**UNIVERSIDADE DO VALE DO RIO DOS SINOS**

**Programa Interdisciplinar de Pós-Graduação em**

# Computação Aplicada

**Mestrado Acadêmico**

Giovani Facchini

# *TorrentU*: Uma Arquitetura Flexível para Obtenção de Informações sobre o Universo de Redes BitTorrent

**Giovani Facchini**

# *TorrentU*: Uma Arquitetura Flexível para Obtenção de Informações sobre o Universo de Redes BitTorrent

Dissertação apresentada como requisito parcial para obtenção do título de Mestre, pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio Dos Sinos

Prof. Dr. Jorge Luis Victória Barbosa
Orientador

São Leopoldo, Fevereiro de 2009

Dedico esse trabalho a todos aqueles que, de alguma forma, contribuíram para sua realização e conclusão.

# AGRADECIMENTOS

Para conclusão do mestrado, foi necessário apoio de pessoas que colaboram tanto com o trabalho quanto com a vida do mestrando, para que este consiga dar seus passos. Nessa caminhada, pude contar com o apoio de muitas pessoas e cito aqui aquelas que mais contribuíram.

Primeiramente, agradeço ao Professor Dr. Marinho Barcellos, meu Orientador. Ele foi o responsável por me motivar a entrar no mestrado e apoiou o trabalho desde seu início até sua conclusão, mesmo não estando mais ligado a instituição Unisinos. Sua dedicação possibilitou a continuidade do trabalho. Incontável é o tempo disponibilizado pelo Marinho para me ajudar. Dessa forma, agradeço imensamente o seu apoio nos meus seis anos envolvidos com a pesquisa (desde a graduação).

Ao Daniel Bauermann, colega do mestrado, que configurou e manteve o ambiente de testes, possibilitando que eu pudesse executar meus experimentos. Sua ajuda foi fundamental para entrega dos resultados dentro do tempo planejado.

Ao Fábio, popularmente conhecido como *Root* – o Laboratorista, por ceder os acessos para que nós pudéssemos executar experimentos no *cluster* do laboratório.

Aos colegas de mestrado pelos momentos de desabafo e por ajudar perceber que as adversidades são apresentadas para todos.

Ao professor Dr. Luciano Paschoal Gaspary pelas discussões a respeito do trabalho e seus rumos.

À minha mãe, Miriam Bertin Facchini, por todo o apoio psicológico e pressão para que eu me concentra-se no trabalho. Além disso, por me ajudar a cuidar do apartamento, pelos almoços e todo o mimo que uma mãe é capaz de fazer.

Ao meu pai, Volmar Luiz Facchini, pelos churrascos do domingo e pelo apoio financeiro.

Ao grupo *Toastmasters*, que me possibilitou grande desenvolvimento pessoal, culminando em uma ótima apresentação na defesa do trabalho de mestrado.

Ao professor Dr. Jorge Barbosa, por fazer o seu papel na transição de orientador para que eu pudesse concluir os requisitos necessários para obter o título de Mestre.

Por fim, a todos meus amigos, colegas da Dell, ex-colegas da HP e ao pessoal do FBI por todos os momentos de laser e diversão, que fazem a vida mais alegre.

## RESUMO

Trabalhos recentes na literatura mencionam a importância de aplicações par-a-par, principalmente BitTorrent, em particular se apoiando em estudos divulgados por provedores de Internet. Apesar da importância dos dados levantados pelos provedores, tais estudos são limitados tanto em abrangência como em profundidade e, em certos casos, não há clareza quanto à metodologia empregada. Este trabalho apresenta *TorrentU*, uma arquitetura escalável e flexível para observação do universo de redes BitTorrent. O objetivo do *TorrentU* é ajudar a elucidar uma série de questões que tem sido levantadas quanto ao volume e teor de compartilhamento de arquivos em redes BitTorrent.

**Palavras-chave**: BitTorrent, P2P

## ABSTRACT

Recently, academic investigations mention the importance of peer-to-peer applications, mainly BitTorrent, focusing on studies published by internet providers as a source of information. Despite the importance of these collected data from internet providers, the studies behind this technique are limited in coverage, deepness, and, in some cases, do not show clarity in the method applied for retrieving the data. This work presents *TorrentU*, an scalable and flexible architecture for the observation of the BitTorrent universe. The main goal is to help on finding answers to a wide range of questions that has been raised regarding volume and types of files and content in BitTorrent networks.

**Keywords**: BitTorrent, P2P

# Lista de Abreviaturas

| | |
|---|---|
| ack | acknowledge é uma mensagem de confirmação utilizada pelo protocolo TCP |
| ADSL | Asymmetric Digital Subscriber Line |
| DoS | Denial of Service |
| HTML | HyperText Markup Language |
| IP | Internet Protocol Address (número que indica o endereço de uma entidade na Internet) |
| LRF | Local Rarest First (Algorítmo utilizado no protocolo BitTorrent) |
| NAT | Network Translation Address |
| P2P | Peer-to-Peer |
| RTT | Round Trip Time |
| TCP | Transmission Control Protocol |
| URL | Uniform Resource Locator |

**Lista de Figuras**

# SUMÁRIO

# 1 INTRODUÇÃO

Muitos são os protocolos que utilizam o conceito de redes par-a-par (P2P) para comunicação e compartilhamento de conteúdo. Dentre estes, o BitTorrent ganhou foco, tornando-se o protocolo mais utilizado pela comunidade mundial para disseminação de conteúdo, segundo estudos recentes, pois a utilização deste protocolo representou, na virada de 2007 para 2008, cerca de 60% do consumo mundial de transmissão de dados [13]. Neste cenário, empresas como a CacheLogic [10] fazem estudos e oferecem soluções para reduzir o impacto que o crescimento da utilização de redes P2P causa aos provedores de Internet. Empresas como esta mostram a influência que redes P2P, com especial atenção ao BitTorrent, têm tanto na indústria de provedores quanto na de criação de conteúdo.

Devido a esse grande percentual de banda dos enlaces que o protocolo BitTorrent utiliza, é possível considerá-lo o método de disseminação de dados mais expressivo e importante. Isso acontece porque o protocolo provê incentivos para que os pares que participam de uma rede BitTorrent colaborem entre si de forma a distribuir o conteúdo desejado sem onerar apenas um ponto da rede [6]. Essa disseminação de dados é feita de forma distribuída, pois os pares que desejam obter o conteúdo precisam colaborar para trocar informações.

Para estudar a dinâmica do protocolo e o seus impactos, encontram-se na literatura três categorias de métodos utilizados para este fim [14]: modelagem analítica, simulação e experimentação. Cada categoria apresenta características importantes e podem ser usadas de forma colaborativa, ou seja, utilização de mais de uma delas para atingir resultados mais consistentes. Uma explicação destas categorias é apresentada a seguir.

A modelagem analítica fornece uma compreensão mais abstrata e global do funcionamento dos protocolos. Sua representação é uma forma aproximada de representar o comportamento que um determinado protocolo irá exercer sobre uma condição específica. As principais vantagens são a facilidade de observar as tendências gerais de comportamento providas pelo modelo e a análise mais abstrata que não se preocupa com questões específicas de implementação e condições externas. A desvantagem desta abordagem é que os modelos, muitas vezes, assumem premissas consideradas fortes (não realistas) que podem comprometer uma parcela do resultado final. Além disto, o modelo mais abstrato não é capaz de prever situações mais específicas de cenários realísticos [3], [8], [18].

As simulações, por sua vez, exploram cenários com mais precisão que a modelagem analítica, pois consideram mais variáveis envolvidas e programa-se a lógica dos protocolos com um nível de detalhe maior [15], [16], [17]. A simulação tem a vantagem de poder variar o nível de detalhe entre experimentos, tomando-se diferentes premissas para cada execução. Isso dá ao pesquisador a oportunidade de descrever o comportamento sobre diferentes níveis de abstração e de melhorar o modelo progressivamente. A possibilidade de cenários explorados através de simulação é maior quando comparada a análise experimental (explorado posteriormente), devido aos problemas logísticos exibidos. A desvantagem da simulação é a necessidade de alto poder de processamento para cenários grandes e complexos. Além disso, existe dificuldade de reprodução dos resultados por outros grupos de pesquisa (quando códigos não são liberados), pois a simulação contém peculiaridades no código de quem a desenvolve.

Por fim, experimentos realizados em ambiente real [2], [5], [7], [9] (controlado ou não) mostram peculiaridades específicas sobre o comportamento dos protocolos que podem não ser previstos utilizando os outros métodos de modelagem. A utilização de ambientes controlados como redes locais com softwares para controle de banda (atraso, largura, perdas, etc) são úteis para aprimorar o protótipo no momento da criação e podem ser utilizados para obter informações iniciais sobre cenários de interesse. Com execuções em ambiente real (Internet de longa distância) é possível levantar informações precisas sobre o ambiente avaliado, pois variáveis como comportamento de usuário, tráfego de fundo, perdas e outros efeitos estão sendo considerados. Outra vantagem é que o resultado de um experimento de rede real é o mais preciso para o cenário explorado. As desvantagens deste tipo de abordagem são a alta influência externa e a falta de controle sobre os experimentos. Problemas de logística como obtenção de hardware, de enlaces disponíveis e controle sobre tráfego de fundo dificulta a execução deste tipo de estudo. Além disso, se existirem outras pessoas executando experimentos no mesmo ambiente, há grande chance de um experimento causar distorções nos resultados do outro.

Nesse contexto, o presente trabalho tem como foco criar uma arquitetura para realização de experimentos em rede real, permitindo "fazer um raio-x" do universo de redes BitTorrent hoje na Internet. Para atingir esse objetivo, escolheu-se realização de experimentos em ambiente real, pois os mesmos permitem obtenção de resultados precisos, além de ser o único método que pode explorar os cenários reais presentes na Internet. Como benefício, este trabalho contribuirá para a melhor compreensão sobre o funcionamento do protocolo em larga escala, bem como do comportamento de seus usuários.

## 1.1 Motivação

Existem publicações e estudos veiculados em sites especializados em BitTorrent, como citado a seguir. O sítio Ars Technica [19] veiculou uma informação que protocolos P2P são responsáveis por 95% do tráfego noturno gerado na Europa no período medido. Outro dado mostra o crescimento do protocolo BitTorrent enfatizando que houve um acréscimo de 24% na quantidade média de pessoas baixando os 200 arquivos considerados mais populares entre o final de 2007 e o início de 2008. Essa informação foi veiculada pelo site Ars Technica que obteve os dados através do BigChampagne [20], empresa especializada em medir popularidade de músicas na Internet. Esses dados estão ilustrados na Figura 1.1.

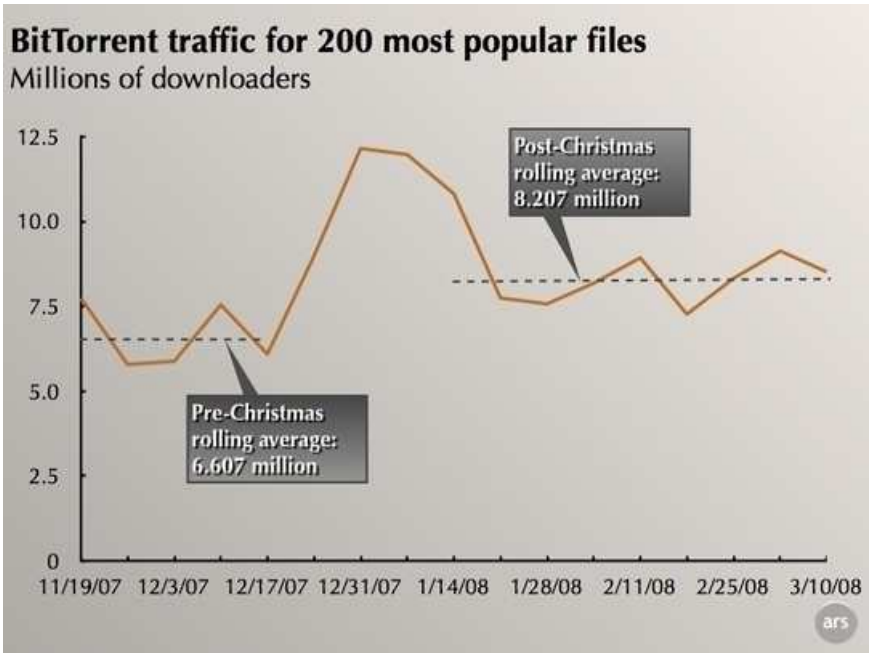


**Figura 1.1: Crescimento do Protocolo BitTorrent [31]**

Além do sítio anterior, o Torrent Freak [21] veiculou um resumo do relatório criado pela empresa ipoque [22] sobre a utilização de BitTorrent na Europa, Oriente Médio e Austrália. O estudo aponta que o tráfego de dados P2P nestas regiões varia de 49% a 83% da quantidade total de dados transmitidos. Isso ressalta a importância do impacto de redes P2P na Internet. A Figura 1.2 ilustra a participação do BitTorrent no total de tráfego das redes P2P para aquelas regiões. Pode-se perceber que o BitTorrent ocupa a maior parte do tráfego medido e é ultrapassado pelo eDonkey apenas na parte sul da Europa.

Atenção ao protocolo BitTorrent não é dada apenas no campo empresarial (indústria de mídia e de comunicação), mas também no campo político, onde governos começam a observar possíveis abusos na utilização dessa tecnologia. Como exemplo dessa corrente, tem-

se o Senador Democrata Joe Biden [23], que propôs investimento de U$ 1 bilhão para detectar distribuição de arquivos ilegais nos Estados Unidos, como pedofilia.

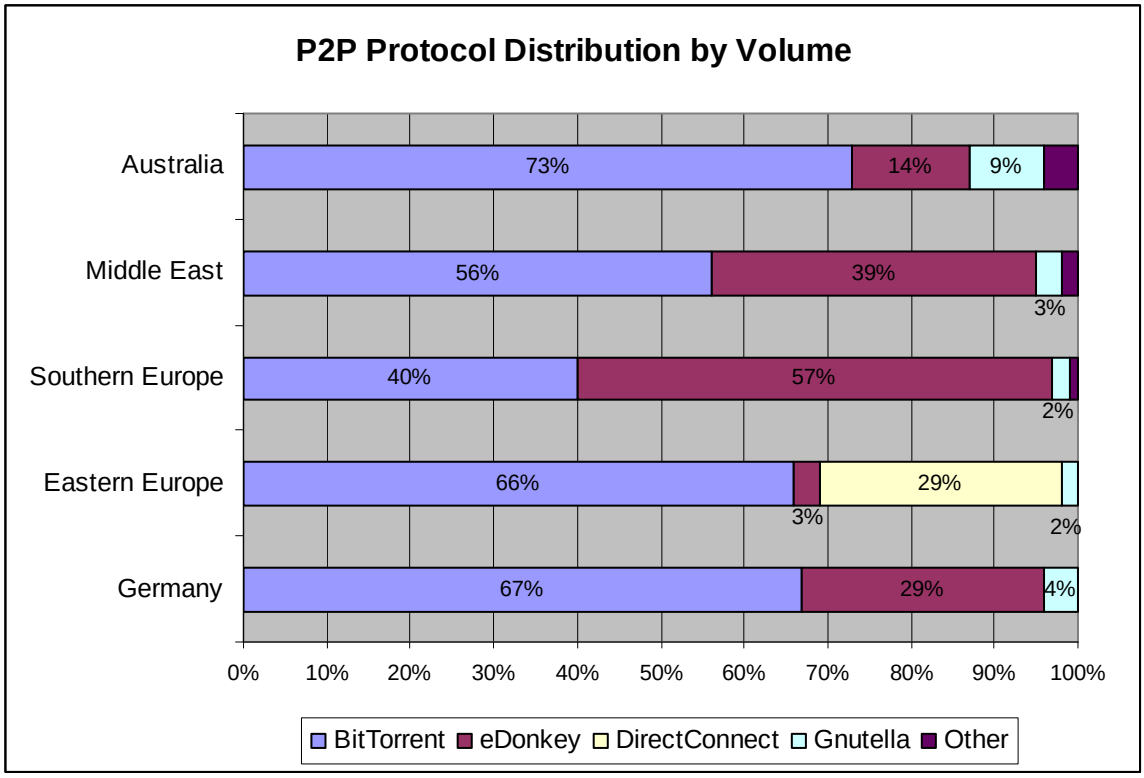


**Figura 1.2: Participação de cada protocolo**

Além das reportagens em sites especializados e das publicações feitas pelas empresas de equipamentos para provedores de Internet, um grande número de trabalhos foi publicado pela comunidade científica sobre a utilização do protocolo BitTorrent [1], [2], [3], [4], [5] e [6]. Isso mostra que o tópico em questão é considerado relevante e tem atraído a atenção de pesquisadores pelo mundo todo. Existe uma gama de pesquisas sobre redes BitTorrent que vão desde a investigação de seus componentes até o efeito que o protocolo apresenta para redes. Como exemplo, tem-se trabalhos que focam na investigação do entendimento sobre o funcionamento do protocolo [1] e [2], em atacar ou subverter o sistema [5] e [15], ou em obter o máximo de retorno possível [3] e [26]. Outra entidade que tem interesse no protocolo é a indústria do entretenimento, que perde dinheiro com a pirataria presente nas redes P2P. Para este grupo, é importante obter conhecimento sobre o atual alcance e difusão do conteúdo transmitido de forma ilegal [47].

Nas seções seguintes, são exploradas a definição do problema e os objetivos e contribuições deste trabalho.

## 1.2 Definição do Problema

Apesar da percepção comum sobre a popularidade de BitTorrent e outros protocolos P2P, não é raro haver discrepâncias entre os resultados obtidos através de medições. Por exemplo, no âmbito do Gnutella, os estudos [55], [1] e [56] apresentaram divergências significativas quanto ao nível de comportamento egoísta. No caso de BitTorrent, em recente conferência internacional na área de P2P (*International Conference on Peer-to-Peer Computing*, 2008), houve sensível divergência entre palestrantes convidados quanto ao volume de uso de BitTorrent em países europeus e na América do Norte, particularmente entre dados apresentados por Schulzrinne [57] e citados por Anja Feldmann [58]. Outro exemplo de divergência são os dados apresentados pela empresa Ipoque [22] e os citados por Pouwelse [59], onde fica claro o efeito da localidade (país) no percentual de tráfego gerado por cada protocolo. Esta discrepância está associada, muito provavelmente, à pouca abrangência dos estudos e/ou à metodologia empregada (restringindo regiões de pesquisa), justificando uma investigação mais profunda.

Um importante aspecto que deve ser observado para o levantamento de dados sobre o universo *torrent* é que as redes do tipo BitTorrent são diferentes das redes Gnutella, Napster e eDonkey. Os pares participantes dessas redes formam um *overlay* único, não particionado, onde um nó pode encontrar qualquer conteúdo ou par existente utilizando os mecanismos de busca presentes nas redes. Uma característica que faz com que redes BitTorrent apresentem desafios de monitoramento maiores que as redes citadas anteriormente é que existem múltiplas redes desconexas, onde não é possível fazer buscas por conteúdo utilizando o protocolo. Dessa forma, os usuários do BitTorrent formam uma rede para compartilhar um conteúdo (arquivo ou conjunto de arquivos) específico, formando um *overlay*, sem realizar interação com usuários de outros *overlays* que compartilham conteúdos distintos. Particularmente, a atividade de compartilhamento se encontra fragmentada em milhares de redes (*overlays*) independentes, dificultando a obtenção de informações globais. Potencialmente, apesar de existirem pares comuns entre redes BitTorrent, cada rede é independente e não possui conhecimento sobre as demais. Esse fato está ligado à falta de um mecanismo de busca de conteúdo, pois o mesmo está fora do escopo do protocolo. Até um determinado conteúdo, como o programa Microsoft Word, pode estar sendo compartilhado em múltiplas redes BitTorrent desconexas, onde pares de uma rede desconhecem os pares da outra.

Esses fatos fazem com que a obtenção de uma visão global sobre a quantidade e tipos de arquivos que são compartilhados em um determinado momento, a identificação de qual o atual nível de espalhamento de um conteúdo específico e qual é a velocidade com que redes BitTorrent conseguem replicar este conteúdo sejam bastante desafiadoras.

O objetivo deste trabalho é a criação de uma arquitetura escalável e flexível que possibilite o estudo do universo de redes BitTorrent em funcionamento em ambiente real, permitindo a exploração do comportamento dos usuários e do estado atual de redes BitTorrent. Além disso, tem-se como objetivo que a arquitetura monitore conjuntos de redes (*overlays*) com base em atributos das mesmas. Dessa forma, será possível selecionar e investigar redes consideradas "interessantes", por exemplo, as que compartilham um determinado conteúdo ou possuem um tamanho específico.

Ao que se sabe, não existe hoje um mecanismo que permita a um pesquisador obter dados atuais do universo *torrent* que contemple uma parcela ampla dos pares e comunidades envolvidas de forma escalável e flexível, filtrando redes, comunidades e *torrents* por seus atributos. Trabalhos atuais não levam em conta a flexibilidade na escolha do que se deseja monitorar e como os dados serão obtidos. As limitações e restrições dos trabalhos relacionados são exploradas posteriormente em maiores detalhes.

## 1.3 Objetivo e Contribuição

Informações sobre compartilhamento de conteúdo no universo de redes BitTorrent podem ser úteis por uma série de razões. Por exemplo, podem ajudar desenvolvedores de aplicações baseadas em BitTorrent a melhorar suas aplicações, ou a criarem novas, de acordo com o comportamento dos usuários, configurações empregadas nos agentes e capacidade da rede. Informações sobre popularidade de conteúdo e preferências de usuários podem ser bastante úteis para amparar campanhas de marketing. A indústria de entretenimento pode identificar o volume de conteúdo sendo compartilhado e estimar perdas financeiras com cópias ilegais [47]. As autoridades, como Polícia Federal e órgãos de repressão ao crime digital, podem usufruir de tais informações com objetivo de identificar usuários que violam direitos autorais, espalham vírus de computador com objetivos maliciosos ou compartilham pedofilia, de forma a coletar evidências contra os mesmos.

Este trabalho propõe uma arquitetura escalável para monitoração e extração de informações do "universo de redes BitTorrent". O foco em tal análise são as informações em nível de protocolo de aplicação, como por exemplo, o número de pares semeadores distribuindo um conteúdo (em uma ou mais redes), percentual de pares com suporte à

criptografia e o tipo de conteúdo (vídeo, filmes, fotos, software, imagens ISO, etc.) compartilhado. Trata-se de um problema bastante desafiador, pois envolve a extração de informações em um cenário de larga escala e descentralizado sobre o qual se possui pouco controle. A arquitetura proposta neste trabalho representa um avanço em relação a trabalhos anteriores [41] e [47], com flexibilidade ao se escolher o nível de profundidade, de abrangência (número de redes observadas), a acurácia (atualização de dados) e custo na utilização de recursos. Além disso, a arquitetura prevê exploração de comunidades.

Atualmente, existem muitos trabalhos que tratam sobre comportamento do protocolo variando-se parâmetros [17] e [26], comportamento de usuários (aliado ao *free-riding* [5] e utilização econômica [3]) e condições de rede. Entretanto, não foram encontrados trabalhos que analisam múltiplas redes BitTorrent reais de forma escalável, que leve em consideração o conteúdo compartilhado destas redes, que são desconexas, que permita seleção do que deve ser monitorado e estabeleça um levantamento de características como replicação e localização dos pares. Este trabalho tem os objetivos específicos listados a seguir:

- Desenvolver uma arquitetura escalável para monitoramento do universo *torrent*;
- Monitoramento de redes e comunidades de maneira flexível (configurando-se o nível de detalhe);
- Permitir que o monitoramento ocorra através de escolhas sobre atributos presentes nas redes;
- Estabelecer um conjunto de informações que podem ser extraídos do universo *torrent*;
- Agregar informações presentes em redes desconexas e múltiplas comunidades;
- Desenvolver um protótipo da arquitetura proposta que permita avaliar custo e impacto;
- Avaliar o custo e impacto da arquitetura.

