UNIVERSIDADE DO VALE DO RIO DOS SINOS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM
COMPUTAÇÃO APLICADA

Estudo das Vulnerabilidades da Arquitetura BitTorrent, Ataques e Contramedidas Possíveis

por

MARLOM ALVES KONRATH

Dissertação submetida a avaliação como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada

Orientador: Prof. Dr. Marinho Pilla Barcellos

São Leopoldo, janeiro de 2007

Ficha catalográfica elaborada pela Biblioteca da
Universidade do Vale do Rio dos Sinos

K82e Konrath, Marlom Alves
Estudo das vulnerabilidades da arquitetura BitTorrent, ataques e contramedidas possíveis / por Marlom Alves Konrath. -- 2007.
94 p. : il. ; 30cm.

Dissertação (mestrado) -- Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2007.
"Orientação: Prof. Dr. Marinho Pilla Barcellos, Ciências Exatas e Tecnológicas".

1. Arquitetura de rede - BitTorrent. 2. Computador - Rede - Segurança. I. Título.

CDU 004.72

Catalogação na Publicação:
Bibliotecária Eliete Mari Doncato Brasil - CRB 10/1184

"L'invention de l'arc avait été l'ouvrage d'un homme de génie: la formation d'une langue fut celui de la société entière. Ces deux genres de progrès appartiennent également à l'espèce humaine." (Marquis de Condorcet)

"In the sciences, we are now uniquely privileged to sit side by side with the giants on whose shoulders we stand." (Gerald Holton)

Agradecimentos

Agradeço primeiro a ter a quem agradecer. Isso significa que não estive solitário nesta caminhada e as pessoas mais importantes para mim também estavam desejando esta vitória...

Tive várias professoras na vida, mas uma em especial. O assunto? AMOR! Foi com ela que aprendi o verdadeiro sentido e gosto desta palavra. Com ela o mundo ficou mais colorido, ainda mais porque seus olhos têm o mais puro azul do céu. Obrigado Grazieli, por todos os ensinamentos, companheirismo, amor e dedicação em todo o tempo que estivemos juntos. AMO VOCÊ!, em letras maiúsculas e garrafais. Obrigado por tudo; e como diz o livro dos livros: "Ainda que eu falasse as línguas dos homens e dos anjos, e não tivesse amor, seria como o metal que soa ou como o sino que tine. E ainda que tivesse o dom de profecia, e conhecesse todos os mistérios e toda a ciência, e ainda que tivesse toda a fé, de maneira tal que transportasse os montes, e não tivesse amor, nada seria." I Coríntios 13:1-2

Ao meu amigo e orientador, Marinho Barcellos, não tenho como agradecer com palavras o exemplo, os ensinamentos e conselhos recebidos. Obrigado pelas excelentes discussões, trocas de idéia, dicas, ensinamentos e puxões de orelha. Quero agradecer ainda a um grande incentivador e amigo, Luciano Gaspary, pelas dicas e ajudas recebidas.

A meu pai, Marcos, e à minha mãe, Eva, um obrigado especial por sempre terem me apoiado na realização dos meus sonhos. Obrigado pelos conselhos, chimarrões e pela união da família. AMO VOCÊS. Aos meus irmãos um agradecimento especial. Foi muito bom ver o amadurecimento de vocês, que passaram de irmãos menores (situação em que era muito mais comum eu ensinar algo) a grandes irmãos (com os quais tenho aprendido e trocado ensinamentos). Beijos para meus avós (Otacília, Epaminondas, Getúlio, Delci e Ari).

As minhas segundas mães, Alice e Izabel, um agradecimento especial. Aos cunhados Alexsandro e Wagner um forte abraço.

Agradeço aos meus colegas do mestrado e da Unisinos, em especial ao amigo Juliano, pela amizade, parceria e bom humor.

Por fim, mas não menos importante, agradeço aos amigos Letícia, Rosângela, Patrícia, Otávio, Sidnei, Cícero, Rodolfo, Lucas, Guilherme, Rafael e àqueles que, com certeza, eu estou esquecendo.

Agradeço a você também, que teve paciência e interesse para chegar até aqui. Espero que a leitura seja proveitosa.

Bon appétit! Messieurs et Mesdemoiselles!

Resumo

BitTorrent é a tecnologia de compartilhamento de arquivos mais popular atualmente na Internet e responsável por fração significativa do tráfego nela existente. Considerando sua importância e adoção em larga escala, a arquitetura BitTorrent precisa ser robusta e resistente a pares maliciosos. Esta dissertação constitui a primeira investigação em termos de vulnerabilidades do BitTorrent, apresentando quatro contribuições principais: (a) auxilia no entendimento do BitTorrent; (b) identifica vulnerabilidades e descreve ataques e contramedidas possíveis à arquitetura; (c) descreve um modelo de simulação discreto, que permite mapear o comportamento do protocolo BitTorrent; e (d) avalia, via simulação, o impacto de dois dos ataques identificados, Mentira de Peças e Eclipse. Os resultados mostram que BitTorrent é suscetível a ataques em que pares maliciosos mentem a posse de peças e tornam-nas mais raras, fazendo com que downloads em geral sejam atrasados em até 50%. Em ataques Eclipse, com uma proporção em torno de 20 pares maliciosos para cada par honesto, a análise demonstra que pelo menos 85% dos downloads falham, independentemente do tamanho do enxame.

Palavras-chave: BitTorrent, P2P, Segurança.

TITLE: "STUDY OF THE VULNERATIBITIES IN BITTORRENT'S ARCHITECTURE, FEASIBLE ATTACKS AND COUNTERMEASURES"

Abstract

BitTorrent is the most popular file sharing technology and associated with a significant fraction of the Internet traffic. Considering its importance and large scale adoption, the BitTorrent architecture should be robust and resistant to malicious peers. This dissertation is the first investigation in terms of vulnerabilities against BitTorrent architecture, providing four important contributions: (a) it helps to increase the understanding of the BitTorrent protocol; (b) it identifies vulnerabilities and describes feasible attacks and countermeasures to the BitTorrent architecture; (c) it presents a discrete event simulation model which maps the behaviour of the protocol; and (d) it evaluates, through simulation, the impact of two proposed attacks, namely Piece Lying and Eclipse. Results show what BitTorrent is vulnerable to cheating peers that lie about having pieces, in order to make them rarer, causing an increase of about 50% in downloading times. Eclipse attacks can cause even more damage, as analysis shows that with a proportion of 20 malicious peers to an honest one more than 85% of the downloads fail.

Keywords: BitTorrent, P2P, Security.

Sumário

Lista de Abreviaturas

AS	Sistema Autônomo
CA	Certificate Authority
CMFSD	Collaborative Multi-File Torrent Sequential Downloading
DHT	Distributed Hash Table
HTTP	Hyper Text Transfer Protocol
ICP	Infra-Estrutura de Chaves Públicas
IP	Internet Protocol
LRF	Local Rarest First
MTCD	Multi-Torrent Concurrent Downloading
MFCD	MultiFile Torrent Concurrent Downloading
MTSD	Multi-Torrent Sequential Downloading
NAT	Network Address Translation
OCR	Optical Character Reader
P2P	Peer-to-Peer
PEX	Peer Exchange
PUS	Published Upload Speed
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TFT	Tit-for-tat
UDP	User Datagram Protocol

Lista de Figuras

Capítulo 1

Introdução

O paradigma Peer-to-Peer (P2P), embora bastante antigo e relacionado ao surgimento da Internet [Sadok et al., 2005], permaneceu até o final da década de 90 apenas como um conceito. Mais precisamente, existem duas arquiteturas básicas no projeto de aplicações de rede e sistemas da Internet: Cliente/Servidor e P2P. O primeiro, predominante por muitos anos, se baseia em um servidor de endereço bem conhecido e que aguarda solicitações de serviço enviadas por clientes. Em contraste, no paradigma P2P os pares (do Inglês *peers*) são funcionalmente iguais (dividem as tarefas de cliente e servidor), relativamente autônomos, possuem fraco acoplamento entre si e nenhuma garantia de que permanecerão disponíveis. Esses fatores introduzem novos desafios e possibilidades como, por exemplo, uma população transiente e a ausência de um ponto central de falhas [Lua et al., 2005, Barcellos and Gaspary, 2006].

Dentre as redes P2P existentes, a aplicação de maior popularidade é o compartilhamento de arquivos. Nessa categoria, BitTorrent[1] [Bittorrent, 2007] é atualmente a mais popular [CacheLogic, 2006]. Em [Barbera et al., 2005] os autores afirmam que a rede BitTorrent foi responsável, em 2004, por aproximadamente 53% do tráfego P2P. Os mesmos autores citam que, em 2005, o tráfego P2P representava aproximadamente 80% de todo o tráfego TCP em *backbones*. No

[1]Nesta dissertação, o termo *BitTorrent* é usado para designar o protocolo, a arquitetura e a tecnologia BitTorrent, intercambiavelmente e dependendo do contexto.

entanto, segundo [Erman et al., 2006], o volume de tráfego BitTorrent apresentou uma leve redução em 2005 devido ao fechamento de vários sites populares. Mesmo assim, sua utilização continuou constituindo uma parcela significativa do tráfego P2P, chegando a representar, em algumas regiões, aproximadamente 60% de todo o tráfego P2P observado [CacheLogic, 2007].

Devido à sua crescente popularização, a arquitetura BitTorrent vem atraindo também a atenção da Indústria. Nos últimos meses, grandes empresas, como Warner Brothers [Helm, 2006], 20^{th} Century Fox, MTV Networks e Paramount [Lily, 2006] indicaram que passarão a distribuir conteúdo através dessa tecnologia. A rede BBC também anunciou em [BBC, 2006] que fechou um acordo para distribuição de seus principais programas através do software agente Zudeo [Zudeo, 2007]. Este último apresenta uma idéia similar ao YouTube [YouTube, 2007], mas com oferta de conteúdo com alta definição e na qual o usuário obtém o mesmo através de uma rede P2P baseada em BitTorrent (via extensão do agente Azureus [Azureus, 2007]). Em outra iniciativa, a NTL, companhia de TV a cabo da Inglaterra, pretende produzir um sistema comercial de download em conjunto com a empresa CacheLogic e o BitTorrent [Paul, 2006].

O foco de uma rede BitTorrent é a distribuição eficiente de arquivos. Dada a sua importância, manifestada através da crescente porção de tráfego Internet que ocupa e do interesse de grandes empresas em utilizá-lo, ataques a esta arquitetura possuem grande impacto potencial. Infelizmente, BitTorrent não foi projetado considerando requisitos de segurança, e possui, portanto, vulnerabilidades passíveis de serem exploradas por atacantes.

Surpreendentemente, os trabalhos sobre BitTorrent encontrados na literatura até o momento concentram-se no desempenho e mecanismos de incentivo à cooperação. A presente dissertação é inovadora porque, ao que se sabe, representa a primeira contribuição em termos de vulnerabilidades de segurança do BitTorrent: não há estudo anterior que ofereça uma avaliação dessas vulnerabilidades, seu impacto ou contramedidas possíveis a serem tomadas. Considerando-se que

BitTorrent começa a ser utilizado também em empresas, como suporte à realização de distribuição de conteúdo, os aspectos de segurança da arquitetura e o nível de segurança que realmente podem proporcionar passam a assumir uma importância ainda maior.

A contribuição principal desta dissertação é identificar as possibilidades de ataques ao protocolo BitTorrent e avaliar como estas podem ser exploradas, visando prejudicar o desempenho dos downloads e a robustez da rede como um todo. Mais especificamente, a dissertação tem as seguintes contribuições:

- aumentar o entendimento sobre BitTorrent, oferecendo uma análise detalhada sobre o funcionamento do mesmo, contrastando especificações, documentação sobre implementações e traços de pacotes coletados;
- identificar o espectro de vulnerabilidades a esta arquitetura;
- determinar o impacto que ataques explorando essas vulnerabilidades;
- sugerir contramedidas que potencialmente minimizem ou anulem os efeitos dos ataques.

Para atingir esses objetivos, o restante da dissertação está organizado da seguinte maneira: o Capítulo 2 define a arquitetura do BitTorrent, seus componentes, a relação entre os mesmos e as estratégias utilizadas para permitir um compartilhamento eficiente de conteúdo. O Capítulo 3 apresenta as vulnerabilidades encontradas na arquitetura e como um atacante pode explorá-las a fim de prejudicar o espalhamento de um conteúdo. O Capítulo 4 sugere contramedidas para mitigar o impacto dos ataques identificados. O modelo de simulação discreta desenvolvido para a investigação do BitTorrent é apresentado no Capítulo 5. A implementação desse modelo é empregada na avaliação de impacto para dois dos principais ataques identificados; os resultados são discutidos no Capítulo 6. O Capítulo 7 apresenta os trabalhos relacionados, analisando artigos na literatura que representam o estado-da-arte sobre BitTorrent. A dissertação se encerra com o Capítulo 8 apresentando as conclusões e trabalhos futuros.

Capítulo 2

A Arquitetura BitTorrent

BitTorrent é uma arquitetura para redes Peer-to-Peer (P2P) cujo foco principal é a distribuição eficiente de conteúdos. Os principais estudos e pesquisas realizados nos últimos anos com a arquitetura BitTorrent são apresentados no Capítulo 7. Neste, é abordada a arquitetura do BitTorrent, bem como a estratégia para distribuição eficiente de arquivos por ele adotada. Primeiramente (Seção 2.1) são apresentados o protocolo empregado no BitTorrent e as estratégias utilizadas buscando-se obter integridade, robustez e o incentivo à cooperação entre os pares. Logo a seguir, a Seção 2.2 discorre sobre o mecanismo empregado na política de seleção de peças. A Seção 2.3 discute como um par encontra outros pares participantes de um "enxame" (*swarm*). O capítulo termina com a Seção 2.4 apresentando extensões do BitTorrent relevantes a esta dissertação.

2.1 Arquitetura e Protocolos

A seguir, são descritos os principais elementos da arquitetura BitTorrent e os protocolos por ela empregados para promover o incentivo à cooperação e eficiência no espalhamento de conteúdo. A obtenção de dados em uma rede BitTorrent, de acordo com [Erman et al., 2005], pode ser dividida em fases, conforme ilustra a Figura 2.1. A primeira delas corresponde à obtenção do arquivo de metadados que representa um determinado conteúdo. Na segunda fase, uma entidade centralizada, chamada

tracker, é utilizada como ponto de encontro (*rendez-vous point*). A última etapa compreende a troca de peças visando à obtenção de todas as partes do conteúdo. Cada uma das três fases é descrita em mais detalhes abaixo.

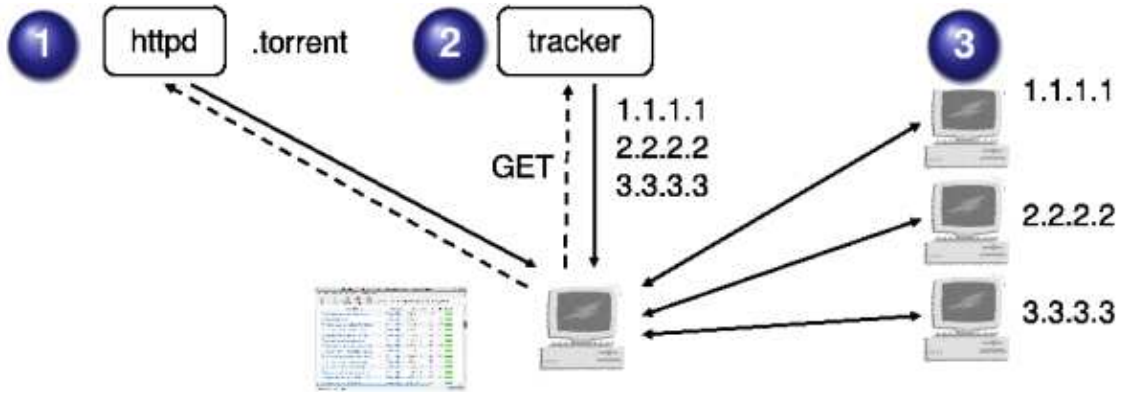


FIGURA 2.1 – Fases de um download no BitTorrent

O "conteúdo" (um único arquivo ou um conjunto deles) a ser disseminado em uma rede BitTorrent é descrito através de um arquivo de metadados e de terminação .torrent. O "torrent", como é popularmente conhecido, é gerado pelo usuário que publica o conteúdo (fase 1), contém informações como o nome e tamanho dos arquivos, *hashes* para os dados, e um ou mais endereços IP de trackers. O tracker é um elemento centralizado que coordena um "enxame" (*swarm*) e auxilia pares a encontrar outros pares no mesmo enxame (fase 2).

Um enxame é um conjunto de pares que faz upload e/ou download de um mesmo conteúdo (fase 3) e conecta-se periodicamente com um tracker. Pares podem ser separados entre "semeadores" (*seeders*), que são pares que possuem uma cópia completa dos dados, e "sugadores" (*leechers*), que são aqueles que ainda não completaram seus downloads. Para o sucesso da rede, é recomendável que, ao tornar-se semeador, um par permaneça no enxame até que sua taxa de contribuição atinja um determinado patamar pelo menos igual ao que recebeu.

O conteúdo publicado e descrito no torrent é organizado em "peças". 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. Quando um par recebe uma peça por completo, notifica – através de uma mensagem de **Have** – aos demais pares com quem está conectado.

Quando um par i recebe informações do campo BITFIELD de outro par j com o qual esteja se conectando, ele realiza uma comparação entre as peças que ele e o par remoto possuem. O comportamento de um par é determinado em função de um conjunto de vetores de *flags*. Mais especificamente, o par i mantém quatro flags para cada par remoto j com quem ele mantém uma conexão. Assumindo que os pares i e j estejam conectados e definindo π_i como sendo o conjunto de peças que o par i possui e π_j as peças de j, as seguintes flags são mantidas pelo par i:

- Interessado[j]: j tem conteúdo que i tem interesse em obter ($\pi_j - \pi_i \neq \emptyset$);
- Sufocado[j]: i está interessado em j, mas não está recebendo de j porque está sendo sufocado pelo mesmo;
- Interessante[j]: i tem conteúdo que o torna interessante para j ($\pi_i - \pi_j \neq \emptyset$);
- Sufocando[j]: i não está enviando conteúdo a j, apesar do interesse de j em obter conteúdo de i.

A flag Interessado[j] expressa o interesse de i em j, e será verdadeira se j possuir uma peça que esteja faltando para i. Toda vez que houver uma mudança de estado em i, uma mensagem será enviada a j, informando-o da mudança.

O par i não realizará *upload* para o par j caso a flag Interessado[j] for falsa. Esta política deriva da lógica de *tit-for-tat*, na medida em que não há vantagem para um par em fazer upload para quem não possa lhe fornecer algo em troca. Há duas exceções a esta regra: uma delas quando o par estiver atuando como semeador, dado que ao possuir todas as peças, o par nunca estará interessado em outros pares. O outro caso é quando o par i está interessado em um número muito pequeno de pares; neste caso, ele pode fazer upload para um par mesmo não estando interessado no mesmo. Mais especificamente, quando a cardinalidade do conjunto Interessado for

menor do que um valor pré-estabelecido, correspondente ao número de uploads que um par deve honrar (usualmente, 4), i pode atender j mesmo quando Interessado[j] é Falso.

Outra estrutura mantida pelo par é o vetor Sufocando. Quando o j-ésimo elemento é verdadeiro, indica recusa temporária de i em realizar upload para j. A cada rodada de σ segundos, i avalia quais foram os η pares com quem ele teve maior taxa de download. Os valores típicos são $\sigma = 10$ segundos e $\eta = 3$ pares. Os η pares dos quais o par recebeu mais download são então marcados como não sufocados, com todos os pares j menos estes melhores recebendo Verdadeiro em Sufocando[j].

Esse mecanismo também deriva da política de tit-for-tat e é utilizado com o objetivo de desencorajar a existência de *free-riders* (pares carona). Cada par é, portanto, forçado a realizar upload a fim de que possa ficar entre os η pares escolhidos para receber download na próxima rodada de cada par com quem interage. Observe-se ainda que o tempo de cada rodada (σ) é independente entre os pares, isto é, enquanto um par está no meio de uma de suas rodadas, outro par pode estar realizando uma avaliação de suas últimas iterações.

A estrutura Sufocado[j] indica se o par i está sendo sufocado pelo par j; esta situação ocorre quando o par i não está, em j, na lista de pares não sufocados (em j, Sufocando[i] = Verdadeiro).

Por fim, Interessante[j] marca se o par i é interessante para j, ou seja, i mantém peças que j deseja, tendo sido i informado por j desta situação via uma mensagem **Interested**. Caso verdadeira em i, é esperado que no par j o valor de Interessado[i] também seja Verdadeiro.

Como se pode notar, cada par mantém informações sobre o seu interesse e possibilidade de fornecer conteúdo para cada par remoto com quem está conectado. Essas informações são enviadas aos pares remotos, sendo que cada par só recebe informações referentes ao estado dele próprio com o par remoto correspondente, assim como os pares remotos informam seu estado ao par local. Há, portanto, um "espelhamento" entre as flags dos pares correspondentes, isto é: Interessado[j] no

par i corresponde a Interessante[i] no par j e Interessante[j] no par i corresponde a Interessado[i] no par j. O mesmo vale para as flags de Sufocado[i] e Sufocando[i] em j, que possuem como correspondentes, respectivamente, Sufocando[j] e Sufocado[j] em i.

Se o mecanismo de tit-for-tat fosse seguido à risca, um par teria grandes chances de ficar trocando peças sempre com os mesmos pares remotos e não teria como descobrir outros com os quais poderia conseguir taxas de upload e download ainda maiores. Novos pares também não teriam a chance de ganhar ao menos uma peça, para que pudessem iniciar seu processo de troca, e sofreriam de "inanição" (*starvation*).

