UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO BRANDÃO MANSILHA

# Investigando Estratégias Otimizadas para Monitoramento Eficiente do Universo BitTorrent

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Marinho Pilla Barcellos
Orientador

Prof. Dr. Luciano Paschoal Gaspary
Co-orientador

Porto Alegre, Março de 2012

**CIP – CATALOGAÇÃO NA PUBLICAÇÃO**

Mansilha, Rodrigo Brandão

Investigando Estratégias Otimizadas para Monitoramento Eficiente do Universo BitTorrent / Rodrigo Brandão Mansilha. – Porto Alegre: PPGC da UFRGS, 2012.

85 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2012. Orientador: Marinho Pilla Barcellos; Co-orientador: Luciano Paschoal Gaspary.

1. Peer-to-peer. 2. BitTorrent. 3. Monitoramento. 4. Otimização. 5. Controle. I. Barcellos, Marinho Pilla. II. Gaspary, Luciano Paschoal. III. Título.

*"Do not pray for tasks equal to your powers;*
*pray for powers equal to your tasks.*
*Then the doing of your work shall be no miracle,*
*but you shall be the miracle."*
— Phillips Brooks

# AGRADECIMENTOS

Agradeço primeiramente à Deus, por ter a quem agradecer, e em seguida, a essas pessoas que foram fundamentais para conclusão do mestrado: aos meus pais, Edison e Venina, pelo amor e exemplo de superação; aos meus irmãos, Ricardo e Raquel e seus respectivos namorado(a), Marília e Juliano, pela força, compreensão e apoio durante este período, que foi de grandes mudanças; à família Mansilha pelo apoio; aos meus orientadores, Marinho e Luciano, pela paciência e confiança; aos meus muitos colegas pelo companherismo dentro e fora do laboratório, e; finalmente, a todas que ajudaram nesse processo.

# SUMÁRIO

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| CGI | Common Gateway Interface |
| CIM | Common Information Model |
| CPU | Central processing unit |
| DHT | Distributed hash table |
| DMTF | Distributed Management Task Force |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| P2P | Peer-to-Peer |
| RFC | Request for Comments |
| RNP | Rede Nacional de Ensino e Pesquisa |
| SQL | Structured Query Language |
| TCP | Transmission Control Protocol |
| URL | Uniform Resource Locator |

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

Trabalhos recentes na literatura indicam que o BitTorrent é o protocolo de compartilhamento de arquivos com maior popularidade, sendo responsável por mais da metade do tráfego P2P em algumas localidades geográficas. Apesar de vários estudos sobre a dinâmica do "universo BitTorrent", até recentemente não existia metodologia para observá-lo sistematicamente.

Um estudo preliminar indicou a existência de múltiplas estratégias de monitoramento, que diferem em termos de objetos observados, conjunto de parâmetros e custos associados, e se sobrepõem em termos de informações extraídas. Um segundo trabalho apresentou uma combinação dessas estratégias na forma de uma arquitetura de monitoramento flexível e escalável.

Nesse contexto, o objetivo da presente dissertação é investigar como otimizar o conjunto de estratégias e seus parâmetros para monitorar eficientemente o universo de redes BitTorrent tendo em vista um dado conjunto de informações a ser observado e recursos computacionais disponíveis. Como solução é proposto um controle de monitoramento adaptativo, que emprega um modelo de programação para otimizar, a cada rodada, o monitoramento considerando o estado percebido da rede. Embora o foco deste trabalho seja redes BitTorrent, é proposto um modelo genérico, que pode ser aplicado a outras redes P2P, aumentando portanto a contribuição desta dissertação.

Os resultados de uma avaliação analítica indicam que o modelo de programação proposto gera soluções ótimas. Além disso, experimentos realizados com instâncias desse modelo geradas aleatoriamente mostram que o mesmo tem potencial para ser aplicado em redes mais complexas que BitTorrent.

**Palavras-chave:** Peer-to-peer, BitTorrent, monitoramento, otimização, controle.

**Investigating Optimized Strategies for Efficiently Monitoring the BitTorrent Universe**

# ABSTRACT

Recent studies in the literature have indicated that BitTorrent is the most popular file sharing protocol, being responsible for more than half of the P2P traffic in some geographical locations. Despite the several studies about the dynamics of the "BitTorrent universe", until recently there has been no methodology to *systematically* observe it.

A preliminary study identified multiple monitoring strategies, which differ in terms of observed objects, set of parameters and associated costs, and overlap in terms of extracted information. A subsequent work has combined those strategies in a flexible, extensible BitTorrent monitoring architecture.

In this context, the goal of the present dissertation is to investigate how to optimize the set of strategies and their parameters for efficiently monitoring the universe of BitTorrent networks, considering a given set of monitoring objectives and available computational resources. As a solution, an adaptive monitoring control is proposed, which employs a programming model to optimize the monitoring at each round, considering the perceived network state. Although the focus of this work is BitTorrent networks, we propose a generic model that can be applied to other P2P networks, thereby increasing the contribution of this dissertation.

The results of an analytical evaluation indicate that the proposed programming model generates optimal solutions. Furthermore, experiments carried out with randomly generated instances of this model show that it has potential to be applied to more complex networks than BitTorrent.

**Keywords:** Peer-to-peer, BitTorrent, monitoring, optimizing, control.

# 1 INTRODUÇÃO

Redes *Peer-to-Peer* (P2P) permitem que recursos sejam compartilhados entre usuários de forma eficiente e escalável. Dentre as aplicações P2P, a de compartilhamento de arquivos é provavelmente a mais popular (KARAKAYA; KORPEOGLU; ULUSOY, 2009). Incontestáveis exemplos de sucesso na Internet, entre eles o BitTorrent, têm sido acompanhados por um grande interesse por parte da comunidade científica (FAN; LUI; CHIU, 2009).

Informações sobre compartilhamento de conteúdo em redes BitTorrent podem ser úteis por uma série de razões. Por exemplo, comercialmente, podem amparar campanhas de marketing baseadas em popularidade de conteúdo, estimar perdas financeiras com cópias ilegais (CHOW et al., 2007) e auxiliar *Internet Service Providers* a investigar métodos que minimizem o custo do tráfego BitTorrent. Também podem auxiliar instituições da justiça a detectar certos tipos de ataques efetuados contra redes BitTorrent e a combater a pedofilia. Além disso, tecnicamente, podem ajudar a melhorar aplicações baseadas no protocolo, ou a criar novos sistemas baseados em BitTorrent, como *streaming* de vídeo (SILVERSTON; FOURMAUX; CROWCROFT, 2008).

Apesar de nos últimos anos ter sido publicada uma gama de estudos sobre redes BitTorrent (STUTZBACH et al., 2009; STUTZBACH; REJAIE; SEN, 2008; SALVADOR; NOGUEIRA, 2008; GUO et al., 2007; DALE; LIU, 2007; ZHAO; STUTZBACH; REJAIE, 2006; POUWELSE et al., 2005; IZAL et al., 2004), ainda pouco se sabe sobre o real funcionamento delas e padrões de comportamento de seus usuários. As informações divulgadas frequentemente deixam a desejar em termos de detalhe, abrangência ou precisão. Por exemplo, relatórios de provedores geralmente apresentam dados em alto nível sobre o tráfego que circula em um ponto de seus *backbones*, refletindo a situação de um país apenas, como em (SCHULZE; MOCHALSKI, 2009). Em trabalhos científicos, medições e análises usualmente focam em pontos específicos. Adicionalmente, não é raro haver discrepâncias entre os resultados apresentados, como por exemplo quanto ao nível de comportamento egoísta em determinadas redes P2P (ADAR; HUBERMAN, 2000; SAROIU; GUMMADI; GRIBBLE, 2002; HUGHES; COULSON; WALKERDINE, 2005).

Três fatores explicam a atual falta de conhecimento sobre a dinâmica do *Universo BitTorrent*. Primeiro, sua escala planetária, com milhões de usuários operando em nível de aplicação (com possível uso de criptografia). A dimensão, complexidade, heterogeneidade e incerteza de tal universo proveem oportunidades limitadas de observação constituindo obstáculos para monitoração efetiva. Segundo, em contraste com outras redes P2P de compartilhamento, como Gnutella, Kazaa e Kad, no BitTorrent a rede é composta por milhões de redes sobrepostas (*overlays*) menores

denominadas "enxames". Cada usuário potencialmente representa múltiplos pares, um para cada exame no qual está participando. Por fim, o protocolo BitTorrent é aberto e não possui uma padronização (por exemplo, RFC) ou especificação formal. Isto levou à criação de diversas implementações distintas de agentes de usuário e extensões ao protocolo, diferentemente das outras redes P2P de compartilhamento, que possuem agente e protocolo únicos. Consequentemente, para monitorar o Universo BitTorrent efetivamente é necessário contemplar os diversos agentes e protocolos, e acompanhar suas mudanças.

Um estudo preliminar (MANSILHA, 2009) indicou a existência de múltiplas estratégias de monitoramento do Universo BitTorrent, que diferem em termos de objetos observados, conjunto de parâmetros e custos associados, e se sobrepõem em termos de informações extraídas. Se combinadas, as estratégias podem fazer parte de uma solução mais abrangente, que pode ser otimizada de acordo com o objetivo do monitoramento.

Para demonstrar essa ideia, em (MANSILHA et al., 2011) foi apresentada uma arquitetura de monitoramento, denominada TorrentU. Essa arquitetura é capaz de monitorar o universo BitTorrent em escala planetária, o que é possível através de uma rede descentralizada de monitores. Adicionalmente, ela é flexível de maneira a permitir a extração eficiente (apenas) das informações desejadas, sem incorrer em custos desnecessários, o que pode ser obtido através da combinação de estratégias de forma ótima a um contexto. Por fim, a arquitetura é extensível para acompanhar a evolução do protocolo e das fontes de informação, o que é possível através de uma estrutura modular e com interfaces bem definidas. O TorrentU tem sido usado, por exemplo, no entendimento da disseminação ilegal de conteúdo através do BitTorrent (SCHMIDT; BARCELLOS; GASPARY, 2011; SCHMIDT et al., 2012). Além disso, um serviço baseado nessa arquitetura está sendo implementado em caráter experimental pela RNP (Rede Nacional de Ensino e Pesquisa) visando identificar uso indevido de sua infraestrutura (GT-UNIT, 2012).

A arquitetura TorrentU demonstra a viabilidade técnica de se combinar diferentes estratégias de monitoramento, mas questões sobre como usar essa potencialidade de forma eficiente permanecem em aberto. Já que existem diversas formas de se obter a mesma resposta, buscar aquela mais eficiente é importante porque a relação "universo monitorado/recursos disponíveis" é, em geral, desfavorável. O objetivo deste trabalho é investigar como otimizar o conjunto de estratégias e seus parâmetros para monitorar eficientemente o universo de redes BitTorrent tendo em vista um determinado conjunto de informações de interesse e um conjunto de recursos computacionais disponíveis.

Planejar monitoramentos eficientes envolve questões gerais de otimização, como definições de alternativas e métricas para compará-las. Desenvolver uma metodologia para responder essas questões torna-se ainda mais desafiador quando considera-se a dinâmica das redes P2P de distribuição, tipicamente suscetíveis a fenômenos como *churn* e *flash crowd*.

Este trabalho propõe como solução um controle adaptativo de monitoramento, no qual um modelo de programação gera planos de monitoramento otimizados considerando o estado percebido da rede. O controle adaptativo permite que a melhor combinação de estratégias seja reavaliada a cada ciclo, e assim adaptar o monitoramento caso o estado percebido da rede tenha sido alterado. O modelo de otimização, por sua vez, viabiliza o modelo de controle adaptativo porque elimina a necessidade

de um humano para escolha da estratégia a ser utilizada a cada rodada.

Embora o foco deste trabalho seja a tecnologia BitTorrent, propõe-se um modelo genérico que pode ser aplicado a outras redes P2P, aumentando portanto a contribuição desta dissertação. Além disso, o modelo pode ser aplicado para outros fins, como por exemplo a avaliação de novas técnicas de monitoramento ou a avaliação da vulnerabilidade a monitoramentos de redes P2P (ou seja, envolvendo questões de privacidade).