A utilização da arquitetura em ambiente real permite que uma investigação dê suporte (ou conteste) os resultados de trabalhos que tiveram foco em simulação ou modelagem analítica, sendo possível verificar se as premissas e variáveis utilizadas condizem com o ambiente de rede real. Os resultados dessa investigação poderão ser utilizados para validar outros experimentos de rede real controlada que, por não utilizarem tráfego de fundo, não reproduzirem o comportamento real do usuário quanto a tempo de semeadura (Tempo de *Seed*) ou não reproduzirem um ambiente de rede próximo ao real, podem gerar distorções nos resultados. Além disso, a arquitetura permite observar a evolução das redes BitTorrent ao longo do tempo, coletando dados atuais e correlacionando métricas de diferentes redes e permitindo a identificação de características dos usuários.

A arquitetura de monitoramento tem inspiração no *crawler* [24] utilizado pelo Google [25], para obter e compilar toda informação disponível na Internet, e em trabalhos que utilizam agentes BitTorrent modificados para coleta de informações [3]. Dessa forma, propor uma forma de se agregar maior conhecimento sobre o universo *torrent*.

## 1.4 Organização do Trabalho

O restante deste documento está organizado em cinco capítulos. O Capítulo 2 revisa os conceitos sobre o protocolo BitTorrent, explorando as informações presentes no universo. O Capítulo 3 apresenta os trabalhos relacionados sobre monitoramento e levantamento de dados sobre redes BitTorrent. O Capítulo 4 apresenta a arquitetura escalável para extração de dados do universo. O Capítulo 5 descreve a avaliação das estratégias de monitoramento, o protótipo desenvolvido e os resultados dos testes de impacto e custo. Por fim, o Capítulo 7 fecha o trabalho com as considerações finais e trabalhos futuros.

# 2 PROTOCOLO BITTORRENT

O protocolo P2P BitTorrent é uma alternativa ao compartilhamento de dados utilizado no tradicional esquema cliente/servidor, pois o modelo centralizado é muito custoso para os seus mantenedores, tem baixa escalabilidade e é um ponto central de falhas. O BitTorrent acaba com essas limitações provendo um protocolo escalável e distribuído. O BitTorrent possui três fases no seu processo de distribuição de dados [27] como podemos ver na Figura 2.1 explicada nos marcadores a seguir.

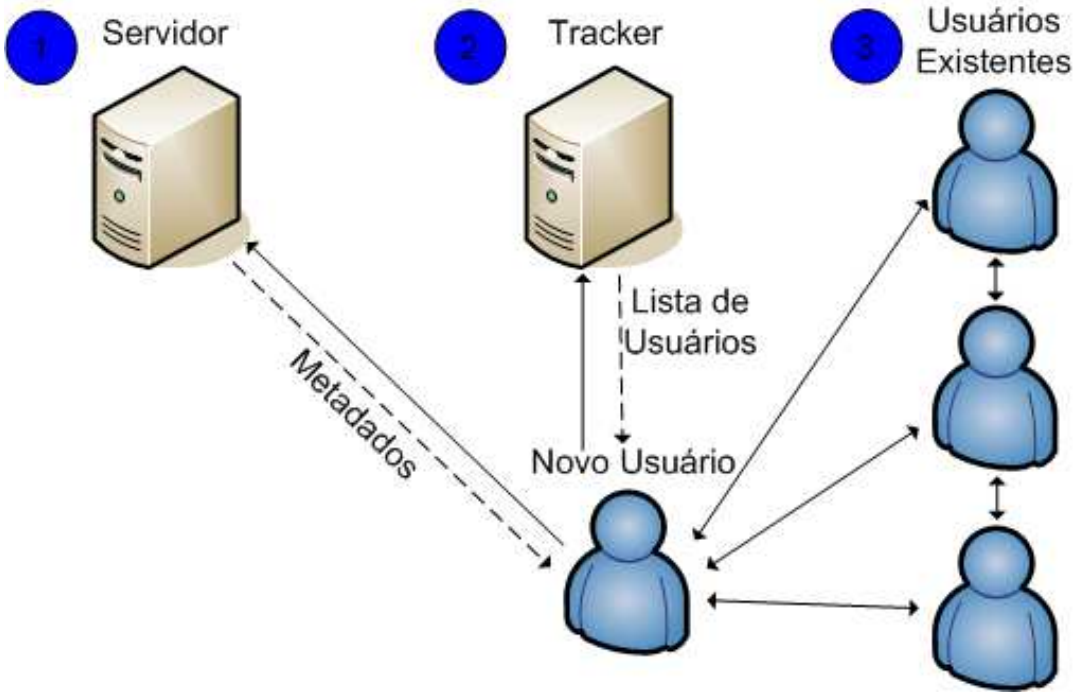


**Figura 2.1: Fases do Protocolo BitTorrent**

- **Fase 1 – Obtenção do Arquivo de Metadados**: o primeiro passo para entrar em um enxame BitTorrent é a obtenção do arquivo de metadados que descreve os nomes e tamanhos dos arquivos a serem adquiridos, o tamanho das peças, o *hash* das peças, os IPs dos rastreadores e outras informações que podem ser utilizadas para extensões ou como informação extra. A obtenção deste arquivo não faz parte do protocolo BitTorrent e pode ser feita por qualquer meio, desde um servidor WEB (método tradicional) até recebimento via e-mail. Para isso, existem comunidades abertas e fechadas, onde as fechadas precisam de permissão, como senhas, para baixar os arquivos de metadados. Popularmente, este arquivo é conhecido como *torrent*, pois esta é a sua extensão no sistema operacional;

- **Fase 2 – Conexão com o Rastreador e obteção da lista de usuários**: uma vez que o arquivo de metadados foi obtido, um agente de usuário como μTorrent [28] pode abrir o arquivo de metadados e obter as informações necessárias para se conectar ao rastreador, que pode utilizar mecanismos de autenticação. Uma vez conectado a um dos rastreadores disponíveis, o agente requisita uma lista com pares que estão participando do enxame. De posse desta lista, o usuário irá passar para a Fase 3;
- **Fase 3 – Conexão aos pares do enxame**: o agente de usuário abre conexões com os pares que participam no enxame e inicia o protocolo de obtenção de dados. Este protocolo é composto por todos os mecanismos utilizados pelo BitTorrent para distribuir o conteúdo entre pares. Cada um destes mecanismos será explicado a seguir.

Como se pode observar nos trabalhos que abordam o funcionamento do BitTorrent [6] e [27], este protocolo possui um conjunto de componentes e entidades que colaboram para atingir o objetivo de distribuição de conteúdo. Os seguintes itens abordam estes componentes:

- **Comunidades**: sítios que armazenam os *torrents,* caracterizando-os. Além disso, algumas comunidades possuem um conjunto de usuários (administradores, moderadores, contribuidores) que participam no processo de publicação e filtragem de conteúdo. As comunidades podem ser abertas ou fechadas (necessitando cadastro), podem armazenar os *torrents* ou apenas indexá-los (apontando para outros sítios), podem conter rastreadores e podem publicar informações sobre os enxames;
- **Rastreador**: elemento central responsável pela organização do enxame. É importante citar que este elemento é um ponto central de falhas (como apontado por [29]) e que os pares presentes na rede confiam incondicionalmente nele. Sua tarefa consiste em realizar a inicialização (*bootstrap*) dos pares novos, fazendo os mesmos se encontrarem. Ele fornece, mediante requisição, uma lista contendo participantes do enxame aleatoriamente escolhidos, informando sobre os endereços IP, identificadores e portas para conexão. Por exemplo, se existe um enxame com um conjunto de mil participantes e um novo par entra neste enxame, o rastreador envia uma lista contendo um subconjunto de cerca de cinquenta identidades aleatoriamente escolhidas. Por fim, o rastreador pode ser utilizado para coleta de informações estatísticas como entrada e saída de pares, taxas de *download* e *upload*, progresso individual de obtenção do conteúdo e tamanho do enxame;

- ***Torrent***: nome dado ao arquivo de metadados que descreve o conteúdo. O mesmo é utilizado pelo agente para obter quais são os arquivos a serem baixados, como estes estão divididos, quais são os *hashes* das peças e quais são os rastreadores disponíveis;
- **Conteúdo, arquivos, peças, *hash* e blocos**: conteúdo é a representação de tudo que será carregado pelo agente. O mesmo é composto por um conjunto arbitrário de arquivos podem ser de qualquer tipo (vídeo, música, programas, etc). O conteúdo está dividido em diversas peças. Cada peça tem um código *hash* associado. O *hash* é utilizado para verificar a integridade da peça. Os pares de um enxame enviam mensagens aos vizinhos informando as peças que possuem e também as que são obtidas no processo de compartilhamento. Esse mecanismo será explicado posteriormente. Cada peça está dividida em blocos. Os blocos não possuem *hash* associado e são utilizados para aumentar a taxa de *download* através de uma técnica de *pipeline*, onde várias requisições a blocos são enviadas em um determinado instante;
- **Semeadores e Sugadores**: O semeador (*seeder*) é um par do enxame que tem o conteúdo completo, ou seja, sua única função é prover dados aos participantes. Destaca-se que para um enxame iniciar, é necessário a presença de pelo menos um semeador que fará o papel de distribuir os dados para os participantes que não tem o conteúdo. Neste cenário, um par é considerado sugador (*leecher*) se não possui o arquivo completo. Neste caso, o papel do par é tanto obter quanto compartilhar o conteúdo;
- **Enxame**: Nome dado ao conjunto de pares conectados para troca de um conteúdo específico, ou seja, o conjunto formado por todos os semeadores e sugadores que utilizam as políticas para controle de conexões, compartilhamento de conteúdo e sufocamento. Para cada conteúdo, existe um enxame. Dessa forma, existem inúmeras redes BitTorrent desconexas que compartilham conteúdos diferentes.

A Figura 2.2 ilustra a relação entre os componentes, apresentando três diferentes cenários para demonstrar tipos de formação possíveis com conteúdos ($c_1$, $c_2$, ...), enxames ($s_1$, $s_2$,...) e pares ($p_1$, $p_2$, ...). Primeiramente, $c_1$ ilustra a situação onde dois ou mais enxames ($s_1$ e $s_2$) distintos compartilham o mesmo conteúdo. Esta situação pode surgir em dois casos, melhor explicados através de exemplos: (a) codificações distintas, MP3 e OGG, de um mesmo álbum de música; ou (b) exatamente o mesmo conjunto de arquivos sendo disponibilizado em duas comunidades distintas, com rastreadores diferentes. Segundo, o enxame $s_3$ compartilhando um conteúdo $c_2$ mostra a situação onde um dado conteúdo é

compartilhado por apenas um enxame. Por último, $s_4$ e $s_5$ representam enxames que tem pares em comum. Dessa forma, a intersecção de $s_4$ e $s_5$ representa pares que estão participando de dois enxames distintos ao mesmo tempo. Apesar desses pares em comum, $s_4$ e $s_5$ são independentes entre si.

Cada uma das três classes de componentes definidas acima possui um conjunto de atributos. A Figura 2.2 apresenta uma lista (não exaustiva) de atributos que podem ser associados a cada classe. Tais componentes servem de fundamento para a definição do modelo de representação de dados, apresentado no Capítulo 4, que descreve como as informações são obtidas.

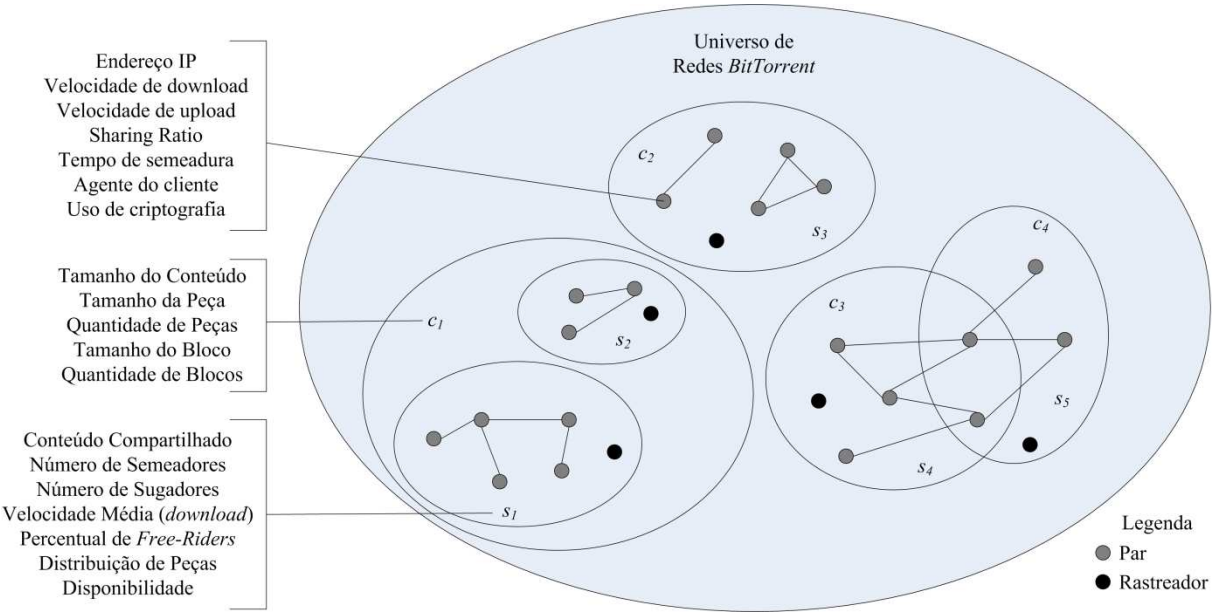


**Figura 2.2: Universo *Torrent***

## 2.1 Política de Conexão com Outros Pares

Inicialmente, para um usuário entrar em um enxame, é necessário conectar-se ao rastreador. Após este processo, o agente requisita uma lista de pares que estão conectados no enxame atual. Essa lista é armazenada localmente, sendo utilizada para conectar-se a outros pares do enxame seguindo políticas para conexão ativa ou passiva como explicado a seguir.

Cada par tem um limite máximo de conexões simultâneas que podem estar estabelecidas em um determinado momento. Esse limite é configurado através de um parâmetro que pode ser manualmente modificado, influenciando na velocidade [30] e conectividade da rede [17]. Segundo Al-Hamra [17], esse parâmetro tem, usualmente, o valor 80. Neste caso, isso indica que um par somente poderá estar conectado a oitenta pares em um determinado tempo *t*. Entretanto, pode-se ter conhecimento um conjunto de identidades muito maior que esse número. O subconjunto da lista completa de pares conhecidos ao qual um

determinado par está conectado é chamado de *Peer Set*. Um par somente irá trocar informações com pares que pertençam a este conjunto.

O limite máximo de conexões é dividido em dois subgrupos [17]. O primeiro refere-se às conexões de saída, ou seja, as conexões que são iniciadas pelo agente local (conexão ativa). O segundo refere-se a conexões de entrada, ou seja, conexões que são iniciadas por agentes remotos (conexão passiva). A utilização de NAT sem roteamento para conexões entrantes impede a utilização de conexão passiva, sendo necessário conectar-se apenas com pares que possuem redirecionamento. Ter o número de conexões de saída menor que o número de conexões máximas é importante para evitar particionamentos de rede e grafos com grande diâmetro [17].

Após receber a lista de pares do rastreador, o par local irá conectar-se aos pares remotos até que o número máximo de conexões de saída tenha sido alcançado. A partir deste momento, espera-se por conexões de entrada. Caso o número de conexões abertas fique abaixo de determinado limite (usualmente 20 [17]), uma nova lista de identidades será requisitada ao rastreador para poder aumentar o tamanho do *Peer Set* a fim de melhorar suas chances de receber o conteúdo desejado.

O rastreador limita a quantidade de pedidos que um par pode fazer a cada intervalo de tempo através de um parâmetro [44]. Toda vez que um par faz o anúncio ao rastreador, a resposta conterá a informação de quanto tempo deve-se esperar antes da próxima requisição.

## 2.2 Política de Seleção de Peças

O BitTorrent usa um algoritmo de seleção de peças conhecido como “Peça Local Mais Rara Primeiro” (*Local Rarest First – LRF*) [6]. Dessa forma, quando um par deseja obter uma peça do conteúdo, ele irá escolher a peça que está menos replicada entre seus vizinhos[6]. O principal objetivo deste algoritmo é evitar que peças desapareçam por baixa replicação dentro do enxame. É importante salientar que a escolha da peça mais rara é feita baseada na visão local do par, pois o mesmo não faz nenhuma consulta para verificar qual a visão dos seus vizinhos sobre o espalhamento das peças.

Além de mitigar o problema de peças desaparecem ou tornarem-se muito raras a ponto de atrasar o carregamento do conteúdo, o algoritmo evita o “problema da última peça”, onde muitos pares precisariam de uma peça considerada a última para finalizar o *download* e a mesma não está disponível nos vizinhos. Somado a isso, a escolha de obter sempre a peça localmente mais rara, faz com que mais pares remotos fiquem interessados pelo conjunto de peças possuído localmente. Esse interesse propiciará a oportunidade de distribuir conteúdo

para os vizinhos de forma que estes forneçam mais peças através do algoritmo *tit-for-tat* [27]. A seleção de peça mais rara é mais efetiva que políticas de seleção de peças aleatórias [26].

Por livre arbítrio, os semeadores iniciais do enxame podem sair do mesmo assim que terminarem de distribuir todo conteúdo pelo menos uma vez, considerando que a partir deste momento o sistema é capaz de se manter sem sua presença. O algoritmo da peça mais rara ajuda a diminuir o impacto da saída do semeador, pois fará com que as peças estejam distribuídas de maneira uniforme (teoricamente) e mitiga o problema de uma peça desaparecer do enxame.

Existem três exceções para a regra do algoritmo de peça mais rara [37]. A primeira delas ocorre quando o par está iniciando o *download* do conteúdo. Neste caso, o estado "iniciando" acontece quando o número de peças obtidas é menor que quatro. Neste momento, opta-se por selecionar qualquer peça aleatoriamente. A segunda exceção ocorre quando o par está finalizando o *download* do conteúdo. Neste caso, entende-se por "finalizando" quando todos os blocos faltantes já foram requisitados. Neste momento, as requisições são enviadas para todo o *Peer Set*. Por último, a terceira exceção acontece quando o par recebeu blocos de uma peça. Neste momento, o par dará preferência por requisitar os blocos faltantes desta peça.

## 2.3 Política de Sufocamento

Sufocamento consiste em um par A parar de contribuir com um par B do enxame. Neste caso, diz-se que o B foi sufocado por A. O sufocamento é utilizado por dois motivos [6]: o primeiro deles consiste em aperfeiçoar a utilização do enlace contribuindo apenas com quem contribui; o segundo consiste em punir os pares que não contribuem.

O primeiro motivo pode ser exemplificado da seguinte maneiro. Se um par mantiver não sufocados todos os pares do conjunto de pares ativos, haverá uma concorrência muito alta pelas peças que o par possui e pela utilização do enlace. Essa alta concorrência no enlace causa congestionamento de rede e, consequentemente, baixo desempenho para o protocolo TCP, pois boa parte do enlace será utilizado para mensagens de controle e haverá alto número de retransmissões por pacotes perdidos. Dessa forma, deixa-se não sufocados apenas um conjunto pequeno de pares para que a utilização do enlace seja maximizada. Usualmente, esse conjunto de pares não sufocados tem tamanho quatro [6]. Além do mais, devido a política de contribuir com os pares que mais contribuem, se o enlace de saída for divido entre muito, existe grande chance do par não estar na lista dos maiores contribuidores de seus vizinhos (inibindo-o de receber peças).

O segundo motivo é manter apenas um subconjunto dos pares ativos não sufocados para punir os pares que não contribuem e incentivar os que contribuem. Desta forma, os pares não sufocados serão aqueles que contribuem mais e os sufocados os que contribuem pouco ou não contribuem.

O algoritmo de sufocamento é executado a cada dez segundos [6]. Neste período, é contabilizado quanto cada par remoto contribuiu com o par local. O par local irá verificar quais são os pares que mais contribuíram e colocar os três que mais contribuíram como não sufocados, não permitindo que os demais baixem conteúdo.

Uma técnica chamada de dessufocamento otimista (*Optimistic Unchoke*) [6] é utilizada juntamente com o sufocamento descrito anteriormente para atingir dois objetivos. O primeiro, e mais importante, é descobrir pares que possam contribuir de forma mais significativa dos que estão contribuindo atualmente. O segundo, que vem em decorrência do primeiro, habilita pares que não tem peças interessantes a seus vizinhos (como na situação em que estão iniciando no enxame) a receberem novas peças de forma a poderem contribuir e entrar na lista de pares não sufocados.

## 2.4 Troca de Informações

O protocolo BitTorrent faz com que pares conectados troquem informações sobre o seu estado atual. A principal informação trocada é a posse das peças do conteúdo compartilhado. Primeiramente, essa troca de informação é realizada no estabelecimento da conexão entre pares. Essa informação está contida em uma estrutura de dados chamada *bitfield*. Esta estrutura é composta por um vetor com tamanho igual à quantidade de peças em que o conteúdo foi dividido, onde cada posição deste vetor indica a posse de uma das peças.

Toda vez que uma conexão é estabelecida, o *bitfield* é enviado por ambos os pares. Essa informação é utilizada para construir um mapa que indica a freqüência de cada peça e quais são os pares remotos que as possuem de forma a prover dados para tomada de decisão no algoritmo LRF apresentado na Seção 2.2, pois esse mapa indica qual é a peça mais rara que o par local não tem e onde essa pode ser buscada.

Como a informação sobre a posse de peças não é uma informação estática, mas dinâmica, existe a necessidade de que esse mapa seja atualizado periodicamente. Essa atualização periódica é enviada através de uma mensagem chamada de *HAVE*. Toda vez que um par obtém uma peça do arquivo que é compartilhado, esse par irá enviar para todos os vizinhos uma mensagem *HAVE* informando que obteve aquela peça. Quando um par recebe uma mensagem deste tipo, ele atualiza o mapa de peças local.

## 2.5 Conclusões

Neste capítulo foram apresentadas as principais políticas utilizadas pelo protocolo BitTorrent para o seu funcionamento, assim como os componentes e seus atributos. Os conceitos aqui apresentados servem de base para a proposta da arquitetura *TorrentU*. O capítulo mostrou como ocorre o processo para compartilhamento de conteúdo utilizando protocolo BitTorrent desde a obtenção do *torrent* e conexão no rastreador até a obtenção do conteúdo desejado. O próximo capítulo apresenta trabalhos que tem como objetivo investigar o comportamento de redes BitTorrent sobre determinadas situações, em função do comportamento de seus usuários e das implementações do protocolo. Os trabalhos foram divididos em grupos baseados nas estratégias utilizadas e objetivos almejados.

# 3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos que utilizam técnicas voltadas ao monitoramento de enxames reais, ou seja, que analisam o BitTorrent operando em redes de longa distância. Podem-se dividir os trabalhos de monitoramento em dois grandes grupos. O primeiro grupo agrega trabalhos que focam em estudar o comportamento do BitTorrent isoladamente, muitas vezes observando um enxame específico. O segundo grupo agrega trabalhos que estudam múltiplos enxames ou comunidades, abrangendo um conjunto amplo do universo.

Cada um dos trabalhos classificados nos grupos utiliza uma técnica para obtenção de dados sobre enxames, pares ou comunidades. A identificação das vantagens e desvantagens de cada estratégia é crucial para determinar quais delas devem ser aplicadas para atingir os objetivos do trabalho, conforme tratado a seguir. As estratégias de monitoramento variam quanto aos seguintes aspectos: o conjunto de informações que podem ser obtidas; a escalabilidade; a intrusão ao enxame; a viabilidade de implantação; a profundidade (nível de detalhes); e a acurácia (atualização) na obtenção dos dados. Nas avaliações, quantifica-se acurácia, intrusão e profundidade utilizando os qualificadores como *alto*, *médio* e *baixo*. Esses qualificadores são utilizados para comparar as estratégias entre elas e não quantificam numericamente os impactos.