Para possibilitar a descoberta de novas taxas de download, a cada κ rodadas um par j é marcado como não sufocado (Sufocando[j] $\leftarrow$ Falso), independente da taxa de download que i obteve de j. O valor típico de κ é 3. Essa política é conhecida como Dessufocamento Otimista (*Optimistic Unchoke*), pois otimisticamente assume que há pares disponíveis com taxas de upload superiores aos atuais mas que precisam ser "encontrados". Para tal, faz com que o par i realize upload para j, visando realizar um futuro download de j em contrapartida.

Em condições normais, a cada momento, somente um par estará sendo otimisticamente dessufocado. No entanto, ocasionalmente um par poderá ficar sufocado por todos os pares com os quais estava trocando conteúdo nas rodadas anteriores. Neste caso, ele acabará sufocado por todos os pares remotos com os quais está conectado e precisará de certo tempo até que o mecanismo de dessufocamento otimista encontre novos pares com os quais possa trocar conteúdo. Quando um par local fica mais de 1 minuto sem receber blocos de um par remoto, ele assume que está sendo "esnobado" (*snubbed*) pelo mesmo. Nessa situação, um mecanismo, chamado "*anti-snubbing*", permite que o par local eleja, temporariamente, mais de um par remoto para receber dados via dessufocamento otimista. Esta tática faz com que o par local consiga se recuperar muito mais rapidamente da situação indesejável de não ter com quem trocar conteúdo.

O conjunto de peças que um par tem disponível vai crescendo ao longo do tempo, e essa informação deve ser repassada aos demais pares. Quando um par i termina o download de uma peça, ele verifica se ela está correta através do *hash* SHA-1 da mesma. Se o conteúdo estiver íntegro, i envia uma mensagem **HAVE** para seus vizinhos comunicando a posse da nova peça. Os pares vizinhos então atualizam suas tabelas de **BITFIELD**, a fim de refletir a nova realidade. Além disso, cada par j, ao receber **HAVE** de i, verifica se o valor da flag Interessado[i] deve ser modificado. O par i também verifica se a posse desta nova peça modifica o fato de estar interessado em algum par remoto. Caso i deixe de ficar interessado em j, por exemplo, a flag Interessado de j é marcada como falsa (Interessado[j] $\leftarrow$ Falso) e j é comunicado via mensagem **NOT_INTERESTED**, para então atualizar sua flag Interessante[i] com o valor Falso.

Cada peça do arquivo original é dividida em sub-peças ainda menores (chamadas Blocos) durante o seu download. Blocos são a unidade mínima de troca de dados entre pares. Esta subdivisão busca aumentar a eficiência na obtenção do conteúdo na medida em que diferentes sub-peças de uma mesma peça podem ser solicitadas para pares distintos. O tamanho típico de cada bloco é de 16KB. Para reduzir a latência e aumentar a eficiência da comunicação, *pipelining* é utilizado. Pipelining é um processo em que várias requisições são solicitadas ao mesmo tempo [Kurose and Ross, 2004], e na medida em que as respostas vêm chegando, novas requisições são geradas.

Na ausência de uma descrição formal do protocolo na literatura, elaborou-se nessa dissertação diagramas de estado e de tempo. Para tal, baseou-se na análise da descrição do protocolo [BitTorrentProtocolSpecification, 2007] e de traços de implementações populares do mesmo [Azureus, 2007, Bittorrent, 2007, μTorrent, 2007], capturados em ambiente controlado. A Figura 2.2 apresenta o diagrama de estados, de forma global e com ênfase na troca de peças. Os elementos principais são descritos a seguir.

Um par inicia no estado START, conforme representado na Figura 2.2. Ao

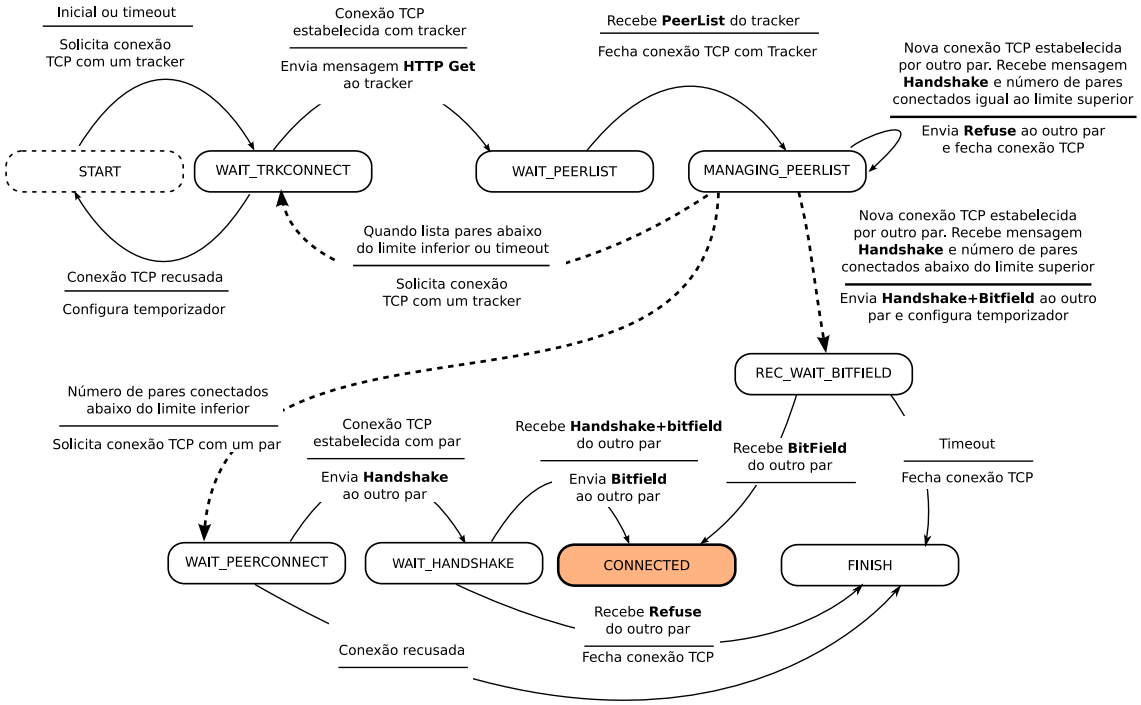


FIGURA 2.2 – Diagrama de estados do BitTorrent

ser iniciado, o par solicita uma conexão TCP com o tracker e passa ao estado WAIT_TRKCONNECT. No estado WAIT_TRKCONNECT, se a conexão TCP é recusada, o par configura um temporizador para uma retentativa e volta ao estado START; se há um *timeout*, ocorre uma nova tentativa. Caso a conexão tenha sido estabelecida com sucesso, o par envia a mensagem **Get** ao tracker e passa ao estado WAIT_PEERLIST. A mensagem **Get** tem, entre seus parâmetros, os campos info_hash e peer_id que contém, respectivamente, um hash do campo que possui as informações sobre o conteúdo e o identificador do par no enxame. Além destas, outras informações são passadas e recebidas em uma solicitação **Get**. Para uma descrição mais detalhada deste passo, veja a Seção 2.3.

No estado WAIT_PEERLIST, se o par recebe uma resposta do tracker com uma lista de até 50 pares [BitTorrentProtocolSpecification, 2007], ele fecha a conexão TCP com este, inicia um temporizador para que o tracker seja reconsultado (tipicamente o tracker informa após quanto tempo o par deverá consultá-lo novamente, embora o par possa consultar o tracker antes deste tempo sem prejuízo) e passa ao estado

MANAGING_PEERLIST. Se a lista retornar vazia ou houver algum erro, o par inicia um temporizador para que o tracker seja reconsultado e, de qualquer maneira, avança para o estado MANAGING_PEERLIST, pois já terá estabelecido uma conexão TCP inicial com o tracker e recebido algumas informações deste. Se o arquivo .torrent possuir mais de um tracker, essas iterações ocorrerão de forma independente e paralela, uma para cada tracker. Além disso, as iterações com um dado tracker podem ocorrer diversas vezes, ou porque não houve retorno para uma mensagem **Get**, ou porque a lista de pares retornou vazia, ou porque o par precisa de mais pares (sua lista ficou abaixo de 20 pares remotos) ou ainda porque houve timeout no temporizador (um par deve estabelecer contato com o tracker periodicamente). Tal é representado no diagrama através da transição de MANAGING_PEERLIST para WAIT_TRKCONNECT.

No estado MANAGING_PEERLIST, há quatro transições possíveis: primeiro, se a lista de pares fica muito pequena ou chega o momento de reconsultar o tracker (timeout), um novo fluxo de execução[1] é criado no estado WAIT_TRKCONNECT, de forma a conectar o par com o tracker e obter mais pares. Segundo, considerando que o par, ao conectar com o tracker, informou seu identificador e pode ser eleito nas próximas listas que o tracker enviar a outros pares, múltiplas conexões podem ser solicitadas por pares remotos; para cada uma, o par local verifica o limite do número de conexões (tipicamente um par não manterá mais do que 55 conexões ao mesmo tempo). Caso dentro do limite (segunda transição), envia mensagem **Handshake+Bitfield**, configura temporizador e cria thread no estado REC_AWAIT_BITFIELD para gerenciar a comunicação com aquele par. No entanto (terceira transição), se o número de pares conectados tiver excedido o limite, uma mensagem **Refuse** é retornada ao par remoto, sendo a conexão TCP com o mesmo encerrada. A quarta possibilidade é a criação de uma thread no estado WAIT_PEERCONNECT, que irá providenciar a conexão com um dos pares da lista. Esta última transição ocorre uma vez para cada um dos pares escolhidos, com base na lista recebida em **PeerList**, até que o número de pares conectados atinja um limite (tipicamente 30 pares) ou

[1] uma nova thread, que é representada por uma seta pontilhada

que não existam mais pares na lista.

No estado REC_WAIT_BITFIELD, a thread aguarda a chegada do **Bitfield**, quando passa ao estado CONNECTED, refletindo a conexão operante com um par remoto. Se a resposta não chegar, ocorre um timeout, levando ao fechamento da conexão e o encerramento da thread. No estado WAIT_PEERCONNECT, a thread aguarda a conexão com o par remoto associado à mesma. Se o estabelecimento de conexão falha, há uma transição para FINISH e a thread termina. Se completada com sucesso, envia uma mensagem **Handshake** e passa ao estado WAIT_HANDSHAKE. Neste estado, a thread pode receber do par remoto uma mensagem **Refuse** ou **Handshake + Bitfield**. No primeiro caso, a conexão TCP com o par remoto é fechada e a thread passa ao estado FINISH, sendo eliminada. No segundo, responde ao par remoto enviando seu mapa de bits em **Bitfield** e passando ao estado CONNECTED.

O estado CONNECTED, marcado no diagrama, concerne às variações de estado que uma thread pode sofrer em função de sua relação com o par remoto pelo qual está responsável de interagir. Essa relação é dirigida por dois diagramas de estado, um para download e outro para upload. O único vínculo entre os mesmos é o processo de escolha de pares, segundo o mecanismo de incentivo, para os quais realizar upload; tal escolha é baseada em reciprocidade em termos de downloads desses pares. As Figuras 2.3 e 2.4 representam os diagramas.

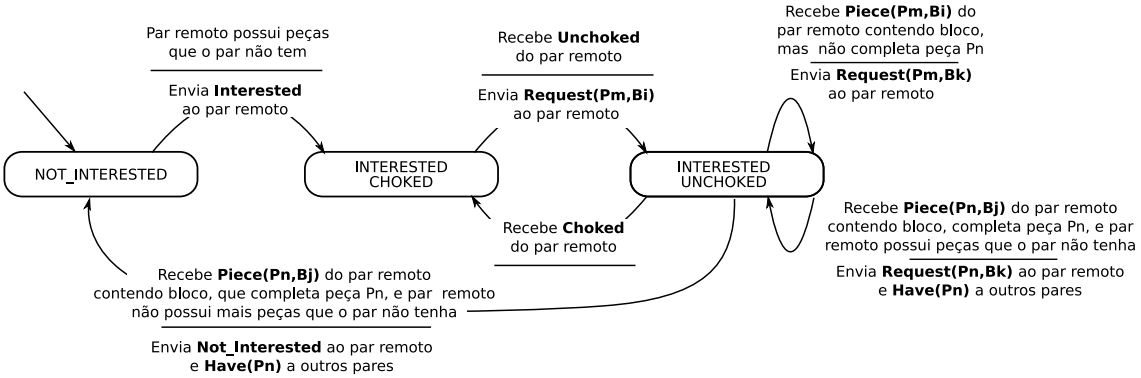


FIGURA 2.3 – Diagrama de estados de thread em relação a downloads de par remoto

Para efeito de download, uma thread em CONNECTED está, inicialmente, no

estado NOT_INTERESTED. Examinando o mapa de bits recebido do par remoto, a thread pode verificar que o par remoto possui uma ou mais peças que o par local não possui. Nesse caso, a thread passa ao estado INTERESTED_CHOKED, e envia uma mensagem **Interested** indicando que o par remoto é considerado "interessante"; entretanto, a thread está sendo sufocada, e assim não pode ainda solicitar peças ao par remoto. No estado INTERESTED_CHOKED, a thread aguarda uma mensagem **Unchoked**; quando isso acontece, é um indicativo do par remoto que o par local está autorizado a solicitar blocos. A thread então envia uma mensagem **Request(Pm,Bi)** e passa ao estado INTERESTED_UNCHOKED. Nesse estado, quatro transições são possíveis. Primeiro, um bloco de uma peça é recebido, porém não completa a peça; uma mensagem **Request(Pm,Bk)** é enviada para requisitar outro bloco dessa mesma peça. Segundo, um bloco de uma peça é recebido, completando-a; neste caso, há ainda duas possibilidades, dependendo se o par remoto possui outras peças que o local não tem. Caso sim, uma mensagem **Request(Pm,Bk)** é enviada solicitando um bloco de outra peça e a thread permanece no mesmo estado. Caso não, envia mensagem **Not_Interested** ao par remoto e passa ao estado NOT_INTERESTED. Em ambos os casos, uma mensagem **Have(Pn)** é enviada a todos os pares conectados, anunciando a disponibilidade da peça.

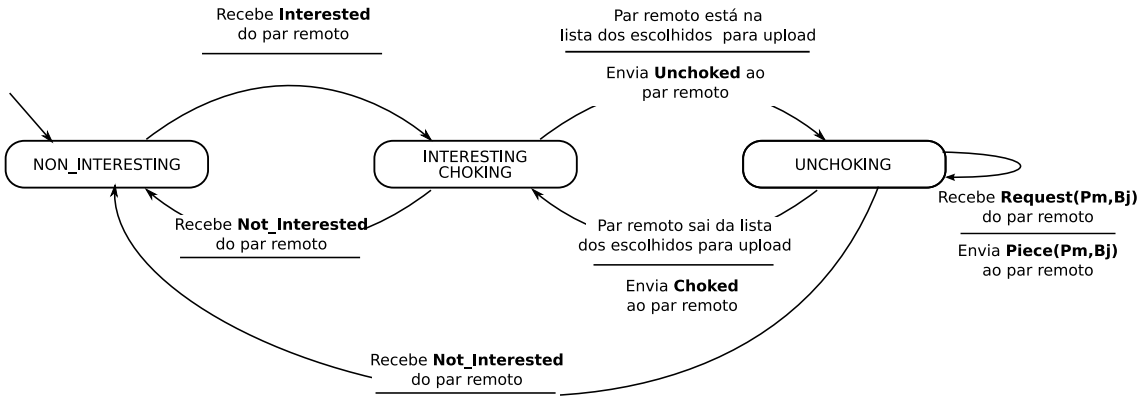


FIGURA 2.4 – Diagrama de estados de thread em relação a uploads para par remoto

Para efeito de upload, uma thread em CONNECTED inicia em NON_INTERESTING, indicando que este par não é interessante ao par remoto. Quando o par recebe

uma mensagem **Interested**, ele passa ao estado INTERESTING_CHOKING: o par local tem peças que o remoto deseja, mas está sufocando o mesmo. Nesse estado, é possível que o par remoto declare, através de uma mensagem **Not_Interested**, não estar mais interessado (por exemplo, porque obteve todas as peças que este par possui).

Quando o par remoto entra na lista dos escolhidos para upload, a thread envia uma mensagem **Unchoked** ao remoto e passa ao estado UNCHOKING. Enquanto nesse estado, o par recebe múltiplas requisições de blocos em mensagens **Request(Pm,Bj)**, enviando a resposta em **Piece(Pm,Bj)**. No momento em que o par remoto sai da lista de escolhidos para upload, a thread envia uma mensagem **Choked** e volta ao estado INTERESTING_CHOKING.

Os diagramas de estado das Figuras 2.2, 2.3, 2.4 definem o espaço de estados e ações de um par, mas não representam a dinâmica da comunicação entre os mesmos. Para tal, ilustra-se na Figura 2.5 um exemplo de comunicação em um enxame. Para ser representável, é mostrada apenas a troca de mensagens entre um tracker e três pares, denominados Semeador, Par1 e Par2. Inicialmente, o Semeador, que possui uma cópia completa do conteúdo, realiza uma conexão TCP (não ilustrada por questões de espaço) com o tracker e informa no mínimo dois campos através de uma mensagem **Get** no formato do protocolo HTTP. Após algum tempo, simbolizado pelo retângulo pontilhado, o Par2 conecta com o tracker, envia uma mensagem **Get** e obtém o endereço do Semeador, pois este está participando do enxame. O Par2 conecta com o Semeador (mensagem **Handshake**), recebe o mapa do par (mensagem **Handshake** + **Bitfield**) e envia o seu (na 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)**.

O recebimento de blocos libera novas solicitações, que ocasionam novos envios e assim por diante. Ao receber todos os blocos de uma peça, há uma verificação de seu hash, a fim de garantir a integridade dos dados recebidos. Se a peça estiver íntegra,

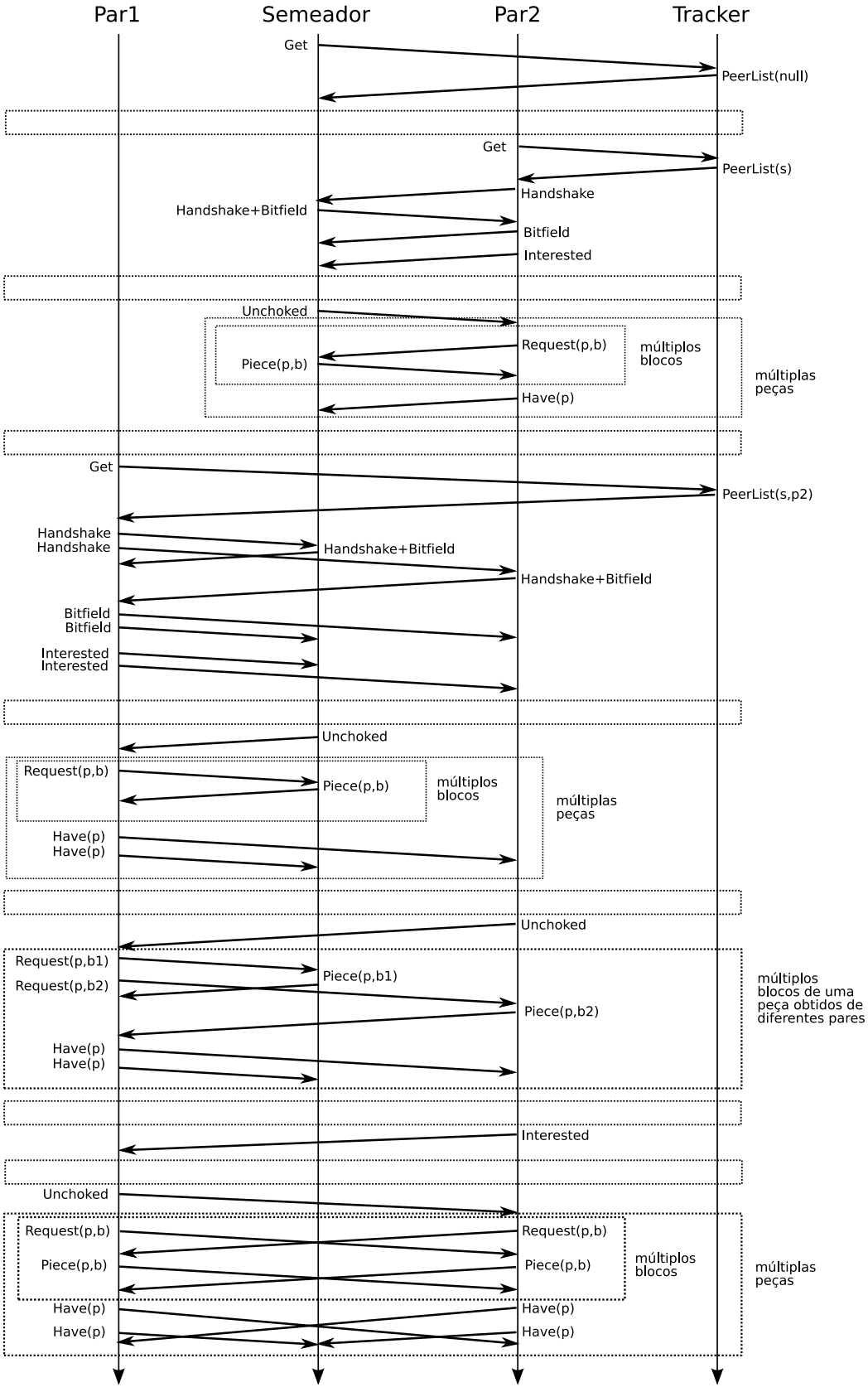


FIGURA 2.5 – Diagrama de tempo ilustrando enxame no BitTorrent

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 tracker, 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 Par1, 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.

