Capítulo 6

Swarming: como BitTorrent Revolucionou a Internet

Matheus B. Lehmann, Rodrigo B. Mansilha,
Marinho P. Barcellos, Flávio Roberto Santos

Abstract

BitTorrent is a file sharing Peer-to-Peer (P2P) application that has achieved great popularity, becoming a de facto *standard for swapping files in the Internet. To achieve such popularity, BitTorrent was built upon a solid scientific contribution, through the proposal of a novel swarming-based protocol, and was supported by an open and sufficiently robust implementation. This chapter collects, organizes, and presents in a top-down, structured way the inumerous advances provided by this technology since its inception, in 2001. The contents are presented in breadth, whenever possible generalizing the useful lessons to other Internet applications. The approach to the subject is top-down and combines theory and practice, analyzing how the proposed policies and mechanisms were actually implemented and to which degree they were successful.*

Resumo

BitTorrent é uma aplicação de compartilhamento de arquivos Peer-to-Peer (P2P) de grande popularidade, tendo se tornado um padrão de facto *para troca de arquivos via Internet. Para atingir tal popularidade, BitTorrent partiu de uma sólida contribuição científica, com a proposta de um protocolo baseado em* swarming*, e aliou-se a uma implementação aberta, funcional e suficientemente robusta do protocolo. Este minicurso coleta, organiza, e apresenta de forma estruturada os inúmeros avanços propiciados por essa tecnologia desde sua introdução, em 2001. Os conteúdos são apresentados de forma abrangente, sempre que possível generalizando as lições a outras aplicações da Internet. O tratamento dado ao tema é* top-down *e combina teoria e prática, analisando como as políticas propostas em artigos foram implementadas e que grau de sucesso tiveram.*

6.1. 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 para diferenciar classes de sistemas de rede: Cliente-Servidor e *Peer-to-Peer*. O primeiro, predominante por muitos anos, se baseia em um servidor com endereço bem conhecido e que fica permanentemente *online* aguardando 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). Além disso, pares são relativamente autônomos, possuem fraco acoplamento entre si e nenhuma garantia de que permanecerão disponíveis *online*. Por um lado, esses fatores trazem vantagens potenciais como escalabilidade e ausência de um ponto central de falhas, mas por outro introduzem novos desafios como, por exemplo, gerenciar uma população transiente de pares [Lua et al. 2005, Barcellos e Gaspary 2006].

Aplicações P2P atingiram grande popularidade, em parte devido às possibilidades de compartilhamento direto de recursos (incluindo arquivos) entre usuários, e em parte pela eficiência de tais sistemas ao lidar com grandes volumes de usuários. Um dos reflexos dessa popularidade é o grande volume de tráfego gerado na Internet por essas aplicações, conforme indicado por [Schulze e Mochalski 2009]. Por exemplo, a Figura 6.1 demonstra a distribuição de "tráfego P2P" nos cinco continentes ao longo de 2008 e 2009. Nota-se que o tráfego P2P mundial corresponde a pelo menos 43% do tráfego geral, podendo chegar até a 70% na Europa.

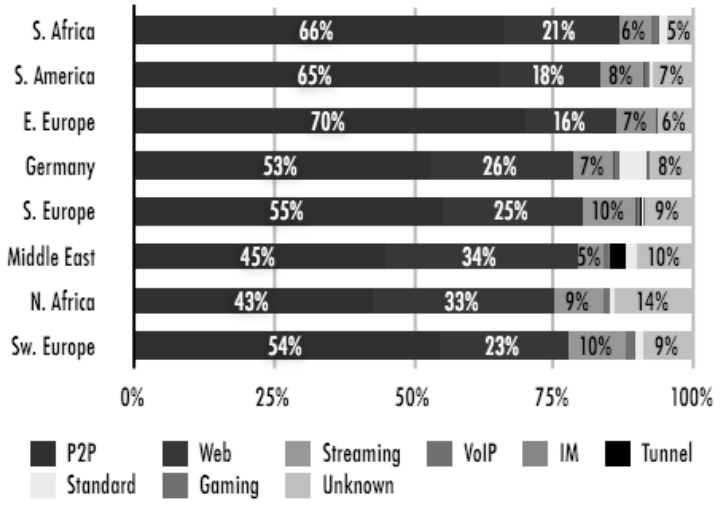

Figura 6.1. Volume de tráfego P2P [Schulze e Mochalski 2009]

Uma fração significativa desse tráfego P2P é oriunda de aplicações de compartilhamento de arquivos. Existem diversas aplicações bem sucedidas nessa classe, porém o BitTorrent é, de acordo com várias evidências, a que possui maior destaque e a que domina em volume de tráfego trocado [Schulze e Mochalski 2009, Sandvine 2010, Envisional 2011].

Com BitTorrent, foram alcançados níveis de compartilhamento de arquivos e colaboração entre usuários inéditos até então. Esse feito foi possível graças às suas diversas inovações, entre as quais destacam-se duas. A primeira é a introdução do conceito de **enxame** (de *swarming*), o que permitia a um usuário contribuir com outros mesmo antes de concluir o download de um arquivo. O segundo aspecto é a separação dos mecanismos de busca de conteúdo e de transferência de dados, o que espalhou o índice de conteúdos (na forma de arquivos de metadados denominados torrent) na Internet, em inúmeros sites e máquinas. Diferentemente de muitas ideias inovadoras que acabaram não vingando, o BitTorrent foi amplamente utilizado e representou uma revolução em aplicações da Internet.

Este capítulo aborda BitTorrent, um tema bastante atual, explicando como e porque essas inovações atraíram significativa atenção tanto da comunidade científica quanto da indústria. Apresenta-se um apanhado geral de resultados de pesquisa e tecnologias relacionadas ao BitTorrent, amparado em exemplos provenientes da vasta gama de implementações disponíveis e em operação atualmente.

O restante do capítulo segue uma organização *top-down*. A Seção 6.2 faz um resgate histórico sobre a evolução do compartilhamento de arquivos P2P e o surgimento do BitTorrent. A Seção 6.3 aborda componentes do BitTorrent com foco nos usuários, incluindo comunidades, agentes e o comportamento de usuários e o seu impacto sobre o BitTorrent. Os diversos componentes que formam o universo BitTorrent são apresentados na Seção 6.4, primeiro de uma perspectiva arquitetural, e após com foco nos protocolos que governam a comunicação entre os componentes. Na Seção 6.5, são tratadas as políticas e mecanismos usados e responsáveis pelo sucesso do protocolo. A Seção 6.6 identifica e discute os principais tópicos de pesquisa que têm sido abordados na pesquisa sobre BitTorrent, evoluindo até o estado-da-arte. A Seção 6.7 apresenta as considerações finais.

6.2. Histórico

Os primeiros sistemas de compartilhamento de arquivos passaram por uma série de melhorias ao longo dos anos. As subseções 6.2.1 e 6.2.2, respectivamente, apresentam um breve histórico da evolução dos sistemas de compartilhamento de arquivos e como se deu o surgimento e evolução do BitTorrent.

6.2.1. Compartilhamento de arquivos

Em 1999, Shawn Fanning e Sean Parker criaram um serviço inovador de compartilhamento de músicas. O serviço, denominado Napster, foi a primeira rede P2P de compartilhamento de arquivos e quebrou definitivamente o paradigma de distribuição de conteúdo. Entretanto, Napster é classificado por muitos autores como um sistema P2P híbrido, porque apesar dos pares trocarem arquivos diretamente entre si, formando uma rede P2P, eles dependem de um servidor central, responsável por indexar os usuários e seus respectivos conteúdos compartilhados. Como o custo do serviço de indexação é mínimo em relação ao custo do armazenamento e distribuição de conteúdos, o Napster obteve boa escalabilidade, alcançando a marca de 25 milhões de usuários e uma coleção de 80 milhões de músicas [Dailytech 2008]. Assim, Napster foi responsável por provar que redes P2P são

viáveis, mas também por atrair a atenção da indústria para a distribuição de conteúdo protegido por *copyright* na Internet. Em 2001, o Napster perdeu uma ação na justiça e foi obrigado a retirar do ar seu serviço de indexação, tornando a rede P2P de compartilhamento indisponível [ABCNews 2001].

A fim de evitar problemas judiciais como aqueles enfrentados pelo Napster, outros sistemas foram desenvolvidos de maneira a descentralizar o mecanismo de indexação e busca de conteúdos. A rede Gnutella [Gnutella 2003] é o primeiro exemplo dessa abordagem completamente descentralizada. Contudo, sua primeira versão oferecia uma qualidade de experiência aquém do desejado, principalmente se comparada ao Napster: a busca era demorada e as respostas variavam a cada interação. Os atrasos eram intrínsecos à abordagem adotada pelo Gnutella na localização de pares com o arquivo desejado: buscas eram transmitidas de um par a um subconjunto de pares vizinhos, e assim sucessivamente, limitando o número máximo de saltos entre pares. Consultas apresentavam grande variabilidade em tempo de resposta e qualidade dos resultados, dependendo das escolhas (aleatórias) de vizinhos. Além disso, era comum não encontrar um arquivo, mesmo que ele estivesse presente na rede, devido ao escopo limitado das buscas.

Para melhorar a qualidade experiência do usuário, um novo conceito, denominado "Super Pares", foi introduzido em 2001 pela rede Kazaa [Yang e Garcia-Molina 2003]. Ele consiste em organizar a rede em dois níveis hierárquicos: o mais baixo, com as funções usuais de compartilhamento, e o mais alto, com a responsabilidade adicional de indexar os conteúdos. O nível mais alto é composto por super pares: um percentual pequeno de pares escolhidos através de um protocolo que leva em consideração a disponibilidade de recursos, a estabilidade e o tempo *online* de cada par. Essa abordagem torna a consulta mais eficiente porque o número de pares alcançados é restrito e as chances desses pares terem disponibilidade para responder à consulta é maior. Os desenvolvedores do Gnutella aplicaram o mesmo princípio de super pares na segunda versão do protocolo [Gnutella2 2003].

Com o sucesso do Kazaa e da nova versão do Gnutella, a indústria de conteúdo obrigou-se novamente a entrar na disputa contra usuários P2P. Porém, dessa vez não havia uma empresa inimiga a ser atacada judicialmente. Por isso, uma nova estratégia foi introduzida: *poluição de conteúdo*. O princípio da poluição é inundar a rede com conteúdos falsos e, dessa forma, frustrar os usuários, dissuadindo-os de usar as redes de compartilhamento para obter conteúdo ilegalmente. Essa forma de "defesa" através de ataques poderia atrair notícias pejorativas e repercutir desfavoravelmente entre os próprios clientes das gravadoras e demais detentoras dos direitos autorais. Não surpreende, portanto, que surgiram companhias (por exemplo, [Overpeer 2002, Mennecke 2004]) que vendiam serviços de combate à cópia ilegal por compartilhamento de arquivos. Embora essa hipótese não possa ser comprovada, acredita-se que a estratégia de lançar ataques de poluição seja utilizada sistematicamente pela indústria cinematográfica e musical desde então.

6.2.2. Surgimento do BitTorrent

Em 2001, Bram Cohen desenvolveu um protocolo para compartilhamento P2P, que ele cunhou BitTorrent, e disponibilizou uma versão funcional de um agente de usuário que implementava o mesmo. O código-fonte, igualmente disponibilizado, permitia que outros

usuários desenvolvessem suas próprias versões do agente de acordo com suas necessidades. Além disso, a estrutura da implementação permitia extensões ao protocolo reduzindo a chance de incompatibilidades com pares executando a versão original. Em 2003, Cohen publicou um artigo seminal [Cohen 2003] descrevendo os princípios básicos do protocolo que ele havia implementado, tratando em particular de questões como incentivos à colaboração entre pares.

Entre as diversas inovações trazidas pelo BitTorrent, dois aspectos foram fundamentais para seu sucesso. O primeiro é a introdução do conceito de *swarming*, o que permitia a um usuário contribuir com outros mesmo antes de concluir o download de um arquivo. Na prática, a largura de banda de upload dos pares passou a ser aproveitada *durante* o processo de download, o que aumentou significativamente a quantidade de pares contribuindo recursos e por conseguinte a eficiência do sistema como um todo [Massoulié e Vojnović 2005]. Em downloads mais longos, havia mais chance de um par contribuir para a disseminação do arquivo em questão.

O segundo aspecto é a separação dos mecanismos de busca de conteúdo e de troca de dados. Na prática, o índice de conteúdos do BitTorrent (metadados descrevendo arquivos, indicação de localização) encontra-se distribuído na Internet, em vários sites e máquinas. A localização de conteúdo passou a ser feita através de máquinas convencionais de busca, conforme descrito a seguir.

Um conteúdo é encontrado na Internet procurando-se por arquivos *torrent*. Um arquivo torrent reúne a descrição e informações necessárias para que os usuários possam obter determinado conteúdo. Em 2002 surgiu o primeiro site cujo objetivo único era simplificar a publicação e a localização de arquivos torrents, agregando interessados em compartilhamento de arquivos via BitTorrent em uma *comunidade* de usuários. O site, denominado *Suprnova*, alcançou grande sucesso e popularidade até dezembro de 2004, quando sofreu um processo judicial. Como resultado, o site teve que ser fechado [TorrentFreak 2006], fato que teve grande repercussão na mídia na época e possivelmente atraiu mais atenção do público ao BitTorrent.

O vácuo deixado pelo fechamento do Suprnova criou uma oportunidade, e logo surgiram novos portais BitTorrent. Além das comunidades de propósito geral, surgiram outras criadas para atingir nichos específicos, de usuários interessados em determinados tipos de conteúdos, como por exemplo distribuições Linux [Linuxtracker 2011] ou gravações amadoras de shows musicais.

Entre as comunidades, duas das mais populares e marcantes foram *Mininova* e *The Pirate Bay*. Com a crescente popularização do BitTorrent, ambas obtiveram mais notoriedade que o Suprnova. No rastro de sua criação, o Mininova foi alvo de um processo judicial, e em 2009 seus administradores foram obrigados a retirar todo conteúdo ilegal do site [TorrentFreak 2009c]. O The Pirate Bay sofreu, igualmente, uma série de processos judiciais, mas de alguma forma sempre conseguiu escapar de punições. Segundo seus defensores, The Pirate Bay representa uma liderança no combate às práticas das grandes empresas e indústria de conteúdo, que procede de forma abusiva ao comercializar obras musicais e cinematográficas.

Já a visão do órgão responsável pela defesa dos direitos autorais nos Estados Uni-

dos, a RIAA – *Recording Industry Association of America* – condena todo tipo de pirataria, que segundo ela gera prejuízo anual em torno de 12,5 bilhões de dólares apenas nos Estados Unidos. Como forma de resposta à pirataria, é utilizada uma abordagem baseada em três pontos: aplicação das leis de *copyright* através de confisco de produtos piratas e processos judiciais, educação das pessoas sobre a lei e formas legais de adquirir conteúdo, e inovação na formas legais de disponibilização de conteúdo digital [RIAA 2011].

Portais como o The Pirate Bay eram abertos à comunidade de usuários em geral, não requerendo registro de usuários para acesso ao portal. É possível estimar que essa "abertura" tenha contribuído significativamente para a rápida popularização das comunidades e de BitTorrent. Por outro lado, a repercussão levou a uma série de problemas, como egoísmo e outros abusos cometidos por usuários não registrados.

Como solução paliativa a esses problemas, surgiram as "comunidades fechadas" ou "privadas". Estas caracterizam-se por apenas permitir o acesso a usuários que tenham sido previamente cadastrados. Além disso, tais comunidades às vezes limitam seu tamanho ao exigir de novos usuários um convite para ingresso na comunidade; entre outras razões, a ideia seria evitar a entrada de usuários interessados em monitorar o compartilhamento e/ou prejudicar o mesmo. Para aumentar o nível de colaboração, tanto em riqueza de conteúdo como em taxa de download, as comunidades fechadas monitoram a quantidade de download e upload e aplicam regras para estimular a contribuição mútua entre pares (reciprocidade) [Zhang et al. 2010].

A expansão no número de comunidades abertas e fechadas veio com a popularização no uso de BitTorrent. Reflexo disso, o volume de tráfego gerado na Internet pelo BitTorrent é substancialmente maior se comparado àquele gerado por outros protocolos de compartilhamento de arquivos. A Figura 6.2 [Schulze e Mochalski 2009] apresenta os percentuais de tráfegos gerados por diferentes sistemas de compartilhamento de conteúdo (P2P ou não) no mundo ao longo de 2008 e 2009. Note-se que o tráfego BitTorrent ocupa maior proporção em todos os continentes, com exceção da América Latina, onde leva pequena desvantagem para um outro sistema de compartilhamento P2P denominado Ares Galaxy [Ares 2002].