Este capítulo está dividido em três seções. A Seção 3.1 apresenta os trabalhos do primeiro grupo, focados na análise de enxames individuais, ao passo que a Seção 3.2 apresenta os trabalhos com foco no estudo de parte do universo *torrent*, incluindo múltiplos enxames ou comunidades. Por fim, a Seção 3.3 fecha o capítulo apresentando um comparativo entre os trabalhos.

## 3.1 Trabalhos com Foco em Monitoramento de Enxames Individuais

Esta seção apresenta o conjunto de trabalhos que tem foco em estudar o comportamento do protocolo BitTorrent através do monitoramento de enxames isolados (bem específicos). Estes trabalhos distinguem-se pela estratégia utilizada para coleta de dados. A primeira estratégia utiliza os *logs* de um rastreador para obter informações, enquanto a segunda utiliza agentes de BitTorrent modificados para monitorar o estado dos pares de um enxame. A Subseção 3.1.1 apresenta o trabalho que utiliza a primeira estratégia enquanto a Subseção 3.1.2 apresenta os trabalhos relacionados a segunda estratégia.

### 3.1.1 Utilização dos *Logs* do Rastreador

O artigo publicado por Izal [2] é um trabalho seminal sobre a avaliação experimental de BitTorrent, sendo não apenas o primeiro trabalho a explorar essa possibilidade, mas também o mais citado dentre os trabalhos aqui discutidos. No referido trabalho, os autores obtiveram *logs* gerados por um rastreador associado a um enxame de grande popularidade, cujo conteúdo compartilhado era uma distribuição Linux. Em tal enxame foram identificadas 180 mil sessões de download nos *logs*, com um pico de 4,5 mil pares no primeiro dia.

A análise do traço do *log* mostrou comportamento de *flash-crowd* no primeiro dia de disponibilização do conteúdo. Após cerca de quinze dias depois desse período, o número de pares no enxame fica estável com menos de 200 pares. As métricas obtidas por aquele trabalho são as seguintes:

- Número de pares no enxame;
- Número de sugadores;
- Número de semeadores;
- Taxa de *download* global;
- Média da taxa de *upload* e *download*;
- Classificação por país/região;
- Quantidade feita de *upload* por semeadores e por sugadores.

De acordo com os autores, o BitTorrent se comporta bem em casos de *flash-crowd* e consegue rapidamente criar novos semeadores aumentando a escalabilidade do sistema. Para o cenário analisado, percebeu-se que a quantidade de dados enviada pelos semeadores foi mais de duas vezes a quantidade enviada pelos sugadores. No mesmo período foi identificado que a quantidade de semeadores variou em torno dos 20%. É importante salientar que essa análise foi feita em um cenário específico que envolvia conteúdo sem restrições de direitos autorais.

Sobre a capacidade dos enlaces, observou-se que a média de *download* ficou acima de 500kb/s. Com essa observação, percebeu-se que boa parte dos enlaces teria uma capacidade mínima comparável ao ADSL. Com o resultado apresentado pela capacidade de *download* total do sistema (800Mb/s no período inicial), concluíram que o BitTorrent é capaz de utilizar a banda disponível nos pares intermediários para aumentar o desempenho do sistema.

Observando as sessões do BitTorrent, aquele trabalho identificou que 11,8% foram sessões simples, ou seja, o par conectou e permaneceu até o fim do *download*. 8,2% foram consideradas multi-sessões, ou seja, sessões em que o usuário saiu do enxame e retornou

posteriormente para conclusão do *download*. O método utilizado pelos autores para obter esses dados revelou que cerca de 81% dos pares abandonaram o enxame e nunca mais voltaram, indicando um alto número de *downloads* incompletos. É importante destacar que os autores citam que o método utilizado pode levar a desvios nos dados. Neste caso, perceberam que estes 81% dos pares representaram apenas 25% dos dados baixados.

Somado a análise de *logs*, um agente modificado foi utilizado para coletar dados participando ativamente do enxame. Esse agente foi utilizado para medir o desempenho para um caso específico, onde utilizaram uma rede de 10Mb/s sem limitação de utilização de enlace. Desta análise concluíram que a taxa de *download* está positivamente relacionada com a taxa de *upload*. Também concluíram que existe um período de "aquecimento", onde é necessário obter algumas peças iniciais para o par poder contribuir e obter uma melhor taxa de *download*. Neste caso, o período foi de 100 segundos.

Como conclusão, demonstraram que o BitTorrent é bastante efetivo e deixaram algumas questões em aberto:

- Qual a política ótima de replicação?
- Como construir um sistema de replicação robusto que garanta que máquinas completem o *download* em algum período no futuro?
- Como proteger estes sistemas de ataques de Negação de Serviço (*DoS*) e pares maliciosos?

### 3.1.2 Agentes Instrumentados para Monitorar o Enxame que Participam

Cita-se dois trabalhos, de mesma autoria, que utilizam agentes instrumentados como estratégia para obtenção de dados de enxames. O trabalho de Legout [37] investiga o algoritmo LRF e de sufocamento para verificar se os mesmos são suficientes para obter justiça e inibir *freeriders*. O objetivo dos autores é afirmar essa sentença e, para este fim, utilizam um agente BitTorrent instrumentado para monitorar enxames reais. É utilizada a versão 4.0.2 da *mainline*, o agente oficial. O ambiente em que os experimentos são executados é o PlanetLab [35].

Duas métricas foram criadas pelos autores. A primeira delas, chamada de entropia, é um indicador sobre o espalhamento das peças na vizinhança de um par. Para calcular essa métrica utiliza-se o tempo que o par local ficou interessando (*interested*) em pares remotos sobre o tempo total. Da mesma forma, é calculado o tempo que os pares remotos ficaram interessados no par local. A segunda delas diz respeito a eficiência do algoritmo de sufocamento, onde os pares remotos foram agrupados de duas formas. Na primeira, agrupa-se

em grupos de cinco pares os que mais contribuíram com o par local (mais fizeram *upload*). Na segunda, faz-se o mesmo agrupamento para os pares que mais baixaram do par local (mais fizeram *download*). A teoria dos autores é que os grupos que mais fizeram *download* e *upload* devem estar fortemente correlacionados.

Os autores classificam um enxame em dois estados. O primeiro, chamado de transiente, acontece quando o semeador ainda não terminou de fornecer uma cópia do conteúdo para o enxame. O segundo momento, chamado de estado estacionário, acontece quando uma cópia já foi fornecida para o enxame e este consegue replicar todas as peças mantendo a entropia com valor próximo ou igual a um. Quando a entropia é baixa, os autores consideram o enxame em estado transiente, pois os pares estão esperando que o semeador distribua peças que eles ainda não possuam.

O segundo trabalho [38], que tem o mesmo autor principal, reforça a avaliação do mecanismo do algoritmo de sufocamento. No segundo trabalho, ao invés de utilizarem os nós do PlanetLab para participar de um enxame real, os experimentos foram conduzidos de forma controlada, sem interferências de usuários externos. As principais conclusões sobre o algoritmo de sufocamento foram as seguintes: (i) facilita a formação de *clusters* entre pares com a mesma capacidade de *upload*; (ii) garante incentivos de compartilhamento efetivos; (iii) mantém alta utilização do enlace de saída.

Aquele trabalho propôs a utilização de quatro métricas baseadas na configuração utilizada para execução dos experimentos. Nesta configuração, optou-se por dividir os pares em três subconjuntos com capacidade de *upload* distintas. Com essa divisão, eles demonstram que os subconjuntos formam *clusters*. Baseado nisso, a primeira métrica foi o Índice de Clusterização, onde cada par computou o tempo que dessufocou determinado subconjunto. Dessa forma, pôde-se perceber que pares pertencentes a um subconjunto dessufocavam por períodos maiores os pares do próprio subconjunto. Ou seja, os pares de alta capacidade de banda dessufocavam por mais tempo eles próprios. Os pares com capacidade média dessufocavam por mais tempo seu grupo e, por fim, o mesmo comportamento ocorria com os pares do subconjunto de pares lentos.

A segunda métrica proposta foi a utilização média global da capacidade de *upload*. Com essa métrica, eles mostraram que, nos cenários estudados, o BitTorrent é capaz de ocupar completamente o enlace de saída. A terceira métrica é a quantidade de dados *uploaded* agregados. Essa métrica indica a quantidade de *upload* que um par fez para todos os outros e, com ela, pode-se ver quais os pares que mais contribuíram com o enxame (neste caso, o subconjunto dos mais rápidos). A quarta e última métrica refere-se ao tempo em que cada par

esteve dessufocando os outros pares. Essa métrica também foi utilizada para comprovar a clusterização entre subconjuntos.

## 3.2 Trabalhos com Foco em Monitoramento do Universo BitTorrent

Os trabalhos apresentados nesta seção têm um foco mais abrangente que os citados na seção anterior, pois monitoram não apenas um enxame separadamente, mas um conjunto deles. Além disso, dados referentes às comunidades BitTorrent são extraídos, de forma a compreender comportamentos de usuários específicos.

Esta seção está organizada em três subseções, que estão divididas pelo tipo de técnica utilizada para obter as informações do universo. A Subseção 3.2.1 apresenta trabalhos que utilizam monitoramento do enlace de rede para obtenção de dados. A Subseção 3.2.2 cita trabalhos que obtém informações através das páginas das comunidades (providas pelos rastreadores). Por fim, a Subseção 3.2.3 tem foco nos trabalhos que obtém informações a partir dos enxames BitTorrent, conectando nos pares participantes.

### 3.2.1 Monitoramento de Enlace de Rede

Cita-se dois trabalhos que utilizam a técnica de monitoramento do enlace de rede. O primeiro, apresentado por Saroiu [32], utiliza essa técnica para monitorar o sistema de transmissão de dados de uma universidade a fim de identificar que aplicações eram as consumidoras da banda. Como seu foco não era em BitTorrent, os resultados apresentados não são relevantes para esse trabalho. Entretanto, destaca-se a técnica utilizada para coleta de dados. Neste caso, a técnica é a mesma utilizada pelo próximo trabalho. Esta técnica inspeciona os dados dos pacotes na tentativa de verificar o que está sendo transmitido.

O segundo trabalho, apresentado por Guo [33], com foco em BitTorrent, tem como técnica o monitoramento passivo do enlace através de um sistema chamado Gigascope [34]. O Gigascope é compreendido por um banco de dados que armazena todo o tráfego que passa por um provedor e fornece uma linguagem para consulta deste tráfego. Baseado nos dados coletados, as seguintes afirmações são feitas:

- É difícil encontrar e carregar um conteúdo depois de certo período de tempo devido à taxa de chegada de novos pares no enxame, pois esta é exponencialmente decrescente. Somado a isso, a falta de semeadores faz com que a disponibilidade do sistema seja prejudicada;
- O desempenho de um agente BitTorrent é instável e flutua de acordo com a variação de pares no enxame;

- Os sistemas existentes são injustos. O nível de contribuição de um par diminui a medida que sua capacidade de *download* aumenta.

O comportamento de sistemas BitTorrent também é modelado analiticamente naquele trabalho e uma nova arquitetura de colaboração é proposta, entretanto como essa abordagem não está dentro do escopo desta dissertação, esses pontos não serão explorados.

Os traços analisados mostraram que mais de 85% dos pares participam em mais de um enxame. Essa observação tem bastante importância no momento da análise dos dados, pois múltiplos agentes ativos (ou um mesmo agente participando de vários enxames) fazem com que haja compartilhamento do enlace de rede e compartilhamento de limites de conexões abertas impostas pelo provedor de rede. Isso afetará taxas de *download* e *upload* e, consequentemente, o comportamento dos usuários nos enxames.

### 3.2.2 Monitoramento dos Rastreadores de Comunidades

Cita-se dois trabalhos que utilizam a técnica de monitoramento de rastreadores. O primeiro deles, apresentado por Belissimo [40], monitora rastreadores sem identificá-los. Assim, não é possível verificar a quem os rastreadores pertencem. Em seus resultados, apresentaram as informações coletadas a partir das páginas disponibilizadas pelos rastreadores. A partir desta coleta, foi criado um conjunto de três métricas sobre os enxames. A primeira delas estudou o tamanho dos arquivos compartilhados, onde identificou-se que a maior parte do conteúdo disponibilizado tinha tamanho similar a de um CD. A segunda delas mostrou a popularidade dos *torrents*, onde identificou-se que a popularidade não segue uma distribuição Zipf. Por fim, a última analisou a escalabilidade das redes BitTorrent, mostrando que o crescimento do tamanho dos enxames também acarreta em crescimento na capacidade de *download*.

As informações fornecidas pelo rastreador mostraram três características referentes aos pares que fazem parte do enxame. A primeira diz respeito às sessões, onde se identificou que um *download* raramente (10%) é terminado na primeira sessão. E que sessões podem durar desde algumas horas até vários dias (35 dias). Na segunda, detectou-se a assimetria dos enlaces de conexão, onde os pares tinham mais capacidade de *download* do que de *upload*. Por fim, identificou-se o período inicial de *flash-crowd* no enxame.

O segundo trabalho, apresentado por Andrade [39], propõe a investigação de comunidades de BitTorrent com a utilização de *crawlers* que fazem a varredura das páginas hospedeiras das comunidades. As páginas contêm informação referente aos enxames que os rastreadores estão coordenando. A partir destas informações, os autores levantaram as

seguintes métricas: (i) percentual de pares que são *freeriders*; (ii) percentual de semeadores no enxame; e (iii) taxa de compartilhamento (*sharing ratio*).

Um comportamento dos usuários mostrado com esse trabalho é o efeito que a "contribuição forçada" tem nos enxames. Das comunidades estudadas, três delas tinham grande percentual de semeadores por enxame. Isso se deve a obrigatoriedade de contribuir imposta pelos donos das comunidades. Da mesma forma, verificou-se que onde não havia restrições para quem não contribuísse, o percentual de semeadores era baixo.

Para verificar o quanto o percentual de semeadores afeta o desempenho dos pares colaboradores e dos *freeriders*, foram executados 21 experimentos utilizando dois agentes. Um dos agentes foi modificado para comportar-se como um *freerider*, sem compartilhar conteúdo. O segundo agente utiliza o protocolo padrão, compartilhando conteúdo e colaborando com o enxame. Dessa forma, eles verificam que se um enxame apresenta uma grande quantidade de semeadores (acima de 60%), os *freeriders* terminam o *downlad* antes que um par colaborador.

É importante destacar que para obter os dados necessários para conduzir este estudo, os rastreadores necessitam reportar esta informação em alguma página pública. Desta forma os estudos ficam limitados a comunidades que disponibilizam este tipo de recurso.

### 3.2.3 Monitoramento de Enxames e Utilização de *Crawlers*

Os trabalhos apresentados nesta subseção têm como foco monitorar o universo *torrent* através dos enxames. Para isso, utiliza-se *crawlers* que varrem sítios escolhidos para obtenção dos arquivos de metadados. De posse dos mesmos, os enxames são contatados para extração das informações. Citam-se três trabalhos que utilizam essa abordagem. O primeiro deles, publicado por Pouwelse e colegas [41] foi embasado em cinco características encontradas em sistemas P2P: (i) a popularidade do sistema, ou seja, o número de usuários; (ii) a disponibilidade dos servidores e rastreadores; (iii) o desempenho do *download*; (iv) o tempo de vida de um conteúdo; e (v) nível de poluição.

Para o desenvolvimento do estudo, foram utilizados dois componentes de *software* com três *scripts* cada. O primeiro componente é responsável pelo monitoramento dos elementos centrais do sistema, ou seja, o rastreador e os servidores que hospedam os arquivos *torrent*. Para tanto, o primeiro *script* tem a responsabilidade de verificar a disponibilidade e os tempos de respostas destes elementos ao longo do tempo. O segundo *script* é responsável por varrer páginas HTML e baixar todos os arquivos *torrent* disponíveis, armazenando-os

localmente. O terceiro *script* recebe os arquivos de metadados obtidos e contata os rastreadores responsáveis pelos enxames para verificar o seu estado.

O segundo componente desenvolvido é responsável pelo monitoramento dos pares dos enxames. O primeiro *script*, chamado de *hunt*, verifica o lançamento de novos arquivos de metadados, executando os *scripts* comentados a seguir. O segundo, chamado de *GetPeer¸* é chamado pelo *hunt* com o endereço do rastreador. Ele é responsável por obter listas de pares participantes do enxame para passar para o próximo *script*. Por fim, o terceiro *script*, chamado de *PeerPing*, é responsável por contatar os pares para verificar o progresso do *download* (utilizando o *bitfield* para acompanhar o progresso) e o *uptime*.

O trabalho teve um foco nos resultados obtidos e não na metodologia empregada. A escalabilidade do sistema de monitoramento não foi avaliada, mas percebeu-se a grande necessidade de poder computacional e banda. Para acompanhar os 90 mil pares, foi necessário um supercomputador com 100 nodos, muito espaço em disco (que foi completamente consumido em um determinado momento) e banda suficiente para monitorar todos os pares uma vez a cada um minuto.

Uma das grandes vantagens daquele trabalho foi a quantidade de pares (IPs) que foram acompanhados para traçar comportamentos. Além disso, muitos enxames e tipos de arquivos foram monitorados, constituindo uma quantidade de dados bastante significativa. Uma desvantagem citada pelos autores é que não foi possível comunicar-se com pares atrás de *firewalls*, não foi possível obter todos os pares que participavam do enxame e, por fim, argumentaram que modificações no protocolo criaram pequenas distorções. Além disso, o estudo limitou-se ao que era disponibilizado pela comunidade Suprnova [42], não verificando outros vastos espaços de disponibilização de *torrents* e sistemas de busca.

O segundo trabalho, apresentado por Iosup e colegas [48], introduz um *framework* para monitoramento de rede BitTorrent chamado de MultiProbe. Este *framework* é composto por três partes. A primeira, de pré-processamento, é responsável por investigar uma página típica de *torrents*, no caso Mininova [49], e colher os *torrents* com maior quantidade de usuários. A segunda parte é o módulo de medições, responsável por medir as redes BitTorrent e armazenar os resultados. Por fim, o último módulo é responsável pelo pós-processamento, onde são identificadas características de banda e localidade.

Para realizar essas medições, foi utilizada uma infra-instrutora computacional bastante poderosa. Primeiramente, foram utilizados cem nodos de um supercomputador para fazer a medição ativa do BitTorrent, ou seja, responsável por obter os pares do rastreador, conectar aos mesmos e monitorá-los de forma pró-ativa. Segundo, cinquenta máquinas do PlanetLab

foram utilizadas para monitoramento iniciado passivamente. Este tipo de monitoramento teve o objetivo de conseguir acompanhar pares que estão atrás de *firewalls* e, por isso, não seriam monitorados pelo supercomputador. Além destes, mais trezentas máquinas do PlanetLab foram utilizadas para contabilizar e medir as rotas entre as máquinas e os pares que conectaram nas outras cinquenta máquinas.

Os resultados obtidos tiveram foco em análise baseada em localização. Primeiramente, foi apresentado um *ranking* do número de usuários e quantidade de dados transmitidos classificado por região e por país. Neste caso, uma constatação interessante foi que o número de pares presentes em um enxame não afeta o *download* médio. Depois, foram analisadas as rotas e o RTT médio para os pares da rede. Por fim, o comprimento das rotas e distâncias entre pares foram medidas.

Por terceiro e último, o trabalho publicado por *Chow* e colegas [47] apresenta uma ferramenta chamada BTM, cujo foco é monitorar "automaticamente" (baseado em regras) enxames BitTorrent para detecção de pirataria. A motivação do trabalho está dentro do contexto de pirataria e das perdas sofridas pela indústria em virtude desta ação. A dificuldade citada pelos autores é identificar e encontrar as redes BitTorrent que compartilham um determinado conteúdo, pois as redes podem ser privadas (fechadas apenas para determinados usuários) ou podem ter os arquivos *torrent* disponíveis apenas através de métodos não públicos como e-mail.

A ferramenta é composta de dois módulos. O primeiro deles denominado *Torrent Searcher* é responsável por encontrar arquivos *torrent* na Internet. Basicamente, o *Searcher* recebe como entrada um conjunto de URLs (neste caso, apenas fóruns) para obter as informações como se fosse um *crawler*. O segundo módulo é denominado *Torrent Analyser* que é responsável por monitorar os enxames referentes aos *torrents* obtidos pelo *Torrent Searcher*. Além dos módulos, existe uma *engine* denominada *ruler* para disparar ações quando alguma característica pré-determinada acontece. Por exemplo, a presença de semeadores na Inglaterra poderia disparar uma ação de monitoramento.

O módulo Searcher é bastante limitado, pois pode tratar apenas de alguns tipos de *sites* (como fóruns) e não é capaz de utilizar mecanismos de busca ou listagem dos sítios de comunidades (onde concentram-se a maior parte dos *torrents*). Para utilizar o módulo em fóruns com autenticação, é preciso entrar manualmente no site para obter o *token* de autenticação (*cookie*) para que o módulo seja capaz de investigar o fórum, dificultando a automação do processo.

Da mesma forma que o módulo anterior, o *Analyser* apresenta grande limitação para monitoramento de pares. A escalabilidade é muito baixa, pois o mesmo somente é capaz de monitorar 50 pares em um determinado momento. Não são tratadas questões como a obtenção de pares junto ao rastreador nem a conexão com pares protegidos por NATs ou *firewall*. Na descrição do módulo, não consta como o mesmo faz a análise dos enxames e nem que tipo de informação ele é capaz de obter.

## 3.3 Comparação

Esta seção apresenta uma comparação entre as cinco técnicas mostradas na seção anterior, quantificando, de forma compreensiva e qualitativa, as técnicas utilizando características que consideramos importantes para avaliação das técnicas. No decorrer da seção, explica-se porque os qualificadores mostrados na Tabela 1 foram adotados. Note que a análise apresentada na tabela tem foco no processo de obtenção dos dados sobre os enxames. Dito isso, definimos as características de avaliação da seguinte forma. Escalabilidade indica se é viável empregar a técnica à medida que a quantidade de pares no ambiente aumenta. Intrusão refere-se ao quanto à técnica afeta o andamento dos enxames ou pares. Viabilidade indica a possibilidade de implantação da técnica para monitorar o universo *torrent*. Profundidade é a quantidade de informações obtidas, que é descrita separadamente através da Tabela 2. Por fim, acurácia representa o quão atual é o dado coletado.

| ***Estratégia*** | ***Escalabilidade*** | ***Intrusão*** | ***Viabilidade*** | ***Acurácia*** |
|---|---|---|---|---|
| **Individuais** | - | - | - | - |
| *Logs* | Alta | Zero | Muito Difícil | Baixa |
| *Agentes Instrumentados* | Baixa | Alta | Difícil | Alta |
| **Universo** | - | - | - | - |
| *Enlace* | Muito Baixa | Zero | Muito Difícil | Muito Alta |
| *Rastreadores* | Média | Zero* | Fácil | Baixa |
| *Enxames* | Média | Baixa | Média | Alta |

**Tabela 1: Comparação entre as Técnicas**

A Tabela 2 apresenta uma lista não exaustiva das métricas ou informações que são obtidas por cada técnica. Nesta tabela, *Sim* indica quando a métrica é obtida e *Não* indica quando a ela não é obtida. *(1)* indica que os dados podem ser obtidos se o fornecedor dos *logs* enviar os *torrents* correspondentes aos arquivos de *log* recebidos. *(2)* significa que os dados não estão previstos na técnica, mas podem ser obtidos caso os arquivos *torrents* sejam relacionados com os dados coletados. O * indica que os dados coletados para aquela métrica referem-se apenas ao nó local, não sendo possível obter a taxa de *upload* para outros pares.

*(3)* indica que os dados podem estar disponíveis, caso o rastreador informe os mesmos em sua página. Neste caso, os dados são os mesmos da técnica que utiliza os *logs*. Por fim, *(4)* significa que apenas é identificada a decisão de dessufocar (dessufocamento otimista), sem obter mensagens de interesse (*INTERESTED*).