2.2 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). Ela se baseia no conceito de escolher primeiro as peças que estão menos replicadas entre pares remotos com os quais o par está conectado, isto é, quando o par j informa a i que este não está mais sendo sufocado, o par i escolhe a peça mais rara (com menor índice de replicação entre os pares vizinhos) existente em j e que i deseja obter. Caso haja mais de uma peça com grau mínimo de replicação, a escolha é aleatória entre as peças mais raras. Durante o download e conforme ilustrado na Figura 2.5, um par solicita vários blocos de uma mesma peça, em paralelo, a pares remotos diferentes.

A escolha da próxima peça ocorre, segundo

[BitTorrentProtocolSpecification, 2007], como descrito a seguir. Seja π_i o conjunto de peças que o par i possui e $\pi_i^{'}$ o conjunto de peças que i ainda não obteve. Seja Δ o conjunto de todas as peças de um conteúdo, tal que: $\pi_i \subseteq \Delta$, $\pi_i^{'} \subseteq \Delta$, $\Delta = \pi_i \cup \pi_i^{'}$ e $\pi_i \cap \pi_i^{'} = \emptyset$. Seja $\Delta^{'}$ o conjunto de peças que os pares remotos conectados possuem e π_j o conjunto de peças que o par j possui. A próxima peça a ser solicitada para j estará no conjunto $\delta = \pi_i^{'} \cap \pi_j$, ou seja, a intersecção do conjunto as peças que i não possui com as que o par j possui, lembrando que $\delta \subseteq \Delta^{'}$ e $\pi_j \subseteq \Delta^{'}$. Seja $\delta^{'}$ o conjunto de peças menos replicadas em δ, com $\delta^{'} \subseteq \delta$. Caso $|\delta^{'}| > 1$, isto é, há mais de uma peça com grau mínimo de replicação, a escolha é aleatória entre as presentes em $\delta^{'}$. Existem três exceções para a regra da LRF:

- quando o par está iniciando o download (situação em que possui menos de 4 peças);
- quando o par está finalizando o download (já requisitou todos os blocos que faltam);
- quando o par está atuando como semeador (possui todas as peças).

Há três vantagens nítidas no uso da dessa abordagem:

- faz com que a robustez da rede aumente, já que todas as peças terão em média a mesma quantidade de réplicas entre os pares, evitando assim o problema da "peça rara" [Legout et al., 2006];
- um par que entre na rede, ao escolher peças mais raras, possui maior chance de possuir conteúdo que outros pares com mais peças não possuem. Com isso, aumentam suas chances de conseguir trocar peças de forma mais rápida;
- por último, esta política também evita o problema da "última peça" [Legout et al., 2006], que ocorre quando um par está no final do processo de download e precisa de uma peça que está pouco distribuída para completar a obtenção do conteúdo. Neste caso, como a política faz com que a distribuição das peças seja praticamente uniforme, um par não vai ter de ficar muito tempo

esperando por uma peça que, utilizando outra política, poderia estar pouco distribuída na rede e, portanto, difícil de ser obtida.

2.3 Política para Conexões com outros Pares

Esta seção aborda aspectos em maior nível de detalhe de como um par interage com o tracker e como se dá a conexão deste com outros pares. No BitTorrent, a identidade de um par é informada através de uma string de 20 bytes. O par gera uma string única a cada vez que participa de um enxame. Se estiver conectado ao mesmo tempo com mais de um enxame, uma identidade diferente é usada em cada enxame. Esta identidade é informada em todas as interações com o tracker e aos pares remotos no início da conexão (mensagem **Handshake**), conforme já visto nas Figuras 2.2 e 2.5. Embora não haja uma regra para a geração de identidades, de acordo com [BitTorrentProtocolSpecification, 2007], há uma convenção de que o identificador deva ser gerado de forma pseudo-aleatória. Nas implementações de agentes[2] populares ([Azureus, 2007, Bittorrent, 2007, μTorrent, 2007]), parte dos bytes do identificador é utilizado para informar o tipo específico do agente e sua versão (por exemplo, um identificador iniciado por "AZ2500" representa o agente Azureus versão 2.5.0.0), sendo o resto completado com bits aleatórios.

Ao contatar o tracker, um par obtém a identidade, o endereço IP e a porta TCP de outros pares em um enxame. O tracker é, literalmente, um elemento central para a arquitetura BitTorrent. Pares que participam de um enxame periodicamente se conectam ao tracker e registram assim sua presença no enxame. Na conexão do par com o tracker, pode ser informado o número de pares que o par local deseja receber (**numwant**), que é por padrão 50.

Tipicamente, os seguintes campos são informados na conexão do par com o tracker [BitTorrentProtocolSpecification, 2007]:

- **info_hash**: hash do campo info do arquivo .torrent (o campo info possui, entre

[2]Preferiu-se o termo "agente" a "cliente" para indicar o software que implementa o protocolo e com o qual o usuário interage.

outras coisas, o hash das peças, o que impossibilita uma conexão com um o arquivo de metadados forjado);

- **peer_id**: identificador de 20 bytes do par, gerado de forma pseudo-aleatória;
- **ip**: opcional, informa o endereço IP no qual o par se encontra. Geralmente utilizado se o IP externo do par de origem é o mesmo IP externo do tracker, como quando os dois estão atrás de um mesmo NAT. Neste caso, para que sejam acessíveis a partir da Internet, ambos serão anunciados com o mesmo IP. Outra possibilidade é o agente estar acessando o tracker através de um servidor *proxy*;
- **port**: porta que o par está escutando por novas conexões;
- **uploaded**: quantidade de conteúdo fornecida a outros pares no enxame;
- **downloaded**: quantidade de dados obtida do enxame;
- **left**: quantidade de dados que ainda precisa ser obtida;
- **event**: o campo pode estar vazio ou conter um de três valores (*started*: primeira requisição ao tracker, *stopped*: agente está deixando o enxame "elegantemente", *completed*: agente tornou-se semeador);
- **numwant**: o número de pares que o par local deseja receber (se omitido, que é o padrão, 50 pares serão retornados).
- **key**: opcional, um identificador que só o par e o tracker conhecem. Pode ser utilizado para um par provar sua identidade caso seu endereço IP mude.
- **trackerid**: opcional. Se o par já se comunicou anteriormente com o tracker, o campo **trackerid** retornado naquela ocasião deve ser informado aqui.

Além da lista de pares, o tracker retorna uma série de informações, incluindo:

- **failure reason**: se aparecer, então todas as outras informações serão suprimidas. Indica o erro que impossibilitou que a solicitação fosse atendida. A mensagem vem em formato de texto na língua inglesa;
- **warning message:** similar ao item anterior, porém os demais campos são preenchidos corretamente;
- **interval**: tempo, em segundos, que o par deve esperar antes de realizar outra solicitação; esse valor também é utilizado para cálculo do tempo que o tracker manterá o IP do agente na lista de pares ativos. Se o tempo exceder este limite mais certa margem, o tracker considera que o par abandonou o enxame sem informar o mesmo. O par pode enviar solicitações em um tempo inferior ao anunciado em **interval** caso precise, mas não é considerada uma boa prática solicitações muito freqüentes;
- **min_interval**: tempo mínimo antes da próxima solicitação;
- **tracker id**: uma string que o agente deve enviar novamente na próxima requisição que fizer. Se o tracker não retornar nenhum valor, mas um anterior já existir, o par deve continuar utilizando o anterior;
- **complete**: número de semeadores presentes no enxame;
- **incomplete**: número de sugadores conectados ao enxame;
- **peers**: o valor é uma lista de dicionário, cada item possuindo os seguintes campos:
  - **peer id**: identificador do par remoto;
  - **ip**: endereço IP do par remoto (pode ser o endereço IPv4, IPv6 ou nome DNS);
  - **port**: a porta TCP na qual o agente está escutando por novas conexões.

O tamanho da lista de pares mantida por um tracker é o tamanho de um enxame: varia tipicamente de dois ou três pares até mais de 1.000.000. Quando

solicitado, o tracker monta aleatoriamente uma lista com até 50 pares (ou o valor solicitado em **numwant**) e envia-a ao par solicitante. Quanto maior a lista global no tracker, maior será a diversidade entre as listas fornecidas pelo mesmo; quando a lista global do tracker contém menos que a quantidade de pares a ser enviada, não há sorteio – basicamente a mesma lista será fornecida para cada novo par.

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 tracker, que responde com o **PeerList**. Segundo, quando a lista decresce abaixo de um limite inferior (tipicamente 20 pares), o par envia novo **Get** ao tracker. Terceiro, quando o par é contatado por outro par no enxame, sendo que o par remoto não constava da lista do par local. Além disso, existem outras extensões disponíveis, de acordo com a implementação do agente BitTorrent, e que podem influenciar esse processo (mais informações na seção seguinte).

2.4 Extensões ao Protocolo

Serão vistas nesta seção algumas extensões existentes ao protocolo BitTorrent, sendo apresentadas apenas as mais relacionadas com essa dissertação: (a) "Troca de Pares" (*Peer-Exchange* – PEX), (b) "SuperSemeadura" (*Superseeding*) e (c) Tabela Hash Distribuída (*Distributed Hash Table*, extensão também conhecida como *TrackerLess*). Destas, apenas a última é considerada uma extensão oficial do protocolo [BitTorrentDHTProtocol, 2007], o que faz com que seja suportada pela maioria dos agentes existentes. As demais são consideradas extensões não-oficiais, não fazendo parte da especificação original do protocolo [BitTorrentOrg, 2007], mas tendo sido implementadas pelos seus agentes mais populares [Azureus, 2007, μTorrent, 2007]. Estas extensões são, dentre as existentes, as que podem ser exploradas por um atacante ou influenciar o desempenho de um ataque, conforme é visto nas Seções 3.4 e 3.8.

Na primeira extensão, Troca de Pares, um par envia a outro informações sobre os pares com os quais está conectado e atualiza esta informação a cada nova conexão

e/ou desconexão. Este protocolo atualmente só funciona entre pares utilizando o mesmo agente, isto é, entre agentes do Azureus e entre agentes do μTorrent, mas não intercambiavelmente entre os dois. A extensão foi desenvolvida com a finalidade de permitir que os pares que já participam de um determinado enxame possam continuar a encontrar uns aos outros mesmo que o tracker fique, por algum motivo, inalcançável.

Já SuperSemeadura é uma técnica através da qual o semeador passa a escolher qual peça um par irá obter. Para tal, ao invés de o semeador anunciar que possui todas as peças na troca do Bitfield, ele "mente" e informa não possuir qualquer peça. Logo após, o semeador passa a enviar mensagens **Have**, simulando como se tivesse acabado de obter uma determinada peça que o par remoto ainda não possui. Isso faz com que o par remoto não tenha outra escolha a não ser solicitar a única peça que julga que o semeador possua e que ele não possui. Após o par remoto completar o download desta peça, uma mensagem **Have** com o identificador de outra peça que o par não possui será enviada pelo super-semeador. Este ciclo se repetirá, peça após peça, até que o par obtenha o arquivo completo ou que, por algum motivo, a conexão entre o par e o super-semeador seja encerrada.

A técnica de super-semear permitiu uma melhora significativa no espalhamento das peças [BitTorrentProtocolSpecification, 2007]. Sem ela, o semeador inicial pode chegar a enviar até 1,8 vezes a quantidade de dados que o conteúdo possui até a última peça ser distribuída. Isto se deve ao fato de que no início, por haver apenas uma cópia de cada peça, os pares realizam uma escolha aleatória e acabam, de tempos em tempos, coincidindo na escolha da mesma peça. Como conseqüência, a peça é enviada duas ou mais vezes e acaba por protelar o envio de outras peças que ainda não foram escolhidas.

Por fim, a extensão Tabela Hash Distribuída, ou TrackerLess, permite que pares atuem como trackers. Com ela, os pares possuem um identificador de 160 bits (**node_ID**); o campo **info_hash**, que também possui 160 bits e é calculado a partir do campo **info** do arquivo .torrent, é comparado com os identificadores dos pares.

Os pares que possuem os identificadores mais "próximos", de acordo com a métrica XOR utilizada pelo Kademlia [Maymounkov and Mazières, 2002], são os pares que atuarão como trackers do conteúdo. Mais de um par com identificador próximo ao do campo **info_hash** é mantido como tracker permanentemente. Isso evita que a saída de um determinado par do enxame faça com que o tracker de um conteúdo deixe de existir.

A rede de sobreposição utilizada é implementada sobre UDP. Pares mantém uma tabela de roteamento contendo os pares mais "próximos", que são utilizados como ponto inicial para buscas na infra-estrutura DHT. Um par, caso deseje obter uma lista de pares de um enxame, irá realizar sucessivas consultas, obtendo sempre identificadores de pares mais próximos ao identificador do campo **info_hash** desejado, até encontrar um ou mais pares que estejam atuando como trackers do conteúdo. Quando encontrá-los, uma mensagem **Get_Peers** é utilizada para obter uma lista de pares, de maneira similar à conexão com o tracker. Não há roteamento de mensagens entre os pares.

Capítulo 3

Estratégias de Subversão/Ataque

O capítulo anterior apresentou a arquitetura BitTorrent, detalhando-se componentes e a relação entre os mesmos. Este capítulo identifica uma série de ataques à arquitetura BitTorrent, explorando vulnerabilidades nela existentes.

Outros trabalhos que abordam segurança em redes P2P, mas não envolvem BitTorrent em específico, são os apresentados em [Wallach, 2002] e [Barcellos and Gaspary, 2006]. No primeiro, são discutidos aspectos relacionados à segurança das camadas de rede e aplicação, principalmente em sistemas P2P estruturados. Técnicas de roteamento redundante e criptografia são propostas como formas de melhorar a segurança dessas redes. O artigo discute questões de roteamento, atribuição de identificadores a pares e possibilidade de "ejeção" de pares da rede, bem como questões de inserção de objetos, justiça e auditoria distribuída.

No segundo trabalho, os autores apresentam um *survey* sobre questões relacionadas a redes P2P e segurança. Nele, é realizado um apanhado geral de resultados de pesquisa e tecnologias ligadas à segurança em redes P2P, amparado em exemplos provenientes de sistemas disponíveis e em operação. São abordados os principais aspectos de segurança, as vulnerabilidades a eles associadas e as implicações da utilização desses aspectos em redes P2P. Um conjunto de propostas já existentes para solucionar os principais problemas de segurança em redes P2P é descrito e problemas ainda em aberto são citados.

No caso de BitTorrent, conforme citado na Introdução, não há registros de

estudos anteriores que tenham pesquisado segurança nessa arquitetura. Como demonstra a Figura 2.1 (Seção 2.1), presente na página 15, a obtenção de um conteúdo pode ser dividida em três fases distintas.

Os ataques pertencentes à primeira fase (obtenção do arquivo .torrent), que envolvem ataques conhecidos à infra-estrutura de servidores *web*, não serão contemplados neste estudo por dois motivos principais: (a) a fase de obtenção do arquivo .torrent não faz uso do paradigma P2P, fugindo portanto ao escopo deste trabalho; e (b) a maioria desses ataques já se encontra documentada em outras fontes.

Os ataques Sybil (Seção 3.1), Eclipsando Pares Corretos (Seção 3.2) e Eclipse Dirigido (Seção 3.3) foram classificados como pertencentes às fases 2 e 3. Primeiramente, eles exploram a facilidade em obter identidades e "envenenam" o tracker (fase 2) para, em seguida, deflagrar uma ação maliciosa (fase 3). Foram identificados como pertencendo somente à fase 2 (obtenção, junto ao tracker, dos pares participantes do enxame), os ataques Criação de um Tracker Forjado (Seção 3.4) e Fornecer um Identificador Falso ao Tracker (Seção 3.5). Já os ataques passíveis de serem explorados apenas na fase 3 (troca de peças e mensagens com outros pares) são: Mentindo a Posse de Peças (Seção 3.6) e Envio de blocos errados (Seção 3.7). Por fim, a influência que as extensões Troca de Pares e Super-Semeadura podem ter nos ataques é comentada na Seção 3.8.

3.1 Sybil

O ataque denominado Sybil, em que um par apresenta-se com múltiplas identidades, é introduzido em [Douceur, 2002]. Neste ataque, um atacante aproveita-se da facilidade em adquirir novas identidades para obter um grande conjunto de identidades virtuais. A maioria dos sistemas P2P utiliza redundância como forma de obter proteção contra falhas em pares, sejam estas intencionais ou acidentais. Utilizando-se do ataque Sybil, um par malicioso pode comprometer o sistema ao controlar uma fração substancial do mesmo, anulando assim o efeito da redundância.

O artigo propõe o uso de autoridades certificadoras de identidade como a melhor forma de proteção contra esse tipo de ataque.

Trabalhos posteriores [Daswani and Molina, 2004, Sieka et al., 2004, Danezis et al., 2005, Bazzi and Konjevod, 2005, Yu et al., 2006, Borisov, 2006, Dinger and Hartenstein, 2006] estudaram os efeitos do ataque Sybil em diversas redes de sobreposição e propuseram diferentes soluções para a resolução do problema. Novamente, não há estudos nesta área envolvendo a arquitetura BitTorrent, tendo os trabalhos até agora desenvolvidos se concentrado, em sua maioria, na rede Gnutella ou em redes estruturadas, baseadas em Tabelas Hash Distribuídas.

Conforme visto na Seção 2.3, no BitTorrent as identidades são geradas de forma pseudo-aleatória e uma nova identidade é gerada cada vez que o par se conecta com um enxame. Para um par local iniciar uma conexão BitTorrent, ele envia uma mensagem de **Handshake** e compara a identidade informada pelo par remoto na resposta com a recebida do tracker (vide Seção 2.1). Se os valores não forem idênticos, a conexão é abortada. No entanto, quando um par local recebe um pedido de conexão de um par remoto, não há como o par local realizar esta verificação. Há duas razões para isso: primeiro, o par que recebe a conexão só conhece um universo limitado de participantes do enxame. Segundo, o tracker não possui um serviço que permita a um par realizar uma checagem da identidade de outro par.

Um atacante, portanto, pode valer-se desta vulnerabilidade e criar várias identidades. Para tal, um mesmo par pode executar um cliente malicioso que permita simular várias identidades diferentes utilizando portas TCP diferentes ou mesmo utilizando um conjunto de endereços IP. Na versão atual do protocolo BitTorrent, um atacante não precisa nem mesmo guardar a identidade que foi utilizada com um par remoto em uma transação anterior, basta gerar uma nova identidade a cada nova conexão.

Pares podem banir um par que se recusa a cooperar, mas nem sempre é fácil identificar quando um par não está cooperando ou quando enfrenta problemas de

conectividade, por exemplo. Alguns clientes permitem certos mecanismos de filtro por IP ou banir pares que estão enviando peças erradas, mas estes mecanismos são facilmente contornáveis ou exigem alguma espécie de intervenção manual.

Assumindo então que um atacante possa executar múltiplos pares compartilhando um mesmo recurso físico (máquina e rede), há uma série de ataques derivados que são potencialmente mais efetivos em BitTorrent. Os ataques descritos a seguir constituem, por si só, formas de subversão, mas todos têm o seu potencial multiplicado com a utilização de Sybil.

3.2 Eclipsando Pares Corretos

O ataque de Eclipse [Singh et al., 2004, Singh et al., 2006] consiste em cercar de maliciosos um determinado par correto, de forma que toda sua comunicação possa ser "interceptada", isolando ou levando o mesmo a crer que ele está corretamente conectado com o resto da rede. No extremo, se um atacante possui suficientes recursos físicos e cria um grande número de identidades, ele pode atacar um enxame ao fazer com que um bando de pares maliciosos participe do mesmo. Além disso, um único conjunto de pares maliciosos pode ser usado para atacar múltiplos enxames.

No contexto deste trabalho, o ataque Eclipse consiste em usar um número suficientemente alto de maliciosos, de forma a fazer com que pares corretos estejam conectados predominantemente ou apenas com pares maliciosos. Conforme visto no Capítulo 2, cada par correto inicia quando possível até 30 conexões simultâneas com outros pares. O número máximo de conexões que um par correto mantém é, por padrão, 55. Portanto, para eclipsar pares corretos são necessários, no mínimo, 55 pares maliciosos. Para ser bem sucedido nestas condições, o atacante terá que criar uma quantidade de identidades tal que, na grande maioria das vezes, somente os seus identificadores sejam escolhidos no sorteio realizado pelo tracker.

Ilustrando, seja ω o conjunto de pares honestos existentes em um enxame e υ o conjunto dos pares escolhidos e retornados a um par pelo tracker em um dado momento. Assumindo que $|\omega| = 101$, $|\upsilon| = 50$ e $\upsilon \subset \omega$, isto é, o enxame é composto

por 101 pares, dos quais 50 aleatórios são escolhidos a cada consulta ao tracker, a probabilidade de um par ser escolhido é $\rho = \frac{|\upsilon|}{|\omega|-1}$, ou seja $\rho = \frac{50}{101-1} = 0,5$: 50% de chance. A subtração de 1 par no denominador deve-se ao fato de que o tracker jamais inclui, no conjunto retornado, o próprio par solicitante, embora este faça parte do conjunto ω.

Assumindo agora que um atacante entre com 400 pares maliciosos em um enxame e que seja ν o conjunto destes pares maliciosos. Esta nova realidade faz com que, a cada vez que houver uma solicitação de uma lista de pares para o tracker, haja 10% de chance de um par ser escolhido ($\rho = \frac{|\upsilon|}{|\nu|+|\omega|-1}$, ou seja, $\rho = \frac{50}{101+400-1} = 0,1$). Se o atacante utilizar 4900 pares maliciosos, o valor de ρ cairia para 1% e assim sucessivamente. Embora o número de atacantes necessários possa ser relativamente alto (como 5000 maliciosos para um enxame de 100 pares), o tráfego gerado pelo par malicioso está restrito ao plano de controle, pois o mesmo não faz efetivamente download nem upload.