Os resultados da avaliação indicam que o modelo gera resultados ótimos para a rede BitTorrent, e que sua complexidade o permite ser computado quando instanciado para outras redes P2P, mais complexas. Para demonstrar o primeiro aspecto, os resultados gerados pelo modelo foram comparados com resultados gerados intuitivamente para uma série de cenários. Para demonstrar o segundo aspecto, foi medido o tempo ocupado para execução de diversas instâncias do modelo geradas de forma aleatória, com tamanhos maiores que BitTorrent.

O restante desta dissertação está organizado como segue. O Capítulo 2 contém um referencial teórico sobre monitoramento de redes BitTorrent. No Capítulo 3 é apresentada a arquitetura de monitoramento que congrega diferentes estratégias de monitoramento. O Capítulo 4 apresenta a solução para otimização de monitoramentos do universo BitTorrent, enquanto que o Capítulo 5 apresenta uma avaliação dessa solução. Por fim, no Capítulo 6 são tecidas as considerações finais e apresentadas propostas de trabalhos futuros.

# 2 UNIVERSO BITTORRENT

Este capítulo apresenta um referencial teórico sobre monitoramento do universo BitTorrent e está organizado como segue. Na Seção 2.1 é apresentada uma visão geral dos componentes da rede, enquanto que na Seção 2.2 são descritos os protocolos de comunicação entre eles. Por fim, na Seção 2.3 são identificadas estratégias de observação do universo discutindo-se trabalhos que apresentam resultados de monitoramento.

## 2.1 Componentes

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

No âmbito deste trabalho, por "conteúdo" subentende-se "conteúdo digital", que é diferente de "conteúdo informacional". Enquanto o primeiro significa um conjunto específico de arquivos digitais, o segundo significa a informação disponibilizada nos arquivos. Por exemplo, dois torrents podem se referir a um mesmo conteúdo informacional, porém a conteúdos digitais diferentes, como no caso de torrents de uma mesma música codificada em duas taxas de compressão diferentes. O conteúdo compartilhado é organizado em peças, o que permite que os pares de um mesmo enxame troquem dados mesmo antes de possuir a cópia completa do conteúdo.

Para participar de um enxame, um agente de usuário utiliza os metadados disponíveis no respectivo arquivo torrent. Esse arquivo contém informações sobre as peças (para cada peça, seu *hash* e tamanho) que formam o conteúdo e arquivos (nomes e tamanhos). Para distribuir um conteúdo, um usuário deve (através de seu agente) gerar um torrent e torná-lo público. Os torrents são tipicamente disponibilizados em sítios dedicados a promover o compartilhamento de arquivos, as comunidades. Além de publicar torrents, algumas comunidades disponibilizam rastreadores. Ou-

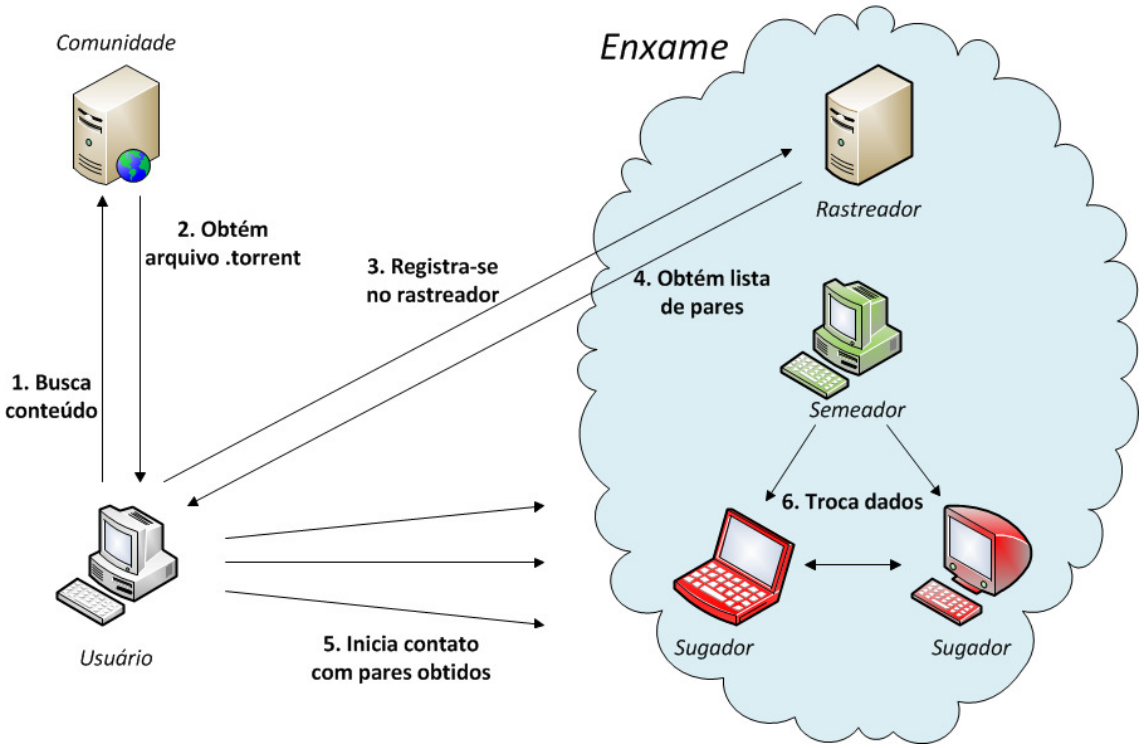



Figura 2.1: Funcionamento do BitTorrent

tras comunidades atuam primariamente como "agregadoras de torrents", indexando informações para permitir buscas e apontando para torrents em outras comunidades. A Figura 2.1 apresenta um exemplo bastante simples da interação entre os elementos. Na prática, múltiplas comunidades podem ser consultadas, assim múltiplos rastreadores podem ser usados em um único torrent.

O universo BitTorrent é ilustrado na Figura 2.2, que representa três cenários diferentes para demonstrar os tipos de formação possíveis com conteúdos, rastreadores e pares. Primeiro, *enxame 1*, que compartilha *conteúdo 1*, mostra o cenário em que um determinado conteúdo é compartilhado por apenas um enxame. Segundo, *enxame 2* e *enxame 3* ilustram o caso em que enxames possuem pares em comum. Note-se que esses enxames são completamente independentes entre si. Por último, *enxame 4* e *enxame 5* exemplificam o caso em que dois ou mais enxames distintos compartilham o mesmo conteúdo. Este cenário pode surgir em dois casos: (a) quando o tamanho das peças é diferente entre os enxames; ou (b) quando o tamanho das peças é igual, porém os conjuntos de rastreadores empregados são mutuamente exclusivos.

## 2.2 Protocolo

Esta seção aborda os conteúdos das mensagens e arquivos do protocolo do BitTorrent, necessários para entendimento do processo de monitoramento. Primeiramente são apresentados os protocolos de comunicação com rastreadores (Subseção 2.2.1), em seguida o protocolo de comunicação entre pares (Subseção 2.2.2) e finalmente a composição de arquivos torrents (Subseção 2.2.3). Uma descrição mais detalhada do protocolo pode ser encontrada em (LEHMANN et al., 2011).

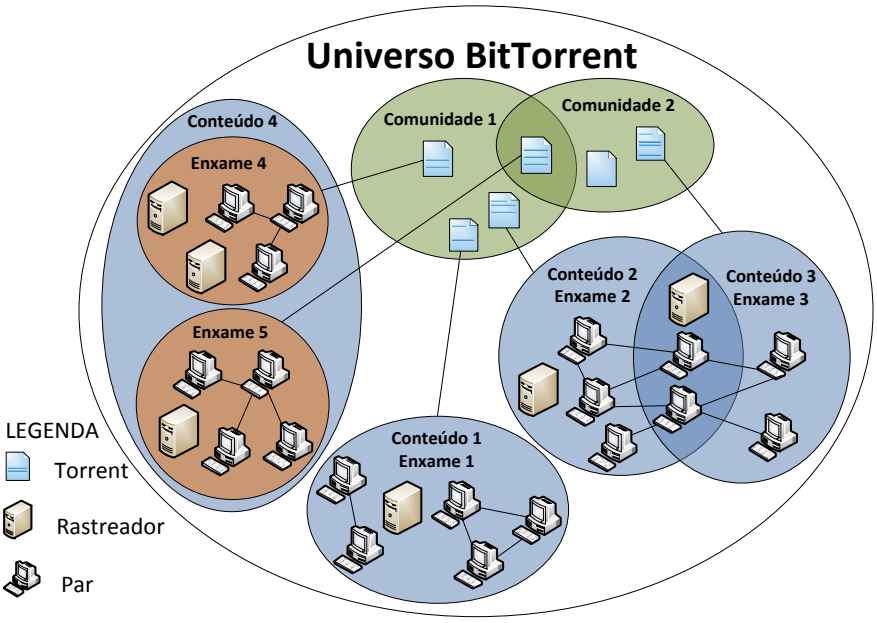



Figura 2.2: Exemplo de componentes e relações no universo de redes BitTorrent

### 2.2.1 Comunicação com rastreador

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

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

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

O contato de um par com o rastreador tem como objetivos atualizar sua situação na lista de pares de determinado enxame e obter endereços de outros pares que

também estão no mesmo enxame. Este contato, denominado *announce*, é previsto em cinco casos, conforme segue:

1. previamente ao *download*, é necessário um primeiro contato com o rastreador, para que o par se registre na lista do enxame e para obter endereços de outros pares;

2. em intervalos regulares, para o par notificar o rastreador sobre sua presença e obter novos endereços de pares;

3. quando a quantidade de pares conectados cai abaixo de um limite inferior, para obter novos endereços de pares;

4. ao completar o *download*, para notificar o rastreador que o par se transformou em um semeador;

5. ao deixar o enxame, para "elegantemente" notificar sua saída.

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

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

#### *2.2.1.1 Announce*

O *announce* é uma URL formada por um endereço de rastreador contido no arquivo de metadados, seguido pelos parâmetros segundo o padrão de métodos CGI (ROBINSON; COAR, 2004). A Tabela 2.1 apresenta todos os parâmetros possíveis para realizar o *announce* e uma breve descrição de cada um. Destes, os campos fundamentais são: *info_hash*, que identifica o enxame; *peer_id*, que é o identificador do par no enxame; e *port*, que indica a porta na qual o par está "escutando" solicitações de novas conexões. Além destes, há os campos *ip* e *numwant*, que são opcionais e podem ser utilizados para passar o endereço IP do par explicitamente e o tamanho da lista de pares a ser retornada, respectivamente. O campo *ip* é usado principalmente se o IP externo do par de origem é o mesmo IP externo do rastreador, enquanto o *numwant* é usado para obter listas maiores ou menores que o valor padrão, igual a 50.

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

Tabela 2.1: Mensagem *Announce*

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

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

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

O rastreador tem como principal objetivo proporcionar o encontro dos pares que desejam compartilhar um mesmo conteúdo. Além de responder ao par requisitante uma lista de pares presentes no enxame, o rastreador pode informar situações de erro, parâmetros de funcionamento e status do enxame. A resposta é um dicionário codificado com *bencode* com uma série de campos, conforme descrito na Tabela 2.2.

Tabela 2.2: Mensagem de resposta ao *Announce*

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

O campo mais importante é o *peers*, que pode ser representado tanto na forma tradicional, quanto na compacta. Na forma tradicional, a lista de pares retornada é representada através de uma lista de dicionários, contendo para cada par, seu *peer id*, endereço IP e porta. Uma forma de redução no consumo de recursos do rastreador é feita representando os pares de maneira compacta. Isso é feito através de uma string contendo 6 bytes para cada par, sendo os quatro primeiros o endereço IP do par, e os dois últimos, a porta em que o mesmo está escutando.

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

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

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

#### *2.2.1.2 Scrape*

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

A resposta de um *scrape* retorna as informações sobre cada um dos torrents requisitados, conforme apresentada na Tabela 2.3, podendo ser apenas um, múltiplos torrents definidos ou todos torrents do rastreador. Para cada um dos torrents são apresentadas as informações de número de semeadores e sugadores presentes no enxame, número de vezes que um *download* foi completado, e, opcionalmente, o nome do torrent.