A primeira técnica, obtenção de informações utilizando os *logs* dos rastreadores, apresenta a melhor escalabilidade entre as técnicas. Isso acontece porque os dados são transferidos *offline*, depois que os eventos ocorreram (sem necessidade de monitoramento contínuo). Devido a não capturar os dados *online*, também não há intrusão. A principal limitação está na viabilidade de implantação, pois, para monitorar o universo *torrent*, é preciso obter os *logs* de todos os rastreadores – uma tarefa muito complicada. Por fim, a acurácia é baixa, pois os pares atualizam o rastreador com intervalos que podem ser muito superiores a quinze minutos [60]. Uma limitação importante é não poder fazer acompanhamento contínuo, pois os *logs* são obtidos posterior os acontecimentos.

| | ***Objeto de Investigação*** | **Enxames Individuais** | | **Universo *Torrent*** | | |
|---|---|---|---|---|---|---|
| | ***Métrica*** | *Logs* | *Agentes Instrumentados* | *Enlace* | *Rastreadores* | *Enxames* |
| **Par** | Endereço IP | Sim | Sim | Sim | (3) | Sim |
| | Taxa de *Download* | Sim | Sim | Sim | (3) | Sim |
| | Taxa de *Upload* | Sim | Não(2) | Sim | (3) | Não |
| | *Sharing Ratio* | Sim | Não(2) | Sim | (3) | Não |
| | Tempo de Semeadura | Sim | Sim | Sim | (3) | Sim |
| | Agente de Cliente | Sim | Sim | Sim | (3) | Sim |
| | Extensões | Não | Sim | Sim | Não | Sim |
| | Tomada de decisão | Não | Sim | Sim | Não | Não(4) |
| **Conteúdo** | Tamanho do Conteúdo | (1) | (2) | Sim | (3) | Sim |
| | Tamanho da Peça | (1) | (2) | Sim | (3) | Sim |
| | Quantidade de Peças | (1) | (2) | Sim | (3) | Sim |
| | Tamanho do Bloco | (1) | (2) | Sim | (3) | Sim |
| | Quantidade de Blocos | (1) | (2) | Sim | (3) | Sim |
| **Enxame** | Nome do Conteúdo | (1) | Sim | Sim | (3) | Sim |
| | Tipo do Conteúdo | Não | Não | Sim | Não | Sim |
| | Número de Semeadores | Sim | Sim | Sim | (3) | Sim |
| | Número de Sugadores | Sim | Sim | Sim | (3) | Sim |
| | Taxa de Velocidade Média | Sim | Sim | Sim | (3) | Sim |
| | Percentual de Free-Riders | Sim | Não | Sim | (3) | Não |
| | Distribuição das Peças | Não | Sim | Sim | Não | Sim |
| | Disponibilidade | Sim | Sim | Sim | (3) | Sim |

**Tabela 2: Métricas Obtidas**

A segunda técnica, utilizar agentes instrumentados, apresenta uma escalabilidade muito baixa devido à necessidade de banda para executar o protocolo completo (com compartilhamento de dados). Devido a esse compartilhamento, essa técnica possui a mais alta intrusão, pois interfere fortemente no andamento do enxame transmitindo dados. Além disso, a baixa escalabilidade faz com que a viabilidade seja de difícil implantação, pois a capacidade de banda necessária é muito elevada. Entretanto, as vantagens dessa técnica estão nas informações obtidas e na acurácia, pois os pares enviam a informação (*HAVE, INTERESTED, CHOKE*) diretamente ao par, sem atraso.

A terceira técnica, monitorar o enlace de rede, possui virtudes importantes, como intrusão zero, devido ao monitoramento ser passivo (nível do enlace). Além disso, é possível obter todas as informações disponíveis (profundidade) e com a melhor acurácia entre as técnicas, pois a captura dos pacotes na rede acontece antes mesmo que estes cheguem ao seu destino. Entretanto, duas desvantagens tornam essa técnica problemática. A primeira dela diz respeito a escalabilidade, pois monitorar todos os fluxos de dados passando em roteadores e *switches* é um processo altamente custoso. A segunda, diz respeito a viabilidade, pois é muito difícil obter acesso a todas as redes de dados existentes no mundo para colocar os monitores em funcionamento. Por fim, uma limitação menor trata da dificuldade em identificar tráfego criptografado.

A quarta técnica, monitorar páginas dos rastreadores, tem limitações dependentes não apenas da capacidade do monitor, mas da largura de banda presente nos sítios dos rastreadores para prover dados. A quantidade de dados necessária para medir o andamento dos enxames é similar ao da quinta técnica. A intrusão no enxame é zero, pois não contata o mesmo. O alto tráfego que pode ser gerado para acessar as páginas, pode gerar algum impacto no rastreador, mas, além de não existir estudo para esse fato, o rastreador bloqueia IPs que tentam contatá-los com uma freqüência elevada. A profundidade e acurácia da informação é, no máximo, a mesma provida pela análise de *logs*. Entretanto, a profundidade pode ser muito menor, visto que rastreadores como o do *The Pirate Bay* não fornecem informações sobre os pares em suas páginas. Apesar da viabilidade teórica (apenas baixar páginas dos rastreadores), a limitação em utilizar essa técnica encontra-se na informação disponibilizada, pois os maiores rastreadores não fornecem informações sobre os enxames.

Por fim, a quinta técnica, monitoramento dos enxames através dos pares e rastreadores, tem características que viabilizam sua implantação em larga escala. Na prática, dois trabalhos já demonstraram a viabilidade na coleta de grande quantidade de informações. Para tal, precisam-se encontrar os enxames e contatar pares e rastreadores. A escalabilidade é

similar a quarta técnica, dado que o custo aumenta linearmente de acordo com a quantidade de pares/rastreadores monitorados. A intrusão é baixa, pois dados não são enviados ao enxame, dado que são feitos apenas pedidos de conexão com troca do *bitfield* e de mensagens *HAVE*. Por fim, a acurácia é alta, pois os monitores são informados pelos monitorados das peças recebidas de forma imediata (*HAVE)* ou na próxima abertura de conexão (configurável).

# 4 ARQUITETURA

O *TorrentU* foi projetado com base nas estratégias apresentadas no capítulo anterior e, particularmente, nas limitações impostas pelas mesmas. O objetivo do *TorrentU* é examinar (monitorar) o universo de redes BitTorrent e permitir a extração de informações de interesse. Tais informações são obtidas na forma de "visões do universo", que representam uma fração do mesmo, e refletem escolhas quanto às informações de interesse. Mais precisamente, define-se visão como o subconjunto do Universo de Redes BitTorrent que está sendo observado segundo critérios pré-estabelecidos (atributos do Universo), como localidade dos pares, tipo de conteúdo, tamanho do enxame, entre outros.

Para efetuar o monitoramento dos enxames existentes, podem ser utilizados um dos quatro modelos apresentados na Taxonomia de Schonwalder [63]. Os modelos são os seguintes: (a) centralizado; (b) fracamente distribuído; (c) fortemente distribuído; e (d) cooperativo. O primeiro modelo (a) centraliza todo o controle e informações em um único elemento, dificultando a escalabilidade. O segundo (b) tem um elemento centralizador (de controle), mas delega as tarefas de monitoramento para gerentes intermediários, aumentando a escalabilidade, mas mantendo o controle. O terceiro (c) apresenta um número maior de gerentes intermediários e estes se comunicam entre si. Isso torna o modelo mais distribuído e necessita maior capacidade de sincronização entre os elementos. Por fim, o último modelo (d) não apresenta centralização, é necessário grande esforço de sincronização e existe, praticamente, a mesma quantidade de gerentes e agentes.

Tendo em vista essas características, a seguir, são apresentados os requisitos para o TorrentU (Seção 4.1), seguido de uma visão geral da arquitetura (Seção 4.2), o modelo de dados que representa o universo *torrent* e os eventos monitorados (Seção 4.3), os componentes do *TorrentU* (Seções 4.4, 4.5, 4.6 e 4.7), as métricas coletadas (Seção 4.8) e a interação entre estes componentes (Seção 4.9).

## 4.1 Requisitos

Definem-se os requisitos em três níveis: externo (visão do usuário); interno (funcionalidades dos subsistemas); e não-funcionais. Primeiramente, os requisitos externos são expressos através da visão do usuário para utilização da solução. Estes são de alto nível e descrevem o que fazer. Por segundo, os requisitos internos representam as funcionalidades dos subsistemas, descrevendo o que cada um deve fazer, ou seja, como a solução opera

internamente. Por fim, os requisitos não--funcionais expressam necessidades voltadas ao desempenho.

Os requisitos externos são os seguintes:

- *permitir a configuração do domínio a ser monitorado*: a solução deve ser capaz de oferecer ao usuário a possibilidade de identificar qual a visão (objeto de estudo) que será monitorada. Para isso, é necessário escolher o que se deseja monitorar, como: as comunidades pesquisadas; o conteúdo (tipo, tamanho, nome); pares participantes (localidade); e a abrangência das informações;
- *identificar as informações a serem obtidas*: indica **o que** será medido. Citam-se, como exemplo, os seguintes atributos: quantidade de *torrents* (enxames) de um dado conteúdo (ou conjunto de conteúdos); quantidade de pares participantes; quantidade de semeadores; localização dos pares e do rastreador; velocidade de disseminação; agente utilizado; e extensões habilitadas.

Os requisitos internos são as funcionalidades dos subsistemas que permitem atender aos requisitos externos. A lista de requisitos internos segue:

- *localizar os arquivos de metadados*: dado o interesse em extrair um certo conjunto de informações de um determinado domínio, o primeiro requisito (ou passo) é encontrar (em comunidades) os arquivos de metadados correspondentes, pois tais arquivos representam o ponto de partida para entrada em um enxame;
- *localizar os pares*: tendo sido encontrado e carregado o conjunto de arquivos de metadados, é possível ingressar nos enxames contactando-se os respectivos rastreadores, e a partir daí obter listas de IPs/portas de pares que participam do enxame;
- *integrar as informações dos diferentes enxames e comunidades*: o armazenamento das informações coletadas de múltiplos enxames em um repositório unificado possibilita consultas e geração de gráficos que demonstram o comportamento de forma ampla, ou seja, sem a necessidade de estudar os enxames separadamente;
- *flexibilidade*: a configuração das estratégias mais adequadas para monitoramento de comunidades e de enxames permite que o sistema possa ser adaptado a cada necessidade. Dessa forma, é possível fazer um *tradeoff* entre as estratégias, identificando qual é a mais adequada para um caso de estudo de acordo com abrangência, acurácia, profundidade e custo;

- *abrangência*: permite que seja possível restringir o universo de pesquisa, por exemplo, dando foco em um tipo de conteúdo, no percentual de pares monitorados dentro dos enxames, monitoramento de pares protegidos por *firewall*, entre outros. Assim, é possível fazer um *tradeoff* entre abrangência e custo para monitoramento.

Por fim, os requisitos não funcionais descrevem as limitações de desempenho que precisam ser tratadas, como descrito a seguir:

- *eficiência*: a arquitetura deve permitir que apenas os dados necessários sejam coletados, evitando desperdício de recursos. Como exemplo, se o usuário selecionou que apenas deseja identificar o número de *torrents* dado um determinado conteúdo, a arquitetura não deve utilizar recursos para monitorar pares ou rastreadores;
- *escalabilidade*: considerando as dimensões estimadas do universo torrent, para que o TorrentU extraia e relacione informações, ele deve empregar algoritmos e procedimentos que sejam, dentro do possível, escaláveis.

Sobre os requisitos acima, note-se que selecionar o conjunto de pares e enxames monitorados de acordo com uma visão específica possibilita entender o comportamento de usuários e o impacto que o tipo de conteúdo tem nos mesmos, bem como a influência geográfica em seu comportamento.

## 4.2 Visão Geral

Conforme indicado nos requisitos acima, a arquitetura deve extrair visões parciais do universo *torrent*, de acordo com uma especificação de “filtro” que contém restrições sobre o universo de busca e também os locais que devem ser observados. Um exemplo disso é a identificação do conteúdo (tipo, tamanho ou comunidade), de enxames ou pares a serem monitorados. Uma visão parcial é obtida em quatro etapas, em ordem crescente de detalhe: (i) identificação dos locais de busca (comunidades, sítios); (ii) identificação dos conteúdos de interesse (selecionando arquivos torrent segundo seus metadados); (iii) identificação dos enxames de interesse, a serem examinados, segundo atributos dos mesmos (tamanho, localidade, entre outros); (iv) identificação de pares de interesse em cada enxame.

Um exemplo de especificação no passo (i) seria identificar as comunidades e sítios que se deseja procurar arquivos de metadados, como o *The Pirate Bay*, selecionadas manualmente. No passo (ii) seria identificar todos os conteúdos associados ao software Adobe Ilustrator, levando à seleção de um conjunto de arquivos *torrent*. Um exemplo do passo (iii) seria a seleção dos enxames abrigados por rastreadores localizados no Brasil, dentre estes que

compartilham o conteúdo selecionado. Já no passo (iv), seria possível identificar a presença de determinado par (pelo seu IP e porta), possivelmente restringindo por características deste par.

Recorde que na Capítulo 2 foram apresentados os atributos para cada um dos componentes básicos (conteúdo, enxame, par) que formam o universo torrent. O objetivo do *TorrentU* é determinar um conjunto de atributos (informações) para uma visão selecionada segundo os quatro passos definidos. Exemplos de atributos de interesse para um conteúdo (informação agregada de múltiplos enxames) seriam a taxa de disseminação, a taxa de sucesso no *download* (após obtenção do arquivo *torrent*) e identidades (IP) dos pares semeadores e sugadores.

Neste conjunto de passos que encontra-se o requisito de flexibilidade. Diz-se flexível, pois permite escolher o que deve ser monitorado baseado em um conjunto de atributos. Essa é uma das diferenças sobre trabalhos anteriores, que não tinham essa possibilidade. É possível, com isso, selecionar as comunidades, enxames ou pares de interesse. Com isso, pode-se monitorar apenas o andamento dos pares presentes no Brasil, a quantidade de pares nos enxames que compartilham um filme ou ainda a velocidade com que novos *torrents* são incluídos em uma comunidade.

Em função dos requisitos estabelecidos, a arquitetura está dividida em dois subsistemas de monitoramento, três repositórios e uma interface para configuração dos subsistemas e geração dos gráficos, conforme ilustrado na Figura 4.1 e comentados a seguir. O primeiro subsistema, denominado *Torrent Crawler* (ou somente *Crawler*), é um *crawler* focado [61] que periodicamente examina sítios de comunidades BitTorrent e carrega arquivos de metadados, que descrevem o conteúdo compartilhado e o endereço IP e porta do rastreador que ajudará os pares do enxame a se encontrarem. O *crawler* alimenta um repositório de arquivos *torrent*, denominado *Universal Torrent Repository* (UTR), utilizado para armazenar as informações coletadas. À medida que o *Torrent Crawler* obtém novos arquivos de metadados e os insere no repositório UTR, a visão global do universo *torrent* se expande, e as consultas e processamento que podem ser executados se tornam mais abrangentes. A partir desse repositório, a *Interface* é utilizada para selecionar os *torrents* dos enxames a serem monitorados de acordo com os atributos configurados, colocando-os no segundo repositório, denominado *Monitored Torrents* (MT). Esse repositório serve de entrada para o segundo subsistema, denominado *Torrent Monitor* (ou somente *Monitor*), que é responsável pelo monitoramento de enxames de interesse. O *Monitor* alimenta o terceiro repositório, denominado *Swarm Information Base* (SIB), com as informações coletadas a partir dos

enxames associados a uma determinada visão (conceito de visão descrito na seção anterior). Por fim, a partir da *Interface*, os dados dos repositórios (SIB e UTR) são lidos e gráficos contendo as informações selecionadas são gerados. As interações entre os subsistemas, repositórios e a Interface são aprofundados na Seção 4.9, após a descrição dos elementos da arquitetura. A seguir, descreve-se o modelo de representação de dados.

## 4.3 Modelo de Dados

As informações obtidas a partir do universo *torrent* são organizadas em um modelo de dados. O mesmo foi criado com base nos componentes de um universo, conforme apresentado na Figura 2.2 (Capítulo 2). O modelo é apresentado na Figura 4.2, usando-se notação UML. Nele encontram-se os seguintes elementos, cada um com seu conjunto de atributos:

- *Conteudo*: indica o conteúdo digital sendo compartilhado, usando para identificação uma combinação de título e do tipo/categoria (note-se que um mesmo conteúdo pode ser publicado de diferentes formas no universo torrent, de acordo com critérios definidos pelos usuários das comunidades). Destaca-se que este modelo não trata de forma automática a associação que arquivos em formatos diferentes podem pertencer ao mesmo conteúdo;

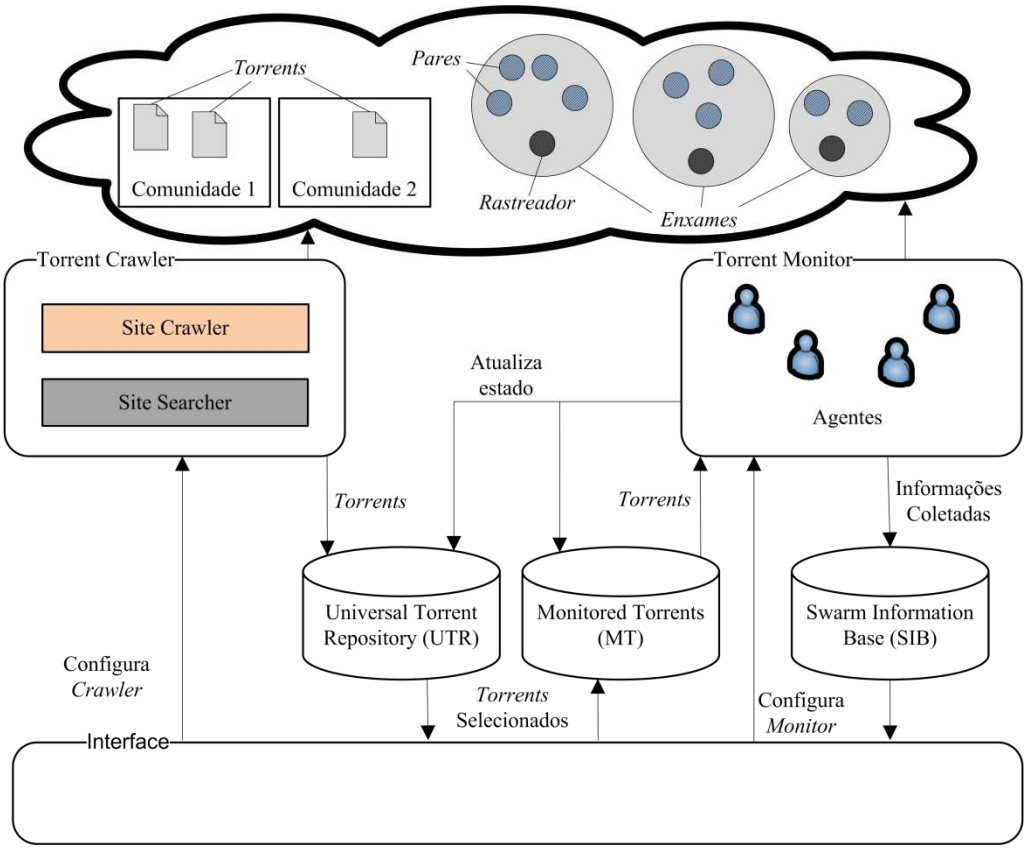


**Figura 4.1: Arquitetura *TorrentU***

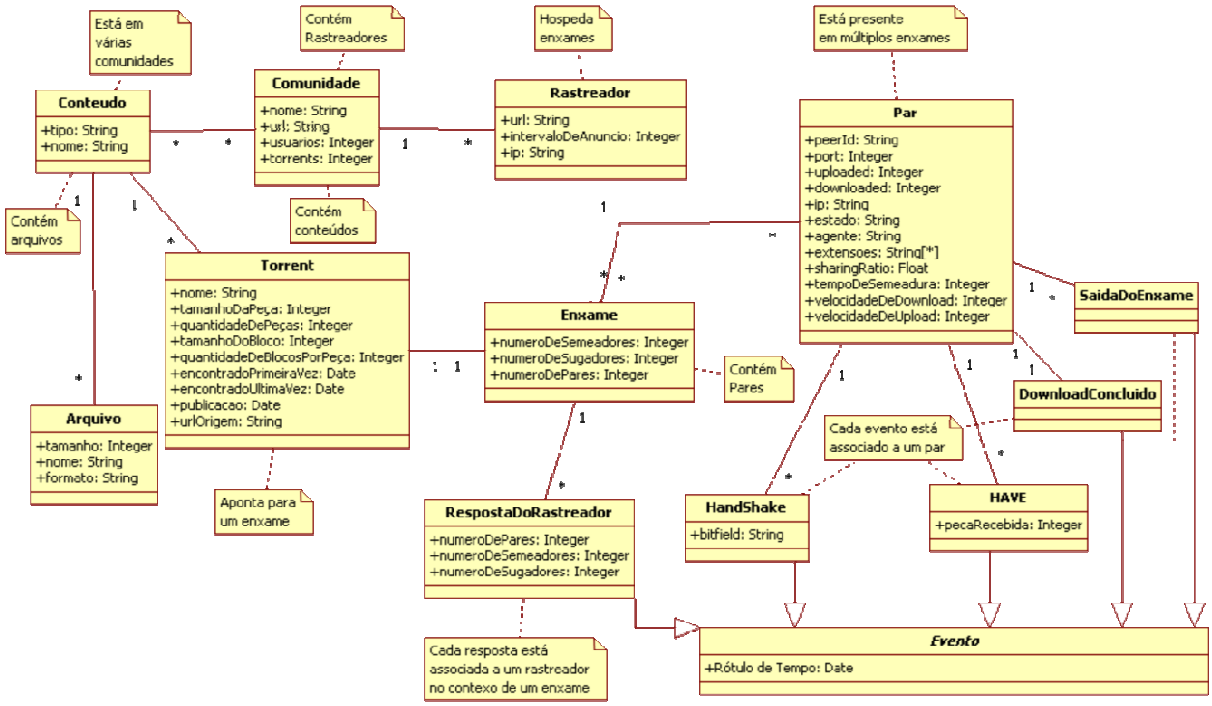


**Figura 4.2: Modelo de Dados**

- *Comunidade*: identifica o sítio em que são encontrados arquivos de metadados e, em alguns casos, rastreadores. Na prática, é o valor informado pelo usuário quando este seleciona quais comunidades monitorar. Comunidades são pontos centrais para busca e obtenção dos *torrents*;
- *Torrent*: informações pertinentes ao arquivo de metadados, que contém o endereço dos rastreadores, descrição do conteúdo, nome dos arquivos, tamanho, *hashes* e tamanho da peça;
- *Rastreador*: informações sobre o elemento central que coordena o enxame;
- *Enxame*: informações sobre o conjunto dos pares mais o rastreador;
- *Par*: informações sobre o elemento participante do enxame interessado em obter o conteúdo;
- *Arquivo*: informações sobre um arquivo que pertence a um Torrent, ou seja, representação do arquivo de dados do conteúdo.

Os elementos apresentados dizem respeito às informações presentes no universo *torrent*. Além disso, é necessário armazenar os eventos monitorados a partir dos enxames de forma a manter um histórico dos acontecimentos. Esse histórico permite que sejam traçados gráficos expressando comportamentos ao longo do tempo. Devido ao requisito de escalabilidade no armazenamento, guardam-se apenas os eventos que foram pré-definidos

como necessários para obter as métricas selecionadas. Dessa forma, os seguintes elementos da Figura 4.2 representam os eventos monitorados:

- *Evento*: classe abstrata que representa um dos eventos que podem ser armazenados. Ela contém o tempo em que a observação do evento ocorreu;
- *RespostaDoRastreador*: representa a resposta que o rastreador fornece sobre um enxame específico. Uma resposta está associada apenas a um enxame, mas um enxame poderá ter várias respostas (que apresentam dados diferentes ao longo do tempo);
- *HandShake*: dados enviados pelo par no momento em que uma conexão é estabelecida. Esses dados representam a quantidade de peças já obtidas pelo par. Assim, cada resposta está associada a somente um par, enquanto um par pode ter várias respostas associadas, mostrando sua evolução no tempo;
- *HAVE*: mensagem enviada pelo par informando qual peça do conteúdo foi recebida. Um par tem várias mensagens associadas e uma mensagem está associada a um par;
- *DownloadConcluido*: evento que representa o momento em que o par terminou de carregar o conteúdo através do protocolo. A associação com o par é 1:1, pois o par baixa o conteúdo apenas uma vez;
- *SaidaDoEnxame*: representa o momento em que o par abandona o enxame. Um par tem várias mensagens associadas e uma mensagem está associada a um par.