Aliado a isso, é necessário considerar duas outras questões: (a) um par só estabelece conexões com um subconjunto de pares da lista recebida do tracker e (b) se um par atingir o limite de conexões (tipicamente 55) permitidas em um mesmo enxame, ele recusará novos pedidos de conexão partindo de pares remotos. O primeiro aspecto faz com que as chances de um par ser escolhido diminuam. Dado que o par inicia tipicamente 30 conexões a partir das 50 identidades recebidas, a chance de um par que apareceu na lista ser escolhido é de $\rho = \frac{30}{50} = 0,6$, ou seja 60%.

O segundo ponto, no entanto, se bem explorado, permite ao atacante diminuir consideravelmente a quantidade de pares necessários em um ataque. Assumindo que λ é o conjunto de pares remotos com os quais um par local está conectado, ao receber a lista do tracker e iniciar suas conexões, a cardinalidade típica de λ será 30 ($|\lambda| = 30$). Assumindo que a cardinalidade máxima de λ seja 55 ($|\lambda| \leq 55$) e que o número de conexões não iniciadas pelo par seja definido pela variável λ', tem-se que tipicamente $\lambda' = 25$.

Para evitar novas conexões de pares honestos, um atacante precisa apenas "consumir" as 25 conexões restantes. Assim sendo, mesmo que o par honesto seja anunciado pelo tracker, ele não irá aceitar novos pedidos por já estar com o seu número máximo de conexões ($|\lambda| = 55$ e $\lambda' = 0$) estabelecido. Um segundo par honesto fica impedido, então, de se comunicar com o primeiro, tendo como conseqüência o isolamento dos dois pares honestos.

3.3 Eclipse Dirigido

Assumindo que o atacante possui restrições quanto a sua capacidade em termos de recursos, é interessante que ele utilize, da melhor forma, a capacidade que possui. Esta restrição é a principal característica que separa este ataque do descrito na seção anterior, onde um atacante possui recursos suficientes para atacar todos os pares honestos presentes no enxame. No caso atual, a atitude mais lógica que um atacante pode tomar é concentrar-se em um subconjunto dos pares existentes no enxame. A principal questão, então, se torna definir como será feita a escolha dos pares a serem atacados.

Tomando como premissa que o atacante possui conhecimento global a respeito dos pares do enxame, isto é, que já realizou pelo menos uma conexão com cada par, a fim de obter seu Bitfield, e que o estado de cada par não mudou desde então, três possibilidades parecem interessantes. Na primeira delas, o atacante irá focar primeiramente nos pares que estão semeando; na segunda, os pares mais rápidos; e na terceira, aqueles com as peças mais raras.

No primeiro caso, a vantagem de conectar-se com semeadores é o fato de estes serem os pares que possuem uma cópia completa do arquivo. Isso prejudicará o enxame de duas maneiras: primeiro, antes mesmo de um eclipse total ocorrer, os pares maliciosos já irão prejudicar o espalhamento das peças; segundo, é fundamental para o ataque que novos pares honestos não consigam conectar-se com os semeadores. Estando conectados, os pares maliciosos terão a possibilidade de realizar download a partir dos semeadores. Se a limitação do atacante residir exatamente na largura

de banda disponível, realizar download com certeza não será uma opção adequada. Caso contrário, o download de conteúdo prejudicaria ainda mais o enxame, pois consumiria a banda de upload dos semeadores.

Para que novos pares não consigam conectar-se com os semeadores, as conexões restantes destes últimos devem ser tomadas pelos pares maliciosos. Em seguida, assumindo que de tempos em tempos os pares honestos que estavam conectados com o semeador fecharão sua conexão com este, por terem tornado-se semeadores também ou por deixarem o enxame, o atacante irá, ao longo do tempo, aumentando o número de conexões com os semeadores. Considerando que, em algum momento, os semeadores ficarão eclipsados e que no futuro deixarão o enxame, este ataque pode, no seu extremo, ocasionar a morte prematura de um enxame caso alguma peça torne-se indisponível. Outra técnica que pode tornar o ataque mais efetivo é viável caso os pares utilizem a extensão do protocolo BitTorrent chamada Peer Exchange. Esta extensão permite que um par informe a outro a sua lista de pares remotos. Se o atacante conectar-se com os outros pares com os quais um semeador está fornecendo peças, este saberá quando os outros pares desconectarem-se do semeador, facilitando então a tarefa de consumir estas conexões.

A segunda estratégia que pode ser adotada é a de tentar eclipsar primeiramente os pares mais rápidos. Por rapidez, neste caso, considera-se a largura de banda de upload do par. Conectando-se com os pares com maior largura de banda, num primeiro momento o atacante fará com que o espalhamento das peças ocorra mais lentamente se parte do upload dos pares rápidos for consumido no ataque. Na impossibilidade disso ocorrer, devido a limitações na velocidade de download do atacante, quando o eclipse total dos pares mais rápidos se concretizar haverá um atraso no espalhamento global das peças.

A última estratégia, conectar-se com os pares com as peças mais raras e eclipsá-los, visa, primeiramente, fazer com que estas peças tornem-se indisponíveis mais rapidamente. Esta técnica pode ser considerada como um avanço (ou ampliação) do ataque para eclipsar somente semeadores. Como os semeadores possuem todas

as peças, logicamente eles estarão entre os pares que possuem as peças menos replicadas. Um atacante pode, então, iniciar seu ataque pelos semeadores e utilizar os pares com as peças menos replicadas em seguida, realizando assim um ataque "em profundidade" até o esgotamento de algum de seus recursos.

Eliminando a premissa que o par possui conhecimento global, a estratégia que se mostra mais plausível é o atacante ir direcionando seu ataque na medida em que vai "conhecendo" mais os pares, isto é, recebendo mais informações sobre os mesmos e acompanhando suas evoluções. Neste caso, os pares seriam ranqueados segundo determinado critério e, na medida em que o ataque fosse evoluindo, o atacante iria eliminando os pares "menos interessantes" e focando-se nos que poderiam causar mais prejuízo ao coletivo.

A escolha do critério deve ser baseada na limitação que o atacante possui. Acredita-se que a estratégia que irá provocar maior prejuízo é àquela na qual o atacante classifica e ataca os pares de acordo com o seguinte critério: primeiro, pares semeadores; segundo, pares com as peças mais raras; terceiro, pares mais rápidos; por fim, pares com maior quantidade de peças obtidas, em ordem decrescente.

3.4 Criação de um Tracker Forjado

Conforme visto na Seção 2.3, ao conectar-se com o tracker, o par envia uma série de informações a este e recebe, em contrapartida, outro conjunto de dados. O tracker identifica a qual enxame um par está se referenciando através do campo **info_hash**, gerado através da aplicação do hash SHA-1 ao campo **info** presente no arquivo .torrent. Entretanto, não é no campo **info** que a informação do tracker (presente no .torrent) fica armazenada. Isso faz com que esta informação possa ser alterada sem que os pares detectem que estão conectando-se a trackers diferentes. Na verdade, este é um comportamento desejado, por dois motivos principais: primeiro, caso o tracker original seja retirado do ar, não é necessário que todos os pares recalculem o campo **info_hash**; segundo, mais de um tracker pode ser definido sem que os pares tenham todos de possuir conexão com os mesmos trackers.

Um atacante, portanto, pode utilizar-se desta característica de forma a criar um tracker forjado, modificar o arquivo .torrent e publicá-lo novamente em outro endereço. Forjado, neste caso, porque sua finalidade não é permitir que os pares honestos encontrem-se. Com o controle do tracker, um atacante pode segmentar um enxame, fornecer identificadores incorretos de pares ou mesmo contribuir para que um par não consiga sair de um ataque Eclipse.

Como o atacante controla o novo tracker, os pares que obtiverem o torrent alterado só se conectarão com quem o atacante desejar. O tracker forjado poderia realizar conexões com o tracker real, a fim de obter uma relação de pares honestos, visando assim não sufocar completamente o download de pares "sob a sua supervisão" e diminuindo as chances de o usuário remoto procurar outro arquivo .torrent para o mesmo conteúdo.

Para utilizar um tracker forjado, o atacante precisa arcar com os custos de mantê-lo funcionando. Embora isso demande o uso de seus recursos, esta estratégia permite que se controle exatamente as informações que um par vai receber. Com um controle maior sobre as conexões de um par, quem está realizando o ataque tem bem mais domínio sobre os seus alvos, assim como pode explorar melhor diferentes cenários.

Por fim, duas extensões existentes ao protocolo – Peer Exchange e Distributed Hash Table (DHT), descritas na Seção 2.4 – podem influenciar este ataque. Na primeira, um par honesto utiliza a extensão para obter a lista de pares dos pares com os quais está conectado. Se o par correto estiver conectado somente com pares maliciosos, ainda sim, o atacante mantém o controle sobre as conexões do mesmo. Mas se, ao contrário, o par sendo atacado estiver conectado com um ou mais pares honestos, a extensão pode ocasionar a descoberta de outros pares honestos.

O uso da Troca de Pares não pode, no entanto, efetivamente evitar o ataque Criação de um Tracker Forjado, pois a troca de informações de pares só é realizada num segundo momento, quando o par local já conectou com o tracker, obteve uma lista de pares e realizou conexões com um (sub)grupo desta lista. Controlando o

tracker, o atacante ainda pode fornecer somente referências a pares maliciosos, que não fornecerão listas de pares fidedignas.

Da mesma maneira, a extensão DHT, também conhecida como Trackerless, não pode evitar este tipo de ataque, ao contrário, pode até facilitá-lo. Por ser uma extensão oficial ela é bastante suportada pelos agentes BitTorrent. Nela, os pares passam a atuar como trackers descentralizados e uma DHT é utilizada como forma de organização de uma rede de sobreposição, tendo como base o sistema Kademlia [Maymounkov and Mazières, 2002].

Para controlar o tracker DHT, uma entidade maliciosa terá de utilizar-se desta rede de sobreposição para conseguir certa quantidade de identidades com identificadores próximos aos do identificador do torrent sendo atacado. Como os identificadores são escolhidos ao acaso, é relativamente simples para um atacante controlar uma fração desejada do espaço de identificadores existentes na rede. Assim, facilmente um atacante pode passar a atuar como tracker de um determinado conteúdo. Como esta extensão permite que pares atuem como o "trackers legítimos" na rede DHT, se pares maliciosos conseguirem tirar o tracker original do ar, passarão a controlar a entrada e a descoberta de novos pares no enxame.

3.5 Fornecer um Identificador Falso ao Tracker

Quando um par conecta-se com o tracker, entre os campos informados, a troca de mensagens inclui um campo opcional chamado **ip** (vide Seção 2.3). Este campo permite informar outro endereço IP que não o utilizado para a conexão com o tracker. Ele foi criado para: (a) permitir que um par realize acesso via proxy e registre o seu endereço IP ao invés do endereço do servidor proxy que está acessando o tracker; e (b) para lidar com situações em que o tracker e o par estão atrás de um mesmo NAT, mas o par não deseja que o tracker registre seu endereço IP reservado.

Um atacante pode valer-se desse campo para adicionar novos pares com endereço IP e porta TCP iguais aos de pares já existentes, mas com identificadores falsos. A vantagem desta abordagem é que ela praticamente não consumirá recursos

do atacante, se comparada com as outras abordagens propostas neste capítulo. Quando o tracker informa as tuplas contendo o identificador, o endereço IP e a porta TCP de diversos pares, o par local estabelece conexões com parte desta lista e verifica se os identificadores retornados pelos pares remotos (mensagem **Handshake**) são idênticos. Se forem diferentes, a conexão é abortada.

Portanto, um atacante poderia realizar várias consultas ao tracker, a fim de obter as tuplas de vários pares honestos e, a partir daí, passar a gerar uma série de identificadores falsos com os campos endereço IP e porta TCP de pares verdadeiros. Novos pares honestos que entrem no enxame tentarão conectar-se com os pares informados, mas, ao receber identificadores diferentes, irão finalizar suas conexões com os mesmos. Um número grande de erros pode acabar fazendo com que alguns clientes, que possuem componentes adicionais de segurança, acabem por banir um IP honesto, por considerá-lo malicioso.

3.6 Mentindo a Posse de Peças

Conforme discutido no Capítulo 2, pares utilizam dois tipos de mensagens para informar a pares remotos sobre a posse de peças. Logo após o estabelecimento da conexão TCP, pares trocam mensagens **Bitfield** contendo um mapa das peças que cada um possui. Após isso, a cada nova peça obtida, o par envia mensagens **Have**, notificando os pares conectados para que atualizem o mapa enviado inicialmente.

Normalmente, os pares buscam uma homogeneidade na quantidade de cópias de cada peça. O ataque Mentindo a Posse de Peças visa reduzir esta homogeneidade. Para tal, um par malicioso irá violar a execução correta do protocolo e anunciar uma peça que não possui, objetivando interferir na escolha da próxima peça pelos outros pares. A interferência ocorre porque, ao anunciar uma peça, o par malicioso está artificialmente aumentando o nível de replicação da mesma e, portanto, induzindo pares corretos a fazer download de outras peças primeiro. Em sua máxima eficiência, o ataque consiste em fazer com que peças se tornem raras a ponto de desaparecer do sistema, decretando assim a morte do enxame.

Vários aspectos são pontos-chave para o atacante, isto é, podem influenciar diretamente a efetividade de um ataque. Os principais são: (a) número de peças a serem mentidas, (b) tamanho do conteúdo, (c) grau de disseminação das peças, (d) quantidade de semeadores, (e) tamanho e número de peças, (f) tamanho do enxame e (g) número de pares maliciosos.

Determinar primeiramente quantas e quais peças mentir são as duas escolhas mais importantes que um atacante terá de fazer ao mentir a posse de peças. Teoricamente, mentir a posse de duas peças é melhor que uma, pois isso aumenta as chances de uma peça desaparecer do enxame. No entanto, mentir 100% das peças não seria em nada efetivo pois reequilibraria os valores de replicação e as escolhas seriam as mesmas de um cenário sem mentiras. A escolha da peça a ser mentida deve ser tal que maximize as chances de seu desaparecimento. A escolha da peça menos replicada parece ser uma decisão natural, mas como os pares possuem uma visão parcial do enxame, esta nem sempre será vista como sendo a menos replicada e pode não ser sempre a melhor escolha.

A realização de um ataque de mentira de peças depende também do segundo aspecto apontado, tamanho do conteúdo que está sendo disseminado, pois para um conteúdo pequeno é pouco provável que este tipo de ataque tenha efetividade. Em arquivos pequenos, normalmente há uma maior quantidade de semeadores [Andrade et al., 2005] e mesmo que o ataque provoque algum atraso, este tende a ser mínimo. Já em arquivos grandes (com mais de 100MB) que, segundo [Stutzbach and Rejaie, 2006], são a maioria em redes BitTorrent, este ataque tende a ser bem mais efetivo, mas exige em contrapartida mais esforço do atacante, pois este terá de permanecer mais tempo no enxame e terá mais recursos consumidos no ataque.

O grau de disseminação das peças constitui outro fator que deve ser levado em consideração. No estágio inicial de um enxame, quando nem todas as peças foram fornecidas pelo semeador inicial, anunciar essas peças pode fazer com que a sua escolha seja protelada, provocando um "trancamento" dos pares ao final do

download, por estes últimos ficarem todos dependentes de peças raras, possivelmente presentes apenas no semeador inicial. Em enxames com as peças bem replicadas, este efeito necessitará de mais tempo para ocorrer. Já em enxames em que algumas peças estão mais replicadas que outras, mentir as peças menos replicadas ajudará a aumentar a heterogeneidade na distribuição das mesmas.

A quantidade de semeadores, influenciada pelas características do enxame e pelo tamanho do arquivo, possui grande impacto na efetividade de um ataque. Em um enxame onde há vários semeadores, o ataque será menos efetivo se comparado a um ataque realizado sobre um cenário no qual os pares deixam o enxame pouco tempo após completar o download. Neste último caso, por exemplo, pares maliciosos poderiam anunciar falsamente a posse de peças que são as mais raras entre os pares com menos peças. Isso faria com que a escolha dessas peças fosse adiada para um momento posterior à saída dos pares com mais peças do enxame.

O tamanho e número de peças definidas em um arquivo torrent afetam a escolha do número de peças a serem mentidas e no tempo que o ataque precisa ser mantido para que possa ser bem sucedido. O tamanho da peça é também fator determinante na escolha da necessidade do par em fornecer blocos incorretos das peças sendo anunciadas, conforme será visto na Seção 3.7.

Por último, outro aspecto que deve ser considerado é o tamanho do enxame, pois quanto mais pares honestos houver, maior será a quantidade de atacantes ou de conexões que cada atacante terá de manter. O tamanho do enxame, portanto, pode servir de base para o atacante calcular o número de pares maliciosos necessários para que o ataque tenha êxito. Estes dois valores possuem uma relação diretamente proporcional, isto é, um enxame maior necessitará de mais pares maliciosos para alcançar o mesmo nível de prejuízo.

Num ataque, a combinação desses fatores deve ser, sempre que possível, reavaliada. Uma alteração no tamanho do enxame ou no número de pares maliciosos, por exemplo, pode demandar uma mudança de estratégia por parte do atacante. No primeiro caso, a entrada ou saída de sugadores irá interferir no grau de replicação

de algumas peças. Isso poderá provocar o aparecimento de novas peças raras, que o atacante poderá passar a mentir, ou poderá fazer com que uma peça mentida deixe de tornar-se pouco distribuída, obrigando também o atacante a redefinir seu alvo. No segundo caso, supondo um ataque distribuído e coordenado, a mudança no número de pares maliciosos também terá influência na efetividade do ataque e poderá ocasionar uma modificação do comportamento dos pares desonestos.

Como pares maliciosos não desejam entregar uma peça sobre a qual estão mentindo, três estratégias são possíveis: (a) manter sempre os outros pares como sufocados; (b) dessufocar um par mas não responder caso um par solicite a peça (ou conjunto de peças) sendo anunciada falsamente, ou (c) enviar lentamente blocos corrompidos, mas não a peça inteira. No primeiro caso, o par malicioso nunca realizará upload para os pares conectados, mas, assim como qualquer par correto, ele poderá ser selecionado para receber conteúdo, via dessufocamento otimista. No caso de não responder a requisições, o dano aumenta, porque pares maliciosos terão informado pares remotos que estes podem solicitar download, mas o par malicioso jamais responderá às solicitações, levando a timeouts no par correto. Já o terceiro caso é o mais doloso, entre as três opções, porque os blocos corrompidos farão com que largura de banda do par correto seja ocupada e provocarão um reenvio da peça inteira, obrigando a retransmissão inclusive dos blocos que estavam corretos. No entanto, neste caso, o par malicioso precisa arcar com mínima, porém suficiente, largura de banda para garantir o download nos pares atacados. Além disso, fornecer todos os blocos de uma peça ao par receptor permite ao mesmo detectar que a peça foi enviada corrompida pelo par malicioso; com isso, o par correto pode decidir não interagir mais com o par malicioso.

3.7 Envio de Blocos Errados

Conforme citado no final da seção anterior, um par malicioso pode optar por enviar um ou mais blocos corrompidos de uma peça. Embora o ataque de enviar blocos errados de uma peça, quando combinado com os ataques de Sybil (Seção

3.1) e Mentir a Posse de Peças (Seção 3.6), tenha o seu impacto ampliado de forma significativa, este ataque por si só já possui um bom grau de prejuízo.

Como a arquitetura BitTorrent possui proteção contra tentativas de adulteração do conteúdo, a peça com os blocos incorretos não será aceita pelo par realizando o download, sendo a mesma descartada e o seu download realizado novamente. 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 o hash presente no arquivo .torrent. Uma das propriedades da função de hash SHA-1, assim como das funções hash em geral, é ser uma função unidirecional ("*One Way*"). 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" ocasionando uma inversão média de aproximadamente 50% dos bits resultantes da função [Wang, 2006].

Portanto, não é possível a um par correto, ao calcular o hash de uma peça e este não ser igual ao anunciado no arquivo .torrent, descobrir qual bloco (ou blocos) está(ão) incorreto(s). Como resultado, o par tem que descartar a peça inteira e realizar o download de todos os seus blocos 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 do torrent.

Alguns agentes 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 honesto 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 enviado anteriormente já ter sido descartado, pois o hash da peça falhou.

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.

3.8 Influência das Extensões ao Protocolo

Esta seção aborda a influência, em ataques, de duas extensões existentes ao protocolo BitTorrent: (a) Peer-Exchange (PEX) e (b) Superseeding. Destas, nenhuma é considerada uma extensão oficial, nem faz parte da especificação original do protocolo [BitTorrentOrg, 2007], mas foram implementadas pelos seus agentes mais populares e são as que podem ser exploradas por um atacante.

Na primeira delas, um par envia a outro informações sobre os pares com os quais está conectado e atualiza esta informação a cada nova conexão e/ou desconexão. A troca da lista de pares entre clientes permite que um atacante:

(a) informe uma lista de pares maliciosos, sem depender da necessidade de anunciar várias identidades no tracker e (b) coletar uma série de informações úteis sobre os seus alvos. Isso permite que o ataque seja mais bem direcionado, pois um par malicioso pode saber exatamente os pares remotos com os quais um par honesto está conectado. Adicionalmente, este mesmo par malicioso receberá atualizações sempre que o par honesto realizar uma nova conexão ou desconexão.