O uso crescente de ferramentas de compartilhamento de arquivos e, em particular, o BitTorrent elevou os custos relativos para os provedores de Internet (*Internet Service Providers* – ISPs) [Bindal et al. 2006]. Anteriormente, usuários ficavam muito menos tempo *online* e usavam na média uma fração muito menor dos recursos que contratavam, permitindo uma multiplexação dos recursos implantados pelo ISP. Com o compartilhamento de arquivos, muitos usuários começaram a demandar uma fração maior da banda contratada. Motivados principalmente pelo aspecto econômico, mas alegando apoio à proteção dos direitos autorais, os ISPs entraram em uma "briga de gato e rato" que estava se formando entre usuários P2P e a indústria de mídia [Piatek et al. 2009]. Nesta linha, a primeira ação tomada pelas ISPs foi aplicar conformação de tráfego (*traffic shaping*), despriorizando, limitando o volume ou até mesmo bloqueando completamente as comunicações realizadas através do protocolo BitTorrent. Nesse contexto, o caso que ganhou maior repercussão foi o do ISP Comcast, que em 2007 passou a aplicar conformação de tráfego a BitTorrent e Gnutella. No caso particular de BitTorrent, o desempenho de

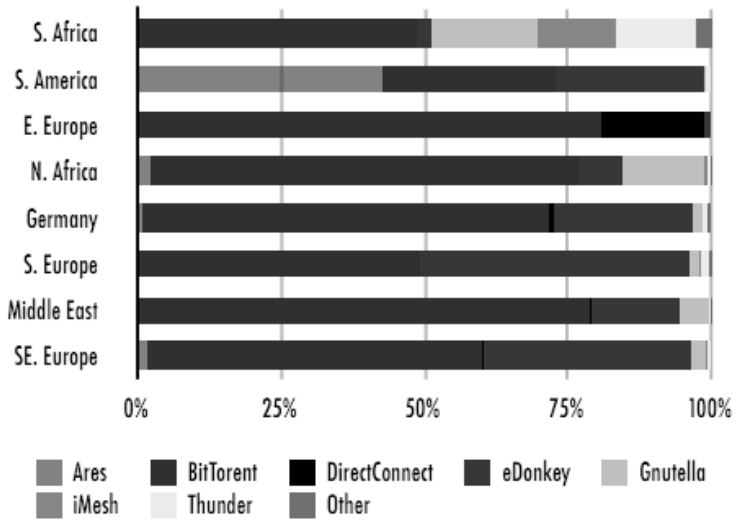

Figura 6.2. Volume de compartilhamento de arquivos [Schulze e Mochalski 2009]

downloads era prejudicado reinicializando-se conexões através de segmentos RST artificialmente construídos e enviados aos agentes de usuário [Weitzner 2008].

Como resposta à conformação de tráfego, os desenvolvedores de agentes de usuário BitTorrent passaram a usar um esquema de seleção dinâmica de portas, dificultando a detecção via faixas de portas. Posteriormente, criaram *plugins* para possibilitar a cifragem de dados na comunicação do protocolo BitTorrent. Tais medidas dificultaram a detecção desse tráfego BitTorrent na rede, embora outras estratégias mais agressivas seriam possíveis, como por exemplo monitorar o padrão de conexões TCP geradas e recebidas, ou simplesmente despriorizar todo tráfego com volume e que não se enquadre em portas bem conhecidas.

A questão sobre atuar ou não em tráfego de usuário de acordo com seu conteúdo faz parte de um debate mais amplo, denominado "neutralidade da rede" [Crowcroft 2007]. Pesquisadores da área de redes de computadores, ISPs e usuários de compartilhamento de arquivos vem debatendo o assunto [Crocioni 2011, Wallsten e Hausladen 2009, Ling et al. 2010, TorrentFreak 2008a, TorrentFreak 2010b, TorrentFreak 2010a].

6.3. Visão do Usuário

Resgatado o histórico, esta seção oferece um primeiro olhar sobre BitTorrent em si, centrado no usuário. São discutidas três questões: comportamento, comunidades BitTorrent e implementações de agentes, através da seguinte estrutura. A primeira subseção discute os tipos de comportamentos de usuário usualmente encontrados em redes BitTorrent. Tais tipos embasam uma discussão sobre as classes de comunidades BitTorrent existentes na Internet, apresentada na segunda subseção. Eles estão também relacionados aos diferentes agentes de usuário existentes, sendo os principais discutidos na última subseção.

6.3.1. Comportamento de usuários

Ao contrário de soluções como RapidShare [Rapidshare 2005] e MegaUpload [Megaupload 2005], em uma rede P2P de compartilhamento de arquivos como BitTorrent não há a figura de um servidor central que armazena arquivos e possui grande quantidade de recursos para dedicar a usuários. Em substituição, cada participante no BitTorrent assume, em maior ou menor grau, o papel de servidor (além do de cliente). Por isso, o sucesso de uma rede BitTorrent está associado a uma relação equilibrada entre demanda de download e oferta de upload por seus participantes. Essa subseção discute o comportamento de usuário quanto ao compartilhamento e as consequências em termos de desempenho e robustez.

Usuários (isto é, pares) podem ser classificados conforme a relação entre upload e download que expressa sua participação na rede. Em um extremo, pares *egoístas*, também conhecidos como "caronas" (de *free riders*) se recusam a cooperar: obtém recursos do sistema, contribuindo o mínimo possível com o mesmo [Locher et al. 2006, Piatek et al. 2007]. No outro, pares *altruístas* colaboram com mais recursos que receberam.

O comportamento altruísta e os altos níveis de cooperação encontrados em redes BitTorrent estão associados a múltiplos fatores. Por exemplo, um par pode inadvertidamente contribuir com upload por várias horas, durante a noite, quando o download encerra sem a presença do usuário [Izal et al. 2004]. Na configuração padrão de agentes, é comum este precisar ser explicitamente fechado após o término do download, caso contrário permanece contribuindo.

Os altos níveis de cooperação em BitTorrent podem também decorrer da separação em múltiplas redes de sobreposição (ou enxames), ao contrário de outras redes de compartilhamento P2P [Hales e Patarin 2005]. Segundo os autores do referido trabalho, usuários podem migrar entre enxames com base no seu desempenho, portanto enxames com alto índice de egoístas tendem a "minguar", perdendo usuários para enxames em que o nível de cooperação for maior.

Além disso, estudos sobre comunidades BitTorrent observaram comportamentos diferentes, entre altruísmo e egoísmo, dependendo do tipo de conteúdo compartilhado [Andrade et al. 2005]. Este mesmo estudo conclui que as comunidades privadas que possuem regras externas de contribuição apresentam igualmente maior nível de altruísmo dos pares.

Subentende-se, portanto, que ao compartilhar um material protegido por *copyright*, o usuário pode se sentir compelido a deixar a rede o mais rápido possível, ou seja, se comportando de forma egoísta. Há outros fatores em potencial [Sirivianos et al. 2007], tais como a assimetria em capacidade de largura de banda tipicamente encontrada em usuários domésticos, com capacidades de download muito superiores às de upload, ou a tarifação por quantidade de tráfego.

O altruísmo é positivo globalmente, porém o sistema precisa lidar com pares que se comportam de maneira egoísta. Uma parte fundamental do BitTorrent é o seu mecanismo de incentivo, que implementa de uma política de reciprocidade. Embora a necessidade de uma política como essa fosse reconhecida anteriormente ao BitTorrent (por exemplo, no Kazaa), nenhum outro sistema foi capaz de implementar um mecanismo de

forma eficaz. O mecanismo, a ser detalhado na Seção 6.5, é viabilizado pelo emprego de *swarming* e considerado um dos fatores que explicam o sucesso do BitTorrent.

O mecanismo de incentivo que busca a reciprocidade entre pares é classificado como TFT – de *Tit-for-Tat* ("olho por olho, dente por dente"). Entretanto, esse atributo foi recentemente questionado, com base no fato que no mecanismo implementado, a reciprocidade não é *proporcional* ao volume de dados contribuído por um par [Levin et al. 2008]. Mais precisamente, os melhores vizinhos são desbloqueados para que recebam dados do par, mas o mecanismo não distingue uma hierarquia de benefício entre eles, contribuindo igualmente aos mesmos. Essa potencial injustiça pode levar a uma situação de altruísmo para determinados pares. Além disso, pares com alta capacidade de upload tendem a se tornar altruístas (contribuem mais do que recebem de outros pares), pois com *Optimistic Unchoking* um usuário contribui com outro sem garantia de reciprocidade [Piatek et al. 2007].

Além da questão sobre o mecanismo de incentivo ser TFT ou não, alguns estudos [Ripeanu et al. 2006, Piatek et al. 2007] contestam a eficácia do mecanismo. Eles propõem que a colaboração entre pares e o baixo nível de pares-carona registrados em certas redes BitTorrent resultam predominantemente de dois fatores não relacionados: o efeito psicológico da penalização a pares-carona, e a utilização de agentes de usuário "*out of the box*", sem ajustar suas configurações ou código para tentar obter vantagens do sistema.

6.3.2. Comunidades

Diferentemente de outros sistemas de compartilhamento de arquivos P2P em que a busca de conteúdo é realizada no próprio sistema (como por exemplo Gnutella e Kazaa), BitTorrent se caracteriza pelo mecanismo de busca ser externo ao protocolo. Para realizar um download, o usuário precisa uma série de informações, como um "ponto de encontro" com outros pares e a descrição do conteúdo (conjunto de arquivos digitais). Essas informações são organizadas em um arquivo de metadados denominado *torrent*. Para cada conteúdo distribuído via BitTorrent, existe um arquivo torrent correspondente; a recíproca não é verdadeira, pois um torrent pode se referir a dados que não estão mais disponíveis na rede. O processo de busca de conteúdo no BitTorrent resume-se à busca por um arquivo torrent que corresponda ao conteúdo desejado.

Um novo arquivo .torrent pode ser criado por qualquer usuário, através de uma facilidade disponível em agentes de usuário. Arquivos torrent podem ser então distribuídos por qualquer meio, por exemplo via email, FTP ou em uma página Web. Para simplificar o intercâmbio de arquivos torrents, surgiram as "comunidades BitTorrent".

As *comunidades* são portais com páginas Web que indexam conteúdos e respectivos arquivos torrents. Essas comunidades tipicamente disponibilizam mecanismos que facilitam a localização de conteúdos, por exemplo sistema de busca, ordenação por popularidade e ordenação por lançamento.

As comunidades podem ser divididas em duas categorias: públicas e privadas (ou fechadas). A primeira categoria engloba todas comunidades abertas em que não existe nenhum tipo de restrição quanto ao uso, tais como Bitsnoop, Mininova e The Pirate

Bay. A segunda, por outro lado, diz respeito a comunidades que restringem o acesso aos conteúdos compartilhados através do registro de seus usuários, como por exemplo: One-BigTorrent e Bitsoup. Algumas comunidades fechadas restringem ainda mais o acesso, permitindo o registro de um novo usuário apenas através de convite (por um usuário já registrado).

As comunidades privadas em geral aplicam regras de uso ortogonais ao protocolo BitTorrent e que visam incentivar os usuários a colaborarem, seja através da publicação de novos conteúdos ou pelo upload de conteúdo. Exemplos de regras são exigir uma relação mínima de upload/download (após um determinado tempo) e forçar um tempo mínimo de compartilhamento após término do download. Cada política possui aspectos positivos e negativos. Um aspecto negativo do primeiro caso é que um par pode ser penalizado caso haja solicitantes insuficientes para alcançar a taxa mínima. O segundo caso resolve esse problema, mas a contribuição pode ser insuficiente se durante o período obrigatório houver poucos solicitantes. Por fim, algumas comunidades oferecem benefícios para usuários que contribuem com a comunidade através de dinheiro.

A existência de comunidades BitTorrent, que disponibilizam torrents, representa uma vantagem sobre as demais tecnologias de compartilhamento de arquivos. Ao contrário dos outros sistemas, no BitTorrent o mecanismo de busca de conteúdo é externo ao protocolo. Isso evita a necessidade de um ponto único de indexação de conteúdo, fator que comprometeu o Napster com processos judiciais. Isso também evita a ineficiência dos protocolos distribuídos de busca baseados em inundação, que tornava a experiência do usuário ruim, como aconteceu com a rede Gnutella.

Uma das questões debatidas legalmente em diversos países é se os portais de comunidades BitTorrent tem ou não culpabilidade na prática de pirataria. Por um lado, os conteúdos protegidos não ficam armazenados nos sites Web das comunidades; por outro, é evidente que os portais estão, ao publicar torrents de conteúdo protegido, contribuindo para que uma violação de *copyright* aconteça [TorrentFreak 2010c].

Além dos processos legais contra os mantenedores dos portais, conforme já discutido, as comunidades representam pontos centrais suscetíveis a ataques anti-pirataria. Como defesa a tais ataques, foram usadas diversas estratégias, incluindo o uso de técnicas de colaboração na identificação de conteúdo falso e a disseminação da base de dados completa de um portal de maneira a permitir que o portal fosse instanciado em outros locais, por outras pessoas. Este foi o caso do The Pirate Bay, que em 2009 teve seus 21,3 GB de dados distribuídos através do próprio BitTorrent. Isso ocorreu como forma de proteger o site devido ao processo de venda e indefinição do futuro da comunidade.

6.3.3. Agentes

Bram Cohen publicou o código-fonte e a especificação do BitTorrent na Internet. Como um diferencial, o protocolo apresentou em sua especificação original a capacidade de ser estendido sem gerar incompatibilidades. Ao contrário de alternativas fechadas anteriores (Napster, Kazaa) ou mesmo abertas (Gnutella), surgiram centenas de implementações de agente de usuário BitTorrent. O fato de o protocolo ser aberto e extensível contribuiu, de forma importante, para o sucesso de BitTorrent. A seguir serão apresentados três dos principais agentes de usuário, escolhidos com base na inovação que trouxeram: Mainline,

μtorrent [μTorrent 2006] e Vuze [Vuze 2003].

O **Mainline** é a primeira implementação do protocolo BitTorrent, e a primeira versão surgiu em 2001. O agente foi desenvolvido na linguagem Python, e era considerada o "padrão" ou Mainline até a versão 5.3 em 2009, quando foi abandonada e substituída pelo μtorrent. A última versão do agente Mainline, 5.3, contava com aproximadamente 45.000 linhas de código, organizadas em torno de 70 módulos. Essa implementação, funcional e eficiente, serviu de base para criação de muitas variantes, tal como BitTornado [BitTornado 2004], e em pesquisas envolvendo metodologia experimental, como por exemplo [Mansilha et al. 2008, Locher et al. 2006, Piatek et al. 2007].

O **μtorrent** é uma implementação que tornou-se famosa por adotar um estilo minimalista, demandando poucos recursos computacionais, e oferecer rápido download para os usuários. Ela foi desenvolvida em linguagem C++ e, após ser comprada por Bram Cohen, passou a ser considerada a versão "oficial" do BitTorrent. Atualmente ele é o agente de usuário mais popular, com mais de 56% de usuários de BitTorrent em 2009, chegando a 100 milhões de usuários em 2011 [TorrentFreak 2009d, TorrentFreak 2011b].

Vuze é uma implementação aberta que tornou-se famosa por incorporar diversas funcionalidades, tal como a visualização da transmissão dos dados. Ela foi desenvolvida na linguagem Java, que contribuiu para sua adoção (multi-plataforma) mas que exibe um custo maior em recursos do que as outras duas citadas anteriormente. Entre as principais novidades introduzidas pelo Vuze estão a busca de conteúdo acoplado à interface e algumas extensões específicas que visam otimizar o desempenho das transferências.

6.4. Universo Torrent

Partindo da visão mais abrangente de usuário discutida na seção anterior, a seguir são introduzidos os componentes que formam a arquitetura BitTorrent, organizados em nível crescente de complexidade. Primeiramente é apresentada uma visão geral dos componentes da arquitetura. Em seguida, é descrito o funcionamento do universo, apresentando detalhes sobre o protocolos executados pelos seus componentes. Assim, a discussão segue para apresentar os protocolos usados pelos rastreadores e depois aqueles usados pelos pares. Finalmente, é discutida a organização dos torrents, e como alguns parâmetros influenciam nos protocolos estudados.

6.4.1. Visão geral

O universo BitTorrent é composto por **enxames**, **pares**, **rastreadores** e **conteúdos**. Um par é um agente de usuário que executa o protocolo e participa de um ou mais enxames, de acordo com o conteúdo que deseja compartilhar. Um par é nomeado semeador, quando possui uma cópia completa do conteúdo, ou sugador, caso contrário. Para ingressar em um enxame, o par tipicamente contata o rastreador e como resposta recebe uma lista de IPs de pares (*peer list*) que participam do enxame. Portanto, o rastreador atua como um "ponto de encontro". Alternativamente, um par pode se valer de uma extensão do protocolo de maneira a diminuir, ou até mesmo evitar, o contato com rastreadores. Há duas extensões que vêm sendo amplamente adotadas; a primeira, denominada *Peer Exchange* (PEX), permite que pares façam diretamente o intercâmbio de lista de pares, e a segunda, geralmente chamada de *trackerless*, permite que os pares se encontrem através de tabelas

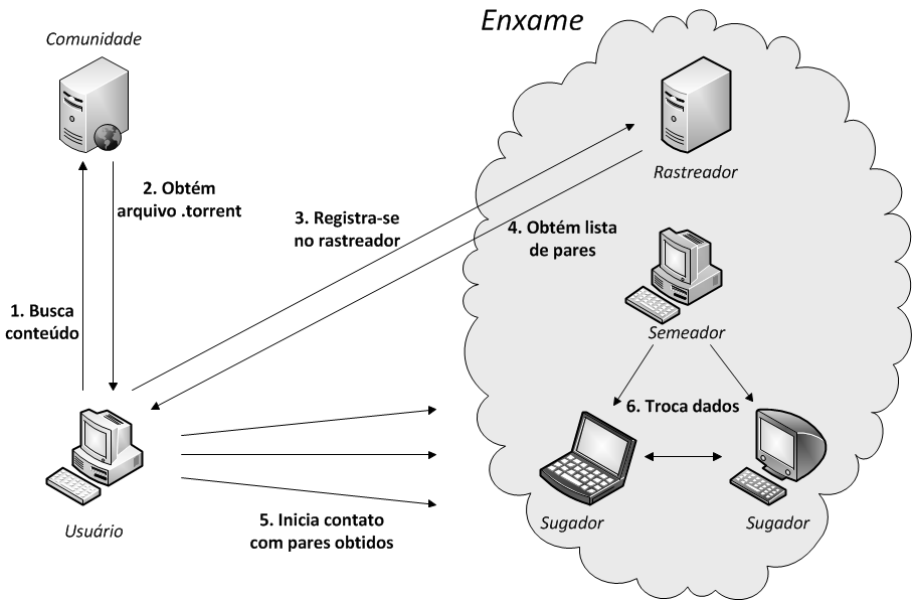

Figura 6.3. Funcionamento do BitTorrent

hash distribuídas (DHTs).