Tabela 2.3: Mensagem de resposta ao *scrape*

| Campo | Descrição |
|---|---|
| files (dicionário) | Lista de arquivos requisitados, contendo os campos abaixo para cada um. |
| complete | Número de semeadores no enxame. |
| downloaded | Número de vezes que o rastreador registrou um término de *download*. |
| incomplete | Número de sugadores no enxame. |
| name | Opcional, nome do torrent. |

### 2.2.2 Comunicação entre pares

A Tabela 2.4 apresenta uma descrição resumida de cada tipo de mensagem trocada pelos pares. A seguir, cada mensagem é descrita em maior nível de detalhe.

O protocolo prevê uma mensagem KEEP-ALIVE para manter a conexão aberta quando não há comunicação entre os pares. Existem as mensagens de controle sem argumento extra CHOKE e UNCHOKE para sinalizar que o par está, respectivamente, bloqueado ou desbloqueado, e INTERESTED e NOT INTERESTED, para indicar interesse ou desinteresse no vizinho, respectivamente.

Em seguida, são utilizadas duas mensagens para atualizar a posse de peças aos vizinhos. A mensagem BITFIELD é enviada uma vez, durante o estabelecimento da conexão, ao longo da interação entre dois pares e serve para que os pares tenham conhecimento de quais peças são possuídas por seus vizinhos. A mensagem possui

Tabela 2.4: Mensagens do Protocolo BitTorrent

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

como argumento um mapa de bits representando a posse ou não de cada uma das peças. A mensagem HAVE serve para atualizar a informação de posse de peças de um par a seus vizinhos, passando como argumento o índice da peça recém completa.

Para a troca efetiva de dados, o protocolo prevê as mensagens REQUEST, usada para a requisição, e PIECE, utilizada para transferir o bloco requisitado. Na requisição, deve constar qual o índice da peça, o byte inicial e o tamanho do bloco desejado. Já na resposta, estão os mesmos campos de índice da peça e byte inicial, além dos dados propriamente ditos.

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

### 2.2.3 Torrent

O arquivo de metadados torrent possui um conjunto de campos, apresentados na Tabela 2.5, com informações sobre o conteúdo compartilhado e como encontrar outros pares (COHEN, 2008). Os dois campos fundamentais são *info* e *announce*. O primeiro é expandido na Tabela 2.6 e apresenta os seguintes campos: *piece length*, a quantidade de bytes em cada peça; *pieces*, a concatenação de todos os *hashes* das peças do conteúdo, que é utilizada para verificar a integridade do conteúdo; *private*, um campo opcional que indica ao agente que ele não deve obter novos pares por outras formas além das listadas no arquivo torrent; *name*, o nome do arquivo; e *files*, que lista os arquivos existentes no conteúdo. O campo *files* define, para cada um dos arquivos, os campos de: *length*, tamanho do arquivo; *md5sum*, soma md5 do arquivo (não utilizado); e *path*, que representa o caminho e nome do arquivo. O segundo campo, *announce*, define qual é a URL do rastreador a ser utilizada para o par entrar no enxame (o compartilhamento sem rastreador será discutido posteriormente).

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

Tabela 2.5: Estrutura do arquivo de metadados

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

Tabela 2.6: Estrutura do campo info

| Campo | Descrição |
|---|---|
| piece length | Número de bytes em cada peça. |
| pieces | Concatenação de todos hashes de cada peça. |
| private | Opcional, se ativado, não permite utilizar outras fontes de pares. |
| name | Nome do arquivo. |
| files | Todos os arquivos existentes com os campos a seguir para cada um. |
| length | Tamanho de cada arquivo. |
| md5sum | Opcional, soma MD5 do arquivo (não é usado). |
| path | Caminho e nome do arquivo. |

## 2.3 Estratégias de Monitoramento

Há diferentes estratégias para extrair informações do universo BitTorrent. Pode-se dividi-las em três grupos de acordo com a fonte de informação. O primeiro, segundo e terceiro grupos de informações referem-se, respectivamente, a comunidades de torrents, aos rastreadores, e aos pares. Para cada grupo é possível definir estratégias específicas de obtenção das informações e relacionar com metodologias utilizadas na literatura, conforme discutido a seguir.

**Comunidades**. De maneira geral, as comunidades BitTorrent existem para fornecer arquivos torrent e permitir a interação entre usuários com os mesmos interesses. Comunidades apresentam variados tipos de informação relacionados aos conteúdos, rastreadores e quantidade de pares. Trabalhos como (ANDRADE et al., 2005; POUWELSE et al., 2005) são exemplos que aplicam a estratégia de monitorar comunidades para obter uma visão ampla do universo. O primeiro investiga enxames pertencentes apenas a comunidades específicas para identificar a popularidade dos *torrents* e a chegada de pares aos enxames. Além disso, os autores avaliam a quantidade de compartilhamento (páginas Web dinâmicas disponibilizadas pelos rastreadores) dessas comunidades para estabelecer padrões de cooperação. O segundo trabalho investiga a disponibilidade dos servidores (de busca, repositórios e rastreadores) disponibilizados. Além disso, também é mostrada a quantidade de *torrents* e arquivos presentes em cada comunidade estudada.

**Rastreadores**. Outra estratégia é monitorar os rastreadores, que disponibilizam informações mais atualizadas e precisas que as comunidades, assim como informações sobre si e sobre os pares (potencialmente apenas um conjunto de endereços IP). Na literatura, são encontradas quatro variações desta estratégia. A primeira é baseada na análise de *logs* do rastreador, quando disponíveis; um exemplo de uso dessa abordagem pode ser encontrado em (IZAL et al., 2004).

A segunda variação da estratégia é observar continuamente o conteúdo das páginas Web dinamicamente geradas pelos rastreadores, assumindo que essa informação

é disponibilizada. A quantidade de informação disponibilizada pelo rastreador de uma comunidade dependerá do tipo de conteúdo que está sendo compartilhado e se a comunidade é aberta ou fechada. Um exemplo de trabalho que adota esta estratégia é (ANDRADE et al., 2005).

A terceira variação da estratégia é contatar os rastreadores como se fosse um par da rede. Desse modo é possível extrair a quantidade de pares/sugadores/semeadores que estão registrados nos rastreadores e obter listas de IP:porta dos mesmos. Um exemplo dessa estratégia é usado em (POUWELSE et al., 2005; ZHANG et al., 2011).

A quarta variação desta estratégia é solicitar para os rastreadores um sumário dos enxames rastreados, que inclui, para cada entrada, as quantidades de semeadores, sugadores e *downloads* completados até então. Em (LE BLOND et al., 2010), esta variação é utilizada para monitorar os rastreadores da comunidade PirateBay.

**Pares**. A terceira e última estratégia é monitorar os pares diretamente. Existem duas abordagens, uma ativa e outra passiva. A abordagem passiva para monitorar redes P2P é usada em (SAROIU et al., 2002; HORNG et al., 2006), e consiste em capturar e processar pacotes que trafegam nos *enlaces* de rede utilizados pelos pares observados. Em contraste, a abordagem ativa se baseia em instrumentar agentes de usuário que se conectam com pares e extraem informações. Há três variações para a abordagem ativa, todas com o mesmo requisito para implementação e utilização: receber como entrada informações sobre os enxames, ou seja, o endereço IP dos pares do enxame ou, alternativamente, do rastreador, a partir do qual pode-se obter endereços de pares. As três variações diferenciam-se quanto ao nível de mensagens trocadas: na primeira (POUWELSE et al., 2005; IOSUP et al., 2006), o agente conecta com pares remotos, efetua o *handshake*, troca o mapa de peças e encerra a conexão em seguida; na segunda, mantém conexão com os pares, mas não efetua troca de dados (YOSHIDA; NAKAO, 2011; KRYCZKA et al., 2011); e na terceira (LEGOUT et al., 2007), um par participa ativamente do enxame, inclusive compartilhando dados.

Individualmente, as estratégias de monitoramento representam uma parte do problema atacado neste trabalho, pois tipicamente limitam-se a partes específicas do universo. Se combinadas, as estratégias podem fazer parte de uma solução mais completa. No próximo capítulo é discutida uma arquitetura que concretiza essa ideia.

# 3 MONITORAMENTO DO UNIVERSO

Este capítulo descreve a arquitetura para monitoramento de redes BitTorrent denominada TorrentU, apresentada em (MANSILHA et al., 2010, 2011). A Seção 3.1 apresenta o modelo de informações projetado para gerenciar os dados observados e a Seção 3.2 descreve a arquitetura conceitual da proposta.

## 3.1 Modelo de Informações

Um modelo de informações do universo BitTorrent foi criado para estruturar, ligar e racionalizar as informações sobre comunidades, torrents, rastreadores, pares, enxames, etc. Ele é baseado em um subconjunto do Common Information Model (CIM), definido pelo Distributed Management Task Force (DMTF), e no modelo P2P, proposto em (DOYEN; FESTOR; EMMANUELNATAF, 2004).

A Figura 3.1 ilustra uma visão simplificada do modelo. As classes em branco contornadas com linhas sólidas e trastejadas designam, respectivamente, classes originadas do CIM e do modelo de Doyen *et al.*. Por outro lado, as classes em cinza são aquelas propostas para expressar os conceitos de BitTorrent e seus relacionamentos.

A classe raiz é a Enabled Logical Element da qual as outras são estendidas. O diagrama segue a notação padrão do CIM. Por exemplo, um usuário que executa o protocolo BitTorrent é um par e pode participar de múltiplos enxames, assim como um enxame pode conter diversos pares. Neste sentido, a classe BitTorrentPeer estende a classe Peer e possui uma relação muitos-para-muitos com a classe Swarm. Da mesma forma, as outras classes (Tracker, File) estão inseridas no modelo refletindo o universo descrito no Capítulo 2. Além disso, a classe TelescopeSetting estende a classe Setting do modelo CIM e armazena as configurações de monitoramento.

## 3.2 Arquitetura de Monitoramento

A arquitetura é composta por dois componentes principais, denominados TorrentU Observer e TorrentU Telescope, que interagem entre si, com o usuário e com o universo, conforme a Figura 3.2. O Observer gerencia um conjunto de Telescopes. Um telescópio pode gerar diferentes resoluções de imagem do universo: desde uma visão abrangente de uma constelação até um olhar mais preciso sobre uma única estrela. Em outras palavras, é possível ter uma visão ampla, considerando um conjunto grande de torrents do Universo e, em seguida, escolher um subconjunto menor para explorar com de forma mais detalhada.

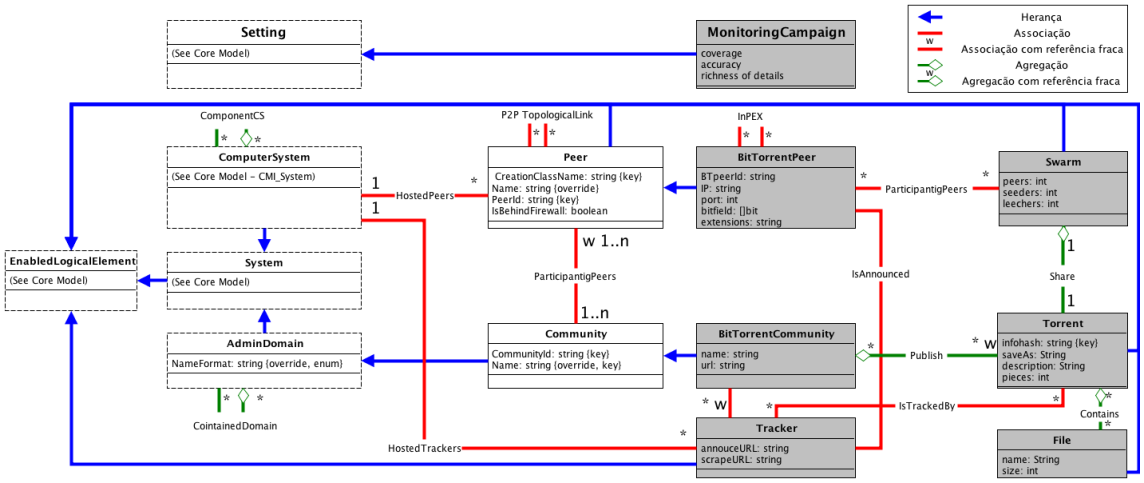



Figura 3.1: Modelo de Informações do BitTorrent

A arquitetura segue o modelo clássico de sistemas de monitoramento de redes. Na prática, Observer e Telescope correspondem, respectivamente, a um gerente e um agente de um sistema de monitoramento típico. Esses componentes são descritos a seguir.