## 4.4 Repositórios

Relembrando, três repositórios fazem parte da arquitetura. O primeiro deles é um repositório universal de *torrents*, armazenando todos os *torrents* encontrados pelo *Crawler* e periodicamente verificados pelo mesmo. Este repositório é denominado *Universal Torrent Repository* (UTR). O repositório armazena parte do modelo de dados descrito na Seção 4.3. Mais especificamente, os objetos *Comunidade*, *Conteudo*, *Arquivo* (dados), *Torrent* e o *Rastreador*.

Cada tupla deste repositório contém as seguintes informações (pertencentes ao modelo de dados): o arquivo de metadados, a data em que o mesmo foi carregado pela primeira vez, data da última verificação de existência do mesmo, lista de URLs onde este arquivo *torrent* foi encontrado, data de publicação do *torrent*, lista dos arquivos presentes no *torrent* (com seu tamanho), URLs dos rastreadores, tamanho da peça, número de peças, categorização (presente em alguns sítios), as comunidades ou sítios em que foi encontrado, data da última verificação

de disponibilidade do enxame e data da última confirmação de disponibilidade (questões tratadas na Seção 4.7).

O segundo repositório mantém como tuplas os *torrents* dos enxames sob monitoração, e por isso representa um subconjunto do repositório universal. Quando um enxame deixa de ser de interesse, ele é removido deste repositório. Diferentemente do repositório anterior, que guarda informação sobre o *torrent* nas tuplas, este repositório apenas referencia o arquivo de metadados, no repositório UTR, que está sendo monitorado e inclui dois rótulos de tempo informando a última vez que o enxame estava disponível (existência de rastreador e pares) e quando a última tentativa dessa verificação foi realizada. Os rótulos de tempo são utilizados para atualizar os campos no repositório UTR, de forma a tornar um *torrent* indisponível para monitoramento (evitando desperdício de recursos) e para armazenar informações sobre disponibilidade. O nome do repositório é *Monitored Torrents* (MT).

O terceiro e último repositório, denominado *Swarm Information Base*, registra de maneira estruturada informações obtidas pelo *Monitor* sobre os enxames observados. Cada tupla do repositório contém informações a respeito do enxame e dos pares. Essas informações estão descritas no modelo de dados apresentados na Figura 4.2. Como a informação a respeito de enxames e pares é bastante dinâmica, ou seja, muda ao passar do tempo, a base é responsável por armazenar os eventos monitorados individualmente através dos elementos *RespostaDoRastreador*, *HandShake* e *HAVE*, como visto na Figura 4.2, para registrar os dados de forma temporal. Estes eventos estão associados da seguinte forma: a resposta do rastreador está associada a apenas um enxame (aquele sendo monitorado); cada elemento *HandShake* e *HAVE* está associado a um único par (provedor desta informação). O armazenamento do momento de coleta dos eventos tem a função de disponibilizar dados que podem ser relacionados em função do tempo, por exemplo, evolução da completude do conteúdo de um par.

## 4.5 *Torrent Crawler*

Identificar os *torrents* presentes na Internet que representem o compartilhamento de um determinado tipo de conteúdo é uma tarefa extremamente desafiadora, pois consiste em varrer os sítios presentes na WEB a procura dos arquivos de metadados. Um exemplo comercial de ferramenta que utiliza mecanismos de busca é o Google [25]. A pesquisa por conteúdo na WEB é abordada por Christopher e colegas [50], onde as dificuldades para encontrar um determinado tipo de conteúdo são exploradas. A área que trata esse tipo de busca é chamada de Recuperação de Informação (*Information Retrieval*).

O objetivo do *Torrent Crawler* é monitorar o universo de arquivos *torrent* disponíveis em comunidades BitTorrent. Para tal, o *Crawler* mantém uma cópia local do *torrent* no repositório UTR. Pode-se adotar duas estratégias para obtenção dos arquivos de metadados: (i) utilizar mecanismos disponibilizados na internet; (ii) implantar uma máquina de busca e indexação focada em *torrents*. Alternativas para o primeiro caso incluem (a) utilização de máquinas genéricas de busca, como *Google* e *Yahoo*; e (b) mecanismos de busca disponibilizados em sítios especializados em *torrents*, como *The Pirate Bay* e *Mininova*.

Considerando o custo e complexidade duas estratégias ((i) e (ii)), optou-se por empregar duas soluções, descritas a seguir. Na primeira delas, um *crawler* (denominada *Site Crawler*) focado recebe como parâmetro o conjunto de sítios a serem investigados, assim como informações de autenticação (para acesso a comunidades fechadas). O *crawler* restringe sua busca nos sítios fornecidos e carrega apenas arquivos de metadados, sem entrar em sítios fora do domínio especificado. Na segunda (denominada *Site Searcher*), utilizam-se os sistemas de busca das comunidades e sítios para obter os arquivos. Esta opção tem o objetivo de permitir uma busca mais restritiva que forneça como entrada palavras chave. Com o segundo método, o custo para armazenamento e procura são menores comparados a utilização do *crawler* varrendo os sítios inteiramente. Entretanto, a definição de palavras-chave para um sítio é uma configuração de médio-prazo. Ou seja, é uma escolha feita de forma estática à instanciação do *crawler*, para otimizar a carga de *torrents* de uma comunidade, e não para refletir buscas de um usuário. Um exemplo de ferramenta que utiliza essa estratégia é o Bit Che [54], onde suas buscas são feitas nas comunidades e sítios mais populares.

A cada varredura, os novos arquivos *torrents* encontrados são carregados e inseridos no repositório, preenchendo-se as informações complementares, como por exemplo, o *timestamp* para registrar a primeira carga (vide seção anterior). Os arquivos já carregados (onde a *URL* na comunidade é a mesma) não são recarregados, mas as informações de disponibilidade na tupla correspondente são atualizadas. Além disso, caso sejam encontrados *torrents* que já tenham sido carregados no banco (onde há um casamento perfeito entre o *infohash* [62] e o rastreador), novas entradas não são adicionadas. Neste caso, a *URL* do novo local é adicionada a base de dados.

Assumindo que o processo de consultar o conjunto completo de sítios (para as duas abordagens) não exceda, no pior caso, tempo $T_c$ minutos, o *Crawler* inicia uma nova varredura com período $P_c$, onde $P_c > T_c$. A sobrecarga de rede gerada por esse processo dependerá do tamanho das páginas HTML carregadas e da quantidade de novos arquivos *torrents* obtidos. Para evitar uma sobrecarga excessiva, particularmente no primeiro acesso,

há um limite superior no número de *torrents* que podem ser carregados de um determinado sítio. Além disso, de forma a refletir as características individuais de cada comunidade (em termos de volume e dinamismo dos *torrents* publicados), nem todo sítio precisa ser examinado a cada ciclo: um parâmetro inteiro é associado a cada sítio e usado para representar a periodicidade de consulta ao mesmo (por exemplo, um valor três faz com que apenas uma consulta seja efetuada a cada três ciclos $P_c$).

O *Crawler* influencia na abrangência no primeiro nível citado anteriormente, mais precisamente no que diz respeito ao número de comunidades e sítios pesquisados. Dessa maneira, a abrangência se dá pela inserção dos sítios para realização da busca. Quanto mais sítios forem pesquisados, maior será a abrangência, mas com isso aumenta-se o custo de obtenção e monitoramento. A abrangência de segundo nível, quantidade de enxames observados dentro de cada comunidade, também é afetada por este subsistema devido às opções de filtro para busca dos arquivos de metadados. Quanto mais restritivo o filtro, menor a abrangência e menor o custo.

Um fato relevante é que podem ser encontrados arquivos de metadados que compartilham conteúdo diferente do anunciado, ou seja, conteúdo que pode ser falso ou corrompido. Identificar conteúdo falso ou corrompido é um desafio apresentado por Liang [53]. São citadas técnicas na tentativa de identificar conteúdo poluído para arquivos de músicas, mas ataques de poluição de metadados e de poluição de conteúdo mais elaborados podem passar despercebidos facilmente. Tratar dados corrompidos ou poluídos foge ao escopo deste trabalho.

## 4.6 *Torrent Monitor*

Este subsistema tem como papel monitorar os enxames selecionados pela *Interface* a partir do repositório UTR. Para cada enxame, haverá uma tupla no repositório *Monitored Torrent*, que foi previamente populado pela *Interface*. O *Monitor* processa sequencialmente cada uma das tuplas do repositório e monitora os enxames associados aos arquivos de metadados de acordo com os parâmetros de configuração recebidos, que estabelecem: abrangência do enxame; monitoração de sugadores e semeadores; intervalo entre o contato com o rastreador; monitoramento do tamanho do enxame; monitoração de pares protegidos por *firewall*, abrangência nestes últimos e percentual de *Sybils* (explorado posteriormente); estratégia de contato com os pares e intervalo entre os contatos; filtro por localidade do rastreador (Brasil, China, etc); filtro por tamanho do enxame (número de pares, semeadores e

sugadores); filtro por par (localidade, tipo de cliente utilizado e extensões) e condições de disponibilidade. Essas configurações são tratadas no decorrer desta seção.

De posse das configurações citadas e dos arquivos de metadados, o *Torrent Monitor* monitora os enxames selecionados e armazena as informações referentes aos eventos recebidos no repositório SIB para posterior processamento. Para isso, a arquitetura precisa levar em conta os seguintes critérios: (i) necessidade de centralização da informação (consolidação); (ii) necessidade de delegação das tarefas para aumento da escalabilidade; (iii) necessidade de sincronização entre os elementos monitores para identificar a responsabilidade de cada um; (iv) custo. Dessa forma, adotou-se o modelo de gerenciamento fracamente distribuído (como descrito no início deste capítulo), pois o consideramos mais adequado as necessidades da arquitetura, como exposto nos elementos (i) a (iv). A Figura 4.3 ilustra a composição hierárquica do modelo de gerenciamento.

No modelo de gerenciamento, o Gerente Global (*Global Manager*) tem uma visão completa dos pares sendo monitorados. Dessa forma, ele mantém uma lista que indica quais pares estão sendo monitorados pelos seus Gerentes Locais (*Local Manager*). Cada Gerente Local também tem uma lista dos pares que são monitorados pelos seus Agentes (*Agent Monitor*). Dessa forma, toda vez que um Agente tentar adicionar um novo par em sua lista de monitorados, ele deve contatar o seu gerente para verificar se este par já está sendo monitorado por outro Agente. Essa consulta se propaga até o Gerente Global, caso o novo par não esteja na lista do Gerente Local. O mesmo comportamento acontece quando um par de um exame tenta conectar em um monitor, pois a conexão somente será aceita se outro Agente não tiver estabelecido uma conexão com o mesmo par.

Definido o modelo de gerenciamento, devem-se empregar os agentes para monitorar os pares. Essa tarefa pode ser executada uma de duas estratégias distintas em pares que não estão protegidos por *firewall*: iniciar o *handshake* e fechar a conexão; ou manter a conexão aberta e esperar o recebimento de mensagens *HAVE*. Os custos referentes às duas estratégias são explorados no Capítulo 5. A primeira estratégia possui um parâmetro de configuração que define o intervalo em que o *Monitor* irá conectar-se aos pares, enquanto a segunda estratégia não possui parâmetros.

Por sua vez, a monitoração de pares protegidos por *firewall* não pode ser realizada através da estratégia de conexão para troca do *handshake*, pois não é possível iniciar uma conexão com par protegido. Para solucionar esse problema, pode-se utilizar uma técnica chamada de Ataque *Sybil* [43]. Este ataque consiste em uma entidade criar e controlar múltiplas identidades lógicas em um sistema, de forma a controlá-lo para seu proveito. No

caso do BitTorrent, o *Monitor* pode instanciar múltiplas identidades no rastreador fazendo-se passar por agentes que tem interesse em compartilhar dados. Dessa forma, pares protegidos por *firewall* recebem essas identidades "falsas" do rastreador e tendem a iniciar a conexão com os agentes de monitoramento, permitindo o acompanhamento do progresso deles. Para utilizar essa técnica, é preciso informar, como parâmetro de entrada, o percentual de *Sybils* em relação ao número de pares que deve ser criado.

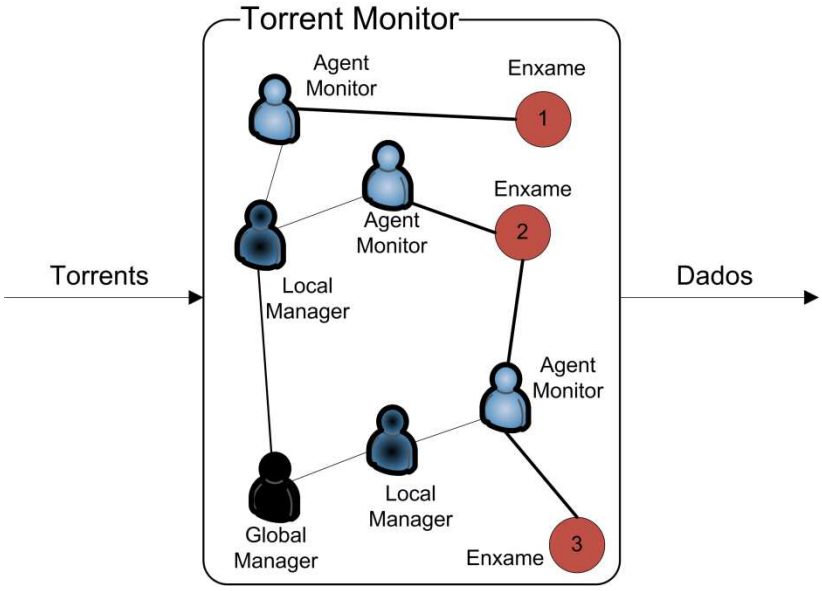


**Figura 4.3: Subsistema *Torrent Monitor***

Segundo o protocolo BitTorrent [44], o número de pares retornados pelo rastreador em uma consulta tem, usualmente, o tamanho cinquenta. Entretanto, um par irá se conectar a apenas 30 desses pares, recusando conexões se estiver com 55 conexões abertas. Baseado nessa informação pode-se estimar que se houver uma identidade de monitor para cada 25 identidades reais, o rastreador irá retornar, na média, duas identidades de agentes por lista de 50 pares. Com esse valor, teríamos uma chance de 85% de um par do enxame iniciar uma conexão remota. Essa probabilidade subiria para 95% se fossem três identidades a cada cinquenta e para 98% se fossem quatro identidades para cada cinquenta.

Além do monitoramento dos pares (protegidos ou não por *firewall*), monitora-se o rastreador para obter duas informações: o número de pares no enxame (sugadores e semeadores) e os endereços destes. Para isso é necessário fazer o anúncio para o rastreador, que responde com essas informações. Esse contato é parametrizado informando o intervalo entre pedidos ao rastreador. Rastreadores não permitem que este pedido seja feito com frequência alta a partir de um mesmo par. Dessa forma, torna-se necessário instanciar

múltiplas identidades para realizar o anúncio, caso a frequência seja mais alta que a permitida pelo rastreador.

Este contato com o rastreador pode não retornar resposta por indisponibilidade ou por desligamento. Dessa forma, configuram-se quantas tentativas e qual o intervalo em que as mesmas são feitas no rastreador antes de considerá-lo desligado ou indisponível. Além disso, um enxame pode não ter pares participantes por um longo período. Assim, as duas situações anteriores configuram casos em que não há utilidade em monitoramento, pois acarreta desperdício de recursos. Para solucionar esse problema, os repositórios que armazenam os arquivos de metadados (UTR e MT) têm marcas de tempo que indicam o último momento de disponibilidade e a última tentativa. Caso um enxame não cumpra as regras de disponibilidade configuradas, ele é marcado como inacessível e não é monitorado.

Depois de escolhida a estratégia e intervalo de monitoramento nos pares e no rastreador, uma série de filtros, obedecendo a restrições informadas pelo usuário, é empregada para selecionar parte do universo *torrent*. O *Monitor* estabelece três tipos de filtros baseados em atributos do rastreador, do enxame e dos pares. O que restringe o rastreador faz a seleção baseado na localização dele. O filtro que seleciona o enxame monitorado restringe a busca utilizando atributos como número de pares totais, de semeadores e sugadores presentes. Por fim, o filtro que atua sobre o par restringe o monitoramento de acordo com a localidade do endereço IP, o agente de cliente utilizado, como *Vuze* e *uTorrent*, e pelas extensões disponíveis no cliente, como PEX e criptografia.

Essas restrições ou filtros influenciam na abrangência do universo monitorado. Além destes, a abrangência é diretamente afetada pelas configurações que indicam como a cobertura do monitoramento deve ser realizada. Para tal, utiliza-se o percentual de pares do enxame, observação de semeadores, sugadores e pares protegidos por *firewall*. Por exemplo, se um enxame tiver 100 pares e definir-se que a abrangência é de 25%, o *Monitor* faz requisições ao rastreador até atingir 25 conexões com pares. Neste cenário, novas requisições são feitas se alguma destas condições acontecer: (i) o enxame crescer; (ii) os pares monitorados deixarem o enxame.

Além da abrangência, o *Torrent Monitor* influencia diretamente na profundidade e acurácia das informações e no custo para obtê-las. O custo é tratado no Capítulo 5. A abrangência é influenciada em dois dos três níveis citados anteriormente. O segundo nível, abrangência dos enxames, é afetado pela quantidade de enxames selecionados a partir do repositório UTR e armazenados no repositório MT. O terceiro nível, abrangência de pares dentro do enxame, é afetado pelo percentual de pares monitorados (através da configuração

mostrada na Seção 4.9). A profundidade é afetada na configuração, onde se define o monitoramento de semeadores, sugadores, tamanho do enxame e progresso dos pares. A acurácia é influenciada pela frequência com que o rastreador é contatado, pela estratégia utilizada para monitoramento dos pares e pelo intervalo em que essa estratégia é utilizada.

## 4.7 Interface

A *Interface* tem a função de comunicar-se com os subsistemas da arquitetura e com os repositórios de forma a representar o papel de ponto central de configuração e controle da solução. A interação que a *Interface* realiza com os subsistemas *Torrent Monitor* e *Torrent Crawler*, que é utilizada para configurar o comportamento dos módulos através de XML, está descrita na Seção 4.9. A presente seção trata especificamente das funcionalidades da *Interface*.

Após configurar o *Torrent Crawler* para obtenção dos arquivos de metadados é necessário selecionar os arquivos armazenados no repositório para monitorar os enxames. Para fazer essa seleção, a *Interface* recebe como entrada as condições que restringem a obtenção de arquivos de acordo com as necessidades de pesquisa. As condições são expressas através dos atributos que compõem as tuplas do repositório UTR (atributos explorados na Seção 4.4). Com isso, é possível selecionar um conjunto de arquivos de metadados no *Universal Torrent Repository*, que permitem monitorar enxames de interesse dado determinadas características, e copiar os mesmos para o repositório *Monitored Torrents*, deixando-os disponíveis para o *Torrent Monitor*.

Após a seleção dos arquivos de metadados e da configuração do subsistema *Torrent Monitor*, os dados monitorados são armazenados na SIB. A partir dos dados da SIB e do UTR, a Interface é capaz de criar correlações entre os valores coletados para os atributos do universo. A próxima seção apresentada métricas e informações que podem ser obtidas e como as mesmas são calculadas.

## 4.8 Definição das Métricas a Serem Monitoradas no Universo

Esta seção apresenta uma lista não exaustiva de métricas e informações propostas em outros trabalhos. Além disso, cria uma métrica para medição do espalhamento das peças baseado no erro quadrático médio. Destaca-se que a taxa de *upload*, *sharing ratio*, percentual de *free-riders* e a tomada de decisão não são obtidos com as técnicas utilizadas por *TorrentU*. As subseções seguintes apresentam as métricas e informações e como as mesmas podem ser obtidas.

### 4.8.1 Taxa Individual de *Download*

A métrica mais citada nos trabalhos de medição (citados por esta pesquisa) é o desempenho na distribuição do conteúdo, ou seja, a taxa em que os dados são transmitidos. A Equação 4.1 apresenta o cálculo da taxa de *download* de um par, onde $V_i(t)$ é a taxa de *download* do par $_i$ no tempo $t$, $Q_i(t)$ é a função que indica quantas mensagens do tipo HAVE o par $_i$ enviou até o tempo $t$, $S$ indica o tamanho de uma peça do conteúdo e $t-1$ indica o período da última medição.

$$V_i(t) = \frac{(Q_i(t) - Q_i(t-1)) * S}{t - (t-1)}$$

**Equação 4.1: Taxa de *Download***

### 4.8.2 Taxa Global de *Download*

Esta métrica indica qual a taxa total acumulada de *download* que todos os pares monitorados atingem. Para coletar essa métrica, utiliza-se a Equação 4.2, onde $V$ é a taxa global de *download*, $V_i$ é a taxa de *download* do par *i* e *N* é o número total de pares sugadores monitorados

$$V = \sum_{i=1}^{N} V_i$$

**Equação 4.2: Taxa de *Download* Global**

Essa informação mostra o impacto que redes BitTorrent causam na transmissão e recebimento de dados. Além disso, é possível mostrar a escalabilidade do sistema colocando no eixo das ordenadas a velocidade total e no eixo das abscissas o número de pares no enxame. Esse gráfico tende a mostrar um comportamento linear crescente, corroborando a escalabilidade do sistema.

### 4.8.3 Taxa Média de *Download*

Essa métrica indica qual a taxa média de *download* de cada par do enxame. Ela pode ser calculada dividindo o valor da velocidade global obtido com a fórmula na Subseção 4.8.2 pelo número total de pares. A Equação 4.3 apresenta o cálculo desta métrica, onde $V$ é a velocidade global e $N$ é o número de pares sugadores total.

$$\overline{V} = \frac{V}{N}$$

**Equação 4.3: Taxa Média de *Download***

### 4.8.4 Capacidade de *upload*

Esse é um valor estimado da capacidade de *upload* do par remoto, pois o valor obtido nesta medição é a capacidade do menor enlace entre o monitor e o par sendo monitorado. Dessa forma, é importante utilizar monitores que tenham uma alta capacidade de transmissão para evitar que o enlace do monitor interfira na medição. A técnica apresentada por Katti [9] pode ser utilizada para esse fim, onde a ferramenta MultiQ foi criada para fornecer esse valor de capacidade do enlace. Ela utiliza o tempo entre as chegadas de pacotes do tipo *ack* para aproximar o valor de capacidade.

Essa métrica pode ser combinada com o desempenho de *download* dos pares para obter-se um gráfico que mostre o quanto o aumento da capacidade de saída do enlace beneficia um par para aumentar seu desempenho de *download*. Destaca-se que essa métrica não representa a taxa de *upload* de um par, pois não é possível obter essa informação com as técnicas utilizadas por *TorrentU*.

### 4.8.5 Distribuição das Peças

Essa métrica mede a eficiência do algoritmo LRF na distribuição das peças entre os pares que participam do enxame. Com os monitores, é possível mapear as peças que cada par do enxame possui e identificar se a distribuição está uniforme (objetivo do algoritmo LRF). Pode-se medir o erro quadrático médio de diferentes enxames na distribuição de peças através da Equação 4.4, onde $P$ é o número de peças total do conteúdo, $O_j$ é o número de peças observado entre os pares monitorados para a peça $j$ e $e$ é o número de peças esperado para cada posição que é obtido com a Equação 4.5.

$$E = \frac{1}{P}\sum_{j=1}^{P}(O_j - e)^2$$

**Equação 4.4: Distribuição de Peças**

$$e = \frac{\sum_{j=1}^{P} O_j}{P}$$

**Equação 4.5: Peças Esperadas**

Com essa métrica de erro é possível comparar diferentes enxames e verificar quais deles tem melhor espalhamento de peças. Outras propriedades podem ser verificadas paralelamente a essa para identificar se existe correlação entre condições do enxame e

espalhamento das peças, como número de semeadores, número de sugadores e tipo de conteúdo distribuído.