Já a técnica de Super-semear pode tornar o ataque Mentindo a Posse de Peças ainda mais efetivo. Na escolha da próxima peça que será anunciada via mensagem **Have** a um par, o super-semeador escolhe um peça ainda não fornecida nenhuma vez, caso exista alguma, ou a mais rara, exatamente como na política da "Peça Local Mais Rara Primeiro" (LRF), vista na Seção 2.2. Neste último caso, se um ataque de mentira de peças for orquestrado, o super-semeador também não irá fornecer as peças sendo mentidas, por julgar que elas já estão devidamente replicadas. Com isso, essas peças se tornarão ainda mais raras, dado que os pares que com certeza a possuem (semeadores) não a distribuirão.

Capítulo 4

Estratégias de Contramedidas

O presente capítulo discute possíveis ações a serem tomadas para reduzir, ou mesmo eliminar, os danos causados pelos ataques identificados no Capítulo 3. A sua organização espelha aquela empregada no capítulo anterior, visando assim facilitar a associação entre a vulnerabilidade identificada e a proposta de contramedida correspondente. Em função dessa estrutura, algumas contramedidas aparecem em mais de uma seção, porque servem como defesa a mais de um ataque.

4.1 Sybil: Gerenciamento de Identidades

O problema de Sybil em redes P2P foi objeto de vários estudos nos últimos anos [Douceur, 2002, Danezis et al., 2005, Yu et al., 2006, Bazzi and Konjevod, 2005, Borisov, 2006, Dinger and Hartenstein, 2006]. Nesses estudos, várias estratégias de proteção contra a possibilidade de um par utilizar múltiplas identidades em outros sistemas P2P foram propostas e testadas, sendo algumas delas implementadas. No contexto de BitTorrent, entretanto, não existem propostas de modificações na arquitetura com o objetivo específico de evitar o ataque de Sybil. Porém, no sentido de deixar BitTorrent mais robusto em relação a identidades e controlar o uso justo de recursos, é possível valer-se do esquema de autenticação disponível no protocolo HTTPS, baseado em criptografia de chave pública, para forçar a identificação de usuários junto ao tracker.

Neste esquema, após gerar um par de chaves, o usuário deve associá-lo a um endereço de e-mail em um servidor web que será responsável pelo serviço. Essa estratégia não evita um ataque Sybil, mas dificulta a criação de um número arbitrário de identidades. Assume-se aqui que o servidor terá proteções contra robôs, como, por exemplo, a obrigatoriedade de o usuário informar os valores que aparecem em uma imagem, e que não são reconhecidos por programas de reconhecimento ótico de caracteres (OCR).

O tracker atua também como uma Autoridade Certificadora (CA). Após autenticar o usuário, ele emite um certificado, válido apenas para a tupla contendo o identificador do enxame e do par (campos **info_hash** e **peer_id**, vide Seção 2.3). Esta abordagem evita que um par, ao contatar com o tracker utilizando uma identidade, tente aproveitar as informações recebidas e utilizar-las em outros identificadores.

Quando dois pares forem estabelecer uma conexão, eles trocarão (através da mensagem **Handshake**) os certificados emitidos pelo tracker. Assumindo que ambos os pares obtiveram a chave pública do tracker de forma segura, eles poderão verificar se a identidade apresentada pelo par foi realmente autenticada junto ao tracker. Esse certificado também deverá ser apresentado ao tracker quando o par necessitar novamente contatá-lo (vide Seção 2.1).

Como vantagens, a utilização de uma Infra-Estrutura de Chaves Públicas (ICP) apresenta-se como uma solução já madura e consolidada, o que facilita a sua adoção no contexto do BitTorrent. Além disso, os usuários poderiam utilizar-se desta infra-estrutura para que todo o tráfego BitTorrent fosse trocado de forma segura. Como desvantagens, a solução proposta implica: (a) a necessidade de um mecanismo de identificação de usuários; (b) uma maior sobrecarga ao tracker, que passa a gerar certificados; e (c) a modificação das extensões Troca de Pares e DHT, apresentadas na Seção 2.4, que precisariam ser adaptadas.

4.2 Eclipsando Pares Corretos: Rotação de Pares

O ataque Eclipse acontece quando um par encontra-se conectado somente com pares maliciosos, ficando impedido de trocar dados com pares honestos (eclipsado). Um dos aspectos que torna o BitTorrent vulnerável a ataques do tipo Eclipse é o fato de que, uma vez estabelecida uma conexão com um par, esta será mantida até que uma das duas partes resolva sair do enxame ou que algum erro ocorra. A estratégia de contramedida explorada nesta seção faz com que um par honesto avalie os pares remotos que mais contribuíram com conteúdo não apenas com vistas ao sufocamento/dessufocamento, mas também para decidir a escolha de pares com os quais ele permanecerá conectado.

O algoritmo de incentivo (tit-for-tat) empregado no BitTorrent faz com que o par forneça conteúdo (realize upload) para os pares que mais contribuíram para ele nas rodadas anteriores. Aliado a isso, o dessufocamento otimista (optimistic unchoke) tem como uma de suas funções auxiliar um par a descobrir pares remotos que possam oferecer taxas de contribuição ainda melhores. Utilizando-se do princípio de reciprocidade do tit-for-tat – privilegiar os pares remotos que mais contribuíram – propõe-se que um par avalie também, periodicamente, a permanência ou não de um par na sua lista de pares ativos.

O esquema proposto é denominado "Rotação de Pares" e consiste em um par realizar, periodicamente, uma avaliação de quais foram os pares remotos que menos trocaram dados com ele e desconectar-se destes. Sugere-se que o par se desconecte de pares que menos contribuíram apenas em situações em que existem outros pares com os quais o local poderia se conectar. A periodicidade da avaliação pode ser definida em rodadas: sendo ψ a quantidade de rodadas que um par irá computar antes de realizar sua avaliação, o valor sugerido para ψ é 180. Como a duração de cada rodada é, em média, 10 segundos, o par fará uma avaliação a cada 30 minutos. Durante a avaliação, o par elege os τ pares que menos contribuíram e encerra sua conexão TCP com estes. O valor sugerido para τ é 25.

A escolha dos valores sugeridos de ψ e τ segue o seguinte raciocínio: o valor

$\tau = 20$ é a diferença entre o número máximo de conexões que um par pode realizar (tipicamente 55) e o número máximo de conexões que o mesmo irá iniciar (tipicamente 30). Eliminando 25 conexões o par retorna à mesma quantidade de conexões estabelecidas ao entrar no enxame, assumindo que o enxame possua mais de 30 pares no momento da entrada. Agindo assim, o par, pelo menos em teoria, não estaria prejudicando seu desempenho, dado que 30 é o número de conexões que ele próprio (o par) busca realizar inicialmente.

Para a escolha do valor de ψ foi considerado o tempo aproximado necessário para que cada par pudesse ter a chance de ser eleito para dessufocamento otimista pelo menos uma vez. Como o tempo de dessufocamento otimista dura tipicamente 3 rodadas, é esperado que um total de 60 dessufocamentos otimistas tenha ocorrido num intervalo de 30 minutos. A escolha deste valor também se deve ao fato de adicionar uma sobrecarga mínima ao par.

Conforme visto na Seção 2.1, existe no BitTorrent um mecanismo chamado *anti-snubbing*, que permite que um par viole temporariamente o limite de 1 dessufocamento otimista, para que possa recuperar-se mais rapidamente da situação em que acaba sendo sufocado por todos os pares remotos. A fim de fazer com que um par eclipsado também se recupere mais rapidamente de um ataque Eclipse, sugere-se uma solução parecida, a ser utilizada no caso de um par não receber contribuição nenhuma, em termos de blocos, de todos (ou da maioria) os pares com os quais está conectado.

Assim, caso não receba dados de pelo menos um número mínimo de pares ϕ, em um dado intervalo de φ rodadas e esteja com todas as suas conexões estabelecidas, um par poderá encerrar conexões antes do tempo previsto em ψ. Sugere-se como valores para ϕ e φ, respectivamente, 5 e 30. Assim, se um par não receber contribuição de pelo menos 5 pares num intervalo de 5 minutos (30 rodadas de 10 segundos), ele assumirá que o tempo de ψ expirou e iniciará o processo de avaliação, término e início de novas conexões.

Para evitar que um atacante tente aproveitar-se desta estratégia e consumir os

τ recursos liberados após uma avaliação, o par pode optar por permitir que apenas $\frac{\tau}{2}$ sejam iniciadas por pares remotos. Com isso o par sempre teria de iniciar pelo menos metade das conexões liberadas, diminuindo a chance de um atacante consumir novamente todos os recursos liberados.

4.3 Eclipse Dirigido: Rotação de Pares

Para ataques eclipse em que o atacante passa a concentrar-se em pares com certas características, o uso da estratégia de Rotação de Pares, proposta na Seção 4.2, também pode ser utilizada como forma de contramedida. A Seção 3.3 apresentou três formas de se realizar um ataque de Eclipse Dirigido: (a) foco nos pares semeadores, (b) foco nos pares mais rápidos e (c) foco nos pares com as peças mais raras. Esta seção mostra, para cada caso, como a Rotação de Pares influenciaria negativamente o ataque.

Para ataques dirigidos aos pares semeadores, os pares podem eliminar, a cada ψ rodadas, os pares para os quais realizaram menos upload, dado que semeadores não realizam download. Esta estratégia obriga um atacante a ter de utilizar seu canal de entrada para evitar ser excluído da lista de pares ativos do semeador. Para um enxame com um número significativo de semeadores, o download de uma grande quantidade deles tornar-se-ia rapidamente impraticável. Mesmo para o cenário de um enxame com poucos semeadores, um atacante teria mais dificuldades em manter um ataque eclipse, pois de tempos em tempos parte de suas conexões com o par atacado seriam desfeitas, obrigando-o a, periodicamente, ter de disputá-las novamente, a fim de consumir os recursos de seus alvos.

No caso de ataque aos pares rápidos, a estratégia de Rotação de Pares é ainda mais efetiva, pois obriga um atacante a manter altas taxas de download, possivelmente exaurindo seus recursos. Novamente, mesmo que o consiga, o atacante terá de periodicamente competir pelos recursos dos pares atacados.

Por último, no ataque direcionado aos pares com as peças mais raras, a abordagem proposta auxilia no espalhamento das peças menos replicadas. Quando

um par com peças raras realizar a rotação e conectar-se com pares honestos, estas peças serão, provavelmente, as primeiras a serem solicitadas. Se novos pares obtiverem as peças mais raras, o atacante terá de aumentar a quantidade de pares sobre os quais está direcionando seu ataque, o que também levará a uma exaustão de seus recursos.

4.4 Criação de um Tracker Forjado: Troca de Informações entre Pares

Para o ataque do tracker forjado, apresentado na Seção 3.4, a contramedida sugerida é a troca, entre os pares, da informação de qual, ou quais, tracker(s) cada um está utilizando. Esta troca objetiva, num primeiro momento, apenas o registro por parte dos pares, da possível existência de outros trackers gerenciando o enxame. Um par permanecerá utilizando o mesmo tracker, ou conjunto destes, independentemente das informações recebidas de outros pares. Somente se a taxa de download ficar abaixo de um determinado limite, considerado impraticável pelo par, é que este pode optar por tentar a conexão com novos trackers. Esta estratégia visa evitar um possível efeito colateral que a troca da informação de trackers poderia ocasionar: um atacante criar um tracker forjado e informá-lo a outros pares através do método aqui sendo proposto.

4.5 Fornecer um Identificador Falso ao Tracker: Validação pelo Tracker

Uma contramedida sugerida para o ataque apresentado na Seção 3.5 é o tracker não aceitar que um mesmo endereço IP e porta TCP possuam mais de um identificador BitTorrent. Como o atacante não tem como saber de antemão qual porta um determinado par irá utilizar antes de este efetivamente entrar no enxame, a segunda tentativa de registrar um identificador BitTorrent diferente, mas com um

endereço IP e porta TCP já existentes, deve ser negada por parte do tracker, que retornará, então, uma condição de erro.

4.6 Mentindo a Posse de Peças: Rotação de Pares

A contramedida para o ataque de mentira de peças pode valer-se da Rotação de Pares proposta na Seção 4.2. Para este ataque em específico, propõe-se uma modificação na forma de avaliação dos pares a serem "descartados". O par, ao invés de levar apenas em consideração a contribuição, pode optar por descartar pares com Bitfield muito parecido. Novamente, a rotação só deverá ocorrer caso todas as conexões do par estejam estabelecidas e existam outros pares no enxame.

No início de um enxame, até que todas as peças sejam disseminadas pelo semeador inicial, é provável que a situação de pares com Bitfields parecidos seja mais comum. No entanto, mesmo nestes casos, a Rotação de Pares não representa prejuízo a um par correto, pois este estará desconectando-se dos pares com Bitfields parecidos, isto é, ainda restarão outros pares com as mesmas peças. Adicionalmente, propõe-se que a contribuição e a homogeneidade de Bitfield sejam avaliadas em conjunto, ou seja, um par não é excluído apenas por possuir um Bitfield parecido com outro, mas também por não ter contribuído com o par local.

4.7 Envio de Blocos Errados: Detecção de Pares Maliciosos

Para o ataque de envio de blocos errados, visto na Seção 3.7, uma contramedida possível é o par tentar identificar os pares que estão agindo de forma maliciosa. Para isso, propõe-se que o par não descarte por completo uma peça quando o seu hash não for idêntico ao do arquivo .torrent.

A fim de identificar pares remotos maliciosos, as seguintes ações são sugeridas: primeiro, o par deverá manter estruturas de dados capazes de armazenar qual par enviou determinado bloco de uma peça. Segundo, se, ao receber todos os blocos, a comparação do hash da peça falhar, o par local não deve descartar esses blocos,

mas sim realizar novas solicitações dos mesmos para um conjunto de pares remotos diferente dos da requisição anterior. Preferencialmente, deve ser escolhido um conjunto de completamente distinto de pares. Terceiro, quando receber novamente os blocos, o par local deve realizar uma comparação entre o conteúdo recebido no primeiro passo e o recém recebido, de maneira a identificar se há discrepâncias. Quarto, quando houver diferenças, uma nova verificação do hash deve ser realizada. Se a comparação mostrar que a peça com o novo bloco está íntegra, o par remoto, que enviou o bloco incorreto no primeiro passo, deve ser adicionado a uma lista de suspeitos. Caso a peça continue inconsistente, este processo deverá ser repetido até que todos os blocos corretos tenham sido recebidos. Por último, após o recebimento de uma determinada quantidade de blocos incorretos de um mesmo par remoto, este último deve ser marcado como malicioso e colocado em uma lista negra pelo par local.

4.8 Ataques com Extensões

Na Seção 3.8 foram vistas as influências nos ataques de duas extensões presentes em alguns agentes BitTorrent. No primeiro, a extensão Peer Exchange é utilizada e, no segundo, o atacante visa aproveitar-se da idéia de SuperSeeding. Para cada caso, uma contramedida é aqui proposta a fim de diminuir ainda mais a efetividade de um ataque.

No caso da troca de pares (Peer Exchange), a contramedida proposta é o par realizar parte das conexões a partir das informações recebidas do tracker e a outra parte com os dados recebidos via Peer Exchange. Isso aumenta a chance de um par conectar-se com pares remotos honestos, dado que o atacante teria de comprometer os dois sistemas caso desejasse causar um dano mais significativo ao enxame.

Já a utilização da técnica conhecida como SuperSeeding permite que um semeador melhore o espalhamento das peças, mas poderia ser explorada por um atacante em conjunto com o ataque Mentindo a Posse de Peças (Seção 3.6) a fim de fazer com que os semeadores deixassem de anunciar um determinado conjunto de

peças. Nesse caso, a contramedida proposta é o semeador continuar anunciando mais freqüentemente as peças tidas como mais raras, mas não deixar que a freqüência de anúncio de uma peça qualquer fique abaixo de um limite inferior. O limite proposto é de $\frac{1}{3}$, isto é, uma determinada peça não pode ter sido anunciada mais de 3 vezes, em comparação com a quantidade de anúncios de qualquer outra peça.

Neste caso, um contador seria associado a cada peça e o "supersemeador" não poderia deixar que o anúncio de peças tidas como altamente distribuídas fosse postergado indefinidamente. Isto significa dizer que, para uma peça poder ser anunciada uma quarta vez, todas as outra já devem ter sido anunciadas pelo menos duas vezes; para o sétimo anúncio de uma peça, todas as outras já devem de ter sido oferecidas, pelo menos, três vezes (caso contrário 7 seria mais de 3 vezes maior que 2); e assim por diante.

Este capítulo discute estratégias para lidar com as vulnerabilidades identificadas no Capítulo 3. Embora as propostas apresentadas no presente capítulo constituam contribuição importante no sentido de projetar contramedidas, é importante que elas sejam devidamente avaliadas via modelagem e simulação.

Capítulo 5

Simulação

Este capítulo visa descrever o modelo de simulação desenvolvido para a investigação do comportamento do BitTorrent, com ênfase na análise de duas das vulnerabilidades apresentadas nesta dissertação. Na Seção 5.1 são descritos os principais componentes do sistema desenvolvido, bem como as premissas simplificatórias adotadas. Logo após, na Seção 5.2, a implementação do modelo de simulação é validada através da comparação entre simulações e resultados obtidos com a execução de experimentos reais em ambiente controlado. Por fim, este capítulo se encerra com a Seção 5.3 demonstrando detalhes mais específicos acerca da implementação realizada e apresentando os parâmetros que podem ser ajustados no simulador a fim de permitir diferentes experimentos.

5.1 Modelo

Para avaliar as vulnerabilidades propostas no Capítulo 3, criou-se um modelo de simulação do protocolo BitTorrent. Assim como os diagramas de estado (Figuras 2.2, 2.3 e 2.4) e tempo (Figura 2.5), o modelo está baseado em uma combinação da especificação do protocolo e da análise de traços de pacotes de implementações populares [Azureus, 2007, Bittorrent, 2007, μTorrent, 2007]. Ele incorpora as características do protocolo citadas no Capítulo 2, com a implementação de trackers, pares e todas as mensagens e estados mostradas nos diagramas daquele

capítulo.

Embora o modelo seja bastante realista, diversas premissas simplificatórias foram adotadas. Primeiro, não há troca efetiva de dados entre pares; portanto, as mensagens de dados possuem tamanho correto, porém carregam apenas a informação de controle necessária. Um par envia mensagens **BLOCK(p,b)** "contendo" um bloco **b** de uma peça **p**. Como não há dados propriamente ditos, não faz sentido usar um mecanismo para detecção de integridade em peças (via hash). Para simular blocos corrompidos, mensagens **BLOCK** incluem um valor booleano que indica se o mesmo é íntegro. Se todos os blocos de uma peça forem íntegros, então se considera que o cálculo de hash da peça obteve êxito.

Além disso, a simulação adota como premissa o protocolo TCP na camada subjacente, mas não simula o mesmo em detalhe. Na implementação real, um bloco é confiavelmente transferido para outro par, segmento a segmento, através da conexão TCP com o mesmo. Na simulação, uma mensagem **BLOCK** contém um bloco inteiro e é enviada de uma vez só, como se o tamanho do segmento fosse o tamanho do bloco e não houvesse fragmentação na camada de IP. Para simular o recebimento confiável de um bloco de dados na camada de transporte, e permitir o envio do próximo, foi acrescentada a mensagem **TCP_ACK(p,b)**, a ser enviada pelo receptor em resposta a uma mensagem **BLOCK(p,b)**.

Pares podem iniciar como semeadores (com uma cópia completa dos dados) ou sugadores (com zero ou mais peças). Pares recebem três parâmetros principais: um tempo de ativação, indicando quando os mesmos devem entrar no enxame; um tempo de semeadura, indicando quanto tempo um par deve permanecer no papel de semeador; e um intervalo de timeout, que é de quanto em quanto tempo um par conecta com o tracker. Assim como outros parâmetros, seus valores são configurados através de um arquivo de descrição da simulação. Os parâmetros restantes para o modelo são o número de peças no conteúdo sendo compartilhado, o número de blocos por peça, e o tamanho de cada bloco em KB. Mais informações sobre estes parâmetros são apresentadas na Seção 5.3.

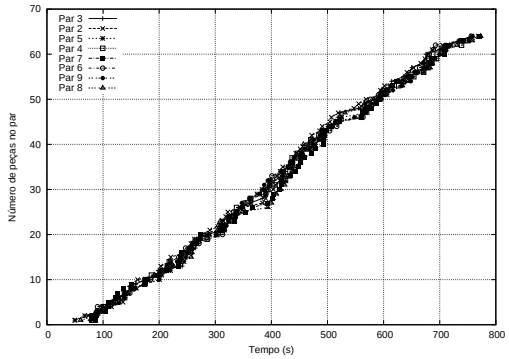
Em termos de topologia, todos os pares estão conectados por um enlace a uma "nuvem" que representa a rede. A capacidade dos enlaces pode ser individualmente configurada, tanto a partir do par como em direção ao par, em termos de largura de banda e latência (esta última simulada através de uma distribuição aleatória uniforme). Na comunicação entre dois pares, a capacidade do enlace de saída do par origem ou de chegada no par destino podem ser limitantes (o mesmo se aplica ao tracker). Não há perdas nos enlaces ou na nuvem porque a abstração adotada é de "canal de comunicação confiável", tal como obtida quando os pares se comunicam através do protocolo TCP.