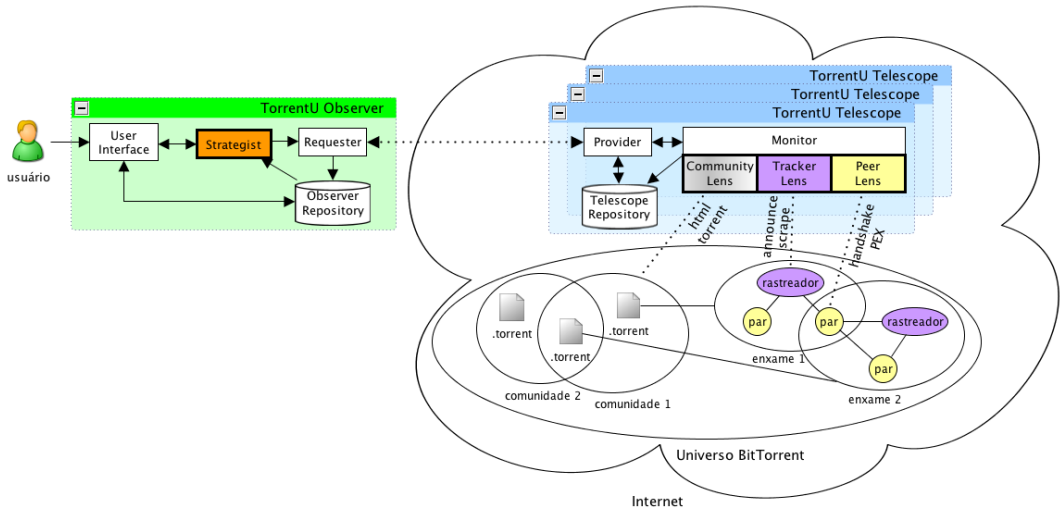



Figura 3.2: Arquitetura *TorrentU*

**TorrentU Observer**. Esse componente coordena um conjunto de um ou mais Telescopes, direcionando-os para pontos de interesse do universo. Ele escolhe estratégias e parâmetros a serem utilizados de acordo com os dados a serem monitorados e Telescopes disponíveis (e suas capacidades). O Observer também atua como estação de gerência, permitindo que o sistema seja configurado e apresentando resultados de monitoramentos obtidos em tempo real (assim como estatísticas e dados históricos). Além da *interface* de usuário, o Observer possui três subcomponentes: *Requester*, *Observer Repository* e *Strategist*. Esses subcomponentes são discutidos a seguir.

O subcomponente Requester implementa um protocolo cliente/servidor para comunicar-se com um Telescope. O protocolo é *statefull*: antes que um usuário em um Observer controle um Telescope, é necessário que o mesmo abra uma sessão com

o Telescope, e se autentique (através de uma chave comum) com o mesmo (mais informações a seguir). Similarmente ao FTP, são usadas conexões separadas, uma para controle e outra para dados. Enquanto a primeira é mantida aberta durante uma sessão, conexões de dados são estabelecidas sob demanda.

O protocolo de controle possui três tipos de mensagens, Telescope Management, Monitoring Tasks e Data Management, comentadas a seguir. Mensagens do tipo Telescope Management são usadas, por exemplo, para verificar qual é a tarefa em execução e finalizar o Telescope. O segundo tipo de mensagem é empregado para instanciar tarefas de monitoramento em Telescopes, assim como recuperar o estado de uma tarefa. Mensagens Data Management, por sua vez, são usadas para obter ou modificar dados obtidos pelos Telescopes. Por uma questão de flexibilidade, uma mensagem Data Management contém um comando/consulta SQL, aplicável às tabelas localmente armazenadas no Telescope. Comandos são tipicamente usados para obter ou remover dados das tabelas de um Telescope. No primeiro caso, os resultados são transferidos através da conexão de dados.

O subcomponente Observer Repository armazena informações obtidas pelo Telescope. Esse repositório é particularmente importante para dados de séries temporais, pois fotografias (por exemplo, um conjunto de atributos do universo) são criadas periodicamente e armazenadas em uma base de dados informacional para posterior recuperação e análise dos dados. Para essa finalidade, um modelo de persistência, derivado do modelo de informações apresentado anteriormente na Figura 3.1, é usado para organizar quantidades potencialmente grandes de volumes de dados recuperados.

O subcomponente Strategist é responsável por otimizar monitoramentos de acordo com objetivos especificados e recursos disponíveis. Esse assunto é tema principal desta dissertação e é discutido em detalhes no Capítulo 4.

**TorrentU Telescope**. Este componente é responsável pelo monitoramento do universo BitTorrent e pelo retorno de resultados de acordo com as requisições enviadas pelo Observer. Ele é composto por três subcomponentes, denominados Provider, Telescope Repository e Monitor. O Provider é a interface de comunicação com o Observer. O Telescope Repository é responsável pelo armazenamento dos dados coletados no Telescope. A exemplo do que ocorre com o Observer Repository, ele também adota o modelo de informações apresentado na Figura 3.1 como forma de persistir, de forma estruturada, os dados coletados. Por fim, o Monitor é o subcomponente que, de fato, contata os elementos do BitTorrent (portais de comunidades, rastreadores e pares). O Monitor é subdividido em três partes, denominadas "lentes" (*lens*), sendo cada uma responsável por monitorar um grupo diferente de elementos do universo: *Community Lens*, *Tracker Lens* e *Peer Lens*. Essa modularização permite que as lentes existentes possam ser substituídas, assim como novas possam ser facilmente incorporadas na arquitetura (sem modificação de seus componentes essenciais). A seguir, cada lente é discutida em detalhes.

### 3.2.1 Community Lens

O Community Lens monitora a publicação de arquivos torrents em um sítio da web. Conforme comentado anteriormente, informações de interesse podem ser extraídas de páginas da Internet ou de arquivos torrent. Para monitorar páginas é necessário o desenvolvimento de *parsers* específicos para cada comunidade, como por exemplo a IsoHunt (ISOHUNT, 2012). Definiu-se uma estrutura bem clara de

maneira a diminuir o custo de elaboração de novos *parsers*.

Um *crawler* focado é utilizado para se obter arquivos torrent de comunidades e sítios que possuem mecanismo de busca de torrents. Normalmente, todos os torrents de um sítio da web seriam gradualmente transferidos, começando pelos publicados mais recentemente até certo ponto nos mais antigos. Entretanto, para reduzir a quantidade total de torrents a serem carregados de um dado sítio, filtros dessas comunidades são usados para reduzir o conjunto de torrents a serem recuperados. O processo de varredura e transferência dos arquivos torrent ocorre em rodadas. A cada rodada, os dados coletados são adicionados ou atualizados no Telescope Repository.

### 3.2.2 Tracker Lens

Os rastreadores são monitorados pelo Tracker Lens. Ele é usado para extrair a quantidade de pares (sugadores e semeadores), endereços IP e portas dos pares, além de informações sobre os próprios rastreadores.

TrackersLens agrega ao Telescópio duas das variações da estratégia de monitoramento dos rastreadores. Uma é anunciar-se (*announce*) para o rastreador como participante de um determinado enxame para obter um subconjunto aleatório da respectiva lista completa de pares. A outra é requisitar um sumário (*scrape*) para obter as quantidades de pares participantes de todos os enxames rastreados. Esta requisição também permite especificar um determinado enxame.

Devido à escalabilidade, os rastreadores impõem um limite à frequência dos pedidos feitos por um mesmo par. Por isso, se a frequência pretendida é maior que a permitida pelo rastreador, é necessário instanciar múltiplas identidades para realizar o anúncio.

O contato com o rastreador pode falhar devido à indisponibilidade do mesmo ou problemas na rede. Além disso, o enxame pode estar sem novos pares ou com nenhum par. Portanto, são estabelecidas regras para determinar a frequência dos contatos em função das respostas do rastreador (verificando sua disponibilidade) e do conteúdo retornado (estimando a existência de pares desconhecidos). Dessa forma, o Tracker Lens pode direcionar seus recursos para os rastreadores que com maior probabilidade podem informar mais pares desconhecidos.

### 3.2.3 Peer Lens

Como o nome indica, o Peer Lens observa pares. Contatando-os, é possível extrair uma série de informações, como o nível de popularidade de determinado agente de usuário, assim como inferir valores como taxas de *download*.

O Peer Lens é um agente de usuário BitTorrent modificado. Para aumentar sua escalabilidade, o Peer Lens usa um esquema *round-robin*: a cada rodada, a lente tenta estabelecer conexões com $p$ pares; uma vez abertas, as conexões são mantidas por um tempo $t$, aguardando mensagens HAVE. Ao término do período, as conexões são encerradas, e o processo reinicia com os próximos $n$ pares da fila circular. Logo que uma conexão é estabelecida, coleta-se informações sobre o par correspondente, como por exemplo versão do agente e disponibilidade e quantidade de peças concluídas. Se a conexão é mantida aberta, é mais fácil acompanhar a evolução da distribuição das peças no enxame e inferir a velocidade de *download* do conteúdo de cada par. Embora existam variações do protocolo que não enviam mensagens HAVE para semeadores, isto não representa problema porque o Peer Lens

anuncia-se como sugador.

Pares que não aceitam o estabelecimento de uma conexão remota, devido a *firewall* ou outra forma de proteção, são marcados como inalcançáveis. Para estes casos, o monitor precisa aguardar até que os inalcançáveis iniciem uma conexão com o Peer Lens. Para aumentar a probabilidade disso acontecer, utiliza-se uma estratégia inspirada em *Sybils* (DOUCEUR, 2002). A lente cria e controla múltiplas identidades lógicas, anunciando-as ao rastreador. Dessa forma, aumenta-se a chance de que os inalcançáveis recebam pelo menos um dos IPs correspondentes ao Peer Lens, portanto aumentando a chance de conexão. Entretanto, essa técnica possui um custo associado: ao anunciar falsas identidades para os rastreadores, é possível que um ou mais pares tentem abrir múltiplas conexões com a lente. Essas conexões extras seriam rejeitadas, mas ainda assim passariam pelos processos de abertura e fechamento. Além disso, há de se considerar uma maior intrusividade no enxame.

A arquitetura de monitoramento apresentada neste capítulo permite empregar diferentes estratégias de monitoramento de maneira flexível. No entanto, explorar essa capacidade adequadamente não é tarefa simples devido ao dinamismo e à complexidade do universo BitTorrent. A proposta para superar tais desafios é o assunto do próximo capítulo.

# 4 PLANEJAMENTO DE MONITORAMENTOS

Este capítulo apresenta uma solução para otimização de monitoramento de redes BitTorrent. Primeiro, define-se um modelo de monitoramento genérico, aplicável a diferentes redes de compartilhamento de arquivos P2P. A seguir, esse modelo genérico é instanciado tendo redes BitTorrent como alvo. Embora o foco desta dissertação sejam redes BitTorrent, define-se um modelo genérico que poderá ser futuramente aplicado a outras redes, ampliando portanto a contribuição deste trabalho.

Este capítulo está organizado da seguinte forma. Considerando o desconhecimento inicial sobre o estado da rede a ser monitorada e as transformações bruscas que ela pode sofrer, é necessário um método adaptativo (Seção 4.1) que busque o monitoramento otimizado à rede P2P alvo. Em seguida, é definido um modelo matemático dos componentes da rede, suas propriedades e métricas, que permitem estabelecer um modelo de programação inteira mista para otimizar monitoramentos (Seção 4.2). Finalmente, discute-se como essa solução genérica pode ser instanciada considerando a rede BitTorrent (Seção 4.3).

## 4.1 Monitoramento Adaptativo

Redes P2P de compartilhamento são dinâmicas por natureza, estando por exemplo sujeitas ao *churn*, ou seja, a chegada ou partida em massa de pares (STUTZBACH; REJAIE, 2006). A eficiência de um sistema de monitoramento de uma rede P2P depende, além dos objetivos propostos, de parâmetros que devem ser ajustados em função das características atuais da rede monitorada, como seu tamanho. Por exemplo, para uma quantidade fixa de recursos, uma rede com 100 pares permitiria, intuitivamente, um acompanhamento muito mais preciso do que uma rede com 10.000 pares.

