tamanho do conteúdo foi aumentado para 640MB e a rede foi configurada com enlaces de capacidade assimétrica. A análise dos resultados foi melhorada também, com informações representando o impacto de um ataque ao longo da vida de um enxame específico. Por fim, tratamos de um novo ataque, de corrupção de

blocos, e analisamos quantitativamente seu impacto.

7. Conclusão

Este trabalho aborda questões de segurança em BitTorrent e apresenta uma avaliação quantitativa do impacto de três ataques: corrupção e mentira de peças e isolamento de pares. O artigo estende um trabalho anterior, em que pela primeira vez foram abordados os aspectos de segurança de BitTorrent. Foi mostrado que mesmo um conjunto modesto de recursos são suficientes para atacar e prejudicar exames BitTorrent nos três ataques estudados.

Os resultados da avaliação demonstraram que BitTorrent é suscetível a ataques em que pares maliciosos mentem a posse de peças e tornam-nas artificialmente mais raras. Este ataque é efetivo em atrasar ou falhar downloads quando existem em torno de 50 pares ou mais atuando, para enxames similares àqueles considerados no estudo. Resultados adicionais foram omitidos devido ao limite de espaço, porém foram avaliados fatores como o número de peças mentidas e o momento em que o ataque inicia. Quando o número de peças mentidas é definido entre 4 e 64 peças, o ataque obtém maior sucesso (sendo 16 o "melhor" número para a configuração considerada); como esperado, quanto antes o ataque iniciar, mais efetivo será o ataque. Ainda, em cenários onde um grande número de pares negam-se a colaborar, e pares corretos estão isolados por maliciosos, os efeitos são ainda mais graves. Quanto ao ataque de corrupção, observou-se que o mesmo é extremamente efetivo quando o intervalo entre ataques individuais é relativamente pequeno (nos casos avaliados, 8s ou menos, porém esse valor dependerá das larguras de banda empregadas e do tamanho da peça).

O artigo pode ser estendido de várias formas. Estamos projetando mecanismos para o BitTorrent de maneira a eliminar ou reduzir o impacto de tais ataques, o que representa um bom desafio de pesquisa. Além disso, apesar da validação dos resultados produzidos pela simulação através de coleta de pacotes de duas implementações populares em um ambiente controlado, nós pretendemos implementar estes ataques em um agente BitTorrent de forma que seja possível medir de forma mais precisa os atrasos produzidos em enxames reais.

Referências

(2007). Azureus website. http://azureus.sourceforge.net/.

(2007a). Bittorrent protocol specification v1.0. http://wiki.theory.org/BitTorrentSpecification.

(2007b). Bittorrent.org. http://www.bittorrent.org/protocol.html.

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.

Barcellos, M. P., Facchini, G., Muhammad, H. H., Bedin, G. B., and Luft, P. (2006). Bridging the gap between simulation and experimental evaluation in computer networks. In *Simulation Symposium, 2006. 39th Annual*, pages 8 pp.+.

Bharambe, A. R., Herley, C., and Padmanabhan, V. N. (2005a). Some observations on bittorrent performance. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS 2005)*, volume 33, pages 398–399, New York, NY, USA. ACM Press.

Bharambe, A. R., Herley, C., and Padmanabhan, V. N. (2005b). Understanding and deconstructing BitTorrent performance. Technical Report MSR-TR-2005-03, Microsoft Research.

Das, S., Tewari, S., and Kleinrock, L. (2006). The case for servers in a peer-to-peer world. In *2006 IEEE International Conference on Communications*, volume 1, pages 331–336, Washington, DC, USA. IEEE Computer Society.

Douceur, J. R. (2002). The sybil attack. In *1st International Workshop on Peer-to-Peer Systems*, pages 251–260, Cambridge, MA, USA.

Eger, K. and Killat, U. (2006). Bandwidth trading in unstructured p2p content distribution networks. In *6th IEEE International Conference on Peer-to-Peer Computing, 2006 (P2P 2006)*, pages 39–48, Washington, DC, USA. IEEE Computer Society.

Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X. (2005). Measurements, analysis, and modeling of bittorrent-like systems. In *Internet Measurement Conference (IMC '05)*, pages 35–48.

Jun, S. and Ahamad, M. (2005). Incentives in BitTorrent induce free riding. In *ACM SIGCOMM Workshop on Economics of Peer-to-Peer systems (P2P-ECON)*, pages 116–121.

Konrath, M. A., Barcellos, M. P., Silva, J. F., Gaspary, L. P., and Dreher, R. (2007). Atacando um enxame com um bando de mentirosos: vulnerabilidades em bittorrent. In *XXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2007)*, volume 2, pages 883–896.

Legout, A., Urvoy-Keller, G., and Michiardi, P. (2006). Rarest first and choke algorithms are enough. Technical report, INRIA.

Liogkas, N., Nelson, R., Kohler, E., and Zhang, L. (2006). Exploiting bittorrent for fun (but not profit). In *5th International Workshop on Peer-to-Peer Systems (IPTPS 2006)*.

Purandare, D. and Guha, R. (2006). Preferential and strata based p2p model: Selfishness to altruism and fairness. In *12th International Conference on Parallel and Distributed Systems, 2006. ICPADS 2006*, volume 1, pages 561–570.

Qiu, D. and Srikant, R. (2004). Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 34(4):367–378.

Shneidman, J., Parkes, D., and Massoulie, L. (2004). Faithfulness in internet algorithms. In *Proc. SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04)*, Portland, OR, USA. ACM SIGCOMM.

Singh, A., Ngan, T.-W., Druschel, P., and Wallach, D. S. (2006). Eclipse attacks on overlay networks: Threats and defenses. In *25th Conference on Computer Communications (INFOCOM 2006)*. IEEE.

Veciana, G. and Yant, X. (2003). Fairness, incentives and performance in peer-to-peer networks. In *Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing*.

Wang, G. (2006). An efficient implementation of sha-1 hash function. In *IEEE International Conference on Electro/information Technology, 2006*, pages 575–579.

Yang, X. and de Veciana, G. (2004). Service capacity of peer to peer networks. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2242–2252.