5.2 Validação

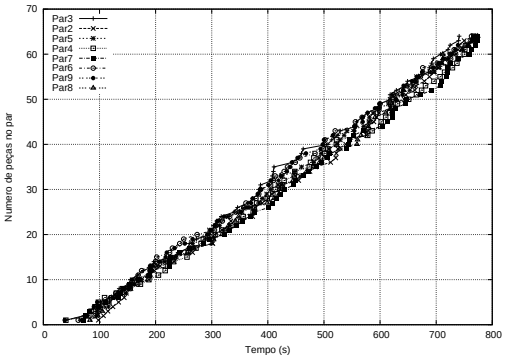
Para permitir a "validação" do modelo de simulação e sua implementação, as saídas geradas pela simulação foram comparadas com resultados colhidos em experimentos com a implementação "principal", também chamada de *mainline* [Bittorrent, 2007] e desenvolvida pelo criador do BitTorrent [Cohen, 2003], e com o Azureus [Azureus, 2007], uma das implementações mais populares do protocolo. A implementação mainline é, seguramente, a mais utilizada em experimentos realizados com o protocolo BitTorrent, sendo freqüentemente utilizada como base em artigos e trabalhos em veículos reconhecidos. A principal razão de sua escolha pela comunidade científica é o fato de ela seguir mais a risca o protocolo e por não possuir extensões adicionais ou otimizações que poderiam interferir no comportamento dos pares, conseqüentemente alterando os resultados finais. Esta implementação é tida como a implementação "mãe" de todas as outras e sempre que a mesma sofre alguma modificação, as outras implementações rapidamente as incorporam.

Para realizar os experimentos em rede real, um ambiente controlado foi montado através de uma rede local isolada, com um concentrador (*Hub*) interligando 10 máquinas Pentium 4 2.8GHz, 512 MB RAM, HD 40Gb e Windows XP. Para evitar interferências, todos os serviços do Windows foram desabilitados; um programa de captura de pacotes foi executado durante 5 minutos para assegurar a inexistência de

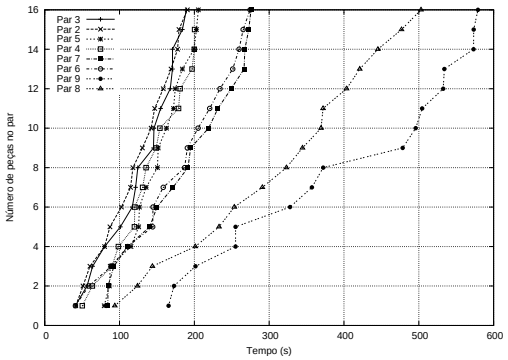
qualquer outro tráfego. Um limitador de banda em nível de *kernel* foi utilizado para a definição das velocidades de download e upload. O Par1 sempre foi o semeador inicial e o Par10 sempre atuou como tracker do enxame. Como pré-análise, foi realizada uma gama de experimentos variando-se fatores importantes como: número de pares (1, 2, 4 e 8 pares); emulação de largura de banda no enlace (40KB/s, 30KB/s, 20KB/s e 10KB/s, de forma homogênea e heterogênea entre pares); tamanho do arquivo (4MB e 16MB); número de peças (16, 64 e 256 peças) e de blocos (4, 16, 32, 64 blocos). Como medida de segurança, o tráfego foi registrado em dois capturadores de pacotes independentes (*tcpdump* e *wireshark*), executados em máquinas distintas.



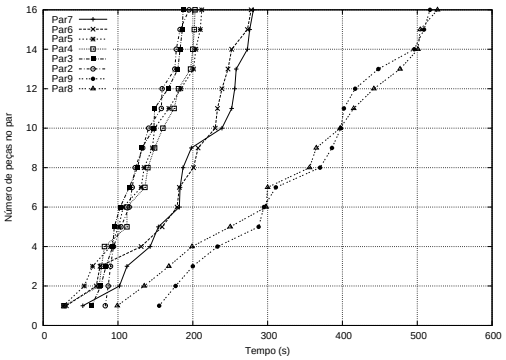
(a) Experimento com 8 pares idênticos



(b) Simulação com 8 pares idênticos



(c) Experimento com 8 pares com taxas diferentes



(d) Simulação com 8 pares com taxas diferentes

FIGURA 5.1 – Comparação entre avaliação experimental e com simulação.

Na Figura 5.1, podem ser observados os resultados obtidos experimentalmente e através de simulação para 8 pares em diferentes configurações. O eixo x representa o tempo, em segundos, enquanto o eixo y representa o número de peças disponíveis

no par. No primeiro caso (Figuras 5.1(a) e 5.1(b)), os pares realizam o download de um arquivo de 16 MB (64 peças de 16 blocos), todos com velocidade igual (pares homogêneos) de 30KB/s. A evolução em relação ao número de peças obtidas ao longo do tempo pode ser observada. A Figura 5.1(a) se refere ao experimento real e a Figura 5.1(b) à simulação. É possível notar que há grande semelhança entre o caso experimental e o simulado.

A fim de verificar a correlação entre o experimento real e a simulação, a média de tempo que os pares levaram para conseguir cada quantidade de peças foi calculada. Ilustrando, seja $t1_1^E$ o tempo que o par 1 levou para obter 1 peça no experimento e $t2_1^E$ o tempo que o par 2 levou para obter uma peça no experimento, independente de qual peça tenha sido. Têm-se então μ_i^E como a média dos tempos que os pares levaram para obter a quantidade i de peças no experimento. Assumindo que o conteúdo possua N peças, foram calculadas as médias para $i = 1..N$. Nos casos das Figuras 5.1(a) e 5.1(b), $N = 64$. A média para cada um dos valores de i foi calculada, tanto para o experimento (μ_i^E), quanto para a simulação (μ_i^S). Posteriormente, estes valores foram comparados e o coeficiente de correlação de Pearson para as médias foi de 0,997526, o que demonstra que as médias apresentam alta correlação entre si.

No segundo caso – Figuras 5.1(c) e 5.1(d) – os pares possuem velocidades diferentes (Pares 1, 2 e 3 com 40KB/s, Pares 4 e 5 com 30KB/s, Pares 6 e 7 com 20KB/s e Pares 8 e 9 com 10KB/s) e realizam o download de um arquivo de 4MB (16 peças de 16 blocos). Novamente, há uma evidente semelhança entre a simulação e o caso experimental. As médias dos tempos obtidos na evolução foram agrupadas segundo a velocidade dos pares, criando assim 4 grupos. O coeficiente de correlação de Pearson para as médias dos pares foi o seguinte: pares 2 e 3 = 0,996382; pares 4 e 5 = 0,995539; pares 6 e 7 = 0,98732 e pares 8 e 9 = 0,9791306. Além disso, os traços gerados pelas simulações foram contrastados com aqueles gerados pelas implementações, no sentido de aumentar a segurança sobre a fidelidade da simulação.

5.3 Implementação

A implementação do modelo de simulação foi realizada em Java, estendendo-se o arcabouço de simulação Simmcast [Simmcast, 2007], um simulador discreto para simulação de protocolos distribuídos. A simulação possui aproximadamente 3500 linhas de código Java, mais a lógica para implementação de nodos hospedeiros (HostNode) e roteadores (RouterNode), oferecidas pelo arcabouço do Simmcast.

Existem dois tipos de nodos na rede, o tracker (classe TrackerNode) e o par (classe PeerNode). Cada nodo possui pelo menos uma thread contendo a lógica de execução; no caso do tracker ou de pares corretos, nas classes TrackerThread e PeerThread, respectivamente. A "nuvem" é representada por um nodo especial de roteamento (classe P2PRouter), e uma thread (classe P2PRouterThread). A classe CheaterNode representa um par malicioso, e abriga a classe CheaterThread, que, por sua vez, implementa a lógica do atacante.

De forma geral e resumida, o fluxo de execução da classe PeerThread segue a lógica descrita na Figura 5.2. Após inicializar e realizar algumas rotinas para configurar o ambiente inicial, o par irá enviar uma mensagem ao tracker, informando sua presença e solicitando uma lista de pares participantes do enxame. Em seguida, o mesmo irá realizar uma chamada bloqueante (a thread será colocada para dormir) até que receba alguma mensagem ou até que o tempo de espera expire. Ao sair da chamada **receive(timeout)**, o par irá verificar se recebeu uma mensagem ou se houve um timeout.

Caso tenha recebido uma mensagem, o par irá tratar a mesma, possivelmente gerando uma ou mais mensagens de resposta. Essas mensagens serão colocadas na fila para serem enviadas e o par irá verificar se o tempo destinado à rodada atual chegou ao fim, o que indica que uma avaliação dos melhores pares deve ser realizada. O controle do envio, bem como o tempo e atraso provocados pelo canal, são realizados pelo Simmcast. Assim como os demais valores padrão, o tempo utilizado na simulação para cada rodada foi o tempo padrão do BitTorrent, ou seja, $\sigma = 10s$ (para mais informações consulte a Seção 2.1). A avaliação dos pares

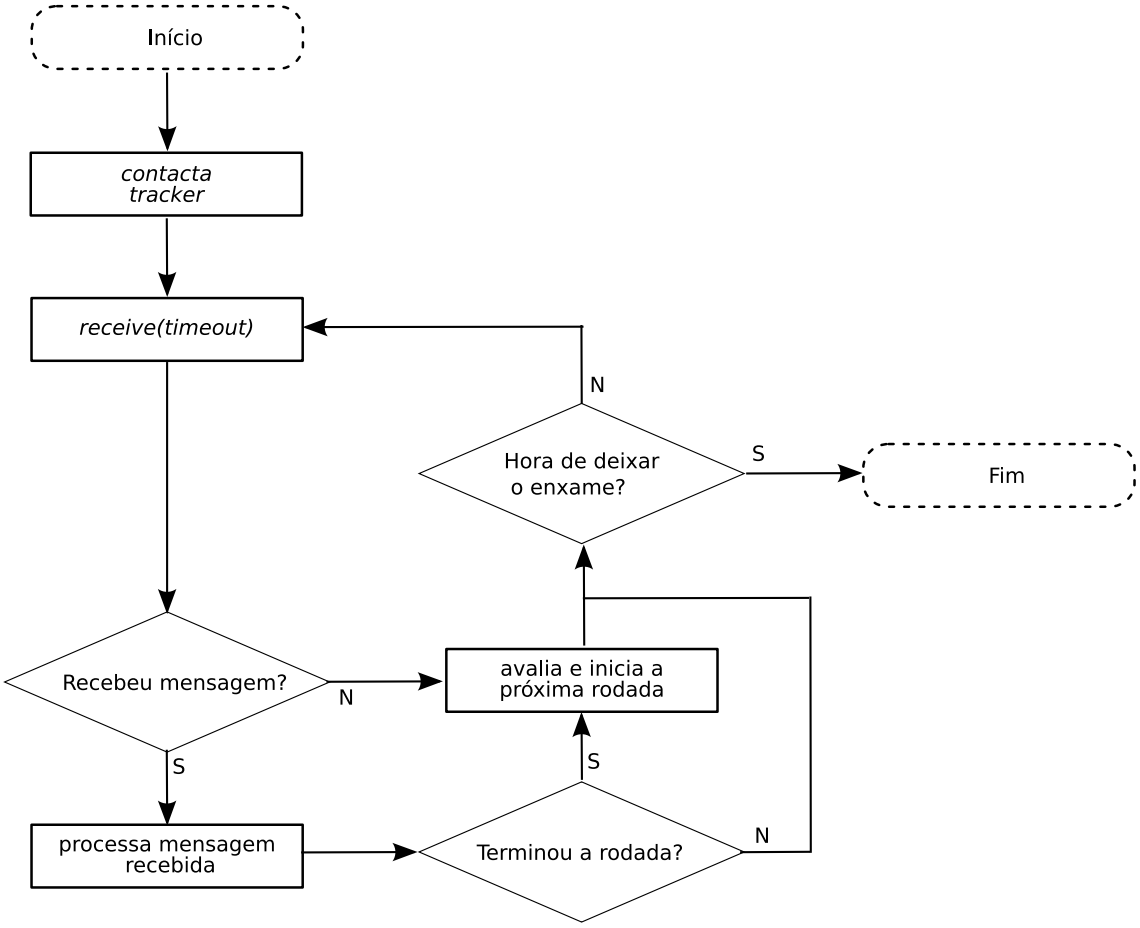


FIGURA 5.2 – Diagrama de fluxo dos pares

é realizada quando ocorre um timeout na espera por mensagens. O timeout visa, justamente, marcar o momento de realizar a avaliação. O processo de avaliação possivelmente também gerará novas mensagens para alguns dos pares remotos com os quais o par local está conectado. Por último, antes de realizar uma nova chamada esperando por mensagens, o par irá avaliar se seu tempo de permanência no enxame não expirou. Caso positivo, uma mensagem de desconexão será enviada a todas as conexões que o par estava mantendo, bem como ao tracker, para que este remova o par da lista de pares que podem ser informados em uma solicitação **Get**.

A lógica empregada pela classe `CheaterThread` é semelhante à apresentada na Figura 5.2, com algumas exceções, dependendo do tipo de ataque que o par malicioso estará realizando. Nos ataques que foram implementados: (a) Mentindo a posse de peças (Seção 3.6) e Eclipsando pares corretos (Seção 3.2), por exemplo, os pares

maliciosos não realizam avaliações por rodadas. Como os pares atacantes não estão trocando conteúdo, não há necessidade de tal procedimento.

Além da lógica implementada nas threads do tracker e dos pares semeadores e sugadores, a simulação é definida pelo conjunto de parâmetros abaixo descritos:

- Número de Semeadores: o método **setPeersSeeder** permite informar a um par que ele atuará como semeador. Um *array* de pares pode ser configurado utilizando-se esta função, a fim de permitir que a simulação inicie com mais de um semeador.

- Número de Sugadores: número de pares que iniciarão sem peças ou com um subconjunto de peças. Os pares podem ser configurados para iniciar com um conjunto aleatório ou definido de peças.

- Número de peças: o método **setSizePiecesSet** define o número de peças existentes no conteúdo que será distribuído na simulação.

- Número de blocos por peça: o método **setBlocksPerPiece** define o tamanho de cada peça. Cada bloco possui o tamanho de 16KB. O tamanho do conteúdo é definido através do número de peças e blocos por peça informados.

- Tamanho dos blocos: o método **setBlockSize** define o tamanho do bloco, em bytes, a ser utilizado na simulação.

- Capacidade dos enlaces: a simulação permite que os pares sejam configurados, individualmente ou em grupo, com largura de banda e latência diferentes, tanto para *uplink* quanto para *downlink*.

- Probabilidade de tornar-se semeador: o método **setSeedingProbability** define a probabilidade de um par tornar-se semeador, e portanto permanecer no enxame pelo tempo informado no parâmetro "Tempo de permanência como semeador".

- Tempo de permanência como semeador: o método **setSeedingTime** define o tempo que o par permanecerá no enxame após tornar-se semeador. Este

parâmetro pode ser passado para os semeadores iniciais, caso estes não devam permanecer no enxame até o término da simulação. Outra possibilidade é a do par permanecer um tempo aleatório, sendo seu valor tirado de uma das distribuições suportadas pelo Simmcast.

- Taxa de entrada de novos pares: se presente, informa a distribuição e os parâmetros utilizados para definir a taxa de chegada de novos pares ao enxame.

Os resultados de uma execução da simulação podem ser coletados de duas formas: detalhada, no traço de pacotes gerado pelo Simmcast (na classe simmcast.trace.SimmcastTraceGenerator), ou mais específica, que são as informações geradas para a saída padrão. O grau de detalhe dessas informações é ajustável. A configuração dos arquivos de simulação, dos ajustes incluindo modificações necessárias para a realização de várias execuções, o processamento dos dados e a consolidação estatística em gráficos são feitos através de scripts em Python.

A versão atual do simulador desenvolvido permite analisar a rede BitTorrent em diferentes cenários e com configurações diversas; novos algoritmos também podem ser facilmente integrados ao modelo. Além disso, é possível estudar-se os ataques já implementados (Mentindo a Posse de Peças – Seção 3.6 – e Eclipsando Pares Corretos – Seção 3.2) e também modelar novos casos.

Para os ataques já implementados, os seguintes parâmetros podem ser configurados:

- Número de Atacantes: informa o número de pares atacantes que entrarão no enxame.

- Número de Sybils: o método **setSybilsToCheat** informa o número de identidades virtuais que um único nodo irá simular.

- Números Mínimo de Conexões: o método **setMinConnections** informa o limite inferior em relação ao número de conexões que um atacante irá manter. Normalmente, este valor é grande o suficiente a fim de fazer com que cada

identidade utilizada pelo atacante possa se conectar com cada um dos pares honestos presentes no enxame.

- Número Máximo de Conexões: o método **setMaxConnections** informa o limite superior em relação ao número de conexões que um atacante irá manter.

- Número de Peças Mentidas: o método **setPiecesToCheat** informa o número de peças que um par ou um conjunto de pares atuando em conluio irá mentir.

- Peças a serem mentidas: a definição de quais peças os pares irão mentir pode ser dar de forma aleatória ou através do arquivo de configuração da simulação (método **setPieceToCheat**). Em ambos os casos, todos os pares irão mentir a posse das mesmas peças.

Capítulo 6

Avaliação do Impacto

Este capítulo apresenta uma avaliação de impacto realizada com o simulador descrito no Capítulo 5. Primeiramente, os cenários de ataque propostos são descritos (Seção 6.1). Logo após, dois dos ataques descritos no Capítulo 3 são avaliados via simulação e os resultados obtidos, discutidos na Seção 6.2. A escolha dos dois ataques deveu-se, principalmente, ao potencial de impacto, à facilidade de implementação dos mesmos no simulador e à restrição de tempo, que impediu que todos os ataques propostos fossem avaliados.

6.1 Cenários Propostos

A fim de permitir uma avaliação mais aprofundada de dois ataques enunciados no Capítulo 3, diferentes cenários são propostos nesta seção. Os ataques visam: (a) prejudicar o espalhamento das peças no enxame e (b) não permitir que um par realize o download do conteúdo. Os resultados foram obtidos através de 12 simulações repetidas com sementes aleatórias diferentes.

No primeiro caso, o ataque Mentindo a Posse de Peças, introduzido na Seção 3.6, foi utilizado por um grupo de atacantes atuando de forma conjunta. O objetivo desse experimento é analisar a influência que o ataque possui no tempo que os pares levarão para realizar o download do conteúdo. As variáveis consideradas foram o número de atacantes e o número de peças mentidas.

No segundo cenário, um atacante utiliza-se de Sybil para realizar o ataque Eclipsando Pares Corretos, definido na Seção 3.2. A variável considerada foi a quantidade de atacantes necessários para provocar a desistência de uma fração significativa dos pares corretos participantes do enxame. Assumiu-se que um par desiste caso não finalize o download em um tempo superior a 10 vezes o tempo médio necessário para obter o conteúdo em condições normais (sem ataque).

A simulação segue o modelo descrito no Capítulo 5. Inicialmente, há apenas um semeador, que é ativado no tempo 0. Sugadores são ativados em tempos distintos, com um atraso uniformemente gerado entre 0 e 10s. Sugadores, quando obtém o conteúdo completo, se tornam semeadores com probabilidade 0,3 (refletindo os 30% de semeadores reportados em [Andrade et al., 2005]); os que se tornam semeadores permanecem assim até o final da simulação, enquanto os demais terminam. Um par recontacta o tracker a cada 30s se estiver com menos de 20 conexões, ou a cada 5 minutos caso contrário.

Escolheu-se compartilhar um conteúdo de 16MB, dividido em 64 peças de 256KB cada (ou seja, 16 blocos). Tais valores foram escolhidos com base em [Legout et al., 2006]. Os enlaces foram configurados de forma simétrica para download e upload, com cenários homogêneos e heterogêneos. O tracker retorna uma lista de até no máximo 50 pares, um valor freqüentemente adotado em implementações.

Nos dois cenários, os pares maliciosos também entraram no enxame com um atraso uniformemente distribuído em um tempo entre 0 e 10 segundos. Em nenhum dos dois casos propostos, os pares atacantes realizam download ou upload de conteúdo. Sua única função é influenciar a escolha dos pares honestos (primeiro caso) ou esgotar o número de conexões dos pares honestos (segundo cenário).

Após a conexão ser estabelecida, o atacante passa a ignorar a maioria das mensagens recebidas de pares honestos. Ele nunca dessufoca outros pares, mas sempre envia uma mensagem **Interested**, passando assim a concorrer nas escolhas que os pares honestos fazem para eleger o par otimisticamente dessufocado.

Pares maliciosos também não estabelecem conexões entre si. Assume-se que os pares atacantes sabem de antemão os parâmetros a serem utilizados nos ataques. Assim, os pares maliciosos evitam desperdiçar seus recursos com conexões entre si. Esta situação é válida no caso de um ataque partindo de um conjunto de máquinas que estejam sob uma mesma gerência. No caso de um conjunto de atacantes distribuídos e sem uma coordenação única, assume-se que anteriormente ao ataque os pares maliciosos obtém as informações acerca das configurações do mesmo.

6.2 Análise dos Resultados Obtidos

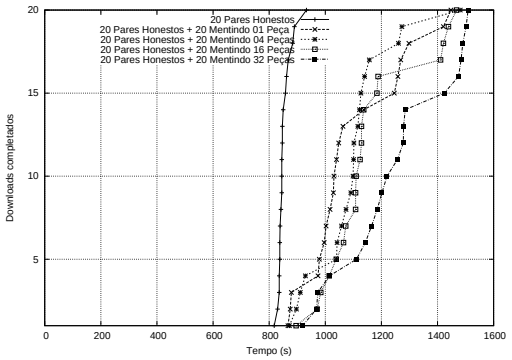
Para avaliar a efetividade do primeiro ataque (Figuras 6.1 e 6.2), foi considerada a métrica do atraso médio no término de downloads dos pares participantes do enxame, expresso através do número cumulativo de downloads. Já no segundo ataque (Figura 6.3), a métrica empregada foi o número de falhas em downloads devido à desistência de sugadores (falha do download para um par).

A Figura 6.1 apresenta dois gráficos que ilustram o impacto do número de peças sendo mentidas por um bando de mentirosos (em igualdade com pares honestos), para enxames com 20 e 80 pares (Figuras 6.1(a) e 6.1(b), respectivamente). O gráfico representa o número de downloads completados em função do tempo, para quatro quantidades diferentes de peças sendo mentidas (1, 4, 16 e 32), além do caso sem atacantes. As curvas mostram que a mentira de peças gera uma sobrecarga significativa, atrasando o download dos pares corretos.