Nesse contexto, propõe-se o uso de uma abordagem adaptativa, em que o monitoramento é dividido em rodadas e ajusta-se o mesmo periodicamente. Mais precisamente, a cada rodada planejam-se os parâmetros do monitoramento de acordo com os objetivos propostos e o estado da rede percebido no momento. Dependendo do resultado do planejamento, pode-se descobrir uma forma mais eficiente de se obter os resultados do monitoramento ou então que não há plano factível e é necessário "abrandar" os objetivos.

A Figura 4.1 ilustra como um monitoramento contínuo é dividido em rodadas. O tempo de monitoramento ($T_m$), que pode ser ilimitado, é organizado em rodadas de tempo $T_r$. No início de cada rodada ocupa-se um determinado tempo para (re)planejar e, caso necessário, adaptar o monitoramento ($T_a$). Esse tempo deve ser menor que $T_r$, preferencialmente muito menor, conforme explicado a seguir.

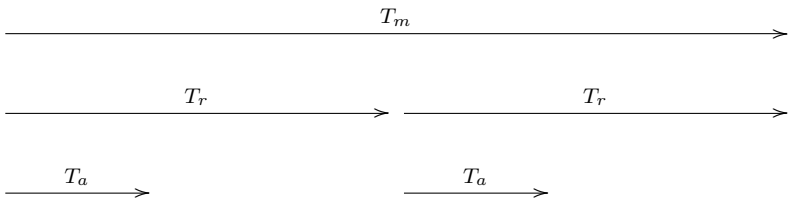



Figura 4.1: Monitoramento em rodadas

A duração de $T_r$ deve ser relativamente longa em relação a $T_a$ porque, intuitivamente, a frequência com que os planos são modificados deve ser bem inferior ao tempo em que um plano permanece efetivo. Por outro lado, quanto menor a frequência de revisão de planos, menos otimizado tende a ser o monitoramento. Dependendo da velocidade com que o estado da rede se altera, o monitoramento pode nunca ser executado de forma ótima. Dado que $T_r$ não deveria ser muito longo, é importante minimizar a duração de $T_a$. O tempo $T_a$ decorre da execução de um algoritmo para tentar encontrar um plano de monitoramento melhor e a potencial instalação do mesmo. Portanto, é necessário um algoritmo de planejamento eficiente. Note-se que, durante $T_a$, o plano em curso não se encontra suspenso, mas sim em adaptação.

## 4.2 Modelo Matemático

Esta seção apresenta um modelo de programação inteira mista para otimizar monitoramentos de redes P2P de compartilhamento de conteúdo. Nesse sentido, são definidos os elementos que compõe o sistema de monitoramento (Subseção 4.2.1), suas propriedades (Subseção 4.2.2) e então, uma série de métricas (Subseção 4.2.3) que servem para especificar funções objetivo para otimizar monitoramentos (Subseção 4.2.4).

### 4.2.1 Elementos do Sistema

Um sistema de monitoramento executa um processo que pode ser resumido da seguinte forma: contatar instâncias de *fontes* para acessar redes de sobreposição (*overlays*), obter instâncias de *unidades* de transferência e extrair instâncias de *atributos* usando determinadas quantidades de *recursos*. **Fontes** são tipos de elementos da rede com os quais se estabelecem conexões em nível de aplicação, como por exemplo par ou servidor de *bootstrap*. As fontes encontram-se organizadas em uma ou mais *overlays*, denominadas enxames no caso da rede BitTorrent. **Unidades** de transferência são tipos de respostas às requisições efetuadas para as fontes, como mensagens de resposta da aplicação ou arquivos, e caracterizam-se por serem indivisíveis e transportarem um conjunto de informações. Cada informação é uma instância de um tipo de **atributo** da rede, como por exemplo número de pares ou de conteúdos compartilhados. Finalmente, **recursos** referem-se a tudo aquilo que é utilizado para realizar o monitoramento, como largura de banda de *download*. A Tabela 4.1 resume os conjuntos de elementos que compõe o sistema de monitoramento.

Tabela 4.1: Conjuntos de elementos que compõe o sistema de monitoramento

| Notação | Nome | Descrição |
|---|---|---|
| $O$ | *Source* | Conjunto de fontes. |
| $U$ | *Unit* | Conjunto de unidades. |
| $A$ | *Attribute* | Conjunto de atributos. |
| $E$ | *Resource* | Conjunto de recursos. |

### 4.2.2 Propriedades do Sistema

Definiu-se uma gama de propriedades sobre os elementos acima definidos, que é suficiente para o posterior estabelecimento de métricas. As propriedades são classificadas em dois grupos, dependendo da etapa do monitoramento em que seus valores são configurados ou atualizados. O primeiro grupo (Subseção 4.2.2.1) representa o tipo de rede monitorada e por isso seus valores são especificados uma única vez, durante a parametrização do sistema. O segundo grupo (Subseção 4.2.2.2) representa o estado do monitoramento e, em contraste com o grupo anterior, seus valores são atualizados a cada rodada do monitoramento.

#### *4.2.2.1 Propriedades do Tipo de Rede*

A Tabela 4.2 apresenta um resumo das propriedades do tipo de rede. De acordo com a simbologia escolhida, cada propriedade é denotada por uma consoante maiúscula indexada pelos respectivos elementos (vogais minúsculas). Cada propriedade pode assumir um valor pertencente a um conjunto de possibilidades: números reais não negativos ($\mathbb{R}^+$), números naturais ($\mathbb{N}$) e valores binários ($\mathbb{B}$). A seguir, cada propriedade é descrita em maior nível de detalhe.

Tabela 4.2: Propriedades que representam o tipo de rede sendo monitorada

| Notação | $\in$ | Nome | Descrição |
|---|---|---|---|
| $R_{u,e}$ | $\mathbb{R}^+$ | *Required* | Quantidade do recurso $e$ utilizado para obter unidade $u$. |
| $N_{u,a}$ | $\mathbb{N}$ | *Instance* | Quantidade de instâncias do atributo $a$ na unidade $u$. |
| $P_{u,a}$ | $\mathbb{B}$ | *Prerequisite* | Atributo $a$ é pré-requisito para obter unidade $u$? |
| $C_{o,a}$ | $\mathbb{B}$ | *Concern* | Tipo de atributo $a$ diz respeito ao tipo de fonte $o$? |

Primeiro, cada tipo de unidade $u$ requer uma quantidade ($R_{u,e}$) para cada tipo de recurso $e$ para ser obtido. Dependendo do recurso modelado, as quantidades podem ser influenciadas pelo sistema de monitoramento ou serem determinadas exclusivamente pelo protocolo da rede a ser monitorada. Um exemplo do primeiro caso é a taxa de ciclos de CPU, que para um mesmo tipo de unidade pode variar para cada sistema conforme, por exemplo, sua linguagem de programação. Um exemplo do segundo caso é a largura de banda de *download*, que para cada tipo de unidade pode ser deduzida a partir da especificação do protocolo e deve ser igual em todos os sistemas.

Segundo, cada unidade $u$ pode conter zero, uma ou mais instâncias ($N_{u,a}$) de cada tipo de atributo $a$. Embora na prática o valor dessa propriedade possa variar conforme a fonte da unidade (por exemplo, quantidade de endereços de pares enviados por cada rastreador em um enxame BitTorrent), no modelo proposto o valor é uma constante que pode refletir, por exemplo, uma média de valores. Essa simplificação evita a necessidade de uma variável para cada instância de fonte de tipo de unidade, que seria necessária para armazenar o respectivo valor. Note-se, adicional-

mente, que o valor da referida propriedade não é influenciado pelo estado da rede (por exemplo, pelo número de pares na rede). Tal aspecto é modelado por outra propriedade, relacionada ao estado da rede, como será visto na próxima subseção.

Terceiro, para obter uma instância de determinada unidade pode ser necessário possuir pelo menos uma instância de um determinado atributo, por exemplo o endereço da fonte da unidade. Um atributo $a$ como esse é dito pré-requisito ($P_{u,a} = 1$) da unidade $u$. Sendo assim, o monitoramento só pode ser iniciado através de um conjunto prévio de instâncias de atributos, e segue restrito pelas dependências entre os tipos de unidades já obtidos (de acordo com os tipos de atributos contidos neles) e os tipos a serem obtidos (de acordo com os tipos de atributos pré-requisitos deles). Um tipo de unidade que não possui atributo pré-requisito é considerado ponto de partida para o monitoramento e na prática a informação necessária para obtê-lo é armazenada como um parâmetro do sistema. Vale destacar que a otimização de um monitoramento pode significar escolher unidades meios (que contém atributos pré-requisitos), mais do que escolher unidades fins (que contém atributo(s) que interessa(m) ao monitoramento).

Quarto e último, cada fonte possui um conjunto de tipos de atributos associados. Por exemplo, uma fonte do tipo "par" tem associado o atributo "endereço do par". Essa propriedade, denominada *Concern*, é anotada em $C_{o,a}$, que assume valor binário 1 (verdadeiro) quando um determinado tipo de fonte $o$ é associado a um tipo de atributo $a$. Um atributo $a$ é associado a uma e somente uma fonte de modo que $\sum_{o \in O} C_{o,a} = 1, \forall a \in A$. Essa propriedade é utilizada para estimar o estado da rede, como será visto na próxima subseção.

#### *4.2.2.2 Propriedades do Estado do Monitoramento*

Para determinar qual a melhor forma de monitorar uma determinada rede é necessário conhecer (ou pelo menos estimar de forma próxima) sua composição naquele momento. Nesse sentido, foram definidas duas propriedades que modelam o estado do monitoramento e da rede monitorada, apresentadas resumidamente na Tabela 4.3. A seguir, cada propriedade é discutida em maior nível de detalhe.

Tabela 4.3: Propriedades que representam o estado da rede monitorada

| Notação | $\in$ | Nome | Descrição |
|---|---|---|---|
| $M_o$ | $\mathbb{N}$ | *Monitored* | Quantidade estimada existente na rede de instâncias da fonte $o$. |
| $S_a$ | $\mathbb{N}$ | *State* | Quantidade estimada existente na rede de instâncias do atributo $a$. |

Primeiro, assume-se que uma rede P2P seja composta por um conjunto de instâncias de fontes participantes dessa rede. Assim sendo, estima-se uma quantidade ($M_o$) de instâncias de cada tipo de fonte $o$. Essa propriedade é útil para estimar (com apoio da propriedade *Concern*) a quantidade de atributos a serem monitorados, como explicado abaixo.

Segundo, assume-se que a rede possua um conjunto de instâncias de atributos relacionadas à(s) sua(s) fonte(s) participante(s). Nesse sentido, estima-se uma quantidade ($S_a$) de instâncias de cada tipo de atributo $a$ na rede monitorada. O valor dessa propriedade é igual ao número de instâncias da fonte $o$ a qual o atributo $a$ está associado ($S_a = \sum_{o \in o} M_o \times C_{o,a}$). Por exemplo, numa rede com 10 pares existem 10 instâncias de endereços de pares. Essas estimativas são úteis para elaborar planos e valorar, a cada rodada, as métricas propostas, como será visto na próxima subseção.

### 4.2.3 Métricas de Monitoramento

Para formalizar objetivos e comparar planos de monitoramento, definiu-se um conjunto de métricas. Dessa maneira, um objetivo (*goal*) define valores limites (mínimos ou máximos, dependendo do caso) para as métricas que devem ser respeitadas pelo monitoramento. Para alcançar um determinado objetivo de monitoramento, podem existir múltiplas opções de plano de monitoramento (*plan*).

Foram definidas três métricas, apresentadas de forma resumida na Tabela 4.4 e discutidas em detalhes a seguir na mesma ordem. Para cada métrica, apresenta-se uma definição em termos gerais, uma definição em termos de objetivo e uma definição em termos de plano de monitoramento.

Tabela 4.4: Métricas de Monitoramento

| Notação | $\in$ | Nome | Descrição |
|---|---|---|---|
| $\alpha_a$ | $\mathbb{B}$ | *Richness* | Atributo $a$ é monitorado? |
| $\beta_a$ | $\mathbb{N}$ | *Coverage* | Quantidade de instâncias de atributo $a$. |
| $\theta_e$ | $\mathbb{R}^+$ | *Cost* | Quantidade de recurso $e$. |

A primeira métrica, *Richness*, refere-se à seleção $\alpha_a$ de tipos de atributos $a$ observados. Logo, a *Richness* do objetivo é a lista mínima de atributos que necessariamente devem ser monitorados. A *Richness* de um plano é a lista de tipos de atributos que seriam em princípio monitorados caso ele fosse executado.