Para participar de um enxame, um agente de usuário utiliza os metadados disponíveis no respectivo arquivo torrent. Esse arquivo contém informações sobre as peças (para cada peça, seu *hash* e tamanho) que formam o conteúdo e arquivos (nomes e tamanhos). Para distribuir um conteúdo, um usuário deve (através de seu agente) gerar um torrent e torná-lo público. Conforme comentado anteriormente, os torrents são tipicamente disponibilizados em sítios dedicados a promover o compartilhamento de arquivos, as comunidades (vide Seção 6.3.2). Além de publicar torrents, algumas comunidades disponibilizam rastreadores. Outras comunidades atuam primariamente como "agregadoras de torrents", indexando informações para permitir buscas e apontando para torrents em outras comunidades. A Figura 6.3 apresenta um exemplo bastante simples da interação entre os elementos. Na prática, múltiplas comunidades podem ser consultadas, assim múltiplos rastreadores podem ser usados em um único torrent.

O universo BitTorrent é ilustrado na Figura 6.4, que representa três cenários diferentes para demonstrar os tipos de formação possíveis com conteúdos, rastreadores e pares. Primeiro, *enxame 1*, que compartilha *conteúdo 1*, mostra o cenário em que um determinado conteúdo é compartilhado por apenas um enxame. Segundo, *enxame 2* e *enxame 3* ilustram o caso em que enxames possuem pares em comum. Note que esses enxames são completamente independentes entre si. Por último, *enxame 4* e *enxame 5* exemplificam o caso em que dois ou mais enxames distintos compartilham o mesmo conteúdo. Este cenário pode surgir em dois casos: (a) quando o tamanho das peças é diferente entre os enxames; ou (b) quando o tamanho das peças é igual, porém os conjuntos de rastreadores empregados são mutuamente exclusivos.

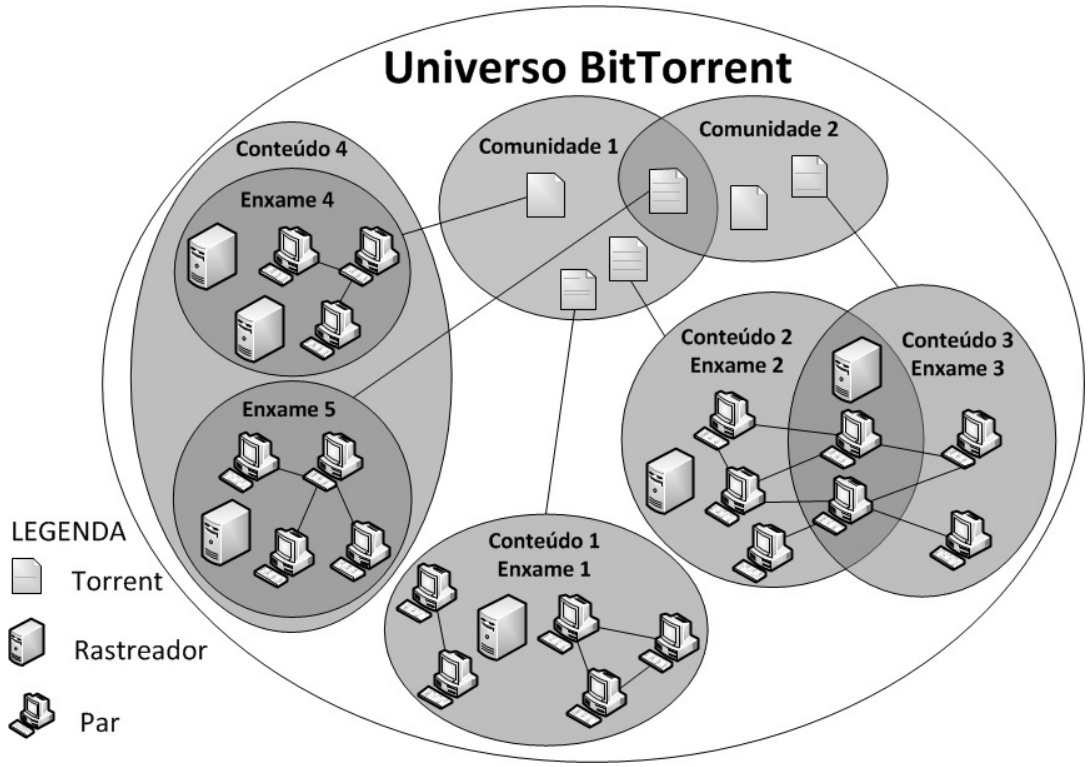

Figura 6.4. Exemplo de componentes e relações no universo de redes BitTorrent

6.4.2. Comunicação com rastreador

Esta subseção aborda aspectos em maior nível de detalhe sobre os protocolos de comunicação entre pares e rastreadores. O rastreador é um elemento centralizado na arquitetura BitTorrent e age como um ponto de encontro entre os pares através de um serviço HTTP/HTTPS que responde a requisições HTTP GET. Este fato não caracteriza um problema a medida que os pares estão distribuídos em centenas ou milhares de enxames e cada enxame é atendido por um conjunto de rastreadores (entre 1 e 8, usualmente). Por outro lado, a recíproca é verdadeira: um mesmo rastreador pode ser responsável por atender um grande número de enxames, o que tipicamente ocorre em rastreadores mantidos por comunidades.

Para cada enxame gerenciado, o rastreador mantém uma lista de usuários participando do mesmo. Como não existe a obrigatoriedade de um par contatar o rastreador ou notificá-lo de sua saída do enxame, a visão do rastreador sobre os usuários presentes no enxame é apenas uma estimativa. Além disso, quando um enxame é gerenciado por múltiplos rastreadores, cada rastreador terá em sua lista apenas um subconjunto de pares, possivelmente com alguma sobreposição entre os conjuntos.

Quando recebe uma solicitação, o rastreador inclui (ou atualiza) o registro do par solicitante na lista de pares do enxame e responde ao requisitante uma lista contendo n pares aleatórios presentes no enxame. O tamanho de um enxame pode variar de nenhum até milhares de pares. No caso de enxames pequenos, quando o número de pares requisitados é menor que o tamanho do enxame, não há escolha de pares e a própria lista é

fornecida. Já no caso de enxames grandes, a medida que cresce o tamanho do enxame, a diversidade entre listas fornecidas é maior, o que aumenta a robustez do sistema, pois não privilegia pares em detrimento de outros.

O contato de um par com o rastreador tem como objetivos atualizar sua situação na lista de pares de determinado enxame e obter endereços de outros pares que também estão no mesmo enxame. Este contato, denominado *announce*, é previsto em cinco casos, conforme segue:

1. no início do download, é necessário um primeiro contato com o rastreador, para que o par se registre na lista do enxame e para obter endereços de outros pares;
2. em intervalos regulares, para o par notificar o rastreador sobre sua presença e obter novos endereços de pares;
3. quando a quantidade de pares ativos cai abaixo de um limite inferior, para obter novos endereços de pares;
4. ao completar o download, para notificar o rastreador que o par se transformou em um semeador;
5. ao deixar o enxame, para notificar "elegantemente" sua saída.

Além do *announce*, os rastreadores atualmente suportam outro tipo de requisição, denominada *scrape*. A resposta a essa requisição informa o número de pares, sugadores e semeadores de um determinado torrent (ou de todos torrents) gerenciados pelo rastreador. Tal requisição é utilizada principalmente por comunidades, com o objetivo de apresentar a popularidade dos enxames a seus usuários.

A seguir são apresentados detalhes dos protocolo de comunicação seguidos pelos rastreadores. Primeiramente é discutido o *announce* e, logo após, o *scrape*.

6.4.2.1. Announce

O *announce* é uma URL formada por um endereço de rastreador contido no arquivo de metadados, seguido pelos parâmetros segundo o padrão de métodos CGI [Robinson e Coar 2004]. A Tabela 6.1 apresenta todos parâmetros possíveis para realizar o *announce* e uma breve descrição de cada um. Destes, os campos fundamentais são: *info_hash*, que identifica o enxame; *peer_id*, que é o identificador do par no enxame; e *port*, que indica a porta na qual o par está "escutando" solicitações de novas conexões. Além destes, há os campos *ip* e *numwant*, que são opcionais e podem ser utilizados para passar o endereço IP do par explicitamente e o tamanho da lista de pares a ser retornada, respectivamente. O campo *ip* é usado principalmente se o IP externo do par de origem é o mesmo IP externo do rastreador, enquanto o *numwant* é usado para obter listas maiores ou menores que o valor padrão, igual a 50.

Os *announces* enviados pelos pares viabilizam que o rastreador, como funcionalidade secundária, registre informações estatísticas sobre o exame. Mais precisamente,

Tabela 6.1. Mensagem *Announce*

Campo	Descrição
info_hash	identificador único do torrent
peer_id	identificador do par requisitante
port	porta em que o par está escutando por novas conexões
ip	opcional, endereço IP do par
numwant	opcional, número de pares desejados
event	opcional, indica a situação do par no enxame
uploaded	quantidade de upload realizado pelo par
downloaded	quantidade de download realizado pelo par
left	quantidade de dados que faltam para terminar o download
no_peer_id	opcional, permite ao rastreador omitir o id dos pares na resposta
compact	opcional, compreende representação compacta de pares
key	opcional, identificador que só o par e o rastreador conhecem
trackerid	opcional, identifica um par que está retornando ao enxame

essas informações são o número de pares (sugadores e semeadores) e o volume de dados (enviados e obtidos) pelos pares. Os campos *event*, *uploaded*, *downloaded* e *left* são parâmetros informados pelos pares para ajudar o rastreador nessa função. O campo *event* serve para notificar o rastreador se o par está iniciando seu download (valor *started*), completou o conteúdo (valor *completed*), saiu do enxame (valor *stopped*) ou se é um contato regular sem evento especial associado (nesse caso, omitido ou valor *empty*). Os outros três campos (*uploaded*, *downloaded* e *left*) informam, respectivamente, ao rastreador quantos dados o par contribuiu com outros, obteve de seus vizinhos e faltam para terminar o download.

Os campos *no_peer_id* e *compact* são utilizados para economizar recursos. O primeiro informa ao rastreador que este pode omitir os identificadores dos pares na resposta à requisição, diminuindo o tamanho da mesma. O segundo sinaliza que o par requisitante compreende uma representação compacta dos pares retornados. Dessa forma, são utilizados 6 bytes para representar cada par, sendo que os quatro primeiros representam o endereço IP e os últimos dois a porta, ao invés de utilizar duas strings (IP e Id) e um int (porta).

Finalmente, os campos *key* e *trackerid* são usados para garantir a autenticidade entre as partes durante a comunicação entre par e rastreador. Eles devem ser preenchidos com valores informados pelo rastreador na resposta do primeiro *announce*, conforme apresentado a seguir.

O rastreador tem como principal objetivo proporcionar o encontro dos pares que desejam compartilhar um mesmo conteúdo. Além de responder ao par requisitante uma lista de pares presentes no enxame, o rastreador pode informar situações de erro, parâmetros de funcionamento e status do enxame. A resposta é um dicionário codificado com *bencode* com uma série de campos, conforme descrito na Tabela 6.2.

O campo mais importante é o *peers*, que pode ser representado tanto na forma tradicional, quanto na compacta. Na forma tradicional, a lista de pares retornadas é representada através de uma lista de dicionários, contendo para cada par, seu *peer id*, endereço

Tabela 6.2. Mensagem de resposta ao *Announce*

Campo	Descrição
peers (dicionário)	lista de pares contendo peer id, endereço IP e porta para cada um
peers (binário)	string da lista de pares usando 6 bytes por par
interval	intervalo em segundos entre requisições regulares feitas pelo par para o rastreador
min_interval	opcional, intervalo mínimo entre requisições
tracker id	string que o par deve enviar de volta nas próximas requisições
failure reason	erro que impossibilitou o atendimento da solicitação
warning message	mensagem de aviso de alguma situação ocorrida
complete	número de semeadores no enxame
incomplete	número de sugadores no enxame

IP e porta. Uma forma de redução no consumo de recursos do rastreador é feita representando os pares de maneira compacta. Isso é feito através de uma string contendo 6 bytes para cada par, sendo os quatro primeiros o endereço IP do par, e os dois últimos, a porta em que o mesmo está escutando.

A respeito do funcionamento do rastreador, o campo *interval* indica qual o intervalo de tempo que o par deve esperar entre sucessivas requisições. Outro campo opcional é o *min_interval* que, quando presente, informa que um agente de usuário não deve realizar requisições mais frequentemente que o indicado.

Para informar eventuais erros na requisição, a resposta pode conter os campos de *failure reason* e *warning message*, que indicam qual erro ocorreu em formato de texto. A diferença entre eles é que quando há *failure reason*, as outras informações são suprimidas, enquanto que no caso de *warning message*, os demais campos são preenchidos corretamente.

Por fim, o rastreador pode retornar dados globais sobre o enxame de acordo com aquilo que é informado por pares em mensagens de requisição. O campo *complete* indica a quantidade de semeadores presentes no enxame, enquanto o campo *incomplete* reporta o número total de sugadores no enxame. Essas informações também podem ser obtidas via requisição de *scrape*, conforme discutido a seguir.

6.4.2.2. Scrape

A URL do *scrape* segue o mesmo formato do *announce*, substituindo-se *announce* por *scrape*. Há no máximo um parâmetro, *info_hash*, para especificar um determinado torrent desejado. Quando este parâmetro é omitido, são retornadas informações sobre todos os torrents do rastreador.

A resposta de um *scrape* retorna as informações sobre cada um dos torrents requisitados, conforme apresentada na Tabela 6.3, podendo ser apenas um, múltiplos torrents

definidos ou todos torrents do rastreador. Para cada um dos torrents são apresentadas as informações de número de semeadores e sugadores presentes no enxame, número de vezes que um download foi completado, e, opcionalmente, o nome do torrent.

Tabela 6.3. Mensagem de resposta ao *scrape*

Campo	Descrição
files (dicionário)	lista de arquivos requisitados, contendo os campos abaixo para cada um
complete	número de semeadores no enxame
downloaded	número de vezes que o rastreador registrou um término de download
incomplete	número de sugadores no enxame
name	opcional, nome do torrent

6.4.3. Comunicação entre pares

Nesta subseção é discutida a interação entre os pares. Para isto, a seção está dividida em duas partes: a primeira define os estágios possíveis durante a comunicação de um par com o rastreador e com outros pares, enquanto a segunda apresenta os fluxos de mensagens entre os pares e com o rastreador.

6.4.3.1. Estados

O funcionamento de um agente de usuário Bittorrent pode ser abstraído e modelado através de um conjunto de diagramas de estado. Os diagramas representam versões daqueles encontrados em [Konrath et al. 2007], definidos de acordo com a descrição do protocolo e traços de implementações populares do mesmo [BitTorrent 2001, μTorrent 2006, Vuze 2003]. Os diagramas representam as múltiplas linhas de execução (*threads*) que podem ser instanciadas de maneira a tratar sincronamente eventos assíncronos, tal como ocorre frequentemente em implementações de agentes de usuário. A Figura 6.5 mostra o diagrama de forma global e com ênfase na troca de peças. Os elementos principais são descritos a seguir.

Um par começa no estado START e, ao ser iniciado, envia uma requisição de *announce* ao rastreador, conforme apresentado na Subseção 6.4.2, passando ao estado WAIT_PEERLIST. Se o arquivo torrent possuir mais de um rastreador, essas iterações ocorrerão de forma independente e paralela, uma para cada rastreador. Neste estado, o par espera uma resposta do rastreador para sua requisição realizada. Quando a resposta contendo uma lista de pares é obtida, são criadas duas linhas de execução. Na primeira linha, o par configura um temporizador para re-consultar o rastreador, passando ao estado START. Na segunda, muda para o estado MANAGING_PEERLIST, na qual a resposta será tratada, iniciando novas conexões com os pares obtidos.