Essa métrica de distribuição fornece valores que podem mostrar a tendência para a disponibilidade do enxame. Um gráfico que mostra essa tendência tem a quantidade de ocorrências da peça mais rara do enxame ao longo do tempo. Se essa linha tender a zero, existe alta possibilidade de o enxame ficar indisponível. Se uma determinada peça tiver zero ocorrência no enxame e não houver semeadores, não é mais possível compartilhar o conteúdo, pois nenhum par chegará ao fim do *download*. Exceção ocorre se um par que possuir a peça faltante retornar ao enxame.

### 4.8.6 Tempo de Semeadura

Essa métrica indica o tempo que um par fica no estado de semeador contribuindo puramente para o enxame. Essa métrica pode ser obtida identificando o momento em que um par se transforma em semeador até o momento em que abandona o enxame, mostrando o grau de colaboração dos pares nos enxames e podendo ser correlacionada com enxames distribuindo diferentes tipos de conteúdos. Para determinar o momento que um par vira semeador, deve-se observar quando o mapa de peças fica completo. A contabilização deste tempo de forma precisa passa por desafios, pois semeadores podem sair do enxame e voltar posteriormente.

### 4.8.7 Número de Semeadores

Indica a quantidade de pares no enxame que possuem o conteúdo completo. A métrica é obtida através da resposta do rastreador no parâmetro *complete* que informa quantos pares no enxame já concluíram o processo de *download*. Essa métrica pode indicar o grau de altruísmo no enxame, onde pares que já completaram continuam presentes para ajudar os demais.

### 4.8.8 Número de Sugadores

Indica a quantidade de pares que não possuem o conteúdo completo no enxame. A métrica é obtida através da resposta do rastreador no parâmetro *incomplete* que informa quantos pares no enxame estão com o arquivo incompleto, ou seja, são sugadores. O monitoramento ao longo do tempo desta métrica indica o interesse e popularidade que um conteúdo tem. Além da popularidade expressa pela alta quantidade de sugadores, a métrica

mostra situações de *flash-crowd* e que o sistema é capaz de suportar grande quantidade de pares ao mesmo tempo.

### 4.8.9 Número Total de Pares

Informa a quantidade de pares presentes no enxame no momento da medição. Para isso, soma-se a quantidade de sugadores e semeadores que o enxame possui. Essa métrica é traçada juntamente com as apresentadas nas subseções 4.8.7 e 4.8.8 para demonstrar a quantidade total de pares envolvidos no enxame.

### 4.8.10 Classificação Regional por IP

A classificação regional por IP identifica a quantidade de IPs que cada região do planeta possui participando nos enxames. Pode-se utilizar os serviços disponibilizados por *Web Hosting Romania* [45] ou *IP Address Locator* [46] para fazer a consulta pela região de um IP. Além de identificar que partes do mundo estão participando nos enxames, é possível identificar quais são as maiores capacidades de enlace disponível para saída (combinando com a métrica exposta na Subseção 4.8.4) e identificar quais regiões possuem o melhor desempenho no *download* de dados (combinando com a métrica apresentada no início desta seção). Por fim, também é possível identificar quais são os maiores contribuidores por tempo de *seed*.

### 4.8.11 Tempo de *Download*

Essa métrica indica o tempo que um par leva para baixar o conteúdo compartilhado. Para obter essa métrica, deve-se monitorar o momento que um par entra no enxame até que o mesmo torne-se semeador. Esse valor pode ser usado para calcular a taxa média de *download* utilizando a Equação 4.6, onde $C$ é o tamanho do conteúdo e $T$ é o tempo decorrido pra baixar o conteúdo.

$$V_m = \frac{C}{T}$$

**Equação 4.6: Taxa Média de *Download***

A dificuldade em calcular essa métrica com precisão encontra-se no fato de o par entrar no enxame e demorar até ser contatado por um dos agentes monitores, ou, caso ele esteja protegido por *firewall*, iniciar a conexão remota com o Agente. Esse tempo entre o par entrar no enxame e ter seu primeiro contato com o monitor não é capturado.

### 4.8.12 Tempo de Vida e Disponibilidade

Essa métrica indica o tempo em que um enxame pode ser considerado útil ou disponível. Essa métrica está presente no trabalho de Pouwelse [41] e consiste em verificar a quantidade de semeadores e sugadores em um enxame ao longo do tempo. Uma vez que existam poucos sugadores e nenhum semeador, de forma que exista um conjunto de peças que não estejam mais disponíveis no enxame, pode-se dizer que o enxame está indisponível.

Como já citado pela métrica de Distribuição de Peças na Subseção 4.8.5, pode-se dizer que o enxame ficou indisponível se alguma das peças desaparecerem do enxame. Essa tendência pode ser vista com um gráfico que mostra a quantidade de ocorrências da peça mais rara ao longo do tempo, pois se tivermos tendência descendente, o enxame poderá estar comprometido.

### 4.8.13 Conteúdo

As informações referentes ao conteúdo, como tamanho do conteúdo, tamanho da peça, quantidade de peças, tamanho do bloco e quantidade de blocos podem ser obtidas diretamente no arquivo de metadados *(torrent*), pois essas informações encontram-se no arquivo segundo a especificação [44].

### 4.8.14 Agente de Cliente

O agente de cliente utilizado pelos usuários pode ser obtido através da mensagem de *handshake* enviada no momento em que a conexão com o par é estabelecida. Oficialmente [69], o campo *peer id* indica qual o cliente utilizado baseado em convenções.

### 4.8.15 Classificação de *Torrents*

Através dos dados contidos no repositório UTR é possível apresentar classificações dos *torrents* obtidos pelo *Crawler*. Essa classificação é realizada através do tipo de conteúdo (classificado pela comunidade). Com essa informação, verifica-se a quantidade de *torrents* existentes para cada tipo de conteúdo. Além disso, é possível identificar os nomes dos *torrents* pertencentes a cada tipo.

### 4.8.16 Contagem dos *Torrents*

A contabilização do número de *torrents* por comunidade, por tipo, ou global identifica o crescimento de uma comunidade específica, dos tipos compartilhados pelos usuários ou do universo *torrent* como um todo. Essa contagem pode ser sobre conteúdos de forma mais

restritiva, selecionando-se através de atributos, como quantidade de *torrents* que possuem algum nome ou tamanho pré-determinado.

## 4.9 Interação

Os componentes da arquitetura possuem interações entre si para configuração, armazenamento de informações e obtenção de dados como descrito a seguir. O subsistema *Torrent Crawler* interage com a *Interface*, de onde recebe as configurações para funcionamento, e com o repositório universal de *torrents* (UTR), no qual armazena os arquivos de metadados encontrados. O recebimento das configurações é realizado através de XML, como exemplificado na Figura 4.4. A interação com o repositório se dá via conectores de banco de dados.

```
<config>
    <sites>
        <site>
            <url>http://www.test.com</url>
            <user>monitor</user>
            <pass>1234</pass>
            <updateInterval>10</updateInterval>
        </site>
        <site>
            <url>http://www.torrents.com</url>
            <updateInterval>20</updateInterval>
        </site>
    </sites>
    <filters>
        <filter>
            <string>search</string>
            <type>movie</type>
            <size> +2GB </size>
        </filter>
        <filter>
            <string>search2</string>
            <type></type>
            <size> +500MB -800MB </size>
        </filter>
    </filters>
    <repository>
        <conector>address:port</connector>
        <schema>user schema</schema>
        <password>pass</password>
        <tableName>name<tableName>
    </repository>
</config>
```

**Figura 4.4: XML de Configuração do *Crawler***

O repositório universal de *torrents* (UTR) interage com três componentes da arquitetura. O primeiro é o subsistema *Torrent Crawler*, de onde recebe os arquivos de metadados para armazenamento, juntamente com as informações contidas dentro dos arquivos. O segundo é a *Interface*, a qual faz a leitura dos arquivos de metadados que serão utilizados para encontrar os enxames que se deseja monitorar e para calcular métricas a

respeito da presença destes nas comunidades. Por último, o UTR recebe atualizações do subsistema *Torrent Monitor* que indica se um enxame está inacessível.

O repositório *Monitored Torrents* (MT) interage com a *Interface* e com o *Torrent Monitor*. A interação com a *Interface* é utilizada para alimentação do repositório com os arquivos de metadados que são utilizados pelo *Torrent Monitor* para encontrar enxames. A interação com o *Monitor* se dá em dois momentos. O primeiro deles para buscar os arquivos de metadados e, por último, para atualizar, periodicamente, os campos do repositório que indicam a última tentativa de contato com o enxame e o último sucesso no contato.

O repositório *Swarm Information Base* interage com dois componentes da arquitetura. O primeiro deles é o *Torrent Monitor*, que usa o repositório para armazenar os dados coletados dos enxames de forma a facilitar o processo de geração de gráficos, mantendo as informações estruturadas com um histórico de eventos. Por fim, o repositório interage com a *Interface* para fornecer as informações que são utilizadas para gerar gráficos e mostrar os dados sobre comportamento coletado pelo *Monitor*.

O *Torrent Monitor* interage com quatro componentes da arquitetura, sendo os três repositórios e a *Interface*. A interação com o repositório UTR acontece para atualizar o campo que indica quando um enxame é considerado indisponível e não será mais monitorado. O repositório de *torrents* monitorados (MT) é contatado em dois momentos, como já descrito. O terceiro repositório, o SIB, é utilizado para armazenar a informação coletada a partir do monitoramento dos enxames. Por fim, a interação com a *Interface* ocorre para configurar o comportamento do *Monitor* utilizando XML, como pode ser visto na Figura 4.5.

Por fim, a *Interface* é responsável pelo contato do usuário com os outros componentes da arquitetura, tornando-se um centralizador, pois comunica-se com todos os componentes. A interação com os subsistemas *Torrent Crawler* e *Torrent Monitor* é utilizada para configurá-los, como exemplo de configuração visto nos dois exemplos de XML mostrados acima. A interação com os repositórios UTR e MT é utilizada para obter os arquivos de metadados do primeiro e colocar no segundo, realizando a seleção baseada em atributos presentes no arquivo, além de utilizar o UTR para cálculo de métricas sobre os arquivos *torrents*. Por fim, a interação com repositório SIB é utilizada para obter os dados coletados a partir dos enxames monitorados para gerar gráficos.

```
<config>
    <abrangency>60%</abrangency>
    <monitorSeeder>yes</monitorSeeder>
    <monitorLeecher>yes</monitorLeecher>
    <updateTrackerInterval>3m</updateTrackerInterval>
    <monitorSwarmSize>yes</monitorSwarmSize>
    <firewallProtectedPeers>
        <monitor>yes</monitor>
        <abrangency>30%</abrangency>
        <sibylsPercentage>8</sibylsPercentage>
    </firewallProtectedPeers>
    <strategy>
        <type>handshake</type>
        <interval>20m</interval>
    </strategy>
    <trackerFilter>
        <locality>
            <country>Brazil</country>
            <country>USA</country>
        </locality>
    </trackerFilter>
    <swarmFilter>
        <size>+100 -2000</size>
        <leechers>+70</leechers>
        <seeders>+10 -500</seeders>
    </swarmFilter>
    <peerFilter>
        <locality>
            <country>Brazil</country>
            <country>Australia</country>
        </locality>
        <clients>
            <agent>Vuze</agent>
            <agent>uTorrent</agent>
        </clients>
        <extensions>
            <extension>PEX</extension>
            <extension>crypt</extension>
        </extensions>
    </peerFilter>
    <availability>
        <retries>10</retries>
        <interval>1 day</interval>
        <conditions>
            <seeders>+0</seeders>
            <tracker>yes</tracker>
            <leechers>+2</leechers>
        </conditions>
    </availability>
    <repositories>
        <UTR>...</UTR>
        <MT>...</MT>
        <SIB>...</SIB>
    </repositories>
</config>
```

**Figura 4.5: XML de Configuração do *Monitor***

# 5 PROTÓTIPO

O desenvolvimento do protótipo de teste foi realizado utilizando a Java BitTorrent API versão 1.1 [64]. Essa API foi criada por Baptiste Dubuis com o objetivo de permitir o desenvolvimento de aplicações que executam sobre o protocolo BitTorrent. A escolha dessa API foi influenciada pelos seguintes aspectos: (i) documentação; (ii) código simples, sem interface gráfica e sem necessidade de bibliotecas externas; (iii) utiliza linguagem popular e multiplataforma; (iv) pode ser executado em linha de comando; (v) apresenta todas as funcionalidades necessárias para desenvolvimento de um monitor.

Dos aspectos citados acima, a documentação da API está disponibilizada em Javadoc, comentando as funcionalidades das classes e seus métodos. Além disso, vários exemplos de utilização da API são providos, facilitando o desenvolvimento. Em relação ao segundo aspecto, o código está segmentado, onde cada classe representa uma funcionalidade. Por exemplo, existe uma classe que trata apenas do envio de mensagens e outra apenas do recebimento. Nesta mesma direção, uma classe trata exclusivamente do contato com o rastreador. Isso faz com que a adaptação de um novo código seja direta, sem a necessidade de grandes modificações no código da API. O terceiro aspecto levantado deve-se a utilização da linguagem Java, com execução multiplataforma com máquinas virtuais disponíveis para vários sistemas operacionais. O quarto aspecto, execução em linha de comando, permite que execuções de experimentos sejam automatizadas, sem a necessidade de intervenção manual entre experimentos. Por fim, a API apresenta todos os tratamentos de mensagens, conexão e codificação necessários para desenvolvimento de um protótipo que interprete as mensagens enviadas pelos pares e pelo rastreador.

O objetivo deste protótipo é avaliar o impacto, nos pares, da estratégia de monitoramento proposta neste trabalho, além do consumo de banda do monitor. Para este fim, foram criados os dois pacotes de classes, como explicados a seguir. O primeiro deles, nomeado *torrentU.logger* contempla as classes utilizadas para registrar os eventos monitorados de forma estruturada. O segundo pacote, denominado *torrentU*, contempla as classes utilizadas para contatar os pares e rastreadores dos enxames. Devido a simplicidade das nove classes utilizadas para registro de eventos e por elas não conterem lógica de protocolo ou monitoramento, elas não serão exploradas.

No pacote *torrentU*, foram criadas as seguintes classes, que representam as atividades de monitoramento:

- **GlobalManager**: classe que dispara a execução dos monitores. Apenas uma instância desta classe é criada, pois ela é responsável por criar múltiplas instâncias da classe que coordena o monitoramento de um enxame e também por controlar o fim da monitoração, desativando os agentes. Essa classe recebe como parâmetro de entrada as configurações de cobertura, intervalo de contato ao rastreador, se semeadores devem ser monitorados e os arquivos de metadados utilizados para encontrar os enxames. O fluxo de trabalho que compõe essa classe pode ser visualizado na Figura 5.1;

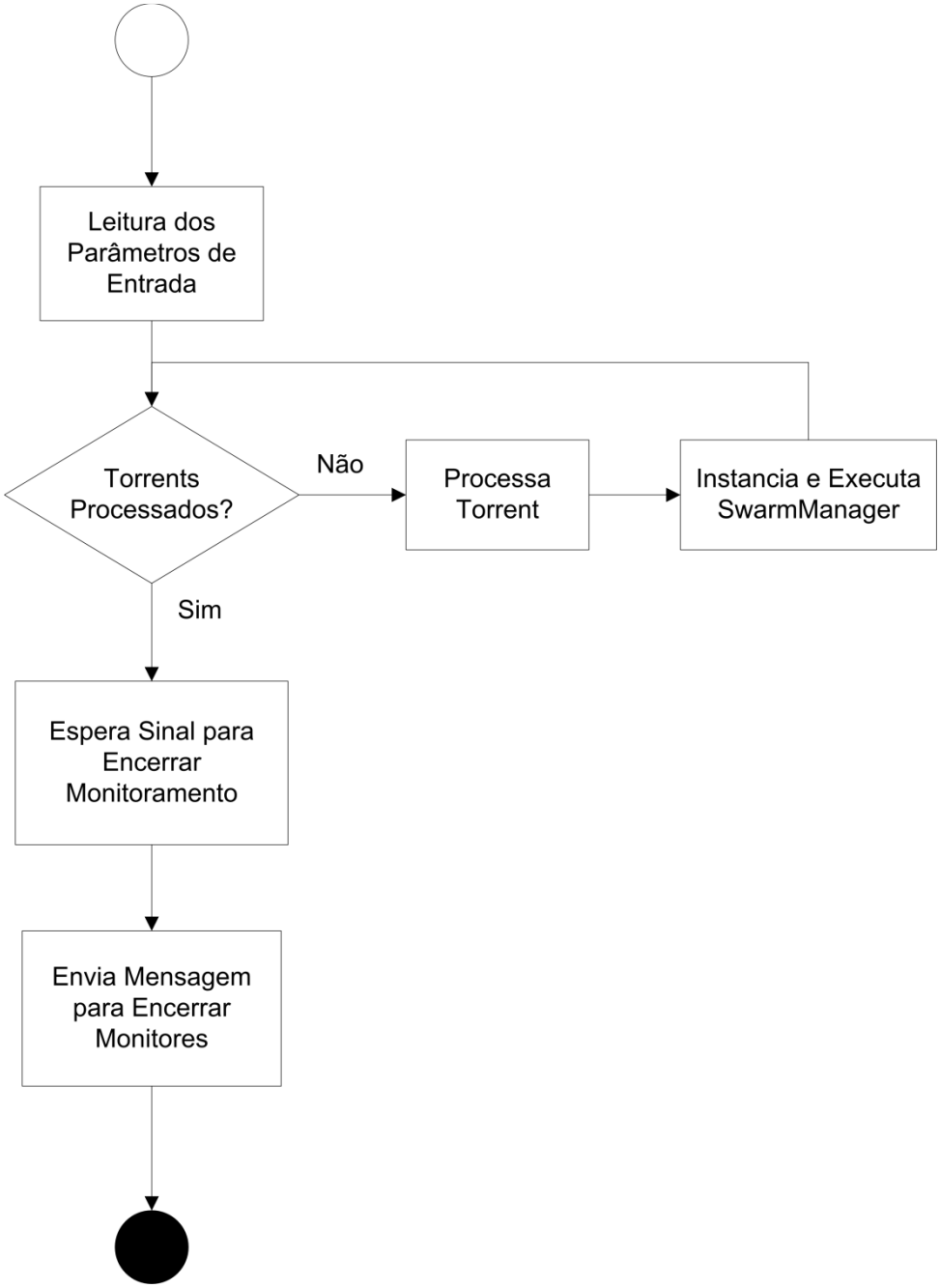


**Figura 5.1: Global Manager**

- **SwarmManager**: classe responsável por gerenciar o monitoramento de um enxame. Múltiplas instâncias dessa classe são criadas pelo GlobalManager, uma para cada enxame. Além disso, a classe é responsável pela criação das instâncias das classes que monitoram cada par e o rastreador, sendo utilizada como ponto central de comunicação entre elas. O fluxo de trabalho desta classe pode ser visualizado na Figura 5.2;

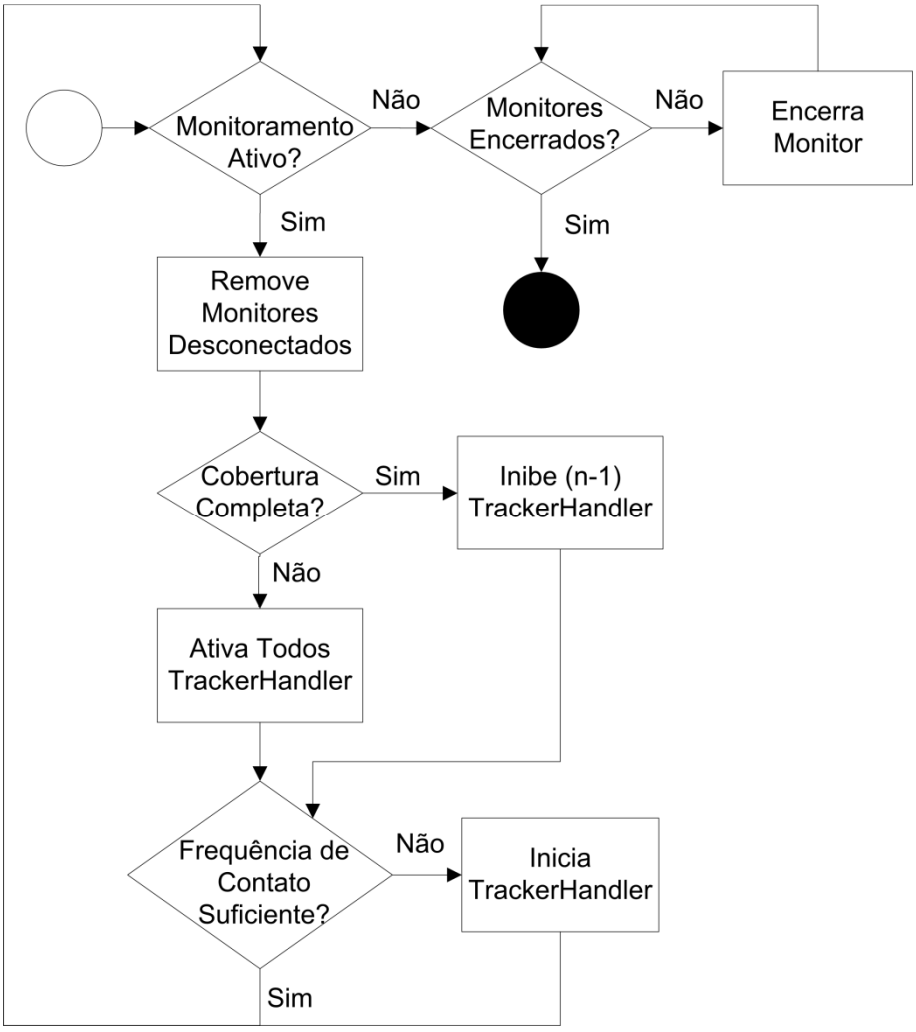


**Figura 5.2: SwarmManager**

- **TrackerHandler**: classe responsável pelo contato com o rastreador para obtenção de pares e acompanhamento do tamanho do enxame. Múltiplas instâncias dessa classe são criadas por um mesmo SwarmManager caso o intervalo entre os anúncios feitos ao rastreador seja menor do que o rastreador restringe. Por exemplo, se o rastreador permite, no máximo, um contato a cada quatro minutos e o intervalo entre os anúncios é de dois minutos, duas instâncias dessa classe são criadas na tentativa de evitar bloqueios por excesso de anúncio (idealmente deve-

se usar IPs diferentes para isso). O fluxo de trabalho dessa classe pode ser visualizado na Figura 5.3;

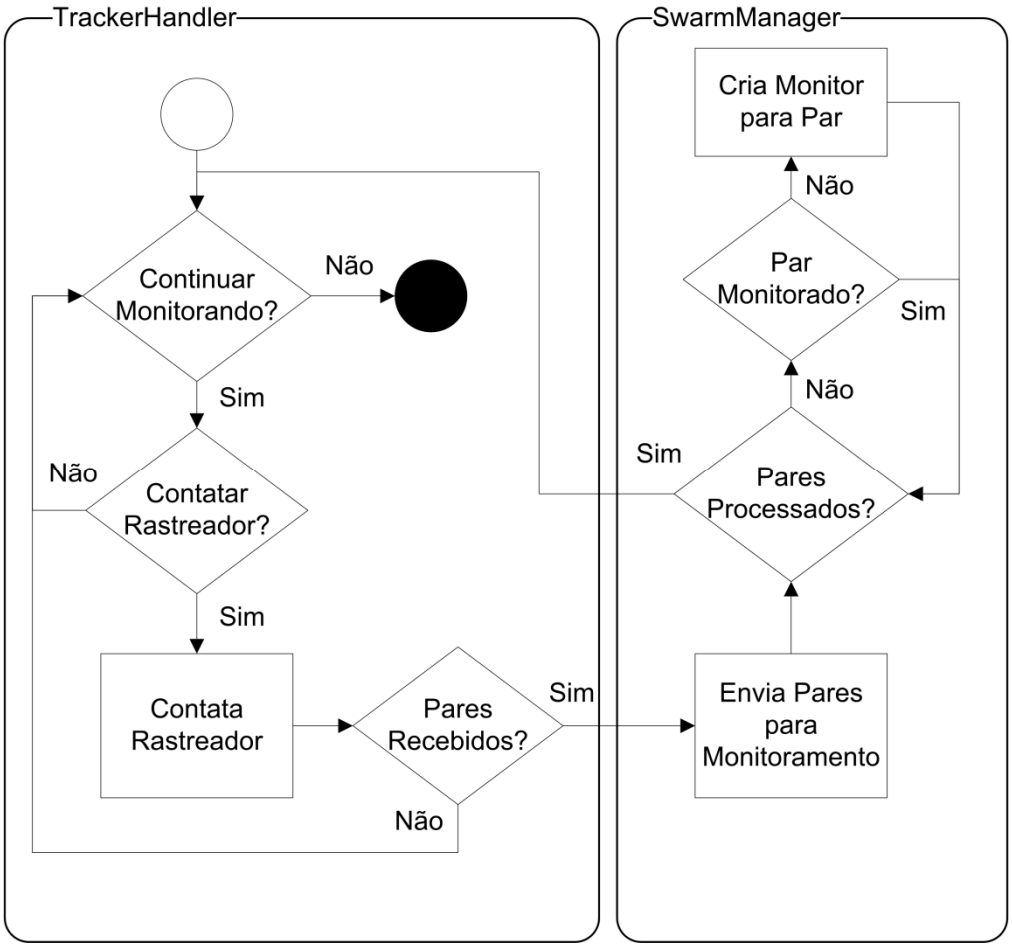


**Figura 5.3: TrackerHandler**

- **ActivePeerMonitor**: classe responsável por contatar um par e acompanhá-lo até sua finalização. Para cada par monitorado uma instância desta classe é criada. Ela recebe todas as mensagens enviadas pelo par monitorado e as armazena segundo a estrutura de dados definida no pacote *torrentU.logger*. O fluxo de trabalho desta classe pode ser visualizado na Figura 5.4;
- **TorrentuPeer**: classe que representa um par monitorado do enxame. Essa classe não contém lógica própria, pois é utilizada para armazenamento de dados a respeito de um par.