A segunda métrica, *Coverage*, refere-se à quantidade $\beta_a$ de instâncias de cada atributo $a$. A *Coverage* do objetivo significa o número mínimo de instâncias de cada atributo que precisam ser obtidas. A *Coverage* do plano determina a quantidade de instâncias de cada tipo de atributo que seriam em princípio obtidas caso ele fosse executado.

A *Coverage* pode ser especificada de duas formas alternativas: amostra da população de instâncias (*Sample*) e filtro lógico (*Filter*). O *Sample* do objetivo significa o percentual mínimo da população de determinado atributo a ser monitorado. O *Sample* do plano é determinado pela divisão da quantidade de instâncias de atributos a serem obtidas pela quantidade estimada existente ($\beta_a^{plan}/S_a$). O *Filter* do objetivo permite restringir o monitoramento às fontes, unidades e atributos de interesse. Essa entrada tem a forma de lista de regras lógicas para cada tipo de atributo de cada fonte e/ou *overlay*, por exemplo, pares localizados no Brasil que compartilham conteúdo com determinado nome. O *Filter* é refletido nos planos através da delimitação da rede observada completa em uma "rede de interesse" através da alteração do número de instâncias estimadas de cada tipo de fonte ($M_o$) e consequentemente do número de instâncias estimadas de cada tipo de atributo ($S_a$) da rede.

A terceira e última métrica, denominada *Cost*, refere-se a todos os recursos que são necessários para executar um monitoramento. Para cada tipo de recurso $e$ do sistema de monitoramento modelado, existe um consumo associado $\theta_e$. O *Cost* do objetivo define os limites máximos de recursos a serem empregados durante o monitoramento. O *Cost* do plano corresponde aos gastos necessários para executar o plano.

Uma vez definidas as métricas, é possível estabelecer maneiras de se avaliar planos. Um determinado plano é considerado factível (função *Feasible*) se atender todos

os objetivos. Mais precisamente, se os valores das métricas *Richness* e *Coverage* do plano forem maiores ou iguais aos do objetivo, e se os valores da métrica *Cost* do plano forem menores ou iguais aos do objetivo, para todos os recursos, conforme Definição 4.1.

$$Feasible(goal, plan) = \left( \begin{array}{cc} & \alpha_a^{plan} \geq \alpha_a^{goal} \\ \wedge & \beta_a^{plan} \geq \beta_a^{goal} \\ & \forall a \in A \end{array} \right) \wedge \left( \begin{array}{c} \theta_e^{plan} \leq \theta_e^{goal} \\ \forall e \in E \end{array} \right) \quad (4.1)$$

### 4.2.4 Otimização de Planos

Para um determinado objetivo de monitoramento, podem existir múltiplos planos de monitoramento factíveis. Assim, é desejável gerar um plano otimizado de acordo com algum critério. Para gerar planos de monitoramentos otimizados, elaborou-se um modelo de programação inteira mista. O modelo é baseado nos conjuntos de elementos e propriedades do sistema apresentados anteriormente. Além disso, o modelo é composto por um conjunto de variáveis, opções de funções objetivo, e um conjunto de restrições, que são apresentados a seguir nessa mesma ordem.

#### *4.2.4.1 Variáveis*

A Tabela 4.5 resume os conjuntos de variáveis do modelo. O primeiro conjunto define a quantidade ($\eta_u$) de instâncias a serem obtidas de cada tipo de unidade $u$. Esse conjunto determina o processo de monitoramento a ser seguido. A partir de um conjunto (de instâncias) de unidades obtidas é possível extrair quantidades de instâncias de atributos. Essas quantidades são armazenadas no segundo conjunto de variáveis ($\beta_{u,a}$) e correspondem à cobertura do plano. A variável $\beta_{\%}$ representa a amostra (*Sample*) do plano. Essa variável é usada para que todos os atributos sejam monitorados de forma proporcional. Por fim, a variável $\alpha_a$ (binária) indica se o tipo de atributo $a$ será monitorado e corresponde à riqueza do plano.

Tabela 4.5: Conjuntos de variáveis

| Notação | $\in$ | Descrição |
|---|---|---|
| $\eta_u$ | $\mathbb{N}$ | Quantidade de unidades $u$ a serem obtidas. |
| $\beta_{u,a}$ | $\mathbb{N}$ | Quantidade de instâncias do atributo $a$ lidas da unidade $u$. |
| $\beta_{\%}$ | $0 \leq x \leq 1 \mid x \in \mathbb{R}$ | Percentual monitorado da rede. |
| $\alpha_a$ | $\mathbb{B}$ | Atributo $a$ é monitorado? |

Como será visto a seguir, essas variáveis permitem calcular valores para as métricas e assim concluir se um determinado plano atende um determinado objetivo. Além disso, elas permitem otimizar o monitoramento visando alguma métrica em particular. Nesse sentido, dependendo da métrica a ser otimizada, pode ser necessário transformar determinadas variáveis em parâmetros (determinando seus valores) para viabilizar a execução do modelo.

#### *4.2.4.2 Objetivos*

Como as métricas apresentam unidades de medida e granulosidades variadas, descartou-se o estudo de uma função objetivo multicritério. Como alternativa, definiu-se uma função objetivo para as métricas *Coverage* e *Cost*. Não foi determinada uma função objetivo para a métrica *Richness* porque considerou-se que ela

não teria aplicação prática, já que essa métrica está relacionada a questões qualitativas do monitoramento e não apresenta graduação linear de valores. Por exemplo, um monitoramento com *Richness=2* não é necessariamente duas vezes melhor que um monitoramento com *Richness=1*. Assim, dependendo do caso, deve ser escolhida uma das opções apresentadas a seguir.

**MaxCoverage**. Em relação à *Coverage*, o objetivo é maximizar o percentual monitorado da rede.

$$\max \beta_{\%} \tag{4.2}$$

**MinCost($e$)**. Em relação ao *Cost*, o objetivo é minimizar as quantidades necessárias de cada tipo de recurso. Para cada tipo de recurso é estipulada uma função específica, já que cada recurso possui unidade de medida diferente. O consumo de um recurso $e$ (como largura de banda) é igual ao somatório da multiplicação do respectivo custo $R_{u,e}$ de cada unidade $u$ pela respectiva quantidade de unidades a serem obtidas, $\eta_u$.

$$\min \sum_{u \in U} R_{u,e} \times \eta_u \tag{4.3}$$

#### *4.2.4.3 Restrições*

**Riqueza mínima**. Todos os atributos da riqueza mínima devem ser monitorados pelo plano.

$$\alpha_a^{plan} \geq \alpha_a^{obj} \qquad \forall a \in A \tag{4.4}$$

**Cobertura mínima**. A cobertura de um atributo corresponde ao somatório da quantidade de instâncias de atributos lidas de cada unidade. A cobertura do plano deve ser maior ou igual à cobertura do objetivo para cada um dos atributos.

$$\beta_a^{plan} = \sum_{u \in U} \beta_{u,a} \geq \beta_a^{obj} \qquad \forall a \in A \tag{4.5}$$

**Custo máximo**. Para cada recurso modelado, o gasto total com todos os tipos de unidades não deve ultrapassar o objetivo. O consumo é calculado conforme explicação da função objetivo *MinCost*.

$$\theta_e^{plan} = \sum_{u \in U} R_{u,e} \times \eta_u \leq \theta_e^{obj} \qquad \forall e \in E \tag{4.6}$$

**Cobertura proporcional**. Os atributos devem ser monitorados de forma proporcional. Tal é garantido por essa restrição, que estabelece que cada tipo de atributo $a$ que está na riqueza do plano ($\alpha_a >= 1$) seja monitorado com uma proporção mínima ($\beta_{\%} \times S_a$). Como não pode haver multiplicação entre duas variáveis em modelo de programação linear, dependendo da função objetivo utilizada, é necessário determinar o valor para um dos tipos de variáveis, $\alpha_a$ ou $\beta_{\%}$. Quando a função objetivo *MaxCoverage* for usada, deve ser estipulado um valor para a variável $\alpha_a$, para cada atributo $a$. Quando a função objetivo *MinCost* for usada, deve ser estipulado um valor para a variável $\beta_{\%}$.

$$\sum_{u \in U} \beta_{u,a} \geq S_a \times \alpha_a \times \beta_{\%} \qquad \forall a \in A \tag{4.7}$$

**Atributos pré-requisitos**. Para todo tipo de unidade a ser obtida, é necessário que todos seus atributos pré-requisitos façam parte da riqueza ($\eta_u > 0 \wedge P_{u,a} = 1 \rightarrow \alpha_a = 1; \forall a \in A; \forall u \in U$). Isso garante uma amostragem mínima para os atributos pré-requisitos. A constante $K$ representa um valor suficientemente grande que garante essa restrição, para cada atributo pré-requisito $a$ de cada unidade $u$ ($P_{u,a} = 1$), da seguinte forma. Por um lado, caso o atributo não faça parte da riqueza então o número de unidades a serem obtidas poderá ser no máximo igual a zero ($\eta_u \times 1 \leq 0 \times K : K \rightarrow \infty$). Por outro lado, caso o atributo faça parte da riqueza então o número de unidades a serem obtidas poderá tender ao infinito ($\eta_u \times 1 \leq 1 \times K : K \rightarrow \infty$).

$$\eta_u \times P_{u,a} \leq \alpha_a \times K \qquad \forall u \in U, \forall a \in A : K \rightarrow \infty \tag{4.8}$$

**Atributos lidos**. Para cada tipo de atributo $a$, é necessário que sejam obtidas quantidades suficientes de instâncias de unidades ($\eta_u$) que transportam esse tipo de atributo ($N_{u,a}$).

$$\beta_{u,a} \leq \eta_u \times N_{u,a} \qquad \forall u \in U, \forall a \in A \tag{4.9}$$

**Atributos existentes**. Toda instância de atributo lida deve existir na rede monitorada.

$$\sum_{u \in U} \beta_{u,a} \leq S_a \qquad \forall a \in A \tag{4.10}$$

**Restrições triviais**. Por fim, o último conjunto de restrições serve para estabelecer que as variáveis devem assumir valores referentes aos respectivos conjuntos.

$$\eta_u \in \mathbb{N}, \alpha_a \in \mathbb{B}, \beta_{u,a} \in \mathbb{N}, \beta_{\%} \in \mathbb{R}^+ \qquad \forall u \in U, \forall a \in A \tag{4.11}$$

## 4.3 Instanciação da Solução para BitTorrent

Esta seção mostra como a solução de otimização mais ampla pode ser aplicada ao BitTorrent, foco desta dissertação, e está organizada como segue. Na Subseção 4.3.1 é discutido como o sistema de controle adaptativo pode ser acoplado a um sistema de monitoramento, tomando como exemplo a arquitetura TorrentU. Na Subseção 4.3.2 é apresentada uma instanciação do modelo de otimização para a rede BitTorrent.

### 4.3.1 Controle do TorrentU

A Figura 4.2 detalha o esquema proposto para realizar monitoramento adaptativo. Primeiramente, apresentam-se os atores envolvidos nesse processo e então, as interações entre eles. O gerente do sistema de monitoramento (Observer) gera planos otimizados a serem executados pelo agente do sistema de monitoramento (Telescope). O componente Strategist possui um subcomponente denominado Planner, que elabora

planos otimizados de monitoramento, e um subcomponente denominado Sensor, que interpreta o estado da rede baseado nos resultados do monitoramento.

As interações entre os atores ocorrem em rodadas dividindo-se o processo contínuo de monitoramento. A cada rodada, os *objetivos de monitoramento* (1) e o *estado da rede* (2) são utilizados para elaborar um *plano de monitoramento* (3). O plano elaborado é passado para o Requester comandar a execução do monitoramento. Os *resultados do monitoramento* são armazenados no Observer Repository e utilizados pelo Sensor para determinar o *estado da Rede*. Se não houver plano factível, uma mensagem é enviada para a User Interface (5).