No estado MANAGING_PEERLIST, o par estabelece ou recebe novas conexões, criando uma nova "linha de execução" para cada contato estabelecido. Para isto, há qua-

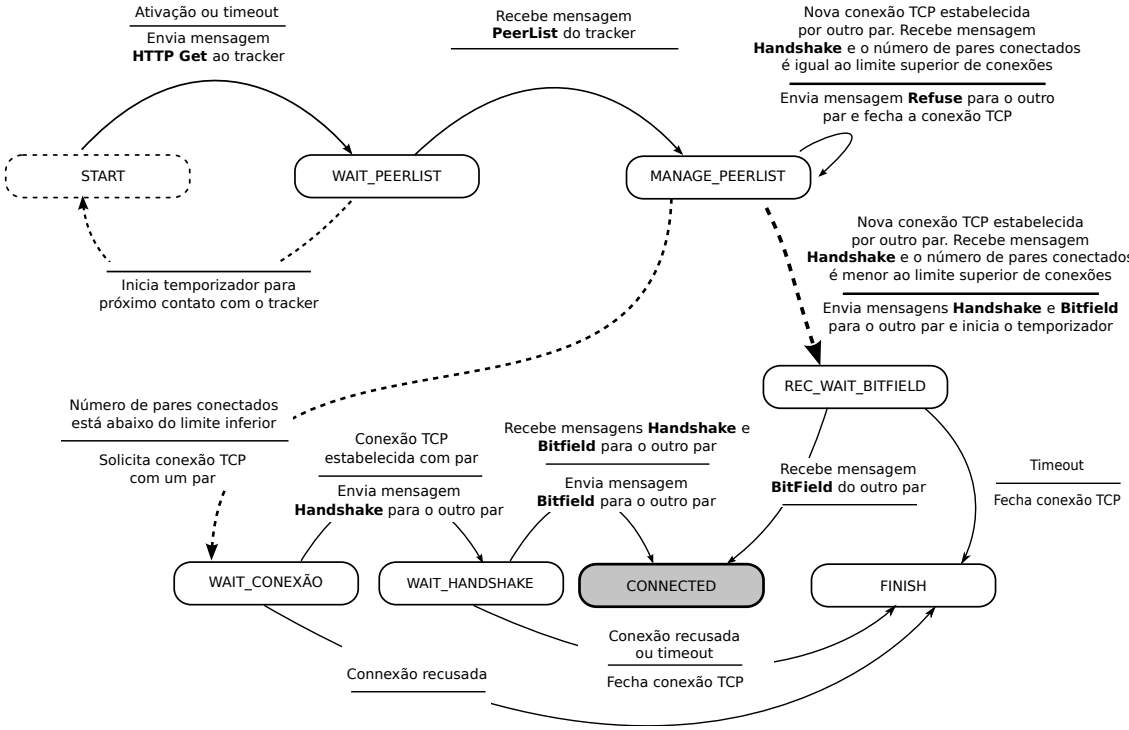

Figura 6.5. Diagrama de estados dos pares

tro transições possíveis: iniciar conexão, receber conexão, recusar conexão e fazer nada. Caso o par não tenha ultrapassado um certo número de conexões, este pode iniciar novas. Isso é feito varrendo a lista obtida do rastreador e requisitando conexões TCP aos pares, passando ao estado WAIT_PEERCONECT. Caso já tenha ultrapassado o limite, o par ignora a lista obtida e mantém-se no estado MANAGING_PEERLIST.

Ao receber uma requisição de conexão vinda de um par remoto, o par local verifica se está apto a receber novas conexões sem ultrapassar o limite máximo de conexões. Desse modo, o par pode tanto aceitar a conexão, passando para o estado WAIT_HANDSHAKE, ou recusar a conexão, fechando-a e passando ao estado FINISH.

Seguindo a linha de estabelecimento de conexão, no estado WAIT_PEERCONECT a conexão pode ser recusada, resultando no término da conexão no estado FINISH. Caso a conexão seja estabelecida, o par envia a mensagem HANDSHAKE e passa ao estado WAIT_HANDSHAKE. Nesse estado, o par espera uma mensagem de HANDSHAKE, juntamente com outra de BITFIELD. Ao receber ambas, envia uma mensagem de BITFIELD, completando a conexão no estado CONNECTED. Caso haja alguma exceção ou timeout, a conexão é fechada, indo ao estado FINISH.

Ao receber uma requisição por conexão de um par remoto, o par recebe a mensagem HANDSHAKE do par remoto e responde com as mensagens HANDSHAKE e BITFIELD, passando ao estado WAIT_BITFIELD. Neste último estado, aguarda o recebimento da mensagem BITFIELD de seu vizinho, para terminar o estabelecimento da conexão e ir ao estado CONNECTED. Em qualquer momento pode haver uma exceção ou timeout, causando a falha da conexão, e fazendo com que ocorra uma transição para o estado FINISH.

O estado CONNECTED, marcado no diagrama, concerne as variações de estado que uma linha de execução 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) e será explorada na Seção 6.5. As Figuras 6.6 e 6.7 representam os diagramas.

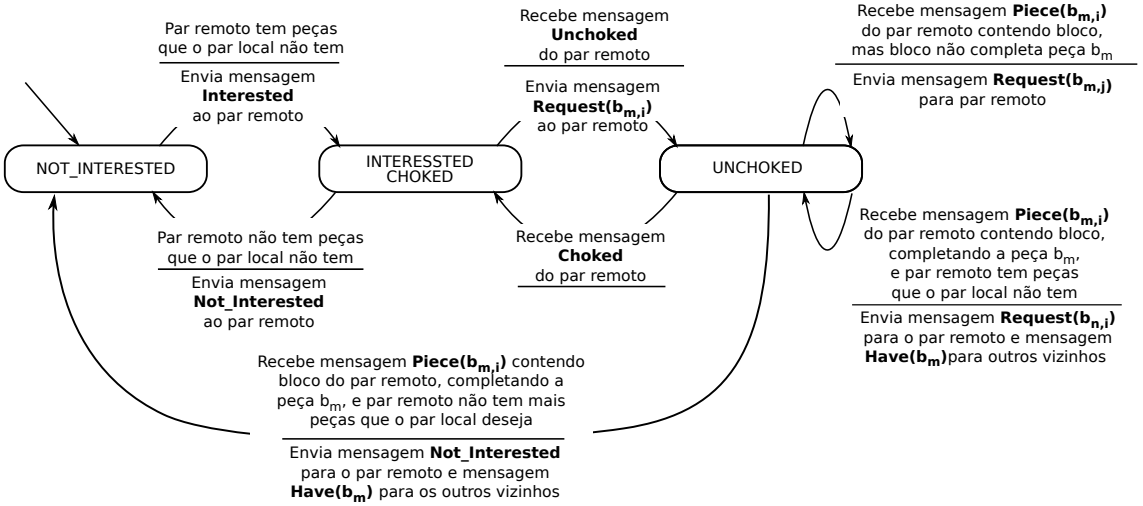

Figura 6.6. Diagrama de estados de linha de execução em relação a downloads de par remoto

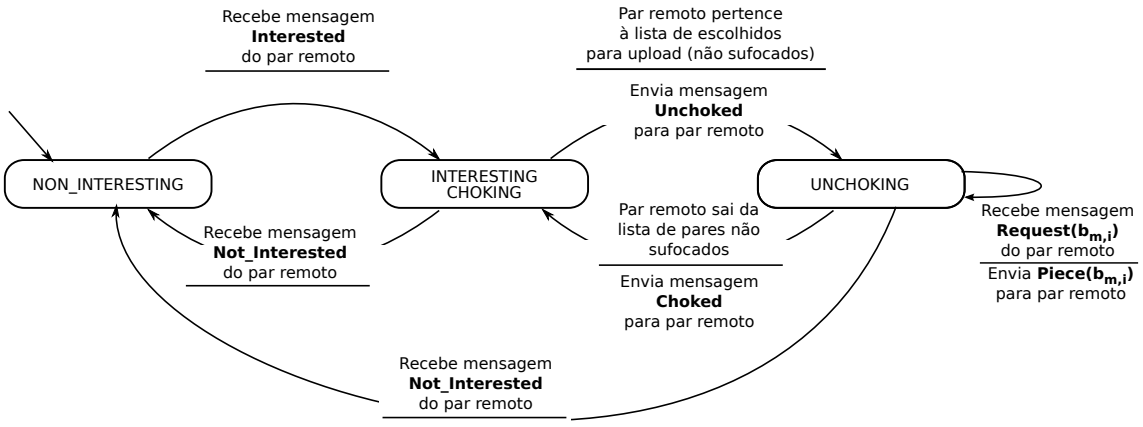

Figura 6.7. Diagrama de estados de linha de execução em relação a uploads para par remoto

Para determinar quais peças devem ser obtidas a seguir pelo par, assim como a quem requisitá-las, cada par mantém um conjunto de informações referentes aos seus vizinhos. Dentre estas, as mais importantes são:

- *bitfield*, um mapa de bits que representa quais peças um vizinho possui;

- uma dupla de flags que sinaliza o interesse do par em seu vizinho e a relação inversa;
- uma dupla de flags que indica se o par está desbloqueado (*unchoked*), isto é, se pode requisitar dados de seu vizinho e a relação contrária.

Ao iniciar a conexão, o *bitfield* está vazio e todas as flags são sinalizadas como falsas, representando que não há interesse entre os pares e que eles estão *choked*, ou seja, não autorizados a requisitar dados. Durante estabelecimento da comunicação, os pares envolvidos trocam mensagens de BITFIELD, permitindo a cada um conhecer o conjunto de peças que o outro par possui. Observe, entretanto, que essa informação enviada pelo par não é confiável; um par pode, por exemplo, anunciar um subconjunto de peças que ele realmente tem, de forma a guiar a solicitação de peças por outros pares. Esta é uma estratégia que será abordada na Subseção 6.5.3.

A Figura 6.6 apresenta os estados existentes e o comportamento do par local em relação a obter dados de um par remoto. Inicialmente, a linha de execução está no estado NOT_INTERESTED. Após o recebimento da mensagem BITFIELD, a estrutura de dados que armazena as peças possuídas pelo vizinho é atualizada, permitindo verificar se aquele par possui uma ou mais peças que o par local não possui. Nesse caso, o par remoto é considerado "interessante" e a linha de execução passa ao estado INTERESTED_CHOKED, enviando uma mensagem INTERESTED. Esta troca de status de "não interessante" para "interessante" pode ocorrer também através de uma mensagem HAVE recebida do par remoto, notificando que este acabou de completar mais uma peça.

No estado INTERESTED_CHOKED, o par local deseja obter dados do par remoto, mas está bloqueado. A sinalização de bloqueado existe para que os pares possam manter sua vizinhança estável e controlar a quem se destina sua contribuição. O fechamento e estabelecimento de novas conexões tem um custo muito elevado e a contribuição com todos os pares é ineficiente, conforme será discutido na Seção 6.5. Nesse estado, o par pode ter obtido uma peça de outro par e com isso perder o interesse num determinado par, notificando-o através da mensagem NOT INTERESTED e retornando ao estado NOT_INTERESTED. Uma segunda opção é o par ser desbloqueado através de uma mensagem UNCHOKED, ficando autorizado a requisitar conteúdo. O par passa ao estado UNCHOKED e envia uma requisição por dados através da mensagem REQUEST($B_{m,i}$), em que $B_{m,i}$ é o i-ésimo bloco da peça m.

No estado UNCHOKED, o par possui quatro transições possíveis:

- par recebe bloco requisitado pela mensagem PIECE($B_{m,i}$), mas não completa a peça, requisitando novo bloco para esta peça REQUEST($B_{m,j}$)
- par recebe bloco requisitado pela mensagem PIECE($B_{m,i}$), completa a peça e continua interessado no par remoto. Então, o par envia requisição para outra peça (mensagem REQUEST($B_{n,i}$)) e notifica seus vizinhos que completou uma peça através da mensagem HAVE(B_m);
- par recebe bloco requisitado pela mensagem PIECE($B_{m,i}$), completa a peça e perde interesse no par remoto. Então, o par envia notificação de não interessado, através

da mensagem NOT INTERESTED, muda seu estado para NOT_INTERESTED e notifica seus vizinhos que completou uma peça enviando a mensagem HAVE(B_m);

- par recebe mensagem CHOKED. Par está novamente bloqueado, cessando a requisição por novos blocos e mudando para o estado INTERESTED_CHOKED.

O upload é representado através da Figura 6.7. No início da comunicação, o par remoto está no estado NON_INTERESTING, sinalizando que não tem interesse no par local. O estado muda somente quando o par local recebe uma mensagem INTERESTED do par remoto, indicando que o primeiro possui uma ou mais peças que o segundo não tem, passando então para o estado INTERESTING_CHOKING.

No estado INTERESTING_CHOKING, o par remoto está interessado no par local, mas ainda está bloqueado. Há duas transições a partir desse estado. Na primeira, o par local recebe uma mensagem NOT INTERESTED, indicando que o par remoto perdeu interesse e retorna ao estado NON_INTERESTING. Na segunda possibilidade, o par local seleciona o par remoto e inclui o mesmo na lista de beneficiados com upload (desbloqueados), notificando-o através do envio da mensagem UNCHOKED e mudando o estado para UNCHOKING.

No estado UNCHOKING, o par recebe requisições REQUEST($B_{m,i}$) e responde enviando o conteúdo requisitado PIECE($B_{m,i}$). O laço continua até que uma das condições se torne verdadeira:

- quando o par local decide bloquear o par remoto, o que é feito através da mensagem CHOKED, passando o estado a INTERESTING_CHOKING;
- quando o par local receber a mensagem NOT INTERESTED, indicando que o par remoto não está mais interessado no par local, trocando para o estado NON_INTERESTING.

6.4.3.2. Mensagens

A seguir são apresentadas as mensagens do protocolo, conforme Tabela 6.4, e discutidos os parâmetros utilizados. Então, na subseção 6.4.3.3, ilustra-se a comunicação entre pares e rastreador através de um fluxo típico de troca de mensagens. O protocolo prevê uma mensagem KEEP-ALIVE para manter a conexão aberta quando não há comunicação entre os pares. Existem as mensagens de controle sem argumento extra CHOKE e UNCHOKE para sinalizar que o par está, respectivamente, bloqueado ou desbloqueado, e INTERESTED e NOT INTERESTED, para indicar interesse ou desinteresse no vizinho, respectivamente.

Em seguida, são utilizadas duas mensagens para atualizar a posse de peças aos vizinhos. A mensagem BITFIELD é enviada uma vez, durante o estabelecimento da conexão, ao longo da interação entre dois pares e serve para que os pares tenham conhecimento de quais peças são possuídas por seus vizinhos. A mensagem possui como argumento um mapa de bits representando a posse ou não de cada uma das peças. A mensagem HAVE serve para atualizar a informação de posse de peças de um par a seus vizinhos, passando como argumento o índice da peça recém completa.

Para a troca efetiva de dados, o protocolo prevê as mensagens REQUEST, usada para a requisição, e PIECE, utilizada para transferir o bloco requisitado. Na requisição,

deve constar qual o índice da peça, o byte inicial e o tamanho do bloco desejado. Já na resposta, estão os mesmos campos de índice da peça e byte inicial, além dos dados propriamente ditos.

Existem ainda duas mensagens, menos frequentes: CANCEL e PORT. A primeira indica o cancelamento de uma requisição previamente enviada, passando os mesmos argumentos de uma requisição; seu uso será melhor explicado na Subseção 6.5.2.2. A segunda é utilizada quando o par implementa a extensão DHT, para indicar em qual porta o nodo DHT do par está escutando e tem como argumento esta porta.

Tabela 6.4. Mensagens do Protocolo BitTorrent

Nome	Descrição
keep-alive	notifica que par continua conectado
choke	sinaliza desautorização para requisitar dados (bloqueado)
unchoke	sinaliza autorização para requisitar dados (desbloqueado)
interested	sinaliza interesse (vizinho tem peças que o par local não tem)
not interested	sinaliza desinteresse (vizinho não tem peças que o par local não tem)
have	notifica nova posse de peça
bitfield	mapa de bits que representa as peças possuídas pelo par
request	requisição para um bloco de uma peça
piece	conteúdo em si, correspondente a determinado bloco/peça
cancel	cancela requisição por um bloco de uma peça
port	notifica porta do DHT

6.4.3.3. Fluxo

Os diagramas de estado das Figuras 6.5, 6.6, 6.7 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 6.8 um exemplo de comunicação em um enxame. Para ser simples e representável, é mostrada apenas a troca de mensagens entre um rastreador e dois pares, denominados Semeador e Sugador.

Inicialmente, o Semeador, que possui uma cópia completa do conteúdo, se registra no enxame e requisita uma lista de pares através da mensagem HTTP GET. Após algum tempo, simbolizado pelo retângulo pontilhado, o Sugador conecta com o rastreador, envia uma mensagem HTTP GET e obtém o endereço do Semeador, pois este está participando do enxame. O Sugador conecta com o Semeador (mensagem HANDSHAKE), recebe o mapa de bits, que indica quais peças são possuídas pelo par, (mensagens HANDSHAKE e BITFIELD) e envia o seu mapa de bits (na mensagem BITFIELD). Como o Semeador possui peças que Sugador não possui, uma mensagem INTERESTED é enviada pelo Sugador para o Semeador.

Quando receber uma mensagem UNCHOKED, o Sugador estará liberado para obter conteúdo a partir do Semeador (desbloqueado). Ele então passará a solicitar blocos de peças através de mensagens REQUEST($B_{m,i}$). O Semeador enviará os blocos solicitados através de mensagens PIECE($B_{m,i}$). O recebimento de blocos libera novas solicitações, que ocasionam novos envios de dados 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, uma mensagem HAVE(B_m) é enviada ao Semeador, para que este atualize a informação do mapa referente ao Sugador. Este ciclo se repete até que o Semeador bloqueie o Sugador, através da mensagem CHOKED.