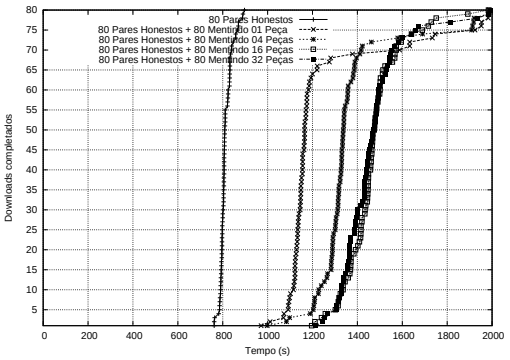
É possível notar que o número de peças mentidas possui impacto negativo e que é diretamente proporcional ao tempo médio de download, isto é, quanto mais peças mentidas, maior o tempo médio de download. Na Figura 6.1(a), sem maliciosos, os pares terminaram em um tempo próximo de 850 segundos. Já com 32 peças sendo mentidas, aproximadamente metade dos pares completa seu download num tempo superior a 1200 segundos, o que representa uma sobrecarga de aproximadamente 50%. Os últimos 6 pares completaram seu download após o tempo 1400.

No caso da Figura 6.1(b), o resultado também foi significativo. Com 16 e 32

peças sendo mentidas, o tempo médio de download pulou de aproximadamente 800 para mais de 1400 em mais de 50% dos casos. Um ponto interessante é o fato de que praticamente não houveram diferenças entre as simulações em que os pares mentiam 16 e 32 peças. Resumindo, é possível concluir que, a partir da mentira de uma única peça, a efetividade do ataque aumenta com o número de peças mentidas, até um ponto em que mentir mais peças se torna inócuo, ou até mesmo contraproducente.



(a) Simulação com 40 pares, metade honestos



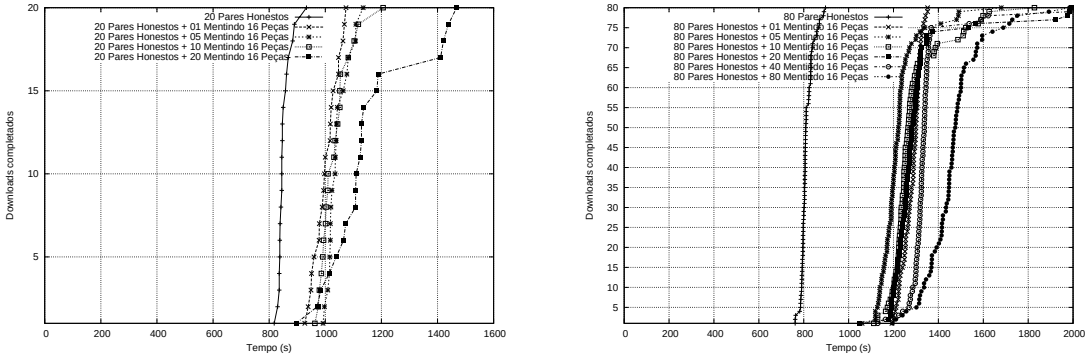
(b) Simulação com 160 pares, metade honestos

FIGURA 6.1 – Impacto da quantidade de peças mentidas nos downloads, com o mesmo número de pares honestos e maliciosos.

Quanto ao número de atacantes, a Figura 6.2 apresenta dois gráficos com o impacto do número de pares maliciosos, em enxames com 20 e 80 pares honestos. O gráfico representa o número de downloads completados em função do tempo, para diferentes números de atacantes. As curvas mostram que o impacto do número de atacantes é significativo, tanto no cenário com 20, como com 80 honestos, causando o atraso dos downloads em geral. Conforme a Figura 6.2(b), a presença de pelo menos um par mentiroso já provoca um atraso de aproximadamente 400 segundos, ou 50% a mais do que o tempo normal de download.

É possível confirmar a intuição que quanto mais atacantes, melhor para o ataque, pois existe uma tendência geral de que as curvas com maior número de atacantes estejam situadas mais à direita no gráfico. Entretanto, é surpreendente que o impacto não seja tão significativo como inicialmente esperado. Exemplificando, na Figura 6.2(b), 1 mentiroso faz com que 60 downloads tenham sido completados no tempo 1200s, enquanto isso ocorre no tempo 815s quando não há ataque; com

80 mentirosos, os primeiros 60 downloads são completados até o tempo 1500s (59 atacantes adicionais resultaram em apenas 300s a mais).



(a) Simulação com 20 pares honestos (b) Simulação com 80 pares honestos

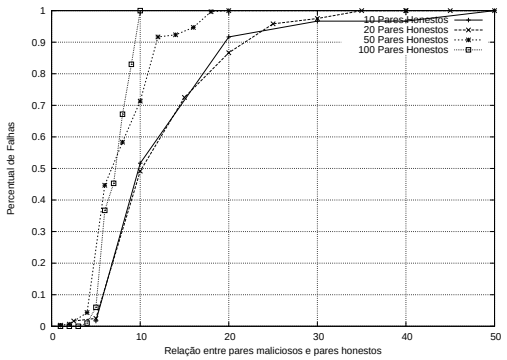
FIGURA 6.2 – Impacto da quantidade de mentirosos nos downloads.

A escolha dos valores de 20 e 80 pares honestos para o tamanho do enxame levou em consideração a quantidade de conexões que um par pode estabelecer. No primeiro caso, representado na Figura 6.2(a), os pares honestos não chegam a esgotar o número de conexões que podem manter com outros pares do enxame (55, vide Capítulo 2). Nesse caso, esperava-se que o ataque de mentira de peças fosse mais efetivo que naquele em que o número de máximo de conexões dos pares honestos é atingido, pois todos os mentirosos certamente conseguiram estabelecer conexões com os pares honestos e, portanto, influenciar melhor suas escolhas. No entanto, esse comportamento não foi observado: a efetividade do ataque com 1, 5 e 10 pares mentirosos foi praticamente a mesma.

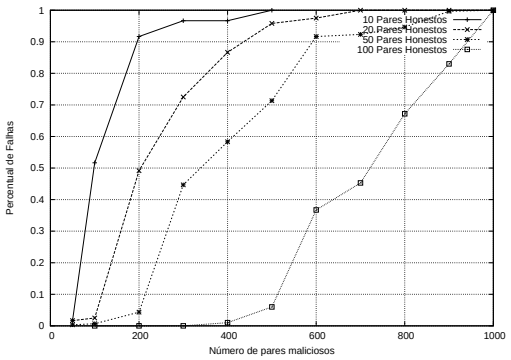
Já no caso da Figura 6.2(b), os pares negaram conexões com outros pares honestos e maliciosos ao terem atingido o limite de 55 conexões simultâneas. Com isto, a influência dos pares mentirosos não foi tão efetiva. No entanto, um efeito adicional ao da mentira das peças, o consumo das conexões do par, acabou por influenciar no tempo de download dos pares honestos. Novamente, quando o número de mentirosos é menor que o de pares honestos, os tempos de download foram muito parecidos. Com um número igual de honestos e mentirosos, o tempo de download começa novamente a apresentar um aumento mas expressivo.

A Figura 6.3 apresenta os resultados de várias simulações para o ataque Eclipse

(vide Seção 3.2). Os dois gráficos ilustram o percentual de downloads falhos em função do número de atacantes (variando-se entre 100 e 1000 maliciosos), para diferentes quantidades de pares honestos.



(a) Impacto da proporção de maliciosos no percentual de downloads falhos, para diferentes tamanhos de enxame.



(b) Impacto da quantidade de maliciosos no percentual de downloads falhos, para diferentes tamanhos de enxame.

FIGURA 6.3 – Impacto da quantidade de mentirosos no percentual de downloads falhos.

Na figura 6.3(a), os resultados são apresentados em termos relativos à relação entre pares maliciosos e honestos, no eixo x, e percentual de downloads falhos, no y. Nessa análise, o tempo de download de um conteúdo varia com as características dos pares do enxame. Assim sendo, o tempo aceitável para "término" de um enxame (após o encerramento do último download) foi determinado em função do tempo (denominado T_{fim}) em que o último par completa o download em um cenário sem ataques. O tempo máximo adotado na Figura 6.3 é igual a $\varepsilon \times T_{fim}$, com $\varepsilon = 10$. O resultado dos mesmos experimentos, mas com números absolutos no eixo x ao invés da proporção, são apresentados na Figura 6.3(b).

Analisando a Figura 6.3, nota-se primeiro que os enxames são em geral afetados pelo ataque de Eclipse e que as curvas apresentam comportamento consistente. Segundo, como demonstrado pela comparação das curvas entre si, o número absoluto de pares tem impacto na efetividade do ataque: as curvas que são proporcionalmente menos afetadas (precisam de uma relação maior de pares atacantes para atingir o mesmo dano) são aquelas com uma quantidade absoluta menor de pares honestos. Terceiro, nota-se que quando a proporção de maliciosos é inferior a 5:1, o efeito dos ataques é inexistente ou negligível. Por outro lado, a partir de 5:1, o efeito de ataques cresce agressivamente; quando é 6:1 em um enxame com 100 pares corretos, os 600 maliciosos são capazes de provocar quase 40% de falhas nos downloads. Por fim, a melhor relação entre custo benefício para um ataque parece se situar em torno de 20:1, pois faria com que pelo menos 85% dos downloads falhassem independentemente do tamanho do enxame. Generalizando, existe um número arbitrário e finito de pares maliciosos que é suficiente para tornar um enxame BitTorrent inútil. Explorando a identidade fraca característica dessa arquitetura, é possível que um usuário mal intencionado automatize o processo de registro e assuma um número bastante alto de identidades virtuais.

Capítulo 7

Trabalhos Relacionados

Este capítulo apresenta uma revisão dos principais trabalhos relacionados a esta dissertação. Conforme já mencionado, até onde se sabe, não há atualmente qualquer trabalho anterior que possua como foco principal o estudo das vulnerabilidades de segurança da arquitetura BitTorrent. Por essa razão, os trabalhos mais proximamente relacionados são aqueles que estudam outras características de sua arquitetura ou funcionamento. Uma quantidade substancial de pesquisa tem sido desenvolvida recentemente buscando estudar diversos aspectos da arquitetura BitTorrent. Boa parte desses trabalhos objetiva determinar se o mecanismo de incentivo empregado, denominado de tit-for-tat, garante justiça no compartilhamento.

Em [Cohen, 2003] a rede BitTorrent é apresentada formalmente. O artigo demonstra o funcionamento dessa rede e como o mecanismo de tit-for-tat é utilizado visando garantir a cooperação entre usuários sem diminuir a troca eficiente de dados entre pares.

Segundo [Jun and Ahamad, 2005], a abordagem do BitTorrent não é totalmente eficaz em desestimular o aparecimento de pares carona (pares que só recebem da rede, mas não contribuem com sua parcela). O artigo propõe a modificação do mecanismo de incentivo, usando como base estudos sobre o problema clássico do Dilema do Prisioneiro com Múltiplas Iterações [Hofstadter, 1983]. Segundo os autores, a falta de um mecanismo de punição para quem não coopera

induz o aparecimento de pares caronas. A solução apresentada propõe cooperar na primeira iteração e repetir, com quem se está trocando peças, o comportamento que este par apresentou na iteração anterior. Através de simulação realizada usando o PlanetLab [PlanetLab, 2007], o artigo mostra que o mecanismo proposto é muito mais eficiente no incentivo à cooperação. Um ponto que deve ser observado, entretanto, é que a solução proposta, embora mais justa em termos de cooperação, apresenta um desempenho global inferior. Em outras palavras, o tempo médio de download dos pares, inclusive os que cooperam, aumenta.

Em [Andrade et al., 2004] é estudado o comportamento dos usuários em termos de cooperação. Dados de uso de redes BitTorrent foram coletados através de múltiplas comunidades, e foram analisados os fatores que influenciam na cooperação entre pares. Particularmente, foram estudados: (a) o comportamento induzido pelo mecanismo de incentivo do BitTorrent e (b) o impacto que o tipo de conteúdo e as políticas da comunidade no qual o par está inserido têm na cooperação. Foram estudadas duas comunidades que empregam mecanismos específicos adicionais para incentivar ainda mais a cooperação. Os resultados mostraram que em enxames com um número relativamente pequeno de pares semeadores, o mecanismo de incentivo penaliza com sucesso os pares carona. No entanto, quando o número de pares com o arquivo completo é grande, os caronas obtiveram tempos médios até menores que pares que cooperaram. Outra conclusão foi que a taxa de semeadores é maior em arquivos com tamanhos menores.

Barbera e outros desenvolvem em [Barbera et al., 2005] um modelo analítico de como atua um par carona em uma rede BitTorrent. O modelo se baseia na perspectiva do usuário e é estruturado usando-se Cadeias de Markov com Tempo Contínuo. Este modelo é válido somente para a fase em que o par está atuando como sugador. O artigo chega ao mesmo resultado de [Andrade et al., 2005] ao demonstrar que a presença de uma quantidade maior de semeadores faz com que o mecanismo de reciprocidade (tit-for-tat) perca sua efetividade. Uma solução citada foi a modificação do algoritmo de escolha dos semeadores, em que seriam eleitos

para receber peças preferencialmente os pares que tivessem boas taxas de upload.

Em [Bharambe et al., 2005a], um estudo através de simulação da rede BitTorrent mostrou que a mesma possui um desempenho muito bom na utilização da largura de banda do uplink e tempo de download. No entanto, o mesmo trabalho afirma que pares com baixa largura de banda realizam sistematicamente mais download que upload. Os autores então propõem e demonstram que modificações no tracker e uma política de reciprocidade mais restritiva ocasionam melhoras significativas na justiça e utilização do canal.

Em [Guo et al., 2005] os autores afirmam que em mais de 85% das vezes pares participam de múltiplos enxames ao mesmo tempo. A análise apresentada mostra que em pares conectados a vários enxames: (a) a disponibilidade torna-se baixa rapidamente, devido à taxa exponencial decrescente de chegada de novos pares; (b) o desempenho do par é instável e altamente flutuante; (c) sistemas existentes podem não garantir justiça (pares com velocidades altas de download tendem a contribuir menos). O artigo então analisa essas questões em ambientes realísticos, segundo os autores, e apresenta um modelo de múltiplos enxames baseado em grafos. O modelo apresentado busca fornecer incentivos para a colaboração entre pares que participam de múltiplos enxames.

O mesmo comportamento (pares conectados a vários enxames) é estudado em [Tian et al., 2006]. Tian e outros apresentam um modelo baseado em fluidos para representar a dinâmica de pares nessas condições. Dois cenários diferentes são estudados: pares realizando download concorrente de mais de um enxame (chamado de *multi-torrent concurrent downloading*, ou MTCD) e pares realizando download de vários arquivos encapsulados em um único torrent (chamado *multifile torrent concurrent downloading, ou MFCD*). O desempenho dos pares nesses dois cenários é avaliado, e os resultados mostram que ele diminui à medida que aumenta a correlação entre os conteúdos presentes nos dois enxames. Duas alternativas são propostas e avaliadas. Para o MTCD, os autores sugerem um modelo (chamado de *multi-torrent sequential downloading, ou MTSD*) onde um par realiza o download

de mais de um torrent, mas participa dos enxames de maneira seqüencial, isto é, um par só participará de um enxame por vez. Para o caso do MFCD é proposta a idéia de os pares colaborarem em "sub-torrents", isto é, um único torrent possui a informação sobre vários arquivos correlacionados. Ao contatar o tracker, o par informa quais arquivos ele possui e quais quer obter. Neste último modelo (chamado de *collaborative multi-file torrent sequential downloading, ou CMFSD*) um par que é sugador em um "sub-enxame" poderá ser semeador em outro e vice-versa. Por último, propõe-se um mecanismo auto-adaptativo que permite correlacionar vários arquivos entre pares. O artigo não apresenta, entretanto, uma forma de determinar o grau de correlação entre conteúdos, sendo este tema indicado como trabalho futuro.

Em [Purandare and Guha, 2006], é proposto e simulado um modelo para BitTorrent que agrupa usuários em categorias e os convence a cooperar. Este modelo usa um esquema de auto-punição e auto-recompensa, e criptografia de chave pública é utilizada como forma de autenticar as informações enviadas. Não é tratado, no entanto, como os pares e o tracker obtém seu par de chaves. Para a troca de conteúdo, os pares são organizados em grupos pelo tracker, de acordo com a largura de banda de upload por eles anunciada (chamada de PUS - *Published Upload Speed*). Um par pode anunciar uma largura de banda menor do que a sua, assumindo que não queira utilizar todo o canal somente para a troca de conteúdo mas, neste caso, será direcionado para pares com largura de banda similar à publicada. Cada par fica responsável por controlar se os pares remotos com os quais está trocando blocos estão honrando a largura de banda informada. Caso negativo o tracker é informado e um alerta é enviado ao par que foi denunciado. Se um limite de delações for atingido, o par é excluído do enxame. Os resultados das simulações mostram que os pares passam a cooperar mais e que há um aumento na utilização do canal, diminuindo conseqüentemente o tempo médio de download.

Já em [Bharambe et al., 2005b], é demonstrada a robustez do BitTorrent, analisando-se os componentes básicos via simulação, para diferentes cargas de trabalho, em termos de utilização de enlace upstream, tempo de download e

justiça; os autores então propõem técnicas para lidar com as situações específicas em que BitTorrent não apresenta comportamento eficiente. No artigo, os autores avaliam os principais comportamentos do protocolo, individualmente e agrupados, e mostram ainda que pares com baixa largura de banda realizam sistematicamente mais download que upload de pares mais rápidos. Duas modificações são propostas: (a) mudanças no tracker, para fornecer listas de pares com velocidades próximas da do par solicitante e (b) uma política de reciprocidade (tit-for-tat) baseada em blocos, onde um par só fornecerá certa quantidade de blocos a mais do que os blocos recebidos. Esta última proposta é usada como alternativa à política de dessufocamento otimista.

Contrastando com as conclusões dos trabalhos anteriores, em [Legout et al., 2006] defende-se que os mecanismos de Local Rarest First (LRF) e tit-for-tat (TFT) são suficientemente bons e que sua substituição não se justificaria, pois outros modelos aumentariam a complexidade da solução e não trariam melhora significativa. Para validar as proposições apresentadas, um cliente BitTorrent foi modificado e experimentos foram executados em enxames reais com diferentes características. Os resultados mostram que a troca do mecanismo de LRF por um de *source code* ou *network code* não traz melhoras significativas e que o algoritmo de sufocamento é robusto contra pares carona, garantindo assim boa justiça de compartilhamento.

Em [Nussbaum and Richard, 2006] apresenta-se um arcabouço para estudar sistemas Peer-to-Peer, denominado P2PLab. O sistema combina técnicas de emulação e virtualização para simular sistemas P2P. No P2PLab a virtualização ocorre em nível de processos, com modificação de alguns arquivos da pilha de protocolos TCP/IP no FreeBSD e o uso do software Dummynet para emulação da rede. Como validação, realiza-se uma simulação com a rede BitTorrent, contendo 160 pares que fazem o download de um arquivo de 16MB, fornecido por 4 semeadores. Como prova de conceito da escalabilidade da solução, apresenta-se um experimento com 5760 pares virtuais localizados em 180 nodos reais.

Em [Stutzbach and Rejaie, 2006] a taxa de entrada e saída de pares da rede P2P, também conhecida como *churn rate*, é estudada. Três sistemas P2P são estudados: Gnutella [Gnutella, 2007], BitTorrent [Bittorrent, 2007] e Kad, uma rede baseada em Kademlia [Maymounkov and Mazières, 2002] utilizada pelo sistema eDonkey [eDonkey, 2007]. O artigo primeiramente aborda outros estudos que apresentaram conclusões contraditórias sobre a taxa de entrada e saída, e demonstra erros comuns que podem levar a conclusões falsas ou imprecisas. Entre as conclusões apresentadas, o artigo mostra que: (a) as propriedades de entrada e saída da rede BitTorrent são significativamente diferentes se comparadas a outras redes; (b) tempos de sessão podem ser ajustados a distribuições Weibull ou log-normal; (c) comportamento anterior não pode ser utilizado como previsão para o comportamento atual em BitTorrent; (d) a disponibilidade de pares mostra uma grande correlação ao longo de dias consecutivos; e (e) pares normalmente permanecem no sistema longo tempo após se tornarem semeadores.

Em [Das et al., 2006] o uso de BitTorrent como mecanismo auxiliar na distribuição de conteúdo por provedores é proposto e avaliado. Os autores criam um modelo analítico para determinar o tempo médio de download, tanto com população transiente como no estado estável. Para a distribuição de conteúdo, servidores bem conectados são alocados dinamicamente, de acordo com a taxa de chegada de novos pares e visando garantir a taxa média de download desejada.

Em [Fan et al., 2006] o uso do método de equações diferenciais estocásticas (EDE) é utilizado para modelar o comportamento do BitTorrent. O modelo permite incorporar o comportamento dos pares na avaliação. Os pares são classificados em três tipos: (a) pares com menos de 50% das peças; (b) pares com mais da metade das peças; e (c) pares semeadores. O modelo é posteriormente validado via simulação e os autores mostram que o novo modelo, se comparado com os até então propostos é o que apresenta os resultados mais precisos. Valores próximos aos reais para itens como média de sugadores, de semeadores, da utilização do canal e o tempo de download podem ser obtidos a partir das equações propostas. Os resultados

observados demonstram que o modelo BitTorrent possui alta escalabilidade, pois o tempo médio de download diminui quando a taxa de entrada de pares aumenta até atingir o ponto de saturar o canal. Outros resultados mostram que a taxa de saída de semeadores, o número médio de conexões de cada par e a largura de banda são parâmetros importantes no tempo médio de download e que, se a taxa de entrada de novos pares for pequena, uma largura de banda maior não necessariamente garante um menor tempo de download.