Com exceção do *TorrentUPeer*, as outras classes são *Threads* que comunicam-se utilizando os métodos das outras classes através das instâncias passadas por parâmetro. Por exemplo, a classe *ActivePeerMonitor* comunica-se com a *SwarmManager* através da instância passada no construtor da *ActivePeerMonitor*. Apesar de o código estar projetado para execução em uma única máquina, o modelo de *Threads* foi criado para facilitar a

portabilidade do código para um ambiente distribuído, onde cada elemento pode estar executando em uma máquina distinta. Como forma de simplificar essa portabilidade, uma ferramenta como o Horb [65] pode ser utilizada, deixando a comunicação entre instâncias, executando em máquinas diferentes, sob responsabilidade da ferramenta.

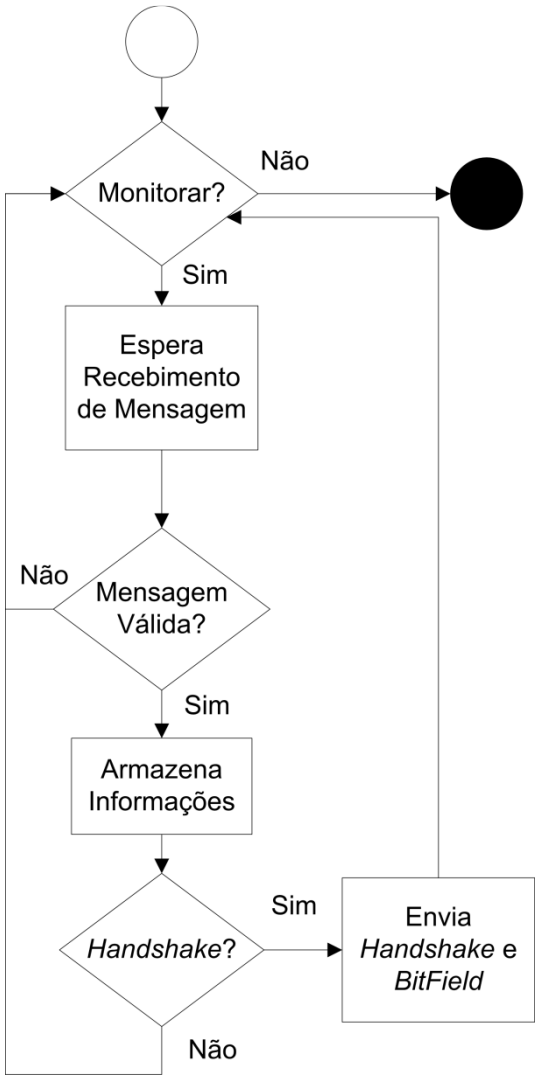


**Figura 5.4: ActivePeerMonitor**

Deste conjunto de classes criados para monitorar um enxame, as classes citadas a seguir são extensões da biblioteca JBitTorrent. A classe TrackerHandler, responsável por contatar o rastreador, é uma especialização da classe PeerUpdater da API, modificando a forma com que as respostas do rastreador são tratadas. A classe TorrentuPeer, que representa um par no enxame, é uma extensão da casse Peer da API, modificada para adicionar métodos de controle de conexão. Por fim, a classe responsável pelo monitoramento de um par, denominada ActivePeerMonitor, é um extensão da classe DownloadTask da API. A principal função desenvolvida nesta classe é o tratamento das mensagens recebidas de um par.

A Figura 5.5 ilustra o relacionamento entre as classes. Pode-se perceber a relação 1-N entre a classe GlobalManager e a SwarmManager. Da mesma forma, essa relação é encontrada entre a classe SwarmManager e as classes TrackerHandler, TorrentuPeer e ActivePeerMonitor.


**Figura 5.5: UML das Classes**

## 5.1 Desafios de Implantação

Ao implantar um sistema de monitoramento em rede real, existe uma série de desafios que precisam ser vencidos. O objetivo desta seção é listar esses desafios de forma a

documentar as principais dificuldades encontradas e que podem atormentar outros pesquisadores. Separa-se as dificuldades em dois grupos.

O primeiro grupo de dificuldades refere-se as diferentes implementações que estão disponíveis para utilização. Isso pode fazer com que hajam desvios do protocolo original, adicionando funcionalidades como o PEX (*Peer Exchange Protocol*). Deve-se tomar cuidado com o tratamento dessas mensagens diferenciadas, pois elas não estão previstas na versão oficial do protocolo. Atenta-se que existem diferentes implementações de rastreadores e estes podem enviar mensagens fora do padrão.

O segundo grupo de dificuldades diz respeito as limitações do ambiente. Aqui se encontram as limitações de rede, de espaço de armazenamento e de configuração de filtros (*firewalls*). As limitações de rede dizem respeito à capacidade de *download* e *upload* necessárias para monitorar todos os enxames desejados. Se a quantidade de banda disponível no enlace não for suficiente para monitorar os enxames, o monitoramento será afetado. Outra característica importante é a quantidade máxima de conexões simultâneas que podem ser estabelecidas, pois alguns provedores de Internet limitam essas conexões, não permitindo que novos fluxos sejam iniciados. Essa característica pode inviabilizar o monitoramento.

As limitações de espaço são visíveis ao longo do tempo, quando se necessita que o monitoramento seja feito por um período mais elevado. Nesse momento, a capacidade de armazenamento disponível torna-se crítica. Por isso, é necessário observar essa necessidade quando se deseja monitorar grande quantidade de enxames por um longo período. Aconselha-se armazenar apenas as informações necessárias para o estudo em pauta, evitando sobre carregar os dispositivos de armazenamento.

Por fim, as limitações de filtro (*firewall*) ocorrem, principalmente, em Universidades e empresas. Nesses locais existem políticas para o uso da banda e o protocolo *BitTorrent*, assim como a utilização de portas altas (acima da 1024), pode ser barrado. Devido a esses filtros, é possível que os monitores não consigam observar nenhum enxame, tornando-se necessário desativar os *firewalls*, abrir exceções ou trocar a rede utilizada.

# 6 AVALIAÇÃO DO MODELO

Este capítulo avalia aspectos do modelo proposto. Primeiramente, a Seção 6.1 apresenta uma análise teórica, utilizando fórmulas de custo, sobre o impacto que os subsistemas *Torrent Crawler* e *Torrent Monitor* causam na rede e nos componentes monitorados. A seguir, a Seção 0 apresenta os aspectos do protótipo desenvolvido para análise de impacto. A Seção 6.2 descreve o ambiente de testes utilizado. Por fim, a Seção 6.3 descreve a análise experimental conduzida para mostrar, na prática, o consumo do *Crawler* (baseado em dados de *torrents*) e do *Monitor* (baseado em experimentos conduzidos em laboratório).

## 6.1 Análise Teórica

A análise teórica tem o objetivo de apresentar os custos do *Torrent Crawler* e do *Torrent Monitor* em operação. Os custos são analisados através de fórmulas que demonstram o impacto causado. A análise sobre o *Crawler* expressa os impactos no enlace de rede para baixar os arquivos de metadados e no armazenamento necessário para guardá-los, enquanto a análise sobre o *Monitor* foca no consumo do enlace de rede para o agente de monitoramento, no aumento percentual no uso de memória e conexões dos pares monitorados.

### 6.1.1 *Torrent Crawler*

Para analisar o efeito dos arquivos *torrent* no custo do *Crawler*, deve-se atentar para a estrutura dos arquivos, descritos da seguinte forma. O arquivo de metadados é composto por um dicionário que descreve os arquivos que fazem parte do conteúdo (com seu caminho e tamanho), o tamanho da peça que divide o conteúdo, os *hash code* das peças (20 bytes cada), a lista de rastreadores e alguns campos opcionais (comentários, data de criação, criador, codificação). Salienta-se que, deste conjunto de elementos, o que realmente influencia no tamanho do arquivo *torrent* é a quantidade de peças e número de arquivos, pois isto faz com que o número de *hashes* e descritores seja maior.

Dessa forma, o custo de rede para a obtenção dos arquivos de metadados de uma comunidade é representado pela Equação 6.1, onde $C$ é o custo total para carregar os arquivos de metadados, $N$ é o número de *torrents*, $CS_i$ é o tamanho do conteúdo $i$, $PS_i$ é o tamanho da peça do conteúdo $i$, $TFS$ é o tamanho dos campos fixos do *torrent* (rastreador, data, criador, etc), $FN_i$ é o tamanho que a representação dos arquivos que compõe o conteúdo $i$ apresentam e $CC$ é o custo fixo de estabelecer uma conexão e requisitar o *torrent* ao servidor.

$$C = \sum_{i=1}^{N} CC + \left(\frac{CS_i}{PS_i} * 20\right) + TFS + FN_i$$

**Equação 6.1: Custo de Rede para *Crawler***

Neste caso, tem-se $CC$ representado pela Equação 6.2, onde $Syn$ é o custo de abertura de conexão, $REQ$ é o custo da requisição do *torrent* para o servidor que o hospeda e $Fin$ é o custo para fechar a conexão. O custo das variáveis $\frac{CS_i}{PS_i}$ e $FN_i$ são as que mudam entre os arquivos, pois os outros custos tendem a serem fixos. Dessa forma, o tamanho dos arquivos de metadados é influenciado pela quantidade de arquivos do conteúdo e pela quantidade de peças.

$$CC = Syn + REQ + Fin$$

**Equação 6.2: Custo para Conexão**

Embora a fórmula contemple o custo para baixar os *torrents*, não é previsto o custo com o carregamento das páginas das comunidades que contém os *torrents*. Esse custo varia com o estilo e tipo de páginas que as comunidades dispõem, assim como o formato de busca (múltiplas páginas, página única com todos os *torrents*).

Por fim, o custo para armazenar os *torrents* localmente é descrito pela Equação 6.3, onde os elementos são os mesmos descritos na fórmula anterior. Note que a principal diferença é a remoção dos custos referentes a conexão com os servidores das comunidades, restando apenas as variáveis que influenciam no tamanho dos arquivos de metadados.

$$C = \sum_{i=1}^{N} \left(\frac{CS_i}{PS_i} * 20\right) + TFS + FN_i$$

**Equação 6.3: Custo para Armazenamento dos *Torrents***

### *6.1.2 Torrent Monitor*

Esta subseção tem o objetivo de ilustrar o impacto que o *Torrent Monitor* impõe no ambiente para realizar o monitoramento. São apresentadas fórmulas contendo o custo teórico das estratégias utilizadas. Para isso, considera-se os elementos (i) par, (ii) rastreador e (iii) monitor. Além disso, consideram-se os recursos de redes utilizados por estes elementos, devido a monitoração, para verificar o impacto em cada enlace. Estimativas para consumo de memória e conexões também são fornecidas.

A partir das duas estratégias de monitoramento de pares apresentadas na Seção 4.6, tem-se, para a estratégia apresentada por Pouwelse [41], o custo de rede para o monitor

representado pela Equação 6.4, onde $C$ é o custo total para o monitoramento, $HS + BF$ é o custo para conectar em um par de um enxame, sendo o primeiro o custo para o *handshake* e o segundo o custo para envio do *bitfield*. $P$ indica o número de pares monitorados e $I$ é o intervalo em que os pares são contatados.

$$C = \frac{(HS + BF) * P}{I}$$

**Equação 6.4: Custo para Monitoramento usando *bitfield***

O custo de rede para monitorar pares utilizando a estratégia proposta neste trabalho dá-se pela Equação 6.5, onde $L$ é o número de sugadores monitorados, $V_i$ é a taxa de *download* do sugador $i$, $H$ é o custo de envio de uma mensagem do tipo *HAVE*, $PS_i$ é o tamanho da peça do enxame que o sugador $i$ participa. $KA$ é o custo de uma mensagem de *Keep-Alive*, $S$ representa a quantidade de semeadores monitorados e $120$ representa o tempo entre os envios de uma mensagem do tipo *Keep-Alive*, usualmente 120 segundos.

$$C = \left(\sum_{i=0}^{L} \frac{V_i \times H}{PS_i}\right) + \left(\frac{KA \times S}{120}\right)$$

**Equação 6.5: Custo para Monitoramento Permanecendo Conectado**

De maneira similar, o custo de rede causado ao **par** pelo monitoramento, utilizando a estratégia de Pouwelse, pode ser expresso pela Equação 6.6, onde os elementos são os mesmos apresentados na Equação 6.4.

$$C = \frac{(HS + BF)}{I}$$

**Equação 6.6: Custo ao Par Monitorado Utilizando *bitfield***

O custo de monitoramento causado ao par utilizando a estratégia proposta neste trabalho é representado pela Equação 6.7, caso o par seja um sugador e pela Equação 6.8, caso o par seja um semeador.

$$C = \frac{V_i \times H}{PS_i}$$

**Equação 6.7: Custo ao Par Monitorado Permanecendo Conectado, caso sugador**

$$C = \frac{KA}{120}$$

**Equação 6.8: Custo ao Par Monitorado Permanecendo Conectado, caso semeador**

Para finalizar a apresentação dos custos de rede, cita-se o custo causado ao rastreador pelo monitoramento devido à obtenção de pares e consulta do tamanho do enxame. Este custo

pode ser definido pela Equação 6.9, onde $C$ é o custo total para contatar o rastreador, $A$ é o custo de um anúncio e $f$ é a freqüência com que os anúncios ocorrem.

$$C = A \times f$$

**Equação 6.9: Custo para o Rastreador**

Assume-se que o monitor apresenta custo de memória e conexões aos pares monitorados não significantes. Isso ocorre porque, nas duas estratégias apresentadas, apenas uma conexão é aberta com um par em dado momento. Sendo assim, se um par tem capacidade de conectar-se com outros cinqüenta, o monitor utiliza 2% da capacidade de abertura de conexões do par. Da mesma forma, dadas as estruturas de dados necessárias para o par armazenar em memória os dados referentes a esse vizinho (o monitor), o aumento no consumo de memória desta estrutura é contabilizado pela Equação 6.10, onde $C$ é o custo de memória total, $C_{n-1}$ é o custo anterior ao monitor conectar no par e $Ne$ é o número de vizinhos ao qual o par está conectado.

$$C = C_{n-1} \times \left(\frac{1}{Ne} + 1\right)$$

**Equação 6.10: Custo para o Rastreador**

## 6.2 Ambiente de Testes

Os experimentos apresentados neste trabalho foram executados no laboratório do Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da Unisinos, apresentado na Figura 6.1. Dentro deste laboratório, foi utilizado um *cluster* composto de nove máquinas. Quatro delas possuindo 2 processadores Intel Xeon 2.4Ghz, 2 GB de RAM, disco IDE 80 GB, placa de rede Gigabit. As outras máquinas possuem 2 processadores Intel Xeon 2.8Ghz, 2 GB de RAM, disco SCSI de 36 GB e placa de rede Gigabit. Todas possuem o sistema operacional Linux, *kernel* versão 2.6, com a distribuição Gentoo 9. Todas as máquinas foram configuradas com múltiplas interfaces virtuais para possuírem múltiplos IPs. Dessa forma, cada máquina tinha mais de 200 IPs configurados.

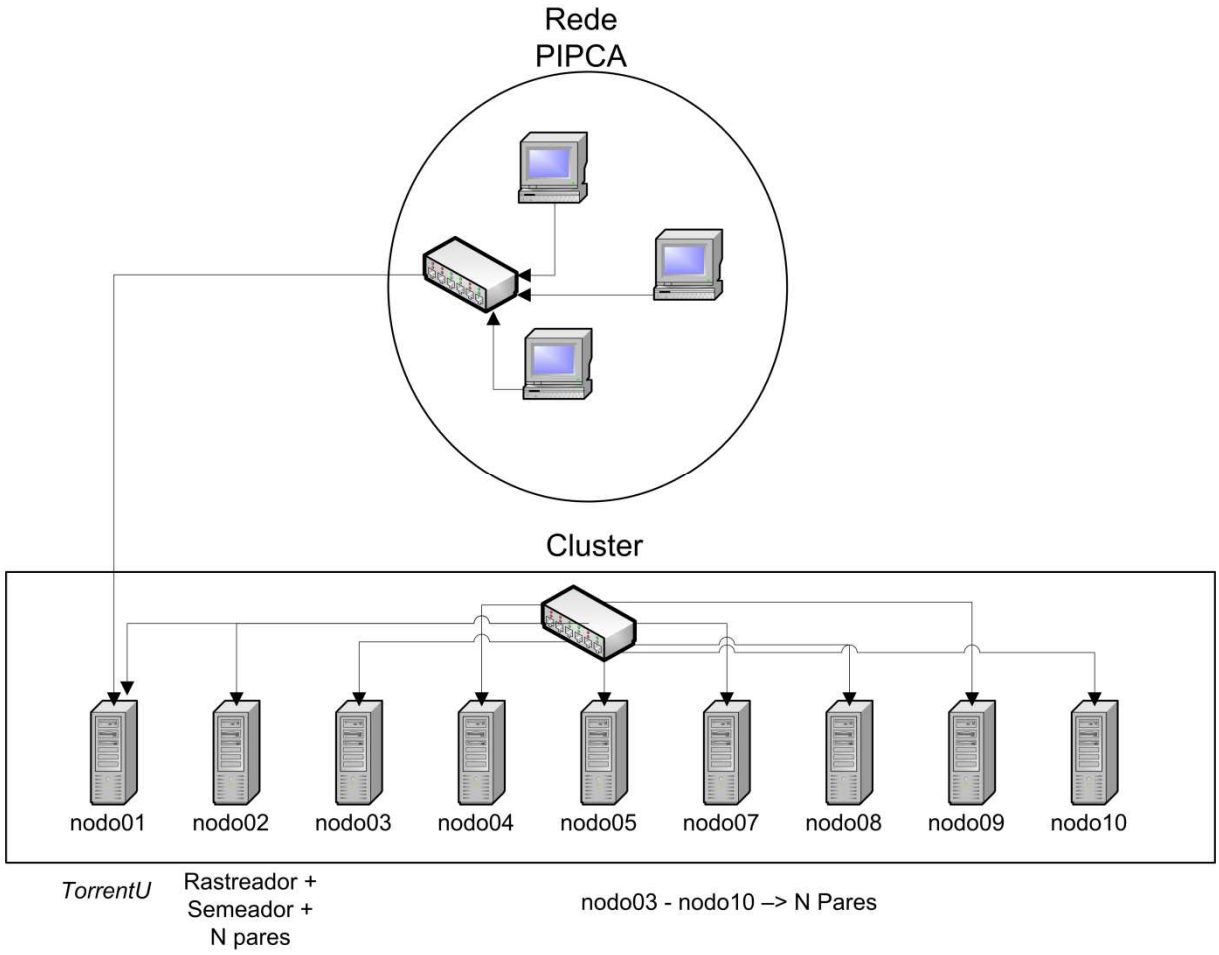


**Figura 6.1: Ambiente de Testes**

Das nove máquinas, uma foi reservada para execução exclusiva do *TorrentU*. Assim, o tráfego que passa por essa máquina é quase exclusivamente causado pelo monitor. Dessa forma, para controlar o tráfego passante pela interface de rede, utilizou-se o programa *tcpdump*. Para armazenar apenas os dados referentes ao monitor, os pacotes recebidos foram filtrados através das portas e IPs que são utilizados nos pares que executam agentes de usuários que conectam no enxame.

Em cada uma das oito máquinas restantes, múltiplas instâncias do agente foram criadas para simular um enxame. O agente de usuário utilizado no teste é o *Mainline* versão 4.4, escrito em *Python*. O rastreador utilizado é o BNBT [66]. Apenas uma instância do rastreador é criada em uma das máquinas, enquanto mais de trinta instâncias de agente são criadas por máquina.

Os pares foram configurados com 1Mbps de capacidade de *download*, 256kbps de capacidade de *upload*. Além disso, a chegada dos pares ao enxame foi realizada com uma distribuição exponencial. No início do enxame foi configurado com semeador com capacidade igual ao dos pares participantes. Esse semeador participava do enxame do início

ao fim dos experimentos. Os pares abandonavam o enxame assim que o *sharing ratio* atingisse a proporção de um.

## 6.3 Análise Experimental

A análise experimental tem o objetivo de identificar aspectos referentes a operação do *Torrent Crawler* e do *Torrent Monitor*. No caso do *Crawler*, é dado foco em arquivos de metadados (*torrents*) encontrados em comunidades. Dessa forma, um levantamento é feito para identificar características dos arquivos e conteúdos compartilhados. No caso do *Monitor*, é dado foco no consumo de banda e no impacto causado no enxame. As subseções seguintes tratam destas duas análises experimentais.