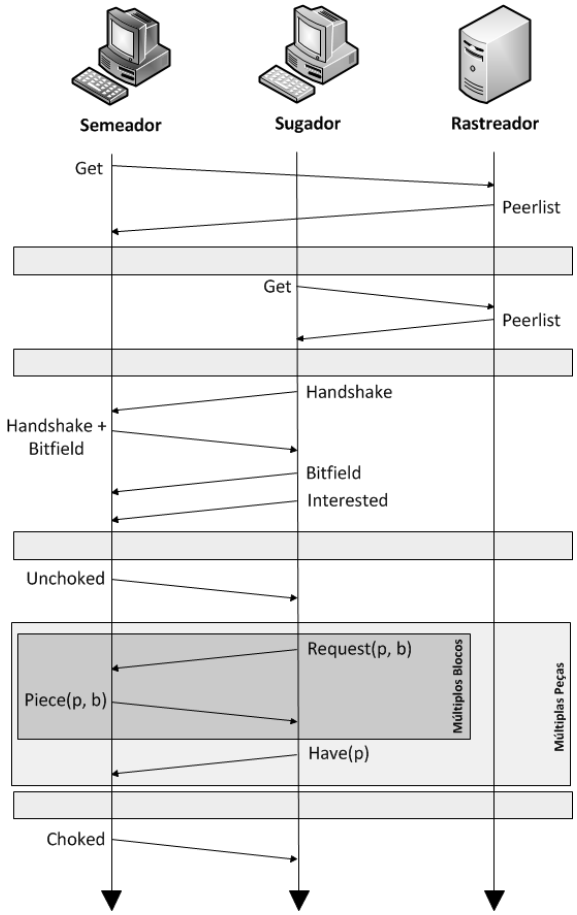

Figura 6.8. Diagrama de tempo ilustrando comunicação

6.4.4. Torrent

O arquivo de metadados torrent possui um conjunto de campos, apresentados na Tabela 6.5, com informações sobre o conteúdo compartilhado e como encontrar outros pares [Cohen 2008a]. Os dois campos fundamentais são *info* e *announce*. O primeiro é expandido na Tabela 6.6 e apresenta os seguintes campos: *piece length*, a quantidade de bytes em cada peça; *pieces*, a concatenação de todos os *hashes* das peças do conteúdo, que é utilizada para verificar a integridade do conteúdo; *private*, um campo opcional que indica ao agente que ele não deve obter novos pares por outras formas além das listadas no arquivo torrent; *name*, o nome do arquivo; e *files*, que lista os arquivos existentes no conteúdo. O campo *files* define, para cada um dos arquivos, os campos de: *length*, tamanho do arquivo; *md5sum*, soma md5 do arquivo (não utilizado); e *path*, que representa o caminho e nome do arquivo. O segundo campo, *announce*, define qual é a URL do rastreador a ser utilizada para o par entrar no enxame (o compartilhamento sem rastreador será discutido posteriormente).

Além desses dois campos, existem outros cinco opcionais: *announce-list*, uma extensão que permite acrescentar uma lista com mais endereços de rastreadores; *creation date*, que define a data de criação do torrent; *comment*, algum comentário em texto do autor do conteúdo; *created by*, nome e versão do programa usado para criar o torrent; e por fim *encoding*, que explicita a codificação usada para gerar os *hashes* da peça.

Tabela 6.5. Estrutura do arquivo de metadados

Campo	Descrição
info	dicionário com a descrição do(s) arquivo(s) do torrent, tais como tamanho de peça, hash das peças, nome e tamanho dos arquivos
announce	URL de announce do rastreador
announce-list	opcional, permite acrescentar mais rastreadores
creation date	opcional, a data de criação do torrent
comment	opcional, comentários do autor
created by	opcional, nome e versão do programa usado para criar o torrent
encoding	opcional, codificação usada para gerar os hashes das peças

Tabela 6.6. Estrutura do campo info

Campo	Descrição
piece length	número de bytes em cada peça
pieces	concatenação de todos hashes de cada peça
private	opcional, se ativado, não permite utilizar outras fontes de pares
name	nome do arquivo
files	todos arquivos existentes com os campos a seguir para cada um
length	tamanho de cada arquivo
md5sum	opcional, soma MD5 do arquivo (não é usado)
path	caminho e nome do arquivo

Conforme mencionado anteriormente, BitTorrent inovou ao utilizar a técnica de *swarming* na distribuição de conteúdo. Um arquivo é dividido em blocos menores, denominados "peças", que consistem a unidade básica de compartilhamento do protocolo. Assim que um par obtém uma peça, ele pode distribuí-la a seus vizinhos, ou seja, ele compartilha um arquivo mesmo sem ter o conteúdo completo. Além disso, a divisão de um conteúdo em pedaços menores permite, de forma simples, que o conteúdo seja obtido de múltiplas fontes. O *swarming* permite um aumento na utilização da capacidade de upload dos pares e leva a uma melhora significativo no desempenho geral do sistema. Por outro lado, exige estruturas auxiliares em cada par para gerenciar múltiplas requisições pendentes a outros pares. A seguir serão discutidos os ganhos alcançados pela utilização da técnica, relacionando desempenho obtido e custo computacional necessário.

Uma peça define dois limites – início e fim – de um segmento de dados do conteúdo sendo distribuído. Ela é a unidade básica de controle que permite aos agentes de usuário trocarem informações entre si sobre a disponibilidade de dados em cada um. Por essa razão, a maioria das estruturas de dados mantidas em agentes refere-se a peças.

Para aumentar a eficiência na transferência de dados, uma peça é subdividida em blocos de tamanho fixo, tipicamente 16 KB. Um agente solicita a transferência de blocos de uma peça a outros agentes, de forma que múltiplos blocos possam ser simultaneamente transferidos de cada par. Ou seja, mais pares podem contribuir (paralelamente) com blocos de uma mesma peça. Além disso, múltiplos blocos podem ser simultaneamente solicitados a um par, formando um pipeline com o mesmo.

Existe uma relação entre o tamanho da peça e o desempenho do enxame [Marciniak et al. 2008]. A quantidade de blocos por peça, ou seja o tamanho de peças, estabelece um limite superior no número de requisições paralelas para download de uma peça. Em um extremo, uma peça de um bloco único implica que cada peça seja obtida de apenas um par, com uma única solicitação. Em outro, o conteúdo inteiro é formado por uma peça subdivida em um grande número de blocos, o que permitiria por um lado que cada bloco fosse obtido de um par distinto, mas por outro desconsideraria o *swarming* e as suas vantagens.

O tamanho de peças possui outros efeitos no desempenho do sistema, como por exemplo afeta a integridade. A integridade do conteúdo é verificada a cada peça carregada através do uso da função *hash* SHA-1 [3rd e Jones 2001]. Após a obtenção de uma peça, o par calcula o *hash* SHA-1 da mesma e compara o resultado com o *hash* correspondente contido no arquivo torrent. A integridade é garantida sobre a peça como um todo e não seus blocos individuais: quando há perda de integridade na peça, não é possível identificar qual ou quais blocos estão incorretos e a peça como um todo deve ser descartada e requisitada novamente. Portanto, quanto maior a peça, maior a quantidade de conteúdo a ser descartada em caso de problemas com a mesma.

Para um dado conteúdo, o número de peças é inversamente proporcional ao tamanho das peças. Assim, o tamanho de peças também impacta na dimensão (custo) de estruturas auxiliares que precisam ser mantidas por agentes de usuário. A gerência de peças e blocos é feita através dessas estruturas e sua manipulação representa um custo computacional ao agente, conforme comentado a seguir. É necessária, por exemplo, uma estrutura para manter um registro das requisições pendentes e ativas, e quais partes do conteúdo já foram obtidas e quais faltam ainda. Um mesmo bloco não deve, em geral, ser solicitado ao mesmo tempo a mais de um par, e todos os blocos devem ser mais cedo ou mais tarde solicitados.

6.5. Políticas e Mecanismos

Esta seção está organizada como segue. Primeiro são tratadas as políticas e os mecanismos utilizados no BitTorrent, tendo em mente a motivação da escolha de cada um e sua aplicabilidade em outros tipos de aplicações da Internet. Segundo, são apresentados os algoritmos, explicando como os mesmos implementam na prática as políticas abordadas anteriormente. Terceiro, são tratadas as extensões do protocolo, que foram incorporadas para aumentar sua eficiência e corrigir deficiências do BitTorrent.

6.5.1. Políticas

Duas políticas serviram de referência no desenvolvimento do protocolo BitTorrent: seleção de pares com quem contribuir e seleção de peças a requisitar. A discussão nesta

subseção é guiada por duas questões básicas: porque a política se faz necessária, e quais aspectos devem ser considerados em sua concretização.

6.5.1.1. Políticas de seleção de pares

Como no BitTorrent não existe nenhum tipo de alocação central de recursos, cada par é relativamente autônomo para determinar suas taxas de upload e de download. Um par tipicamente mantém conexões abertas com dezenas de outros pares, seus vizinhos. Não seria eficiente pulverizar a (quase sempre relativamente menor) capacidade de upload entre dezenas de vizinhos, considerando a operação do protocolo TCP, seu controle de congestionamento e sobrecargas envolvidas. Portanto, é necessário que um par selecione periodicamente um subconjunto de vizinhos para contribuir. Cada par utiliza um algoritmo de *choking* que implementa uma variação de *Tit-for-Tat* para beneficiar pares que contribuem mais.

Do ponto de vista do sistema, um bom algoritmo de seleção de pares deve maximizar a utilização global de recursos disponíveis e ser resistente a pares que não contribuem [Cohen 2003]. Para alcançar esses objetivos, são considerados três aspectos, discutidos a seguir: reciprocidade, evitar fibrilação e descoberta de melhores pares dentre os vizinhos.

A reciprocidade está relacionada à justiça na rede: um par deve retribuir a aqueles pares que lhe contribuem. Ou seja, ao escolher vizinhos para upload, um par seleciona aqueles que lhe fizeram upload. Isso torna o enxame mais resistente a pares egoístas porque, quando um par não contribui, suas chances de obter dados são diminuídas. Assim, os pares são incentivados a contribuírem, aumentando a disponibilidade global de peças no sistema e contribuindo para a robustez do enxame.

O segundo aspecto está relacionado à manutenção consistente de taxas de downloads. Nesse sentido, é necessário evitar que um par autorize e desautorize rapidamente o download de um vizinho (ou seja, pares consecutivos de CHOKE e UNCHOKE). Esse fenômeno é conhecido como "fibrilação" e sua consequência seria o desperdício de largura de banda com requisições ignoradas.

O terceiro aspecto está associado ao fato de um par ter que avaliar dinamicamente o potencial de upload seus vizinhos e selecionar os melhores. Para tal, um par toma a iniciativa de contribuir fazendo upload a outro par (ou seja, faz uma sondagem), e então nos momentos seguintes avalia a reciprocidade do mesmo (quanto aquele vizinho oferece de upload). Mais precisamente, para não ficar preso a um máximo local de desempenho, o par tenta conhecer as capacidades de upload de seus outros vizinhos, visando encontrar pares que contribuam mais que os atuais. Idealmente, um par testaria todas as possibilidades, mas "testar rapidamente todos os vizinhos" levaria provavelmente à fibrilação.

6.5.1.2. Políticas de Seleção de peças

Um par divulga, e recebe, informações sobre a disponibilidade de peças entre seus vizinhos. O conjunto de peças disponíveis para solicitação depende da disponibilidade entre

os vizinhos que lhe autorizam download. Uma política é empregada para escolha de quais peças serão solicitadas primeiro. Tal política deve ser eficiente do ponto de vista global do enxame, e para alcançar este objetivo, é necessário considerar dois aspectos: equilíbrio na distribuição de peças e prioridade de conclusão.

O objetivo de buscar equilíbrio na distribuição de peças é evitar que uma peça torne-se rara a ponto de deteriorar o desempenho do enxame. No caso extremo, um peça poderia tornar-se extinta, levando à falha do enxame. Como não há um elemento central coordenador, este equilíbrio global deve ser alcançado a partir de ações locais de cada par. Além disso, quanto mais equilibrada for a distribuição, maiores serão as chances dos pares se manterem "interessantes" uns aos outros, maximizando a chance de interação e consequentemente a utilização das capacidades.

O segundo aspecto, conhecido como *strict priority*, implica priorizar a conclusão do download de uma peça em andamento, em detrimento do incio do download de nova peça. O objetivo é aumentar a quantidade de peças disponíveis tão logo quanto possível e consequentemente as chances de o par tornar-se interessante a seus vizinhos, ou seja, com peças que eles não possuem.

6.5.2. Mecanismos

Esta subseção se concentra nos mecanismos e algoritmos utilizados para atender aos requisitos mencionados na subseção anterior. Dois algoritmos são discutidos a seguir: *unchoking*, referente à seleção de pares, e *piece picker*, referente à seleção de peças.

6.5.2.1. Desbloqueio (*Unchoking*)

O algoritmo de *Unchoking* é responsável por determinar quais os pares serão autorizados a requisitar blocos. Ele é composto pelos seguintes mecanismos, discutidos a seguir: *Tit-for-Tat*, períodos de avaliação, *Optimistic Unchoking* e *anti-snubbing*.

O mecanismo *Tit-for-Tat* (TFT) é responsável pela reciprocidade entre os pares. Ele avalia a contribuição dos vizinhos e escolhe aqueles que mais forneceram dados para retribuir-lhes, autorizando-os a requisitar dados (*unchoke*). Desse modo, os pares são incentivados a contribuir para aumentarem suas chances de serem escolhidos. Quando o par é semeador, o algoritmo é diferente, havendo duas variantes, como segue. A primeira seleciona os pares que possuem melhores taxas de download para aumentar a velocidade da disseminação do conteúdo. A segunda percorre a lista de vizinhos circularmente (esquema *round-robin*), proporcionando uma divisão justa da capacidade de upload entre todos vizinhos.

A avaliação e escolha são feitas em *rodadas*, tipicamente representando intervalos de 10 segundos. Isso evita a fibrilação na comunicação, permitindo que os vizinhos tenham tempo para requisitar dados e contribuir de volta.

O princípio do TFT é priorizar pares que são capazes de oferecer maior taxa de upload ao par local. Se ele 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.

Para possibilitar a descoberta de pares com taxas de upload superiores, a cada κ rodadas um par é marcado como desbloqueado, independente da taxa de download que o par local obteve daquele outro par. O valor típico de κ é 3. O par assim realiza upload para outro par, visando realizar um futuro download daquele par em contrapartida. Esse mecanismo é conhecido como *Optimistic Unchoking*), pois otimisticamente assume que há pares disponíveis com taxas de upload superiores aos atuais mas que precisam ser descobertos.

Em condições normais, a cada momento, somente um par remoto estará desbloqueado via *Optimistic Unchoking*. No entanto, ocasionalmente um par poderá alocar mais "vagas" via esse mecanismo. A situação surge quando o par local não é, em nenhum dos pares para quem ele faz upload, um dos quatro melhores contribuidores, nem foi selecionado pelo *Optimistic Unchoking* de um par remoto. Nesse caso, o par local estará bloqueado por por todos os pares remotos com os quais está conectado e precisará de certo tempo até que o *Optimistic Unchoking* 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, o mecanismo "*anti-snubbing*" permite que o par local eleja, temporariamente, mais de um par remoto para receber dados via *Optimistic Unchoking*. Essa tática faz com que o par local consiga se recuperar muito mais rapidamente da situação indesejável de não ter com quem trocar conteúdo.

6.5.2.2. Piece Picker

Conforme discutido anteriormente, o desempenho do enxame e de seus pares depende da distribuição equilibrada das peças. O mecanismo responsável por isso é denominado *Local Rarest First* (LRF). Como o próprio nome indica, esse mecanismo prioriza a peça local mais rara, considerando os vizinhos do par local. Obter a peça mais rara ajuda, naturalmente, a aumentar sua disponibilidade. A escolha é "local" porque os pares conhecem a disponibilidade das peças de apenas um subconjunto de pares do enxame, seus vizinhos. A expectativa é que o comportamento local de todos os pares seja refletido na realidade global do enxame.

O LRF é eficaz em distribuir de forma equilibrada as peças, porém não é a solução ideal quando um par está iniciando ou terminando o seu download, como explicado a seguir. Na fase inicial (enquanto um par possui menos de 4 peças), um par não possui a informação completa da sua vizinhança (que está sendo formada) e por isso não consegue montar uma "fotografia" da disponibilidade das peças. Além disso, o par não possui peças para retribuir, o que aumenta a necessidade de completar o quanto antes suas primeiras peças. Por isso, o BitTorrent aplica nesse caso uma estratégia aleatoria, *Random First*, sobre as peças disponíveis. Ela é executada até o par alcançar a um limiar de n peças completadas, tipicamente 5.

Durante a fase final do download de um par, denominada *endgame*, também é utilizada uma estratégia diferente. Ela consiste em solicitar, de forma redundante, a trans-

ferência de todos os blocos ainda não obtidos a *todos* os pares vizinhos (considerando, naturalmente, o subconjunto de pares remotos para os quais o par local está autorizado a solicitar download e que possuem os blocos desejados). O *endgame* é geralmente ativado após um par enviar sua última requisição, ou seja, quando todos os blocos do conteúdo foram obtidos ou já foram requisitados a vizinhos. Uma vez que um bloco é recebido, mensagens de cancelamento são enviadas aos vizinhos com requisição ativa para o respectivo bloco. O uso de força bruta não gera desperdício significativo porque sua duração é relativamente curta.