Em [Erman et al., 2005] as características das sessões do tráfego BitTorrent são apresentadas. O artigo descreve a infra-estrutura desenvolvida para a coleta dos dados, a metodologia utilizada na modelagem e os resultados obtidos. Os traços em nível de aplicação foram coletados no Instituto de Tecnologia de Blekinge e em um provedor local da Suíça, onde treze medições independentes foram realizadas. Os resultados levaram em conta apenas a fase em que um par está atuando como semeador no enxame e mostraram que a taxa de entre chegadas (*interarrival rate*) pode ser modelada através de uma distribuição hiper-exponencial, que o tamanho e duração das sessões são altamente correlacionados e podem ser modelados através de uma distribuição lognormal.

O artigo traz ainda dois diagramas de tempo, mostrando a conexão e a troca de peças entre pares, mas ambos apresentam inconsistências em relação à definição do protocolo em [BitTorrentProtocolSpecification, 2007] e aos traços de pacotes coletados durante os experimentos controlados. No primeiro diagrama – procedimentos para a conexão entre pares – a figura apresentada pelos autores mostra a mensagem de **BitField** sendo trocada num segundo momento, após o procedimento de **Handshake**, quando na verdade o BitField é informado como resposta ao Handshake. No segundo diagrama, os autores mostram as requisições por blocos sendo realizadas antes de o par receber uma mensagem de dessufocamento. No entanto, um par só pedirá blocos a outro após receber uma mensagem de dessufocamento (**Unchoke**), até porque não há sentido em um par pedir antes. Se um par local solicitasse blocos a um par remoto antes de ser dessufocado por este último,

poderia acontecer a situação de o par já ter obtido a peça quando o par remoto o dessufocasse. Como os pares evoluem ao longo do tempo, as requisições por blocos só são realizadas quando um par possui a expectativa de receber conteúdo de outro, isto é, não está sendo sufocado por este. Outro ponto que os diagramas não mostram é que podem existir intervalos de tempo arbitrariamente longos entre as mensagens. Na presente dissertação, a Figura 2.5 fornece um diagrama de tempo mais completo e apurado. As inconsistências identificadas nos diagramas em [Erman et al., 2005] demonstram a complexidade de entendimento do protocolo BitTorrent e refletem a ausência de uma documentação extensa sobre o seu funcionamento. Neste contexto, os diagramas e as explicações no Capítulo 2 são uma contribuição desta dissertação.

Os mesmo autores do artigo anterior apresentam, em [Erman et al., 2006], um estudo das características de tráfego do BitTorrent. Traços em nível de enlace e os mesmos dados coletados no estudo acima foram utilizados como base para a análise. Os resultados mostram que a taxa entre chegadas (interarrival rate) pode ser modelada segundo uma distribuição hiper-exponencial de segunda ordem, ao passo que a duração e tamanho das sessões podem ser modeladas com distribuições Weibull e log-normal misturadas, os tempos de reposta podem ser modelados por misturas de distribuições log-normal dual (*dual log-normal mixture*) e a taxa de requisições pode ser modelada segundo uma distribuição dual gaussiana.

No trabalho apresentado em [Daoying et al., 2006], os autores concentram sua atenção em três mensagens do protocolo BitTorrent: **Have**, **Request** e **Piece**. Estas mensagens servem para, respectivamente, anunciar a posse de uma peça, solicitar um bloco e enviar um bloco solicitado. Estas três mensagens são as mais utilizadas enquanto um par está participando de um enxame. Como forma de melhorar o desempenho, os autores sugerem a criação de um novo tipo de mensagem, chamado de **MultiHave**, no qual uma mensagem contendo o anúncio da posse de várias peças é enviada em intervalos regulares. O intervalo sugerido é de 5 segundos, metade do tempo de uma rodada, isto é, um par irá receber em torno de duas mensagens **MultiHave** entre cada avaliação realizada. Além disso, no modelo proposto, os pares

podem utilizar um esquema híbrido, com mensagens de **Have** para alguns pares e **MultiHave** para outros, utilizando o desempenho de cada par como métrica.

Uma avaliação, em nível de rede de diferentes mecanismos de distribuição de conteúdo é apresentada em [Das and Kangasharju, 2006]. No artigo, a arquitetura BitTorrent é comparada com arquiteturas tradicionais de distribuição em Unicast, Multicast e com um novo modelo analítico proposto. Este modelo foi criado para representar a quantidade de tráfego gerada pelas diferentes abordagens. O custo, que é o fator avaliado, leva em consideração somente o número de saltos entre Sistemas Autônomos (AS) necessários para distribuir um arquivo para todos os pares interessados. Como forma de avaliação, vários experimentos foram realizados utilizando-se um simulador de topologias de rede entre AS denominado BRITE [Medina et al., 2001]. O modelo proposto, chamado de *Peer-Assisted Content Distribution Network*, é baseado na lógica do BitTorrent, em que pares recebem mas também fornecem conteúdo, mas diferencia-se no fato de a origem do conteúdo residir em determinados servidores estrategicamente localizados. O tracker por sua vez, quando solicitado, devolve uma lista dos pares "mais próximos" ao par requisitante. Os resultados mostram que este mecanismo deve ser utilizado com cautela, mas que um modelo cuidadosamente planejado pode apresentar desempenho próximo ao de sistemas multicast.

Em [Iosup et al., 2006] é abordado o uso da informação de localização geográfica dos pares. Os autores buscam: (a) verificar a relação existente entre a rede de sobreposição criada e as milhares de redes que compõem a Internet e (b) caracterizar o comportamento dos usuários nelas existentes. Para isto, um arcabouço de medição de sistemas P2P, que realizam compartilhamento de arquivos, foi proposto e implementado. Após, este arcabouço, chamado de MULTIPROBE, foi utilizado para realizar medições na rede BitTorrent. As principais conclusões do artigo são: (a) a maioria dos pares estão localizados na Europa, mas o tráfego gerado pela arquitetura está globalmente espalhado; (b) pares estão, em geral, bem conectados em termos de características geográficas; (c) na média, a velocidade de

download é de 500 kbps; (d) o número médio de conexões que um par mantém independe do tamanho do enxame; e (e) BitTorrent apresentou uma mudança no uso de portas TCP de valores fixos para dinâmicos.

Em [Eger and Killat, 2006], os autores concentram-se no esquema de troca do BitTorrent (pares fornecendo upload uns aos outros em troca de download). Conforme trabalhos anteriores, os autores mostram que em alguns casos o mecanismo de incentivo não garante justiça na utilização dos recursos. Dois novos mecanismos de incentivo são propostos, ambos baseados em modelos econômicos, onde os pares ganham crédito ao fornecer upload e gastam-no ao realizar download. Dos mecanismos propostos, o primeiro, chamado *Resource Pricing*, baseia-se apenas na informação local do par e o segundo, denominado *Reciprocal Rate Control*, utiliza também as taxas de download informadas pelos outros pares. Em ambos os casos, pares carona são punidos mais severamente que no esquema empregado atualmente no BitTorrent (tit-for-tat), tendo o primeiro modelo se mostrado mais justo também na alocação dos recursos fornecidos por pares altruístas (semeadores no caso do BitTorrent).

Os trabalhos mais fortemente relacionados com esta dissertação são os apresentados em [Luan and Tsang, 2006] e [Liogkas et al., 2006]. No primeiro deles, o foco reside na questão da distribuição dos blocos entre os pares. Embora seja senso comum que o protocolo BitTorrent deva fornecer um espalhamento mais ou menos homogêneo das peças, o estudo mostra, através de simulação, que esta premissa não é verdadeira e que: (a) algumas peças podem tornar-se raras enquanto outras estão bem distribuídas e (b) as peças tendem a aglomerar-se de acordo com a topologia da rede. Um esquema baseado em source coding é proposto e avaliado. Este modelo mostra-se superior em termos de desempenho, mas em alguns casos ainda não é efetivo em evitar o aparecimento do problema das peças raras. Na conclusão, os autores sugerem ainda outras duas formas de aliviar o problema do espalhamento desproporcional das peças: (a) as peças necessárias por pares com maior capacidade de download e upload deveriam ser prioritárias, visando assim aumentar a sua

velocidade de espalhamento e (b) pares com maiores taxas de upload deveriam passar a fornecer conteúdo a mais de 5 pares remotos ao mesmo tempo.

Já Liogkas e outros [Liogkas et al., 2006] apresentam três técnicas que um par poderia empregar no sentido de aproveitar mais da rede do que contribuir para a mesma. Na primeira delas, um par pode escolher tentar realizar download apenas de semeadores, na segunda realizar download apenas dos pares mais rápidos, e na terceira anunciar falsamente peças com a intenção de permanecer interessante para os pares remotos e obter download destes através do dessufocamento otimista. A avaliação do impacto dessas técnicas é realizada através de experimentos em dois cenários diferentes: enxames privados utilizando o PlanetLab [PlanetLab, 2007] e enxames públicos disponíveis na Internet. Os resultados mostram que somente em alguns poucos casos as técnicas empregadas apresentaram ganho significativo para pares desonestos. Casos envolvendo combinação das técnicas não foram estudados.

Tal como o trabalho apresentado por Liogkas e outros, a presente dissertação explora técnicas em que um par não respeita o protocolo e "mente" (sobre a disponibilidade de peças) ou se recusa a cooperar. Diferentemente dele, no entanto, o anúncio falso de peças é utilizado aqui para prejudicar, ou mesmo evitar, o espalhamento de conteúdo. Além disso, outros cenários são neste trabalho explorados, como aqueles no qual a quantidade de atacantes é muito superior a de honestos, causando um "isolamento" dos pares honestos (vide Seção 3.2).

Capítulo 8

Considerações Finais

Esta dissertação trata de vulnerabilidades em BitTorrent, apresentando quatro contribuições principais. Primeiro, permite aumentar o entendimento sobre BitTorrent, documentando o comportamento de implementações populares e criando diagramas que auxiliam na compreensão do mesmo. Segundo, são identificadas vulnerabilidades e descritos ataques a esta arquitetura, com contramedidas sendo sugeridas para cada um dos casos. Trabalhos anteriores se concentram na avaliação do mecanismo de incentivo, enquanto este representa a primeira contribuição em termos de tratar ataques ao BitTorrent. Terceiro, foi criada uma simulação discreta, que permite mapear o comportamento do protocolo BitTorrent, tendo sido ela validada comparando-se seus resultados com uma avaliação experimental em ambiente controlado. A implementação e todos os scripts utilizados para a preparação, execução e análise dos resultados obtidos serão disponibilizados sob licença Creative Commons [CreativeCommons, 2007]. Quarto, o impacto de dois dos principais ataques identificados foi avaliado via simulação. A escolha desses ataques deveu-se, principalmente, ao impacto potencial que apresentaram.

Os resultados da avaliação demonstraram que BitTorrent é suscetível a ataques em que pares maliciosos mentem a posse de peças e tornam-nas mais raras, fazendo com que downloads em geral sejam atrasados em até 50%. Mostrou-se que esse ataque é efetivo ainda que seja utilizado um número reduzido de maliciosos. Entretanto, mesmo em cenários extremos onde o número de mentirosos é igual ao

número de pares corretos, não foram registrados casos em que um par não completa o download. Em contraste, tais falhas foram amplamente verificadas em ataques de Eclipse, em que o número de pares maliciosos suplantou largamente o número de honestos.

Este trabalho pode ser estendido de várias formas. Primeiro, não há como garantir que todos os ataques possíveis foram identificados; outras vulnerabilidades não foram tratadas nesta dissertação, por uma questão de foco; e uma investigação mais aprofundada do comportamento do BitTorrent, sob diferentes condições, pode permitir a descoberta de outras possibilidades de ataques.

Segundo, a avaliação de impacto foi, nesta dissertação, restrita a um conjunto de cenários simples, predominantemente homogêneos. Nesse contexto, vislumbra-se como continuidade ao trabalho a investigação de cenários mais ricos, particularmente com enxames de vida longa que se reciclam através de várias "gerações" de semeadores e sugadores e com ataques combinando duas ou mais estratégias de subversão.

Terceiro, o trabalho deve ganhar prosseguimento também através da validação dos mecanismos propostos como contramedidas, de maneira a avaliar a efetividade dos mesmos em eliminar ou reduzir o impacto dos ataques. Embora as propostas apresentadas nesta dissertação constituam contribuição científica relevante, é importante que os ataques e contramedidas sejam devidamente avaliadas via modelagem e simulação.

Bibliografia

[Andrade et al., 2004] Andrade, N., Brasileiro, F., Cirne, W., and Mowbray, M. (2004). Discouraging free riding in a peer-to-peer cpu-sharing grid. In *13th IEEE Symposium on High Performance Distributed Computing (HPDC'04)*, pages 129–137.

[Andrade et al., 2005] Andrade, N., Mowbray, M., Lima, A., Wagner, G., and Ripeanu, M. (2005). Influences on cooperation in bittorrent communities. In *Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems (SIGCOMM 2005)*, pages 111–115.

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

[Barbera et al., 2005] 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 and Gaspary, 2006] Barcellos, M. P. and Gaspary, L. P. (2006). *Fundamentos, Tecnologias e Tendências rumo a Redes P2P Seguras*, volume 1, chapter 4, pages 1–57. Ed PUC-RIO.

[Bazzi and Konjevod, 2005] Bazzi, R. A. and Konjevod, G. (2005). On the establishment of distinct identities in overlay networks. In *Proceedings of the 24th annual ACM symposium on Principles of distributed computing*, pages 312–320, New York, NY, USA. ACM Press.

[BBC, 2006] BBC (2006). Bbc moves to file-sharing sites. http://news.bbc.co.uk/2/hi/technology/6194929.stm.

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

[Bittorrent, 2007] Bittorrent (2007). Bittorrent website. http://www.bittorrent.com.

[BitTorrentDHTProtocol, 2007] BitTorrentDHTProtocol (2007). Bittorrent dht protocol. http://www.bittorrent.org/Draft_DHT_protocol.html.

[BitTorrentOrg, 2007] BitTorrentOrg (2007). Bittorrent.org. http://www.bittorrent.org/protocol.html.

[BitTorrentProtocolSpecification, 2007] BitTorrentProtocolSpecification (2007). Bittorrent protocol specification v1.0. http://wiki.theory.org/BitTorrentSpecification.

[Borisov, 2006] Borisov, N. (2006). Computational puzzles as sybil defenses. In *6th IEEE International Conference on Peer-to-Peer Computing, 2006 (P2P 2006)*, pages 171–176.

[CacheLogic, 2006] CacheLogic (2006). Cachelogic and bittorrent announce strategic partnership to advance the adoption of peer-assisted content delivery. http://www.cachelogic.com/home/pages/news/pr070806.php.

[CacheLogic, 2007] CacheLogic (2007). Cachelogic: advanced solutions for p2p networks. http://www.cachelogic.com.

[Cohen, 2003] Cohen, B. (2003). Incentives build robustness in bittorrent. In *Proceedings of the 1st Workshop on the Economics of Peer-to-Peer Systems*, pages 116–121, Berkeley, CA.

[CreativeCommons, 2007] CreativeCommons (2007). Creative commons. http://creativecommons.org/.

[Danezis et al., 2005] Danezis, G., Lesniewski-Laas, C., Kaashoek, F. M., and Anderson, R. (2005). Sybil-resistant dht routing. In *10th European Symposium on Research in Computer Security (ESORICS 2005)*, LNCS 3679, pages 305–318. Springer.

[Daoying et al., 2006] Daoying, H., Jianyong, L., Jianchun, L., Chengren, L., Jie, Z., and Xiancen, G. (2006). Multihave: The extension of peer protocol in bittorrent system. In *5th International Conference on Grid and Cooperative Computing Workshops, 2006 (GCCW '06)*, pages 39–43.

[Das and Kangasharju, 2006] Das, S. and Kangasharju, J. (2006). Evaluation of network impact of content distribution mechanisms. In *Proceedings of the 1st international conference on Scalable information systems (InfoScale 2006)*, New York, NY, USA. ACM Press.

[Das et al., 2006] 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.

[Daswani and Molina, 2004] Daswani, N. and Molina, H. G. (2004). Pong-cache poisoning in guess. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 98–109. ACM Press.

[Dinger and Hartenstein, 2006] Dinger, J. and Hartenstein, H. (2006). Defending the sybil attack in p2p networks: taxonomy, challenges, and a proposal for self-registration. In *First International Conference on Availability, Reliability and Security (ARES 2006)*, pages 756–763.

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

[eDonkey, 2007] eDonkey (2007). edonkey2000 - overnet website. http://edonkey2000.com/.

[Eger and Killat, 2006] 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.

[Erman et al., 2005] Erman, D., Ilie, D., and Popescu, A. (2005). Bittorrent session characteristics and models. In *Proceedings of the 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, pages P30/1–P30/10.

[Erman et al., 2006] Erman, D., Ilie, D., and Popescu, A. (2006). Bittorrent traffic characteristics. In *International Multi-Conference on Computing in the Global Information Technology, 2006 (ICCGI 2006)*, pages 42–42, Washington, DC, USA. IEEE Computer Society.

[Fan et al., 2006] Fan, B., Chiu, D.-M., and Lui, J. C. S. (2006). Stochastic differential equation approach to model bittorrent-like p2p systems. In *2006 IEEE International Conference on Communications*, volume 2, pages 915–920.

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

[Guo et al., 2005] 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.

[Helm, 2006] Helm, B. (2006). Bittorrent goes hollywood.

[Hofstadter, 1983] Hofstadter, D. R. (1983). The prisoner's dilemma computer tournaments and the evolution of cooperation. *Scientific American*, pages 16–26.

[Iosup et al., 2006] Iosup, A., Garbacki, P., Pouwelse, J., and Epema, D. (2006). Correlating topology and path characteristics of overlay networks and the internet. In *6th IEEE International Symposium on Cluster Computing and the Grid Workshops, 2006*, volume 2, pages 10–10.

[Jun and Ahamad, 2005] 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.

[Kurose and Ross, 2004] Kurose, J. F. and Ross, K. W. (2004). *Computer Networking: A Top-Down Approach Featuring the Internet*. Pearson Benjamin Cummings.

[Legout et al., 2006] Legout, A., Urvoy-Keller, G., and Michiardi, P. (2006). Rarest First and Choke Algorithms Are Enough. Technical report, INRIA.

[Lily, 2006] Lily (2006). Bittorrent strikes digital download deals with 20th century fox, g4, kadokawa, lionsgate, mtv networks, palm pictures, paramount and starz media.

[Liogkas et al., 2006] 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)*.

[Lua et al., 2005] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93.

[Luan and Tsang, 2006] Luan, H. and Tsang, D. H. K. (2006). A simulation study of block management in bittorrent. In *Proceedings of the 1st international conference on Scalable information systems (InfoScale 2006)*, New York, NY, USA. ACM Press.

[Maymounkov and Mazières, 2002] Maymounkov, P. and Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *1st International Peer-to-Peer Symposium (IPTPS 2002)*, pages 53–65.

[Medina et al., 2001] Medina, A., Lakhina, A., Matta, I., and Byers, J. (2001). Brite: an approach to universal topology generation. In *9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001*, pages 346–353.

[μTorrent, 2007] μTorrent (2007). μtorrent website. http://www.utorrent.com/.

[Nussbaum and Richard, 2006] Nussbaum, L. and Richard, O. (2006). Lightweight emulation to study peer-to-peer systems. In *20th International Parallel and Distributed Processing Symposium, 2006 (IPDPS 2006)*, pages 8 pp.+, Washington, DC, USA. IEEE Computer Society.

[Paul, 2006] Paul, R. (2006). Cable company tries out content distribution with bittorrent.

[PlanetLab, 2007] PlanetLab (2007). Planetlab website. http://www.planet-lab.org.

[Purandare and Guha, 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.

[Sadok et al., 2005] Sadok, D., Kamienski, C. K., Souto, E., Rocha, J., Domingues, M., and Callado, A. (2005). Colaboração na internet e a tecnologia peer-to-peer. In *XXV Congresso da Sociedade Brasileira de Computação (SBC 2005)*, pages 1407–1454.

[Sieka et al., 2004] Sieka, B., Kshemkalyani, A. D., and Singhal, M. (2004). On the Security of Polling Protocols in Peer-to-Peer Systems. In *Peer-to-Peer Computing (IEEE P2P 2004)*, pages 36–44.

[Simmcast, 2007] Simmcast (2007). Simmcast: an object-oriented simulation framework for protocol and network research. http://inf.unisinos.br/~Simmcast/.

[Singh et al., 2004] Singh, A., Castro, M., Druschel, P., and Rowstron, A. (2004). Defending against eclipse attacks on overlay networks. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC*.

[Singh et al., 2006] 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.

[Stutzbach and Rejaie, 2006] Stutzbach, D. and Rejaie, R. (2006). Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC 2006)*, pages 189–202, New York, NY, USA. ACM Press.

[Tian et al., 2006] Tian, Y., Wu, D., and Ng, K.-W. (2006). Analyzing multiple file downloading in bittorrent. In *International Conference on Parallel Processing, 2006 (ICPP 2006)*, pages 297–306, Washington, DC, USA. IEEE Computer Society.

[Wallach, 2002] Wallach, D. S. (2002). A survey of peer-to-peer security issues. In *International Symposium on Software Security*, pages 42–57.

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

[YouTube, 2007] YouTube (2007). Youtube website. http://www.youtube.com/.

[Yu et al., 2006] Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. (2006). Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 267–278, New York, NY, USA. ACM Press.

[Zudeo, 2007] Zudeo (2007). Zudeo website. http://www.zudeo.com/.