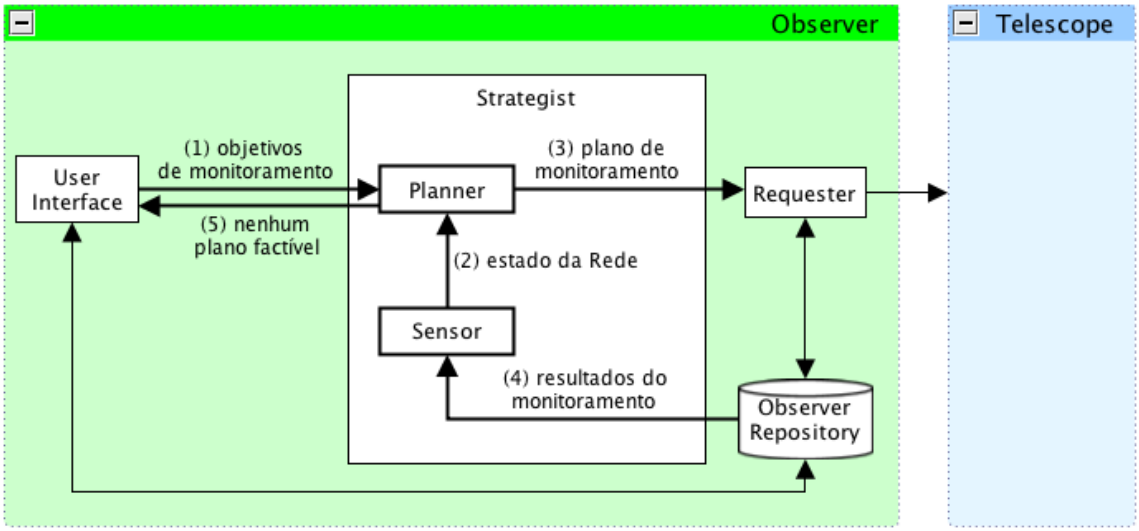



Figura 4.2: Controle Adaptativo de Monitoramento

### 4.3.2 Modelo do TorrentU

Esta subseção apresenta a instanciação do modelo matemático proposto considerando a arquitetura de monitoramento TorrentU e está organizada como segue. Primeiramente, são apresentados os elementos do sistema (Subseção 4.3.2.1), então, as propriedades desses elementos (Subseção 4.3.2.2).

#### *4.3.2.1 Elementos do TorrentU*

Conforme discutido no Capítulo 3, o TorrentU possui três lentes que são capazes de monitorar os tipos de fontes apresentados na Tabela 4.6, com exceção da fonte Swarm, que representa as *overlays* da rede BitTorrent. Embora essa "fonte" não forneça nenhuma unidade, ela foi modelada dessa maneira porque possui atributos que podem ser monitorados.

Tabela 4.6: Tipos de fontes contatáveis pelo TorrentU ($O$)

| Fonte ($o$) | Descrição |
|---|---|
| Community | Sítio da Internet que indexa arquivos torrents e páginas sobre conteúdos compartilhados. |
| Tracker | Ponto de encontro de pares. |
| Peer | Agentes de usuário que executam o protocolo BitTorrent. |
| Swarm | *Overlay* da rede BitTorrent. |

Através de cada lente do TorrentU é possível obter um conjunto de unidades. As comunidades enviam a unidade Html, que representa uma página do sítio contendo

informações sobre o enxame, e a unidade Torrent, que representa o arquivo de metadados de mesmo nome. Os rastreadores retornam as unidades Scrape e Announce, que representam as respectivas mensagens do protocolo, discutidas no Capítulo 2. Os pares retornam as unidades Handshake e Pex. A primeira representa as duas mensagens trocadas para estabelecimento da conexão em nível de protocolo - *Handshake* e *Bitfield*. Embora a mensagem *Bitfield* pudesse ser modelada como uma unidade a parte, optou-se pela junção por fins de simplificação, já que a separação não implicaria economia significativa de recursos. A unidade Pex representa a mensagem da extensão do protocolo de mesmo nome. Note que além dessas, outras unidades poderiam constar no modelo se fossem consideradas no sistema de monitoramento outras extensões do protocolo, como por exemplo DHT.

Tabela 4.7: Tipos de unidades recuperáveis pelo TorrentU ($U$)

| Unidade ($u$) | Descrição |
|---|---|
| Html | Arquivo no formato HTML que contém informações sobre um determinado enxame. |
| Torrent | Arquivo no formato Torrent. |
| Announce | Resposta ao anúncio efetuado para um rastreador. |
| Scrape | Resposta à requisição Scrape efetuada para um rastreador. |
| Handshake | Mensagem de conexão em nível de protocolo BitTorrent com par remoto. |
| Pex | Mensagem enviada por um par, que contém lista de pares conectados e desconectados. |

A explicação sobre os elementos do modelo é continuada pelos tipos atributos que podem ser monitorados pelo TorrentU, listados na Tabela 4.8. Eles encontram-se organizados de acordo com o que se referem: enxame (*overlay* do BitTorrent), par e rastreador. A maioria dos atributos foi definida através da análise do protocolo de comunicação, conforme apresentado no Capítulo 2. As exceções, cujos nomes são assinalados com "*", armazenam a conclusão de algum estágio do processo de monitoramento. Por exemplo, o atributo Peer connected* significa que foi possível estabelecer conexão em nível de protocolo com o par remoto.

Por fim, foram modelados dois tipos de recursos, Download e Connection, conforme a Tabela 4.9. Como os nomes indicam, o primeiro recurso significa largura de banda de *download* e o segundo significa o estabelecimento de conexão TCP. Como o foco deste trabalho é análise das estratégias de monitoramento, independente das suas implementações, outros recursos tipicamente considerados em avaliação de sistemas, como memória e CPU, não foram modelados porque não são estritamente relacionados ao protocolo.

#### *4.3.2.2 Propriedades do TorrentU*

Relembrando, na Seção 4.2.2 foram definidas as 6 propriedades de um modelo genérico de monitoramento de rede de compartilhamento P2P: *Cost*, *Instance*, *Prerequisite*, *Concern*, *Monitored* e *State*. A primeira propriedade, *Cost*, representa a quantidade de recursos necessários para obter as unidades. Os valores dessa propriedade são parâmetros do modelo. Os valores apresentados na Tabela 4.10 servem como exemplo e foram estipulados da seguinte forma. Os valores de Download são médias arredondadas de valores que foram obtidos através de medições. Quanto à conexão, é necessária uma para cada unidade obtida.

Os dados referentes à propriedade *Instance*, que modela as quantidades de instâncias de atributos disponíveis em cada unidade também são parâmetros do sistema. Os valores apresentados na Tabela 4.11 servem como exemplo e foram estipulados

Tabela 4.8: Atributos monitoráveis pelo TorrentU ($A$)

| | Atributo ($a$) | Descrição |
|---|---|---|
| Enxame | Infohash | Identificador único de um enxame. |
| | Leechers | Número de sugadores no enxame. |
| | Seeders | Número de semeadores no enxame. |
| | Content | Nome usado para gravar arquivo digital. |
| | Votes | Votos dos usuários da comunidade. Positivo e Negativo. |
| | Encoding | Encoding do arquivo de meta dados. |
| | Creation date | Data em que o arquivo torrent foi criado. |
| | Publication date | Data em que o arquivo torrent foi publicado. |
| | Comment | Campo de comentários disponível no torrent. |
| | Published by | Usuário que publicou o torrent na comunidade. |
| | Created by | Agente de usuário usado para gerar arquivo torrent. |
| | File names | Arquivos que compõe o conteúdo digital. |
| | File sizes | Tamanhos dos arquivos que compões o conteúdo digital. |
| | File md5sum | Md5sum dos arquivos. |
| | File paths | Caminhos para diretórios do arquivos. |
| | Number of pieces | Número de peças do conteúdo. |
| | Piece hash | Concatenação de todos hashes de cada peça. |
| | Piece length | Número de bytes em casa peça. |
| | Announce Swarm* | Announce realizado para o enxame? |
| Par | Peer address | Endereço IP e porta de um par. |
| | Peer address removed | Endereço IP e porta de um par que saiu do enxame. |
| | Peer connected* | Foi estabelecida conexão em nível de protocolo?. |
| | Peer reachable* | Foi estabelecida conexão TCP? |
| | Peer id | Identificador do par retornado pelo rastreador. |
| | Client | Agente de usuário usado pelo par. |
| | Extension | Extensões habilitadas pelo par remoto. |
| | Bitfield | Mapa de peças disponíveis pelo par remoto. |
| Rastreador | Tracker address | Endereço URL ou IP de um rastreador. |
| | Announce interval | Intervalo mínimo permitido para realizar *announces*. |
| | Failure reason | Mensagem de erro retornada pelo rastreador. |
| | Warning message | Mensagem de aviso retornada pelo rastreador. |
| | Scrape interval | Intervalo mínimo permitido para solicitar scrapes. |

Tabela 4.9: Conjunto de recursos ($E$)

| Recurso ($e$) | Descrição |
|---|---|
| Download | Largura de banda de *download*. |
| Connection | Conexão TCP. |

Tabela 4.10: Custos das Unidades de Transferência ($R_{u,e}$)

| Recurso ($e$) | Unidade ($u$) | | | | | |
|---|---|---|---|---|---|---|
| | Html | Torrent | Scrape | Announce | Handshake | Pex |
| Download | 30 KB | 25 KB | 50 MB | 467 B | 316 B | 467 B |
| Connection | 1 | 1 | 1 | 1 | 1 | 1 |

da seguinte forma. A maioria dos valores foi obtida através de análise da especificação do protocolo, conforme explicado no Capítulo 2. A exceção é a quantidade de instâncias do atributo Infohash na unidade Scrape, que varia conforme o número de enxames rastreados pela fonte. O valor do modelo (730.708) é o mesmo utilizado nos experimentos, conforme será descrito no Capítulo 5.

Os valores da propriedade *Prerequisite*, que modela quais atributos são pré-requisitos para solicitar cada unidade, são apresentados na Tabela 4.12. Os dados foram obtidos através de análise do protocolo, como segue. Para obter as unidades Html e Torrent nenhum atributo é necessário; a informação usada na prática ("endereço na comunidade") não é modelada porque essas unidades são consideradas pontos de partida do monitoramento. Para obter um Announce são necessários o atributo

Tabela 4.11: Quantidade de instâncias de atributos contidas nas unidades ($N_{u,a}$)

| | Atributo ($a$) | Unidade ($u$) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Html | Torrent | Scrape | Announce | Handshake | Pex |
| Enxame | Infohash | 1 | 1 | 730.708 | 0 | 0 | 0 |
| | Leechers | 1 | 0 | 730.708 | 1 | 0 | 0 |
| | Seeders | 1 | 0 | 730.708 | 1 | 0 | 0 |
| | Content | 1 | 1 | 0 | 0 | 0 | 0 |
| | Votes | 1 | 0 | 0 | 0 | 0 | 0 |
| | Encoding | 0 | 1 | 0 | 0 | 0 | 0 |
| | Creation date | 0 | 1 | 0 | 0 | 0 | 0 |
| | Publication date | 1 | 0 | 0 | 0 | 0 | 0 |
| | Comment | 0 | 1 | 0 | 0 | 0 | 0 |
| | Published by | 1 | 0 | 0 | 0 | 0 | 0 |
| | Created by | 0 | 1 | 0 | 0 | 0 | 0 |
| | File names | 0 | 1 | 0 | 0 | 0 | 0 |
| | File sizes | 0 | 1 | 0 | 0 | 0 | 0 |
| | File md5sum | 0 | 1 | 0 | 0 | 0 | 0 |
| | File paths | 0 | 1 | 0 | 0 | 0 | 0 |
| | Number of pieces | 0 | 1 | 0 | 0 | 0 | 0 |
| | Piece hash | 0 | 1 | 0 | 0 | 0 | 0 |
| | Piece lenght | 0 | 1 | 0 | 0 | 0 | 0 |
| | Announce Swarm | 0 | 0 | 0 | 1 | 0 | 0 |
| Par | Peer address | 0 | 0 | 0 | 200 | 0 | 25 |
| | Peer address removed | 0 | 0 | 0 | 0 | 0 | 25 |
| | Peer connected* | 0 | 0 | 0 | 0 | 1 | 0 |
| | Peer reachable* | 0 | 0 | 0 | 0 | 1 | 0 |
| | Peer id | 0 | 0 | 0 | 0 | 1 | 0 |
| | Peer client | 0 | 0 | 0 | 0 | 1 | 0 |
| | Extension | 0 | 0 | 0 | 0 | 1 | 0 |
| Rastreador | Tracker address | 8 | 8 | 0 | 0 | 0 | 0 |
| | Announce interval | 0 | 0 | 0 | 1 | 0 | 0 |
| | Failure reason | 0 | 0 | 0 | 1 | 0 | 0 |
| | Warning message | 0 | 0 | 0 | 1 | 0 | 0 |
| | Scrape interval | 0 | 0 | 1 | 0 | 0 | 0 |

Tracker address e também o atributo Infohash, enquanto que para obter uma unidade Scrape basta o endereço do rastreador. Para conectar-se com um par é necessário o endereço dele e também o Infohash do enxame que ele participa. Por fim, é necessário estar conectado com um par para obter uma unidade Pex.