6.5.3. Extensões ao Protocolo

Extensões cumprem importante papel no BitTorrent, pois corrigem questões não antevistas na especificação original do protocolo e ao mesmo estendem o protocolo com novas funcionalidades. O fato do BitTorrent definir uma maneira de ser estendido foi fundamental para o surgimento de diversas inovações. Esta forma é, na prática, uma área reservada de 8 bytes no cabeçalho da mensagem HANDSHAKE, para indicar quais extensões o agente de usuário compreende [Cohen 2008b]. Quando dois pares compreendem mutuamente determinada extensão, eles podem iniciar uma comunicação específica seguindo o respectivo protocolo.

Nesta subseção são descritas as principais extensões ao protocolo BitTorrent, considerando relevância, uso e inovação. As descrições que seguem são baseadas na especificação oficial do protocolo [WikiTheory 2011] e do Vuze [WikiVuze 2011]. Inicialmente, são abordadas três extensões cujo foco é melhorar o processo de descoberta de novos pares e estabelecimento de conexões com os mesmos: *Distributed Hash Table* (DHT) [Loewenstern 2008], *Peer Exchange* (PEX) [WikiTheory 2008], *Local Peer Discovery* (LPD). Em seguida, são discutidas duas extensões que visam aumentar a eficiência do protocolo: *Fast Extension* [Harrison e Cohen 2008] e *SuperSeeding* [Hoffman 2008].

6.5.3.1. Distributed Hash Table

A extensão *Distributed Hash Table* (DHT) [Loewenstern 2008] possibilita a um par encontrar outros sem a necessidade de um rastreador. Esse modo de operação é denominado *trackerless*. Os pares que usam essa extensão compõem um "rastreador distribuído", cuja comunicação está baseada em uma tabela *hash* distribuída que armazena tuplas de forma descentralizada. Cada par implementa um nodo da DHT e pode armazenar informações referentes a um conjunto de enxames. A DHT é baseada no Kademlia [Maymounkov e Mazières 2002] e implementada sobre o protocolo UDP.

Cada agente de usuário BitTorrent com suporte à DHT representa um nodo da mesma. A DHT é organizada em um espaço de 160 bits, e cada nodo possui um identificador desse tamanho. A cada conteúdo está associada uma tupla, cujo identificador é o *infohash*, e sua informação, a lista de pares que estão compartilhando o conteúdo. Existe um nodo que é responsável pela tupla, determinado de acordo com a proximidade entre o *infohash* e os identificadores de nodos, observando uma métrica XOR no roteamento. Para manutenção da topologia, um nodo mantém uma lista de nodos conhecidos, com informações tanto de pares próximos como pares mais afastados, para facilitar saltos

longos e aumentar a eficiência.

As operações realizadas na DHT procuram o nodo mais próximo ao responsável pela tupla, contatando-o com a requisição. A resposta pode ser a informação solicitada (quando o destinatário possui a informação) ou um conjunto de nodos mais próximos do destinatário (caso contrário). Esse procedimento de requisição é realizado iterativamente pelo par requisitante até que a informação seja retornada. O protocolo prevê quatro mensagens para a gerência da DHT, conforme a Tabela 6.7.

Existem duas grandes DHTs operando com BitTorrent atualmente: uma referente ao Vuze, e outra ao Mainline. Cada uma das DHTs funciona independentemente da outra, como dois rastreadores separados. Cada DHT serve como rastreador dos torrents de seus participantes.

Tabela 6.7. Mensagens da extensão DHT

Nome	Descrição
PING	Verifica se o nodo está online.
FIND_NODE	Usado para obter informações sobre um nodo. Retorna ou as informações do nodo ou os *k* nodos online mais próximos
GET_PEERS	Obtém pares associados a um infohash. Retorna uma lista de pares que estão compartilhando o conteúdo ou os *k* nodos online mais próximos
ANNOUNCE_PEER	Anuncia que o par que fez a requisição está fazendo o download de um determinado conteúdo, informando a porta utilizada.

6.5.3.2. Peer Exchange

Assim como a DHT, a extensão *Peer Exchange* (PEX) [Vuze 2003, Wu et al. 2010] é uma opção para descoberta descentralizada de novos pares. Ela possibilita que pares troquem listas de pares (ativos e inativos) diretamente entre si, diminuindo assim a dependência em relação ao rastreador. Diferentemente da DHT, o PEX serve apenas como complemento, pois precisa de um outro meio para obtenção de uma lista inicial de pares. O intercâmbio de listas entre pares que suportam PEX ocorre em ciclos regulados temporalmente.

O PEX, originalmente implementado no agente de usuário Vuze (6.3.3), se tornou amplamente popular em agentes modernos. Não existe um padrão para o mesmo, mas há uma convenção [WikiVuze 2010] que determina limites, como número máximo de pares ativos e inativos (50), e o intervalo mínimo de 1 minuto entre mensagens.

6.5.3.3. Local Peer Discovery

O *Local Peer Discovery* (LPD) é uma extensão não oficial do protocolo BitTorrent cujo objetivo é localizar vizinhos que estejam na mesma rede local. Isso garante benefícios tanto para o usuário quanto para os ISPs. Para o usuário, o compartilhamento de um conteúdo dessa forma aumenta o desempenho do agente ao permitir a transferência direta

usando vazão abundante e latência baixa típicas de redes locais, ao invés de competir por largura de banda individualmente com outros pares em enlaces de acesso.

Para os ISPs, essa extensão economiza recursos, pois todo o tráfego em uma rede local estará por definição confinado em um único ISP. De forma similar, existem outras extensões amigáveis a provedores de Internet (*ISP-friendly*), que tem como objetivo priorizar a comunicação com pares que estão presentes no mesmo ISP ou nos ISPs mais próximos. Na Seção 6.6 são discutidos trabalhos sobre esse tópico.

6.5.3.4. Fast Extension

O *Fast Extension*, proposto inicialmente pelo agente Vuze, tem como objetivo principal acelerar a fase inicial de download de um par recém chegado ao enxame. Nesta etapa, um par possui poucas peças, se alguma, para fazer upload. Portanto, é muito difícil se tornar interessante a outros pares (isto é, ter peças que eles não tem) ao ponto de contribuir com outros pares e fazer com que eles atuem de forma recíproca, desbloqueando o par local.

A extensão altera a semântica de algumas mensagens do protocolo original e adiciona uma nova mensagem de rejeição à requisição de peça. A extensão é implementada através da criação de cinco novas mensagens, conforme apresentado na Tabela 6.8.

Tabela 6.8. Mensagens da Fast Extension

Nome	Descrição
HAVEALL	Notifica que possui todas as peças
HAVENONE	Notifica que possui nenhuma peça
SUGGEST PIECE	Sugere uma peça
REJECT REQUEST	Notifica que uma requisição de bloco não será atendida
ALLOWED FAST	Notifica que envia determinada peça mesmo se outro bloqueado

6.5.3.5. Superseeding

A "supersemeadura" (*superseeding*) é uma extensão que visa melhorar o desempenho dos semeadores que possuem largura de banda restrita. A ideia é racionalizar a distribuição de peças até que surjam novos semeadores no enxame. Diferentemente de um semeador normal, que notifica a posse de todas as peças a outros pares, o "supersemeador" (*superseeder*) se disfarça como um par normal, que não possui nenhuma parte do conteúdo.

Quando um par conecta ao supersemeador, este último lhe anuncia a disponibilidade de uma única peça, ou que nunca foi enviada ou que é muito rara. Dessa forma, o par tem como única opção, caso desbloqueado, solicitar a tal peça ao supersemeador. Este último não notifica o recebimento de uma nova peça ao par em questão, esperando até que esta peça seja vista em outro par do enxame vizinho ao supersemeador.

Com essa estratégia, o supersemeador distribui bem as peças e limita o gasto dos recursos escasso com pares que não contribuem com o enxame. Estudos como [Chen et al. 2008] apontam que, na maioria dos casos, o supersemeador poupa em torno de 20%

de upload realizado, até que surjam os primeiros semeadores no enxame se comparado a um semeador normal.

6.6. Estado da Arte

Nesta seção é apresentado, de forma sintética, um apanhado do estado-da-arte em relação ao BitTorrent. O primeiro tópico de pesquisa tratado diz respeito a estudos, tanto experimentais como de modelagem analítica, com o objetivo de compreender melhor a eficiência e funcionamento do protocolo em diferentes cenários, bem como o comportamento de usuários do sistema. O segundo tópico consiste na investigação sobre a justiça (*fairness*), os mecanismos de incentivo à colaboração entre pares e como eles lidam com os ditos pares egoístas. O comportamento egoísta por parte de usuários impacta negativamente no BitTorrent, e em sistemas P2P em geral, porém há uma forma mais nociva ainda, que é o comportamento malicioso de pares que executam ataques de negação de serviço. O terceiro tópico abordado nesta seção é o estudo de vulnerabilidades do BitTorrent, e o quarto, propostas de mecanismos de contra-medidas a esses ataques. Por fim, no último tópico, são discutidos trabalhos que analisam a relação entre ISPs e usuários de BitTorrent e propostas de soluções para tornar a convivência mais vantajosa para ambas as partes.

6.6.1. Medições e Modelagem

Segundo Cohen, o BitTorrent foi criado primariamente como um esforço de engenharia. Assim como outros sistemas de larga escala P2P, é muito difícil entender adequadamente como eles funcionam e prever as interações complexas entre pares [Hales e Patarin 2005]. Isso gerou uma demanda pela comunidade científica por medições e análises de BitTorrent, que permitissem melhor entendimento do protocolo. A seguir, serão discutidos trabalhos, experimentais e analíticos, que buscaram preencher esta lacuna. A discussão é iniciada pelos de caráter experimental.

Os trabalhos [Izal et al. 2004, Pouwelse et al. 2005] foram os primeiros dessa classe, apresentando dados referentes a enxames reais, e discussões sobre o desempenho de download, disponibilidade, integridade e chegada em massa de pares. Mais recentemente, as investigações buscaram aumentar o escopo e o período dos monitoramentos, a fim de oferecer dados mais representativos e atualizados.

Nessa linha, [Zhang et al. 2010] apresenta medições sobre comunidades privadas que indicam que estas comunidades oferecem melhor desempenho para os usuários que comunidades públicas. Em [Le Blond et al. 2010] são discutidos métodos mais eficientes de monitoramento e exploram tais métodos para observar que a privacidade dos usuários é vulnerável. Em [Zhang et al. 2011], é apresentado um monitoramento com nove meses de duração de quatro comunidades abertas, rastreadores e DHTs, tanto do Vuze, como do μtorrent. Em [Hossfeld et al. 2011] são apresentadas medições de enxames reais focadas em características relevantes para o desenvolvimento de mecanismos de otimização de tráfego entre ISPs, assunto que é tratado na Subseção 6.6.5.

Outros trabalhos experimentais focam especificamente na avaliação de desempenho e funcionamento dos algoritmos de seleção de peças e pares. Nesse contexto, em [Legout et al. 2006] os algoritmos são avaliados em ambiente real quanto à robustez do sis-

tema, e identificadas áreas de melhoria para o protocolo. A estratégia de seleção de pares é avaliada, em ambiente experimental controlado, em [Legout et al. 2007]. No referido trabalho observa-se, em particular, o agrupamento de nodos de acordo com similaridade da largura de banda dos pares. Outro estudo experimental [Dale e Liu 2007], realizado em ambientes real e controlado, apresenta uma avaliação específica do algoritmo de seleção de peças, a partir do monitoramento da evolução da distribuição das peças nos enxames.

Os estudos analíticos, por sua vez, se baseiam em observações do mundo real para propor modelos matemáticos que reflitam o desempenho de um enxame sob diferentes métricas, local ou globalmente. Em [Qiu e Srikant 2004], o enxame é descrito globalmente através de um modelo de fluídos, onde é estudado o impacto da variação de parâmetros e mecanismos em seu desempenho. Seguindo nessa linha, em [Tian et al. 2006] é apresentada uma modelagem matemática que foca no *período estável* do enxame, analisando os mecanismos oferecidos pelo protocolo. Considerando apenas o desempenho de um par local, cadeias de Markov são utilizadas para modelar o comportamento de pares-carona em [Barbera et al. 2005]. Trabalhos que propõem estratégias para esse comportamento são discutidos na próxima subseção.

6.6.2. Justiça

Trabalhos relacionados à justia estudam a possibilidade de modificar o protocolo BitTorrent a fim de "abusar" de outros pares que executam o protocolo conforme esperado. Basicamente, são propostos algoritmos que quebram a regra de reciprocidade.

Em [Locher et al. 2006], apresenta-se uma proposta e implementação de par-carona no contexto de BitTorrent, no qual um par malicioso apenas recebe conteúdo de outros pares e nunca contribui de volta. Para isso, o autor introduz o agente de usuário *BitThief* e discute formas de burlar o mecanismo de reciprocidade do BitTorrent.

Seguindo nessa mesma linha, o agente BitTyrant [Piatek et al. 2007] foi desenvolvido para demonstrar e estudar o *free-riding*. Nesse trabalho, é apresentada a construção do agente proposto com suas devidas justificativas, além de resultados comprovando sua eficácia. O BitTyrant procura determinar o montante exato de contribuição necessário para maximizar sua taxa de download adaptando dinamicamente a taxa de upload alocada para os vizinhos. Os resultados indicam que o agente modificado alcança isoladamente um ganho médio de 70%, mas que se aplicado universalmente poderia causar um colapso nos enxames.

Em [Carra et al. 2011], os autores avaliam em detalhes diversos mecanismos empregados pelo BitTyrant para identificar suas contribuições para o desempenho do agente. Os resultados indicam que o ganho deve-se principalmente ao número de conexões estabelecidas, mais do que à sub-utilização do canal de upload. Curiosamente, o BitTyrant mostrou-se altruísta e particularmente eficiente na disseminação do conteúdo, especialmente durante as fase iniciais do processo de distribuição. Contudo, o ganho obtido com o uso de um único agente desaparece quando o mesmo é adotado globalmente, causando grande perda de eficiência no enxame.

Uma estratégia diferente é omitir a posse de peças, opção investigada em [Levin et al. 2008]. Os autores propõem uma variante de agente BitTorrent que controla

racionalmente quais peças são anunciadas com o objetivo de manter seus vizinhos mais tempo interessados e contribuindo com dados. Desse modo, o agente modificado tem um controle mais estrito sobre a distribuição de peças para sua vizinhança, manipulando-a como se fosse um "leilão" para melhorar seu próprio desempenho. O trabalho também argumenta que o BitTorrent não implementa de fato uma política TFT. No modelo atual, a largura de banda de upload de um par é dividida igualmente entre os pares desbloqueados, independentemente de sua contribuição. Em função disso, é proposta uma divisão da largura de banda de upload de um par proporcionalmente ao volume de dados que cada vizinho contribui, tornando a interação mais justa.

Mais recentemente, pesquisadores buscaram combater as estratégias citadas acima, alegando que elas podem causar morte precoce do enxame por falta de contribuição dos usuários [Roy e Zeng 2009, Roy et al. 2010]. Os resultados indicam que o tratamento ortogonal das políticas de escolha de peças raras e de pares incentivam a manipulação racional das informações, permitindo a participação de pares desleais. No referido trabalho, é proposta uma solução baseada na unificação do algoritmo de peças raras com o algoritmo de escolha de pares. É discutido também um potencial ataque que, segundo os autores, poderia comprometer a maioria dos mecanismos TFT existentes. A análise demonstra como a sub-notificação de peças, conforme proposto em [Levin et al. 2008], poderia levar à morte do enxame. Como solução, é proposto o prTorrent, uma variante de BitTorrent. A mesma emprega uma formulação estratégica baseada na raridade das peças para otimizar os incentivos em um enxame, promovendo um comportamento "verdadeiramente cooperativo". Outros tipos de comportamento malicioso, além do egoísta, são abordados na subseção seguinte.

6.6.3. Ataques

Nesta subseção são discutidos trabalhos que tratam de ataques com o objetivo de impedir o download de outros pares (Negação de Serviço). Para tal, serão comentados possíveis ataques a redes BitTorrent e suas consequências.

Existem diferentes estratégias que um par pode usar para provocar negação de serviço em um enxame, incluindo enviar conteúdo embaralhado e subverter as mensagens de controle do protocolo. Dada a escala de redes BitTorrent, o impacto de um ataque provocado por um par isolado será inerentemente limitado, considerando que em um dado momento um par estará interagindo (conectado) com apenas uma fração da rede. Uma forma de se aumentar o impacto de ataques é subvertendo o esquema de identidades empregado no BitTorrent, permitindo a um único agente se fazer representar simultaneamente por múltiplas identidades falsas, denominadas *Sybils*. Na literatura, o ataque Sybil [Douceur 2002] foi utilizado como base para diferentes ataques, conforme discutido a seguir.

Os autores em [Konrath et al. 2007] e [Mansilha et al. 2007] identificaram três possíveis ataques ao BitTorrent: Eclipse, Mentira em Massa e Corrupção de Peças. O primeiro ataque, originalmente definido em [Singh 2006], consiste em cercar de maliciosos os pares corretos, de forma que estes últimos sejam "eclipsados" da rede. Para que esse ataque tenha sucesso, é necessário que exista um número bastante alto de Sybils: dezenas deles para cada par correto.