### 6.3.1 *Torrent Crawler*

Como forma de verificar o tamanho dos arquivos de metadados, das peças e do conteúdo compartilhado, foi carregado, da comunidade Mininova [49], 10021 *torrents*. Esses arquivos foram mensurados e analisados, resultando nos dados apresentados na Tabela 3. Note que "Armazenamento" é o custo total para guardar os dez mil *torrents* carregados. Analisando algumas comunidades BitTorrent e assumindo que as médias calculadas para os arquivos de metadados encontradas na comunidade Mininova se mantém verdadeiras, tem-se a necessidade de armazenamento expressa na Tabela 4 para guardar os *torrents* das referidas comunidades. Destaca-se que não se tem previsão de quantos *torrents* repetidos entre as comunidades podem existir, o que poderia diminuir a necessidade de armazenamento para guardar todos os *torrents*. Obviamente, a necessidade de utilização de banda para baixar o conteúdo encontrado nas comunidades é maior do que expresso nas tabelas, pois envolve o carregamento das páginas HTML dos sítios que hospedam os dados.

| | **Tamanho do Conteúdo** | **Tamanho da Peça** |
|---|---|---|
| **Máximo** | 184 GB | 8 MB |
| **Mínimo** | 44 bytes | 16 KB |
| **Média** | 1,08 GB | 909 KB |
| **Desvio Padrão** | 3,76 G | 1,16 M |
| | | |
| **Armazenamento** | 221 MB | |

**Tabela 3: Dados dos *Torrents***

| | ***Torrents*** | **Armazenamento** |
|---|---|---|
| **The Pirate Bay [52]** | 1250368 | 27 GB |
| **Mininova [49]** | 1020457 | 22 GB |

| | | |
|---|---|---|
| **Btjunkie [51]** | 1500000 | 32 GB |
| **Torrentz [68]** | 3804861 | 82 GB |
| **Isohunt [67]** | 1759297 | 38 GB |

**Tabela 4: Comunidades**

### 6.3.2 *Torrent Monitor*

A experimentação realizada para avaliação do *Torrent Monitor* tem o objetivo de mostrar o custo de banda para monitoramento de um enxame e o impacto que o *Monitor* causa nos pares. Antes da execução dos experimentos, partiu-se de duas premissas: (i) o custo para monitoramento cresceria linearmente com o aumento do número de pares no enxame; e (ii) não haveria impacto significativo nos tempos de *download* dos pares devido ao monitoramento.

Primeiramente, apresentam-se os resultados sobre o custo causado a rede pelo monitor. A investigação foi feita permitindo liberdade do número de pares no enxame, onde foi configurado com os seguintes valores {50, 100, 150, 200, 250}. Dado que o aumento do número de pares monitorados acarreta um aumento no recebimento de mensagens *HAVE* e *Keep-Alive*, espera-se aumento no consumo de recursos neste cenário (como demonstrado na análise teórica da Seção 6.1).

No gráfico a seguir, o eixo horizontal (abscissas) representa o número de pares presentes no enxame monitorado. O eixo vertical (ordenadas) representa a quantidade de dados, em *bytes*, geradas na rede pelo monitoramento. O gráfico da Figura 6.2 apresenta o custo de rede onde a peça que dividia o conteúdo compartilhado tinha tamanho de 1MB. Neste gráfico, percebe-se o aumento do custo de monitoramento juntamente com a presença de mais pares no enxame, ou seja, existe uma correlação positiva entre as variáveis.

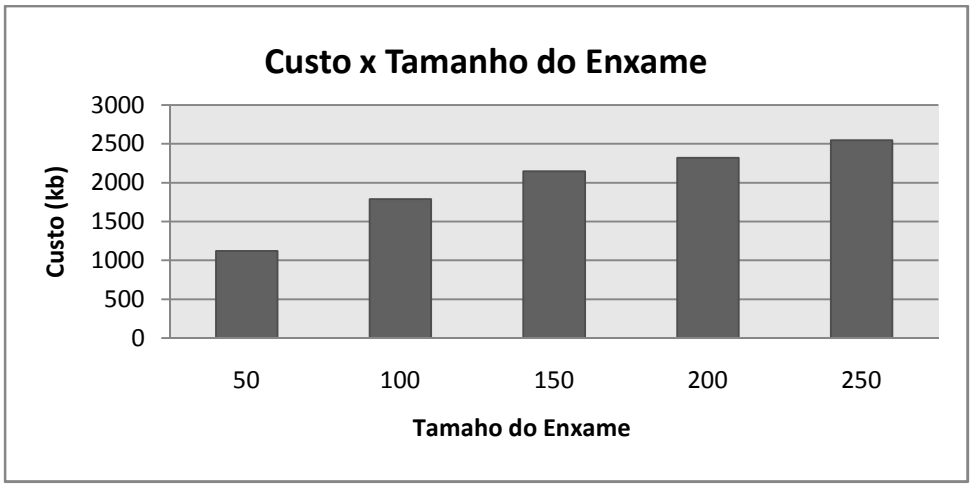


**Figura 6.2: Custo do Monitor com Peça de 1MB**

A Figura 6.3 mostra a influência que o tamanho da peça tem, quando outros parâmetros são fixos (exceção ao número de pares), no custo causado pelo monitor. Cada conjunto de barras na figura ilustra o custo de experimentos com o mesmo número de pares, mas variando-se o tamanho da peça. Note que o aumento na quantidade de peças de um conteúdo também é responsável por aumentar o custo, pois mais mensagens do tipo HAVE são recebidas pelo monitor.

Apesar de existirem oito vezes mais peças nos enxames com tamanho de peça 128kB se comparado ao de 1MB, a proporção de custo ficou em 3,6 vezes maior no enxame de 50 pares. Já no enxame de 250 pares, essa mesma proporção ficou em 2,3 vezes, demonstrando que o crescimento do custo não segue, na mesma proporção, o crescimento do número de peças (confirmações) recebidas.

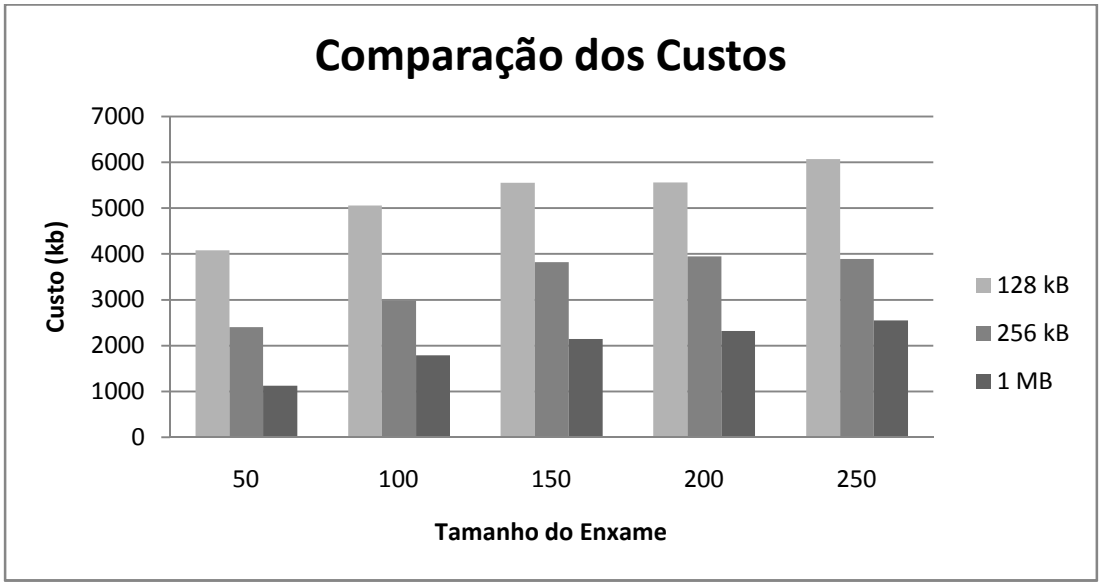


**Figura 6.3: Comparação de Custos**

Por fim, apresentam-se os resultados obtidos na análise de impacto causado pelo monitor. O objetivo desta análise é identificar se o *Torrent Monitor*, utilizando a estratégia apresentada neste trabalho, causa algum impacto para o enxame em execução a ponto de atrasá-lo (os pares demorarem mais para concluir).

Para essa análise, experimentos foram executados com 250 pares no enxame, tamanho de peça de 1MB, tamanho do conteúdo de 64MB, chegada de pares seguindo distribuição exponencial, um semeador e pares contribuindo até atingirem *sharing ratio* de um. A diferença entre os experimentos é a presença do monitor no enxame. Em um deles o monitor está presente, monitorando todos os pares e no outro o monitor não está presente.

A Figura 6.4 apresenta o impacto causado pelo monitor. Neste gráfico, o eixo horizontal (abscissas) representa o número de pares que completaram o carregamento do

conteúdo no enxame e o eixo vertical (ordenadas) representa o tempo, em minutos, em que o par completou. Destaca-se que os pares estão ordenados pelo tempo em que terminaram de carregar o conteúdo. A figura mostra que, confirmando a hipótese inicial de que o monitor não causa impacto no enxame, não existe atraso significativo nos tempos de *download* do conteúdo. Percebe-se que as linhas estão sobrepostas, indicando que os tempos são praticamente os mesmos.

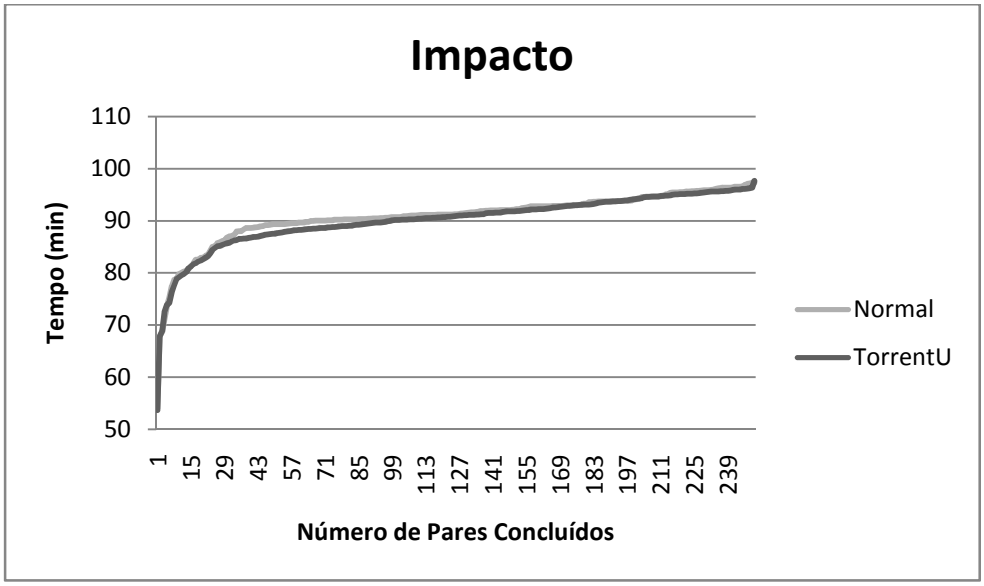


**Figura 6.4: Impacto do Monitor no Enxame**

# 7 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o *TorrentU*, uma arquitetura flexível para extração de informações sobre o universo de redes BitTorrent. A arquitetura abrangente trata aspectos referentes a obtenção e classificação dos arquivos de metadados, assim como o monitoramento de pares e rastreadores.

O Capítulo 2 apresentou o funcionamento do protocolo BitTorrent introduzindo os conceitos sobre sua operação e caracterizando informações sobre o Universo. O Capítulo 3 apresentou os trabalhos relevantes para área de monitoramento e investigação de redes BitTorrent, comparando-os segundo critérios estabelecidos. O Capítulo 4 apresentou a arquitetura flexível de monitoramento *TorrentU*, utilizada para investigar o universo de redes BitTorrent. Por fim, o Capítulo 5 apresentou avaliação teórica e experimental dos componentes da arquitetura.

As principais contribuições deste trabalho são: (i) classificação e comparação de trabalhos experimentais sobre BitTorrent; (ii) apresenta uma versão consolidada do funcionamento do protocolo BitTorrent, caracterizando informações disponíveis no Universo; (iii) aponta métodos para coleta de métricas; (iv) propõe uma arquitetura flexível para monitoramento do universo de redes BitTorrent; e (v) analisa questões referentes a custo e impacto que o *Crawler* e o *Monitor* causam a rede e sua necessidade de armazenamento.

A arquitetura proposta neste trabalho representa um avanço em relação a trabalhos anteriores [41] e [47], com flexibilidade ao se escolher o nível de profundidade (quantidade de informações obtidas junto às entidades monitoradas), de abrangência (número de redes observadas), a acurácia (atualização de dados) e custo na utilização de recursos. Além disso, a arquitetura prevê exploração de comunidades, onde são analisados os padrões de comportamento dos usuários na inserção de conteúdo, assim como o crescimento das comunidades.

Como trabalhos futuros, propõem-se a execução de experimentos de maior escala, onde enxames com mais de mil pares serão investigados. Essa execução permitiria que uma avaliação com maior nível de detalhe do impacto que o número de pares causa no custo de rede seja feita, além de investigar impactos que outros parâmetros dos enxames podem causar. Como exemplo de parâmetros cita-se o número de semeadores inicias, suas capacidades, tamanho do conteúdo e distribuição de chegada de pares.

Após avaliar mais profundamente impactos em rede controlada, serão executados experimentos em rede real, com o objetivo de identificar o comportamento de usuários de

enxames de diferentes conteúdos e comunidades, avaliando influências que sua localização pode trazer.

# Bibliografia


[1] Saroiu, S.; Gummadi, P.; Gribble, S. **A Measurement Study of Peer-to-Peer File Sharing Systems**. In Proceedings of Multimedia Computing and Networking, 2002.

[2] Izal, M.; Urvoy-Keller, G.; Biersack, E.W.; Felber, P.A.; Hamra, A. Al; Garces-Erice, L. **Dissecting BitTorrent: Five months in a torrent's lifetime.** In Proceedings of the 5th Passive and Active Measurement Workshop, Apr. 2004.

[3] Piatek, Michael; Isdal, Tomas; Anderson, Thomas; Krishnamurthy, Arvind; Venkataramani, Arun. **Do incentives build robustness in BitTorrent?** NSDI, April, 2007.

[4] Bellissimo, A.; Shenoy, P.; Levine, B. N. **Exploring the Use of BitTorrent as the Basis for a Large Trace Repository**. University of Massachusetts, Tech. Rep., 2004.

[5] Locher, Thomas; Moor, Patrick; Schmid, Stefan; Wattenhofer, Roger. **Free Riding in BitTorrent is Cheap**. Fifth Workshop on Hot Topics in Networks (HotNets-V), November, 2006.

[6] Cohen, Bram. **Incentives Build Robustness in BitTorrent**. In Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, June 2003.

[7] Madhyastha, Harsha V.; Isdal, Tomas; Piatek, Michael; Dixon, Colin; Anderson, Thomas; Krishnamurthy, Arvind; Venkataramani, Arun. **iPlane: an information plane for distributed services**. In Proceedings of the 7th symposium on Operating systems design and implementation, 2006.

[8] Guo, Lei; Chen, Songqing; Xiao, Zhen; Tan, Enhua; Ding, Xiaoning; Zhang, Xiaodong. **Measurement, analysis, and modeling of BitTorrent-like systems**. In proceedings of ACM SIGCOMM Internet Measurement Conference, (IMC'05), October, 2005.

[9] Katti, Sachin; Katabi, Dina; Blake, Charles; Kohler, Eddie; Strauss, Jacob. **MultiQ: automated detection of multiple bottleneck capacities along a path**. In proceedings of the 4th ACM SIGCOMM conference on Internet measurement, 2004.

[10] CacheLogic. http://www.cachelogic.com. Accessed in April of 2008.

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

[12] Napster. http://free.napster.com/. Accessed in April of 2008.

[13] Barbera, M.; Lombardo, A.; Schembra, G.; and Tribastone, M. **A markov model of a freerider in a bittorrent P2P network**. In IEEE Global Telecommunications Conference (GLOBECOM '05), volume 2, pages 985-989.

[14] Mansilha, R.B.; Barcellos, M.P.; Brasileiro, F.V. **TorrentLab: investigating BitTorrent through simulation and live experiments**. IEEE Symposium on Computers and Communications (ISCC'08). July 6 - 9, 2008.

[15] Konrath, M. A.; Barcellos, M. P.; Silva, J. F. da; Gaspary, L. P.; Dreher, Rafael. **Atacando um Enxame com um Bando de Mentirosos: vulnerabilidades em BitTorrent**. In: XXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2007), p. 883-896.

[16] Konrath, M. A.; Barcellos, M. P. ; Mansilha, R. B. **Attacking a Swarm with a Band of Liars: evaluating the impact of attacks on BitTorrent**. In: The Seventh IEEE International Conference on Peer-to-Peer Computing (IEEE P2P 2007).

[17] Al-Hamra, Anwar; Legout, Arnaud; Barakat, Chadi. **Understanding the Properties of the BitTorrent Overlay**. Technical Report (inria-00162088, version 1 - 12 July 2007).

[18] Qiu, Dongyu; Srikant, R. **Modeling and performance analysis of BitTorrent-like peer-to-peer networks**. SIGCOMM Comput. Commun. Rev., 2004.

[19] Ars Technica – The art of technology. http://arstechnica.com/.

[20] BigChampagne Online Media Measurement. http://www.bigchampagne.com/.

[21] Torrent Freak. http://torrentfreak.com/p2p-traffic-still-booming-071128/

[22] ipoque. http://www.ipoque.com/

[23] http://www.news.com/8301-10784_3-9920665-7.html.

[24] Sherman, Chris; Price, Gary. **The Invisible Web: Uncovering Information Sources Search Engines Can't See**. Cyberage Books, 430 pages, September 2001.

[25] Google. http://www.google.com.

[26] Bharambe, Ashwin R.; Herley, Cormac; Padmanabhan, Venkata N. **Analyzing and Improving a BitTorrent Networks Performance Mechanisms**. INFOCOM 2006.

[27] Konrath, Marlom Alves. **Estudo das vulnerabilidades da arquitetura BitTorrent, ataques e contramedidas possíveis**. 94p. 2007. Dissertação.

[28] µTorrent – The Lightweight end Efficient BitTorrent Client. http://www.utorrent.com/.

[29] Harrington, J.; Kuwanoe, C.; Zou, C. **A bittorrent-driven distributed denial-of-service attack.** In 3rd International Conference on Security and Privacy in Communication Networks (SecureComm 2007), September 2007.

[30] Sirivianos, Michael; Park, Jong H.; Chen, Rex; Yang, Xiaowei. **Free-riding in BitTorrent with the Large View Exploit**. In 6th International Workshop on Peer-to-Peer Systems (IPTPS 2007).

[31] http://arstechnica.com/news.ars/post/20080417-bittorrent-use-soars-as-mpaa-fights-on-against-p2p-sites.html.

[32] Saroiu, Stefan; Gummadi, K. P.; Dunn, Richard J.; Gribble, Steven D.; Levy, Henry M. **An Analysis of Internet Content Delivery Systems**. Proceedings of Fifth Symposium on Operating Systems Design and Implementation, 2002.

[33] Guo, Lei; Chen, Songging; Xiao, Zhen; Tan, Enhua; Ding, Xiaoning; Zhang, Xiaodong. **Measurements, analysis, and modeling of BitTorrent-like systems**. In Proceedings of the Internet Measurement Conference 2005 on Internet Measurement Conference.

[34] Cranor, C.; Johnson, T.; Spatscheck, O. **Gigascope: a stream database for network applications**. In Proceedings of ACM SIGMOD (Junho 2003).

[35] Chun, B.; Culler, D.; Roscoe, T.; Bavier, A.; Peterson, L.; Wawrzoniak, M.; Bowman, M. **PlanetLab: An Overlay Testbed for Broad-Coverage Services**. In SIGCOMM CCR, Julho 2003.

[36] Gnutella. http://www.gnutella.com/

[37] Legout, Arnaud; Urvoy-Keller, G.; Michiardi, P. **Rarest First and Choke Algorithms Are Enough**. Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement. 2006.

[38] Legout, A.; Liogkas, N.; Kohler, E.; Zhang, L. **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.

[39] Andrade, Nazareno; Mowbray, M.; Lima, A.; Wagner, G.; Ripeanu, M. **Influences on Cooperation in BitTorrent Communities**. Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems.

[40] Bellissimo, Anthony; Shenoy, Prashant; Levine, Brian N. **Exploring the Use of BitTorrent as the Basis for a Large Trace Repository**. Technical Report 04-41, Department of Computer Science, University of Massachusetts, June 2004.

[41] Pouwelse, J. A.; Garbacki, P.; Epema, D. H. J.; Sips, H. J. **The Bittorrent P2P File-Sharing System: Measurements And Analysis**. 4th Int'l Workshop on Peer-to-Peer Systems (IPTPS), 2005.

[42] Suprnova – The Universal BitTorrent Source. http://www.suprnova.com/

[43] Douceur, J. R. **The Sybil Attack**. In 1st International Workshop on Peer-to-Peer Systems, Cambridge, MA, USA, March 2002, pp. 251-260.

[44] BitTorrent Unofficial Protocol Specification. http://wiki.theory.org/BitTorrentSpecification.

[45] http://www.dl.ro/index/utils.country/lang/en

[46] IP Address Locator. http://www.geobytes.com/ipLocator.htm.

[47] Chow, K. P.; Cheng, K. Y.; Man, L. Y.; Lai, P. K. Y.; Hui, L. C. K.; Chong, C. F.; Pun, K. H.; Tsang, W. W.; Chan, H. W.; Yiu, S. M. **BTM - An Automated Rule-based BT Monitoring System for Piracy Detection**. In Second International Conference on Internet Monitoring and Protection 2007, July 2007.

[48] Iosup, A.; Garbacki, P.; Pouwelse, J.; Epema, D. **Correlating Topology and Path Characteristics of Overlay Networks and the Internet**. In 6th IEEE International Symposium on Cluster Computing and the Grid Workshops, 2006, vol. 2, 2006, p. 10.

[49] Mininova: The Ultimate BitTorrent Source! http://www.mininova.org/

[50] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. **An Introduction to Information Retrieval**. Cambridge University Press. Cambridge, England. 2008.

[51] btjunkie – The Largest BitTorrent Search Engine. http://btjunkie.org/

[52] The Pirate Bay – The World`s largest BitTorrent Tracker. http://thepiratebay.org

[53] Liang, J.; Kumar, R.; Xi, Y.; Ross, K. **Pollution in p2p file sharing systems**. In The 24th Conference on Computer Communications (INFOCOM 2005), vol. 2, Miami, FL, USA, March 2005, pp. 1174-1185.

[54] Convivea – Bit Che. http://www.convivea.com/product.php?id=2

[55] Adar, E; Huberman, B. **Free riding on gnutella**. First Monday (Peer-Reviewed Journal of the Internet), vol. 5, no. 10, October 2000.

[56] Hughes, D.; Coulson, G.; Walkerdine, J. **Free riding on gnutella revisited: the bell tolls?** Distributed Systems Online, IEEE, vol. 6, no. 6, 2005.

[57] Schulzrinne, H. **Engineering peer-to-peer systems**. Invited Talks. http://www.p2p08.org/program/sessions/1-invited-talk-i/P2P08-keynote.pdf/at_download/file. 2008.

[58] Aggarwal, V. and Feldmann, A. **Locality-aware p2p Query Search with ISP Collaboration**. Networks and Heterogeneous Media. 2008.

[59] Pouwelse, J. A.; Garbackia, P.; Epemaa, D.; Sipsa, H. **Pirates and Samaritans: A decade of measurements on peer production and their implications for net neutrality and copyright**. Telecommunications Policy, vol. 32, no. 11, pp. 701-712, December 2008.

[60] AzureusWiki. **Scrape**. http://azureuswiki.com/index.php/Scrape

[61] Chakrabarti, S.; Berg, M.; Dom, B. **Focused crawling: A new approach to topic-specific web resource discovery.** Computer Networks, vol. 31, pp. 1623-1640, 1999.

[62] Cohen, Bram. **The BitTorrent Protocol Specification**. 2008. Website. http://www.bittorrent.org/beps/bep_0003.html

[63] Schonwalder, J.; Quittek, J.; Kappler, C. **Building Distributed Management Applications with the IETF Script MIB**. IEEE Journal on Selected Areas in Communications, vol. 18, no. 5, pp. 702-714, May 2000.

[64] Dubuis, B. (2008). Java BitTorrent API. http://sourceforge.net/projects/bitext.

[65] Satoshi, Hirano. **HORB: Distributed Execution of Java Programs**. In Proceedings of the International Conference on Worldwide Computing and Its Applications. 1997. Pg 29-42.

[66] Hogan, T. (2008). **BNBT Easy Tracker**. http://bnbteasytracker.sourceforge.net

[67] isoHunt – The BitTorrent and P2P Search Engine. http://isohunt.com/

[68] Torrentz – Torrents Search Engine. http://www.torrentz.com/

[69] Harrison, David. **Peer ID Conventions**. BEP 20, version 11031, 2008. http://www.bittorrent.org/beps/bep_0020.html.