A ideia por trás do segundo ataque é evitar a homogeneidade na quantidade de cópias de cada peça. Um par malicioso pode anunciar peças que não possui, com o objetivo de interferir na escolha da próxima peça pelos outros pares. Ao anunciar uma peça que não possui, o par malicioso está artificialmente aumentando o nível de replicação da mesma e induzindo pares corretos a fazer download de outras peças primeiro. Em outras palavras, o ataque consiste em fazer com que peças se tornem raras ao ponto de desaparecer do enxame. Um pequeno conjunto de máquinas é suficiente para aumentar bastante o impacto desses ataques.

O terceiro ataque é denominado Corrupção de Peças [Mansilha et al. 2007]. Os dois ataques discutidos anteriormente demandam poucos recursos em termos de largura de banda, pois requerem apenas a gerência de conexões TCP e mensagens de controle do protocolo. No ataque de Corrupção, pares maliciosos fazem uploads de blocos incorretos, contaminando a integridade de dados de peças carregadas. Dada a sistemática atual de verificação de peças no BitTorrent (via função *hash*), um par correto não tem como identificar quais blocos estão incorretos. Por consequência, precisa descartar a peça por inteiro e realizar seu download novamente. Os autores verificaram através de simulação [Mansilha et al. 2008] que poucos atacantes são suficientes para causar um impacto devastador em um enxame.

Em [Dhungel et al. 2008] são estudados ataques ao semeador inicial do enxame com objetivo de minimizar sua capacidade enviar blocos a outros pares. Se um atacante conseguir descobrir e reagir suficientemente rápido ao surgimento de novos enxames, ele será capaz de evitar que os sugadores obtenham o conteúdo completo. São considerados dois ataques de negação de serviço: largura de banda e conexões, sendo este último uma variação de [Konrath et al. 2007]. O trabalho também discute métodos para descobrir enxames em estágios iniciais. Os autores concluem que o BitTorrent é potencialmente vulnerável a ataques aos semeadores iniciais, e fornecem orientações para melhorar a segurança do BitTorrent nesse aspecto.

Os ataques acima estão baseados na subversão do protocolo executado por agentes na montagem da topologia e na troca de peças. Outra classe de ataques, bem distinta, visa a fase de busca de conteúdo, antes que o torrent seja obtido. A "poluição de conteúdo" é um ataque de negação de serviço muito comum em redes P2P de compartilhamento de arquivos, ocasionado pela inserção de identificadores inválidos para conteúdos, induzindo o sistema a falhas no mecanismo da busca [Liang e Naoumov 2006]. Em [Cuevas et al. 2010], é apresentado um estudo sobre os publicadores de arquivos torrents nas comunidades. É demonstrado que uma pequena fração de publicadores são responsáveis por 67% dos conteúdos e 75% dos downloads. Esse grupo é dividido em duas classes. A primeira classe, responsável pela poluição, é composta por "agências anti-pirataria" e publicadores maliciosos, que disponibilizam um grande volume de conteúdo falso e espalham *malware*, respectivamente. A segunda classe é composta por empresas que recebem incentivos financeiros para publicar conteúdo.

Em [Santos et al. 2010], descreve-se um levantamento sobre índices de poluição junto a administradores de comunidades BitTorrent, que estimaram que 25% do conteúdo diário publicado necessita de algum tipo de moderação. Isso está consistente com [TorrentFreak 2009b], que apresenta relatos sobre a presença de conteúdo poluído em tais

comunidades.

6.6.4. Contra-Medidas

Os ataques descritos na subseção anterior motivaram a investigação e proposta de mecanismos de contramedida, para aumentar a segurança do BitTorrent. Primeiramente são comentados os mecanismos encontrados no protocolo original, assim como em implementações populares. Em seguida, são discutidos trabalhos que introduzem mecanismos de proteção contra comportamento malicioso, como aqueles apresentados na subseção anterior.

Conforme descrito na Seção 6.5.2.1, BitTorrent inclui um mecanismo ("*antisnubbing*") para acelerar a busca de melhores pares quando a comunicação se encontra estagnada, tal como ocorre no ataque de Eclipse. Entretanto, em princípio o mecanismo não fecha conexões. Em relação a peças corrompidas, os agentes BitTorrent mais populares possuem mecanismos de contra-medida, denominados usualmente de "*IP filters*", que punem pares que enviam peças corrompidas. A alternativa mais agressiva é banir todos os pares que contribuíram para uma peça. Um problema com esse esquema é que um par não malicioso que contribui para o par local com grande quantidade de dados, e consequentemente maior número de peças, tem maior chance de ser punido. Isso prejudicaria o próprio par local. Um mecanismo mais sofisticado, adotado no Vuze, consiste em monitorar quantas vezes cada par contribui para peças corrompidas, e comparar com a quantidade de peças corretas que este par tem enviado.

Em [Barcellos et al. 2008] são propostas duas contramedidas, denominadas Rotação de Pares e Anti-Corrupção, para combater os ataques comentados Eclipse, Mentira em Massa e Corrupção de Peças. Em termos gerais, a Rotação de Pares busca identificar os pares que constantemente permanecem "inativos", sem solicitar nem enviar blocos. Tais pares são considerados "suspeitos" e temporariamente desconectados, se os recursos que ficarão disponíveis habilitarem estabelecer conexões com pares potencialmente melhores.

No caso da Corrupção, emprega-se uma estratégia baseada em reputação para identificar pares maliciosos. Quando os pares são suspeitos de contribuírem com blocos corrompidos para uma peça, eles têm sua reputação diminuída, e aumentada contribuem com blocos de uma peça íntegra. Quando a reputação de um vizinho atinge um limiar mínimo, ele é desconectado e colocado em quarentena, similarmente ao caso anterior. As contra-medidas propostas foram implementadas e tiveram sua eficácia avaliada através de simulações. Para isto, cenários representativos foram montados, comparando situações em que a presença de ataque, assim como o mecanismo de contra-medida, são variados. Os resultados sugerem que os algoritmos propostos possam ser incorporados ao protocolo como contra-medidas, tornando implementações de agentes de usuário e aplicações derivadas mais seguras.

Para lidar com o problema da poluição de conteúdo em BitTorrent, [Santos et al. 2010] propõe um mecanismo, denominado *Funnel*. O seu objetivo é controlar a disseminação de conteúdo poluído em comunidades BitTorrent privadas. Funnel considera votos positivos e negativos atribuídos aos conteúdos, para classificá-los como autênticos ou poluídos. O princípio consiste em controlar de forma automática a distribuição das cópias de acordo com os votos emitidos: quanto maior a proporção de votos negativos, menos

downloads concorrentes são autorizados. Além disso, o Funnel dispõe de um mecanismo de incentivo para que os usuários emitam seus testemunhos sobre os conteúdos recuperados. A implantação depende da implementação do mecanismo em agentes populares e da participação de usuários para votarem em conteúdo que eles fizeram download.

Os autores em [Borch et al. 2010] descrevem *Closed Swarms* (CS), um sistema baseado em BitTorrent que acrescenta um controle de acesso a enxames, distinguindo pares autorizados e não-autorizados. Os resultados apresentados sugerem que usuários legítimos, confiáveis, conseguem um serviço de maior qualidade em relação aos não autorizados.

6.6.5. ISP

Redes BitTorrent oferecem robustez e escalabilidade para distribuição de conteúdo graças aos enormes recursos computacionais, de armazenamento e de comunicação disponibilizados coletivamente pelos pares participantes. Em contrapartida, o tráfego cada vez maior gerado por P2P e, em particular BitTorrent, tem gerado uma pressão sem precedentes nos ISPs, devido ao uso de largura de banda em enlaces dos ISPs com o restante da Internet.

A formação da topologia de um enxame é aleatória: um par recebe listas de pares escolhidas de forma randômica pelo rastreador. Quando um par recebe uma lista de IPs e estabelece vizinhança com pares em outros ISPs, é possível que existissem melhores alternativas de pares, no mesmo ISP. Do ponto de vista do ISP, isso gera um desperdício de recursos nos enlaces de comunicação entre ISPs. Para diminuir esse tráfego, tem sido amplamente estudada a exploração da localidade dos pares (*P2P locality*); tais trabalhos serão discutidos a seguir, iniciando pelos que são aplicáveis a redes P2P em geral, e seguido por trabalhos que discutem estratégias específicas para BiTorrent.

O trabalho seminal [Karagiannis et al. 2005] foi o primeiro a sugerir o uso da localidade em sistemas P2P a fim de reduzir a carga sobre as relações inter-ISP. Os autores mostram traços reais que indicam o potencial da localidade (particularmente, a existência de uma correlação espacial e temporal das solicitações de conteúdos) e avaliam várias soluções. Mais recentemente, [Xie et al. 2008] propôs a arquitetura P4P, que permite a cooperação entre provedores e aplicações P2P, incluindo BitTorrent. É demonstrado através de simulação e experimentos em ambiente real que o uso do P4P permite tanto uma redução do tráfego externo para os ISPs como acelerar na média o download dos pares. Outras propostas que exploram a cooperação entre ISPs e usuários são [Aggarwal et al. 2007, Aggarwal et al. 2008].

Sobre BitTorrent, em particular, [Bindal et al. 2006] introduz o conceito de "seleção tendenciosa de pares". A ideia é que as listas de pares enviadas pelos rastreadores tenham em sua maioria pares do mesmo ISPs do par solicitante, para aumentar as chances das conexões serem estabelecidas no mesmo ISP. Através de simulações de uma variedade de cenários, é verificado que o desempenho do BitTorrent se mantém próximo ao ótimo, enquanto o tráfego entre ISPs é reduzido drasticamente.

Em [Blond et al. 2011] os autores apresentam um estudo experimental detalhado sobre a implementação de localidade através de rastreador modificado. Primeiro é descrita a proposta dos autores para modificação do protocolo. A novidade em relação à proposta

anterior é a introdução de uma opção que força o rastreador a enviar pares de outros ISPs, e que deve ser utilizada para aumentar a conectividade da rede quando necessário. O protótipo é avaliado em ambiente controlado e os ganhos alcançados são projetados considerando dados obtidos de medições do universo BitTorrent. Essas projeções indicam a possibilidade de redução de tráfego inter-ISP em até 40%.

Rastreadores modificados para explorar localidade também foram alvo de estudo em [Wang et al. 2010], porém através de monitoramentos de enxames e pares reais. A análise sobre os enxames mostra uma ampla adoção de múltiplos rastreadores, e os autores concluem que isso prejudica a eficácia dos rastreadores modificados: como os pares escolhem o rastreador aleatoriamente, não existe garantia que será selecionado um rastreador modificado. Uma segunda análise, dessa vez sobre a correlação entre ISPs e rastreadores dos pares, verificou que na prática a grande maioria dos pares se concentra em uma parcela relativamente pequena de rastreadores populares. Os autores concluem que para a estratégia proposta ser efetiva, seria necessário que os rastreadores populares cooperassem com os ISPs.

Diferentemente das abordagens baseadas na modificação de rastreadores, [Choffnes e Bustamante 2008] apresenta o Ono, uma extensão de BitTorrent que usufrui de uma infraestrutura de CDN para localização dos pares na rede, a fim de agrupar aqueles que estão próximos uns dos outros. Resultados de uma avaliação experimental em grande escala mostram uma redução no tempo de conclusão do download dos pares e também no tráfego inter-ISPs.

Por fim, a proposta em [Lin et al. 2010] requer apenas a alteração do agente de usuário. A ideia é classificar os pares vizinhos em dois grupos, locais (mesmo ISP) e remotos (outro ISP), e usar essa informação nos algoritmos de envio de mensagem INTERESTED e de seleção de peças. Os agentes modificados solicitam primeiro peças que estão disponíveis nos pares no mesmo ISP, complementando o conjunto de peças com solicitações a pares em ISPs remotos. Para avaliar a proposta, foram realizados experimentos em ambiente real no PlanetLab, que mostraram uma redução do tráfego inter-ISPs aliada a uma melhoria no tempo de download.

6.7. Conclusões

Existem diferentes definições de P2P. Umas se concentram na arquitetura (ausência de servidor ou recursos centrais), outras na colaboração entre usuários (compartilhamento de recursos de computadores nas bordas da Internet), ou ainda nas possibilidades criadas pelas aplicações (identificação de vida extra-terrestre, compartilhamento de música, filmes e software, *streaming* ao vivo). Até hoje, na visão do público leigo, o termo "*Peer-to-Peer*" parece estar associado ao compartilhamento de arquivos. Tal se deve à importância do mesmo no contexto de P2P: foi indiscutivelmente a "aplicação matadora", responsável pela imensa popularização desse modelo na Internet.

Nesse contexto, Napster e Gnutella foram os precursores, e deram início a uma guerra de "gato e rato" entre usuários e a indústria de conteúdo. Essa guerra teve diversas batalhas, tais como os processos judiciais contra o Napster e a introdução de poluição de conteúdo na rede Kazaa.

O BitTorrent foi lançado em meio a essa disputa, em 2001, trazendo pelo menos duas inovações: o uso de *swarming* e a separação do mecanismo de busca do protocolo da parte de download. Elas contribuíram para resolver uma série de questões intrínsecas às redes P2P. Por exemplo, o uso de *swarming* criou as condições necessárias a um mecanismo de incentivo que fosse efetivo, ao passo que a divisão da rede em enxames aumentou a robustez dos sistemas de compartilhamento de arquivos contra ataques de negação de serviço. Além disso, o protocolo foi distribuído com código fonte e permitiu que a comunidade contribuísse para a sua evolução.

BitTorrent foi uma aplicação revolucionária para a Internet por conta do estilo de compartilhamento que introduziu e massificou, tendo impacto em diferentes setores da sociedade. No campo científico, introduziu diversas inovações, conforme a análise do estado-da-arte, na Seção 6.6. Os princípios estabelecidos pelo BitTorrent continuam a influenciar, de maneira importante, o desenvolvimento de novas tecnologias para a Internet, tais como sistemas P2P de mídia contínua (*streaming*) [Dana et al. 2005, Vlavianos et al. 2006].

O impacto de BitTorrent se deu também no âmbito da Indústria, forçando os ISPs, provedores de conteúdo e detentores de direitos autorais em geral a repensar seus modelos de negócio. Essa mudança de paradigma gerou novas oportunidades; por exemplo, BitTorrent está sendo usado para disponibilizar conteúdos em larga escala, como atualizações de softwares [TorrentFreak 2009a], distribuição de jogos [TorrentFreak 2008b] e a distribuição legítima de filmes [TorrentFreak 2011a]. Além disso, influencia o desenvolvimento de novas tecnologias em empresas, como a propagação de atualizações a milhares de servidores do Twitter distribuídos pelo mundo [Engineering 2010]. Por fim, novos produtos estão sendo lançados pela indústria, tais como o BitTorrent DNA [Inc. 2011a] e dispositivos com suporte a BitTorrent [Inc. 2011b].

Referências

[3rd e Jones 2001] 3rd, D. e Jones, P. (2001). US secure hash algorithm 1 (SHA1). RFC 3174, Internet Engineering Task Force.

[ABCNews 2001] ABCNews (2001). Napster shut down. Disponível em http://abcnews.go.com/Technology/story?id=119627.

[Aggarwal et al. 2008] Aggarwal, V., Akonjang, O. e Feldmann, A. (2008). Improving user and isp experience through isp-aided p2p locality. In *INFOCOM Workshops 2008, IEEE*, pp. 1 –6.

[Aggarwal et al. 2007] Aggarwal, V., Feldmann, A. e Scheideler, C. (2007). Can isps and p2p users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 37:29–40.

[Andrade et al. 2005] Andrade, N., Mowbray, M., Lima, A., Wagner, G. e Ripeanu, M. (2005). Influences on cooperation in bittorrent communities. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, P2PECON '05, pp. 111–115, New York, NY, USA. ACM.

[Ares 2002] Ares (2002). Ares website. Disponível em http://www.ares.net/.

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

[Barcellos et al. 2008] Barcellos, M. P., Bauermann, D., Sant'anna, H., Lehmann, M. e Mansilha, R. (2008). Protecting bittorrent: design and evaluation of effective countermeasures against dos attacks. In *27th International Symposium on Reliable Distributed Systems, IEEE SRDS 2008*.

[Barcellos e Gaspary 2006] Barcellos, M. P. e Gaspary, L. P. (2006). *Fundamentos, Tecnologias e Tendências rumo a Redes P2P Seguras*, volume 1, capítulo 4, pp. 1–57. Ed PUC-RIO.

[Bindal et al. 2006] Bindal, R., Cao, P., Chan, W., Medved, J., Suwala, G., Bates, T. e Zhang, A. (2006). Improving traffic locality in bittorrent via biased neighbor selection. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, ICDCS '06, pp. 66–, Washington, DC, USA. IEEE Computer Society.

[BitTornado 2004] BitTornado (2004). Bittornado website. Disponível em http://bittornado.com/.

[BitTorrent 2001] BitTorrent (2001). Bittorrent website. Disponível em http://www.bittorrent.com/.

[Blond et al. 2011] Blond, S. L., Legout, A. e Dabbous, W. (2011). Pushing bittorrent locality to the limit. *Computer Networks*, 55(3):541–557.

[Borch et al. 2010] Borch, N. T., Michell, K., Arntzen, I. e Gabrijelcic, D. (2010). Access control to bittorrent swarms using closed swarms. In *Proceedings of the 2010 ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking*, AVSTP2P '10, pp. 25–30, New York, NY, USA. ACM.

[Carra et al. 2011] Carra, D., Neglia, G., Michiardi, P. e Albanese, F. (2011). On the robustness of bittorrent swarms to greedy peers. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1.

[Chen et al. 2008] Chen, Z., Chen, Y., Lin, C., Nivargi, V. e Cao, P. (2008). Experimental analysis of super-seeding in bittorrent. In *IEEE International Conference on Communications, 2008. ICC '08*, pp. 65 –69.

[Choffnes e Bustamante 2008] Choffnes, D. R. e Bustamante, F. E. (2008). Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pp. 363–374, New York, NY, USA. ACM.

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

[Cohen 2008a] Cohen, B. (2008a). The bittorrent protocol specification. Disponível em http://www.bittorrent.org/beps/bep_0003.html.

[Cohen 2008b] Cohen, B. (2008b). The bittorrent protocol specification. Website. http://www.bittorrent.org/beps/bep_0003.html.

[Crocioni 2011] Crocioni, P. (2011). Net neutrality in europe: Desperately seeking a market failure. *Telecommun. Policy*, 35:1–11.

[Crowcroft 2007] Crowcroft, J. (2007). Net neutrality: the technical side of the debate: a white paper. *SIGCOMM Comput. Commun. Rev.*, 37:49–56.

[Cuevas et al. 2010] Cuevas, R., Kryczka, M., Cuevas, A., Kaune, S., Guerrero, C. e Rejaie, R. (2010). Is content publishing in bittorrent altruistic or profit-driven? In *Proceedings of the 6th International COnference*, Co-NEXT '10, pp. 11:1–11:12, New York, NY, USA. ACM.

[Dailytech 2008] Dailytech (2008). Napster sinking fast, tries to fight off ice cream store owner. Disponível em http://www.dailytech.com/Napster+Sinking+Fast+Tries+to+Fight+Off+Ice+Cream+Store+Owner/article12547.htm.

[Dale e Liu 2007] Dale, C. e Liu, J. (2007). A measurement study of piece population in bittorrent. *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pp. 405–410.

[Dana et al. 2005] Dana, C., Li, D., Harrison, D. e Chuah, C.-N. (2005). Bass: Bittorrent assisted streaming system for video-on-demand. In *IEEE 7th Workshop on Multimedia Signal Processing, 2005*, pp. 1 –4.

[Dhungel et al. 2008] Dhungel, P., Heiz, X., Wu, D. e Ross, K. W. (2008). The seed attack: Can bittorrent be nipped in the bud? Relatório técnico.

[Douceur 2002] Douceur, J. R. (2002). The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pp. 251–260, London, UK. Springer-Verlag.

[Engineering 2010] Engineering, T. (2010). Murder: Fast datacenter code deploys using bittorrent. Disponível em http://engineering.twitter.com/2010/07/murder-fast-datacenter-code-deploys.html.

[Envisional 2011] Envisional (2011). An estimate of infringing use of the internet. Disponível em http://documents.envisional.com/docs/Envisional-Internet_Usage-Jan2011.pdf.

[Gnutella 2003] Gnutella (2003). Gnutella rfc. Disponível em http://rfc-gnutella.sourceforge.net/.

[Gnutella2 2003] Gnutella2 (2003). Gnutella2 website. Disponível em http://g2.trillinux.org/.

[Hales e Patarin 2005] Hales, D. e Patarin, S. (2005). Computational sociology for systems "in the wild": the case of BitTorrent. *IEEE Distributed Systems Online*, 6(7).

[Harrison e Cohen 2008] Harrison, D. e Cohen, B. (2008). Fast extension. Disponível em http://www.bittorrent.org/beps/bep_0006.html.

[Hoffman 2008] Hoffman, J. (2008). Superseeding. Disponível em http://www.bittorrent.org/beps/bep_0016.html.

[Hossfeld et al. 2011] Hossfeld, T., Lehrieder, F., Hock, D., Oechsner, S., Despotovic, Z., Kellerer, W. e Michel, M. (2011). Characterization of bittorrent swarms and their distribution in the internet. *Computer Networks*, 55(5):1197 – 1215.

[Inc. 2011a] Inc., B. (2011a). Bittorrent delivery network accelerator (dna). Disponível em http://www.bittorrent.com/dna/.

[Inc. 2011b] Inc., B. (2011b). Bittorrent devices. Disponível em http://www.bittorrent.com/devices/.

[Izal et al. 2004] Izal, M., Urvoy-Keller, G., Biersack, E., Felber, P., Al-Hamra, A. e Garcés-Erice, L. (2004). Dissecting bittorrent: Five months in a torrent's lifetime. In Chadi, editor, *5th International Workshop, PAM 2004*, volume 3015 / 2004, pp. 1–11. Springer Berlin / Heidelberg.

[Karagiannis et al. 2005] Karagiannis, T., Rodriguez, P. e Papagiannaki, K. (2005). Should internet service providers fear peer-assisted content distribution? In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pp. 6–6, Berkeley, CA, USA. USENIX Association.

[Konrath et al. 2007] Konrath, M. A., Barcellos, M. P. e Mansilha, R. B. (2007). Attacking a swarm with a band of liars: evaluating the impact of attacks on bittorrent. In *P2P '07. The Seventh IEEE International Conference on Peer-to-Peer Computing, 2007*. IEEE.

[Le Blond et al. 2010] Le Blond, S., Legout, A., Lefessant, F., Dabbous, W. e Kaafar, M. A. (2010). Spying the world from your laptop: identifying and profiling content providers and big downloaders in bittorrent. In *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, LEET'10.

[Legout et al. 2007] Legout, A., Liogkas, N., Kohler, E. e Zhang, L. (2007). Clustering and sharing incentives in bittorrent systems. In *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 301–312, New York, NY, USA. ACM Press.

[Legout et al. 2006] Legout, A., Urvoy-Keller, G. e Michiardi, P. (2006). Rarest first and choke algorithms are enough. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pp. 203–216, New York, NY, USA. ACM.

[Levin et al. 2008] Levin, D., LaCurts, K., Spring, N. e Bhattacharjee, B. (2008). Bittorrent is an auction: analyzing and improving bittorrent's incentives. *SIGCOMM Comput. Commun. Rev.*, 38(4):243–254.

[Liang e Naoumov 2006] Liang, J. e Naoumov, N. (2006). The index poisoning attack in p2p file sharing systems. *Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM 2006*.

[Lin et al. 2010] Lin, M., Lui, J. C. S. e Chiu, D.-M. (2010). An isp-friendly file distribution protocol: Analysis, design, and implementation. *IEEE Trans. Parallel Distrib. Syst.*, 21:1317–1329.

[Ling et al. 2010] Ling, F.-Y., Tang, S.-L., Wu, M., Li, Y.-X. e Du, H.-Y. (2010). Research on the net neutrality: The case of comcast blocking. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pp. V5–488 –V5–491.

[Linuxtracker 2011] Linuxtracker (2011). Linuxtracker website. Disponvelemhttp: //linuxtracker.org/.

[Locher et al. 2006] Locher, T., Moor, P., Schmid, S. e Wattenhofer, R. (2006). Free riding in bittorrent is cheap. In *Fifth Workshop on Hot Topics in Networks (HotNets-V)*, Irvine, CA, US.

[Loewenstern 2008] Loewenstern, A. (2008). Bittorrent dht protocol. Disponível em http://www.bittorrent.org/beps/bep_0005.html.

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

[Mansilha et al. 2008] Mansilha, R., Barcellos, M. P. e Brasileiro, F. (2008). Torrentlab: Um ambiente para avaliação do protocolo bittorrent. *XXVI Simposio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC 2008)*, pp. 1–14.

[Mansilha et al. 2007] Mansilha, R., Konrath, M. A. e Barcellos, M. P. (2007). Corrupção, mentiras e isolamento: avaliação de impacto de ataques a bittorrent. pp. 1–14.

[Marciniak et al. 2008] Marciniak, P., Liogkas, N., Legout, A. e Kohler, E. (2008). Small is not always beautiful. In *Proceedings of the 7th international conference on Peer-to-peer systems*, IPTPS'08, pp. 9–9, Berkeley, CA, USA. USENIX Association.

[Massoulié e Vojnović 2005] Massoulié, L. e Vojnović, M. (2005). Coupon replication systems. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '05, pp. 2–13, New York, NY, USA. ACM.

[Maymounkov e Mazières 2002] Maymounkov, P. e Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pp. 53–65, London, UK. Springer-Verlag.

[Megaupload 2005] Megaupload (2005). Megaupload website. Disponível em www.megaupload.com/.

[Mennecke 2004] Mennecke, T. (2004). Retspan seeks to have suprnova eliminated. Disponível em http://www.slyck.com/news.php?story=602.

[μTorrent 2006] μTorrent (2006). μtorrent website. Disponível em http://www.utorrent.com/.

[Overpeer 2002] Overpeer (2002). Overpeer website. Disponível em http://www.overpeer.com.

[Piatek et al. 2007] Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A. e Venkataramani, A. (2007). Do incentives build robustness in bittorrent? In *Proceedings of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 2007)*, Cambridge, MA. USENIX.

[Piatek et al. 2009] Piatek, M., Madhyastha, H. V., John, J. P., Krishnamurthy, A. e Anderson, T. (2009). Pitfalls for isp-friendly p2p design. In *Eighth ACM Workshop on Hot Topics in Networks (HotNets VIII)*.

[Pouwelse et al. 2005] Pouwelse, J. A., Garbacki, P., Epema, D. H. J. e Sips, H. J. (2005). The bittorrent p2p file-sharing system: Measurements and analysis. In *4th International Workshop on Peer-to-Peer Systems, IPTPS*.

[Qiu e Srikant 2004] Qiu, D. e Srikant, R. (2004). Modeling and performance analysis of bittorrent-like peer-to-peer networks. *ACM SIGCOMM Computer Communication Review*, 34(4):367–378.

[Rapidshare 2005] Rapidshare (2005). Rapidshare website. Disponível em www.rapidshare.com/.

[RIAA 2011] RIAA (2011). Riaa website. Disponível em http://riaa.com/.

[Ripeanu et al. 2006] Ripeanu, M., Mowbray, M., Andrade, N. e Lima, A. (2006). Gifting technologies: A bittorrent case study. *First Monday*, 11(11).

[Robinson e Coar 2004] Robinson, D. e Coar, K. (2004). The common gateway interface (CGI) version 1.1. RFC 3875, Internet Engineering Task Force.

[Roy e Zeng 2009] Roy, S. e Zeng, W. (2009). prtorrent: On establishment of piece rarity in the bittorrent unchoking algorithm. In *P2P '09. IEEE Ninth International Conference on Peer-to-Peer Computing, 2009*, pp. 252 –261.

[Roy et al. 2010] Roy, S. D., Knierim, T., Churchman, S. e Zeng, W. (2010). Design issues of the prtorrent file sharing protocol. In *Proceedings of the 7th IEEE conference on Consumer communications and networking conference*, CCNC'10, pp. 338–339, Piscataway, NJ, USA. IEEE Press.

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

[Sandvine 2010] Sandvine (2010). Fall 2010 global internet phenomena report. Disponível em http://www.sandvine.com/downloads/documents/2010GlobalInternetPhenomenaReport.pdf.

[Santos et al. 2010] Santos, F., da Costa Cordeiro, W., Gaspary, L. e Barcellos, M. (2010). Choking polluters in bittorrent file sharing communities. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pp. 559 –566.

[Schulze e Mochalski 2009] Schulze, H. e Mochalski, K. (2009). Internet study 2008/2009, https://portal.ipoque.com/. *Ipoque*, pp. 1–14.

[Singh 2006] Singh, A. (2006). Eclipse attacks on overlay networks: Threats and defenses. *Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM 2006*.

[Sirivianos et al. 2007] Sirivianos, M., Park, J. H., Chen, R. e Yang, X. (2007). Free-riding in bittorrent with the large view exploit. In *6th International Workshop on Peer-to-Peer Systems (IPTPS 2007)*, Bellevue, WA, US.

[Tian et al. 2006] Tian, Y., Wu, D. e Ng, K. (2006). Modeling, analysis and improvement for bittorrent-like file sharing networks. *Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM 2006*, pp. 1–11.

[TorrentFreak 2006] TorrentFreak (2006). Suprnova.org: Two years since the shutdown. Disponível em http://torrentfreak.com/suprnovaorg-two-years-since-the-shutdown/.

[TorrentFreak 2008a] TorrentFreak (2008a). Comcast ordered to stop bittorrent traffic interference. Disponível em http://torrentfreak.com/comcast-ordered-to-stop-bittorrent-traffic-interference-080711/.

[TorrentFreak 2008b] TorrentFreak (2008b). Ea choose bittorrent for warhammer online distribution. Disponível em http://torrentfreak.com/ea-choose-bittorrent-for-warhammer-online-distribution-080813/.

[TorrentFreak 2009a] TorrentFreak (2009a). Asus uses bittorrent to boost software downloads. Disponível em http://torrentfreak.com/asus-uses-bittorrent-to-boost-downloads-090720/.

[TorrentFreak 2009b] TorrentFreak (2009b). Fake axxo torrents bombard bittorrent. Disponível em http://torrentfreak.com/fake-axxo-torrents-bombard-bittorrent-090313/.

[TorrentFreak 2009c] TorrentFreak (2009c). Mininova deletes all infringing torrents and goes 'legal'. Disponível em http://torrentfreak.com/mininova-deletes-all-infringing-torrents-and-goes-legal-091126/.

[TorrentFreak 2009d] TorrentFreak (2009d). utorrent still on top, bitcomet's market share plummets. Disponível em http://torrentfreak.com/utorrent-still-on-top-bitcomets-market-share-plummets-090814/.

[TorrentFreak 2010a] TorrentFreak (2010a). Comcast can block bittorrent again, court rules. Disponível em http://torrentfreak.com/comcast-can-block-bittorrent-again-court-rules-100406/.

[TorrentFreak 2010b] TorrentFreak (2010b). Net neutrality wont prevent bittorrent blocking. Disponível em http://torrentfreak.com/net-neutrality-wont-prevent-bittorrent-blocking-10-01-29/.

[TorrentFreak 2010c] TorrentFreak (2010c). The pirate bay appeal verdict: Guilty again. Disponível em http://torrentfreak.com/the-pirate-bay-appeal-verdict-101126/.

[TorrentFreak 2011a] TorrentFreak (2011a). Paramount pictures partner with bittorrent release movie. Disponível em http://torrentfreak.com/paramount-pictures-partner-with-bittorrent-release-movie-110317/.

[TorrentFreak 2011b] TorrentFreak (2011b). utorrent & bittorrent hit 100 million monthly users. Disponível em http://torrentfreak.com/utorrent-bittorrent-hit-100-million-monthly-users-110103/.

[Vlavianos et al. 2006] Vlavianos, A., Iliofotou, M. e Faloutsos, M. (2006). Bitos: Enhancing bittorrent for supporting streaming applications. *Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM 2006*.

[Vuze 2003] Vuze (2003). Vuze website. Disponível em http://www.vuze.com/.

[Wallsten e Hausladen 2009] Wallsten, S. e Hausladen, S. (2009). Net neutrality, unbundling, and their effects on international investment in next-generation networks. *Review of Network Economics*, 8(1):6.

[Wang et al. 2010] Wang, H., Liu, J., Chen, B., Xu, K. e Ma;, Z. (2010). On tracker selection for peer-to-peer traffic locality. *P2P '10. 2010 IEEE Tenth International Conference on Peer-to-Peer Computing*, pp. 1 – 10.

[Weitzner 2008] Weitzner, D. (2008). Net Neutrality... Seriously this Time. *Internet Computing, IEEE*, 12(3):86–89.

[WikiTheory 2008] WikiTheory (2008). Peer exchange conventions. Disponível em http://wiki.theory.org/BitTorrentPeerExchangeConventions.

[WikiTheory 2011] WikiTheory (2011). Bittorrent protocol specification v1.0. Disponível em http://wiki.theory.org/BitTorrentSpecification.

[WikiVuze 2010] WikiVuze (2010). Peer exchange. Disponível em http://wiki.vuze.com/w/Peer_Exchange.

[WikiVuze 2011] WikiVuze (2011). Wikivuze website. Disponível em http://wiki.vuze.com.

[Wu et al. 2010] Wu, D., Dhungel, P., Hei, X., Zhang, C. e Ross, K. (2010). Understanding peer exchange in bittorrent systems. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pp. 1 –8.

[Xie et al. 2008] Xie, H., Yang, Y. R., Krishnamurthy, A., Liu, Y. G. e Silberschatz, A. (2008). P4p: provider portal for applications. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pp. 351–362, New York, NY, USA. ACM.

[Yang e Garcia-Molina 2003] Yang, B. e Garcia-Molina, H. (2003). Designing a super-peer network. *Data Engineering, International Conference on*, 0:49.

[Zhang et al. 2010] Zhang, C., Dhungel, P., Wu, D., Liu, Z. e Ross, K. (2010). Bittorrent darknets. *Proceedings of 29th IEEE International Conference on Computer Communications, INFOCOM 2010*, pp. 1 – 9.

[Zhang et al. 2011] Zhang, C., Dhungel, P., Wu, D. e Ross, K. (2011). Unraveling the bittorrent ecosystem. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1.