Tabela 4.12: Atributos pré-requisitos das unidades ($P_{u,a}$)

| Atributo ($a$) | Unidade ($u$) | | | | | |
|---|---|---|---|---|---|---|
| | Html | Torrent | Scrape | Announce | Handshake | Pex |
| Tracker address | 0 | 0 | 1 | 1 | 0 | 0 |
| Infohash | 0 | 0 | 0 | 1 | 0 | 0 |
| Peer address | 0 | 0 | 0 | 0 | 1 | 0 |
| Peer connected* | 0 | 0 | 0 | 0 | 0 | 1 |

A última propriedade do tipo de rede, *Concern*, indica o tipo de fonte que cada atributo se refere. O valor $C_a$ para cada atributo $a$ equivale à classificação apresentada na Tabela 4.13. Por exemplo, o atributo Infohash se refere à fonte Swarm, enquanto que o atributo Peer address se refere à fonte Peer, e o atributo Tracker address se refere à fonte Tracker. A seguir são tratadas as propriedades referentes ao estado da rede monitorada pelo TorrentU.

A rede BitTorrent é composta por um conjunto de *overlays* denominadas enxames. Os valores da propriedade *Monitored*, que reflete estimativas das quantidades de fontes, podem ser obtidos através da contagem dos respectivos identificadores. Mais precisamente, cada enxame é identificado pelo atributo Infohash, enquanto que rastreadores e pares, participantes dos enxames, são identificados por seus endereços,

Tabela 4.13: Fonte a qual o atributo se refere ($C_{o,a}$)

| | Atributo ($a$) | Fonte ($o$) | | |
|---|---|---|---|---|
| | | Swarm | Peer | Tracker |
| Enxame | Infohash | 1 | 0 | 0 |
| | Leechers | 1 | 0 | 0 |
| | Seeders | 1 | 0 | 0 |
| | Content | 1 | 0 | 0 |
| | Votes | 1 | 0 | 0 |
| | Encoding | 1 | 0 | 0 |
| | Creation date | 1 | 0 | 0 |
| | Publication date | 1 | 0 | 0 |
| | Comment | 1 | 0 | 0 |
| | Published by | 1 | 0 | 0 |
| | Created by | 1 | 0 | 0 |
| | File names | 1 | 0 | 0 |
| | File sizes | 1 | 0 | 0 |
| | File md5sum | 1 | 0 | 0 |
| | File paths | 1 | 0 | 0 |
| | Number of pieces | 1 | 0 | 0 |
| | Piece hash | 1 | 0 | 0 |
| | Piece lenghts | 1 | 0 | 0 |
| | Announce swarm | 1 | 0 | 0 |
| Par | Peer address | 0 | 1 | 0 |
| | Peer address removed | 0 | 1 | 0 |
| | Peer connected* | 0 | 1 | 0 |
| | Peer reachable* | 0 | 1 | 0 |
| | Peer id | 0 | 1 | 0 |
| | Peer client | 0 | 1 | 0 |
| | Extension | 0 | 1 | 0 |
| | Bitfield | 0 | 1 | 0 |
| Rastreador | Tracker address | 0 | 0 | 1 |
| | Announce interval | 0 | 0 | 1 |
| | Failure reason | 0 | 0 | 1 |
| | Warning message | 0 | 0 | 1 |
| | Scrape interval | 0 | 0 | 1 |

ou seja, pelos atributos Tracker address e Peer address, respectivamente. Além disso, a quantidade de pares pode ser obtida através da soma dos atributos Leechers e Seeders. Conforme discutido anteriormente, os valores da propriedade *State* são estimados a partir das quantidades de enxames, pares ou rastreadores, dependendo do que o atributo se refere (propriedade *Concern*, anotada em $C_{o,a}$).

#### *4.3.2.3 Planos do TorrentU*

Um plano de monitoramento define uma quantidade a ser recuperada de cada tipo de unidade. Quando pelo menos uma instância de um determinado tipo de unidade deve ser recuperada por um plano, esse tipo de unidade é dito integrante da combinação (de unidades) desse plano. Assim, para um conjunto de unidades ($U$) existe uma gama de combinações, que é igual ao seu conjunto potência ($P(U)$).

A gama de combinações é restringida pelos atributos pré-requisitos das unidades. As combinações que não monitoram todos os atributos pré-requisitos necessários podem ser trivialmente descartadas, sendo consideradas *inviáveis*. Um exemplo é a combinação composta pelas unidades Torrent e Handshake, porque ela não inclui o atributo Peer address, pré-requisito da segunda unidade. Um contra exemplo é a combinação composta pelas unidades Torrent, Announce e Handshake: ela é dita *viável* porque contém todos os atributos pré-requisitos (Peer address, Tracker address e Infohash) em sua riqueza.

A Figura 4.3 apresenta o grau de riqueza ($\sum_{a \in A} \alpha_a$) correspondente a cada combinação (os nomes das unidades foram abreviados para as respectivas quatro primeiras

letras). Por exemplo, a riqueza de uma combinação composta apenas pela unidade Html é igual a 8, enquanto que a riqueza de uma combinação composta apenas pela unidade Torrent é igual a 14. Quando juntas, essas unidades formam uma combinação com riqueza igual a 19. Note-se que esse valor não é igual à soma das riquezas porque as unidades da combinação possuem atributos em comum. As combinações com pelo menos duas unidades e inviáveis foram omitidas da referida figura.

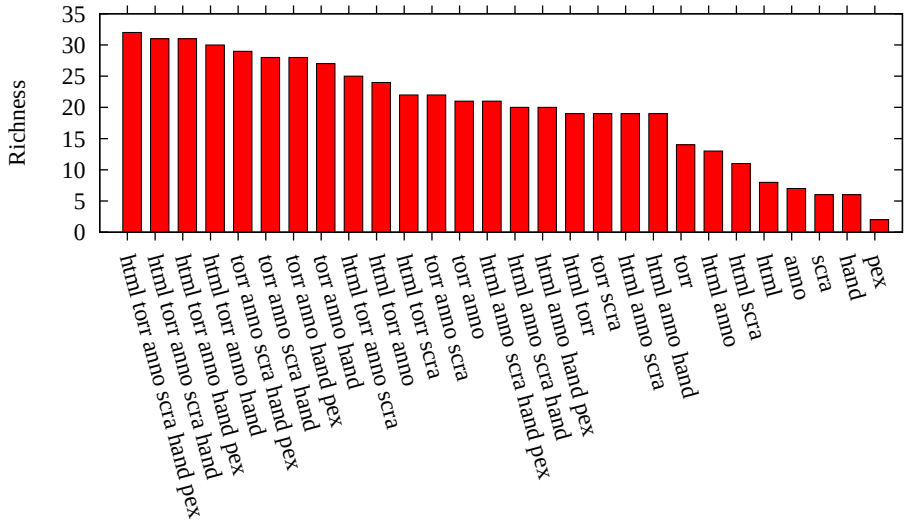



Figura 4.3: Riqueza de combinações de unidades

Como visto, o grau de riqueza de um monitoramento depende da combinação de unidades do plano. Por sua vez, o grau de cobertura de um monitoramento depende do número de instâncias de unidades a serem recuperadas, bem como o tamanho do universo a ser observado. Já a quantidade de instâncias a serem recuperadas impacta no custo do plano. Assim, o processo de otimização pode ser resumido na escolha adequada da combinação de unidades do plano, como será discutido no próximo capítulo.

# 5 AVALIAÇÃO

A presente dissertação propõe um controle adaptativo de monitoramento, onde a cada rodada um modelo matemático gera planos de monitoramento otimizados considerando o estado percebido da rede. Embora o foco do trabalho seja redes BitTorrent, é proposto um modelo genérico, que pode ser aplicado a outras redes P2P.

Devido à amplitude da proposta, que envolve tanto aspectos teóricos de modelagem matemática como aspectos práticos de implementação de *software*, foi necessário restringir o escopo de avaliação da solução. A avaliação do modelo matemático foi priorizada devido ao seu caráter inovador em comparação com o modelo de controle adaptativo empregado.

Este capítulo apresenta uma avaliação do modelo de programação proposto para otimizar monitoramentos de redes P2P de compartilhamento. Nesse sentido, primeiro são avaliados os resultados de otimizações considerando a rede BitTorrent (Seção 5.1) e então, a viabilidade de se usar o modelo para redes mais complexas (Seção 5.2).

Os resultados apresentados nessa avaliação foram gerados pelo *solver* CPlex (IBM ILOG CPLEX OPTIMIZER, 2012). Um modelo (função objetivo MaxCoverage) e um conjunto de entradas de exemplo, em linguagem de programação GNU MathProg (GLPK , GNU LINEAR PROGRAMMING KIT)[1], são apresentados no Anexo 1 e no Anexo 2, respectivamente.

## 5.1 Rede BitTorrent

Nesta etapa da avaliação é demonstrado que o modelo gera planos ótimos, de acordo com critérios intuitivos. Nesse sentido, em diversos cenários os planos gerados pelo modelo foram comparados com uma série de combinações de unidades, como explicado a seguir.

Cada cenário simula um *estado da rede* e foi elaborado de acordo com a seguinte metodologia. Extraiu-se uma lista de 730.708 enxames e respectivas quantidades de pares (total de 6.382.385) de uma unidade *Scrape*[2]. Essa lista foi ordenada de forma decrescente pelo número de pares. Ao longo da avaliação, cada estado da rede representa uma quantidade de enxames e o respectivo somatório das quantidades de pares.

[1]Essa linguagem é mais simples e pode ser traduzida de forma automatizada para linguagem interpretada pelo CPlex (OPL - Optimization Programming Language)

[2]Fornecida pelo rastreador http://ipv6.tracker.harry.lu/scrape/ em 14 de janeiro de 2012

A comparação entre resultados gerados pelo modelo e combinações de unidades é amparada por uma gama de gráficos (por exemplo, a Figura 5.2). Cada gráfico apresenta um conjunto de curvas e um conjunto de pontos. As curvas são geradas de acordo com uma função (Equação 5.1, explicada adiante) que determina a quantidade de recursos necessários para se obter uma determinada combinação de unidades. Por sua vez, os pontos representam resultados gerados pelo modelo. Os gráficos permitem argumentar sobre três aspectos: (i) o modelo retorna quantidades realísticas de unidades de cada combinação, o que é ilustrado pela proximidade entre pontos e curvas; (ii) os resultados são intuitivamente corretos, pois os pontos se aproximam da curva referente à melhor combinação (que possui menor custo, ou permite maior cobertura) e; (iii) relação entre métricas de monitoramento, como riqueza e custo, e cobertura e custo.

A função que gera a curva representando a quantidade necessária do recurso $e$ é igual ao somatório da quantidade a ser obtida de cada tipo de unidade $u$ multiplicada pelo seu respectivo custo (Equação 5.1). A quantidade a ser obtida de cada tipo de unidade $u$ é determinada pela razão máxima entre a quantidade de atributos disponíveis no estado da rede e a quantidade de atributos contidos em cada unidade (Equação 5.2).

$$\theta_e = \sum_{u \in U} \eta_u \times R_{u,e} \tag{5.1}$$

onde,

$$\eta_u = \max_{a \in A} S_a / N_{u,a} \qquad \forall a \in A \mid N_{u,a} > 0 \tag{5.2}$$

Cada função objetivo definida foi avaliada de forma particular, como segue. Primeiro, discute-se como a riqueza e a cobertura impactam na otimização do *Cost* (Subseção 5.1.1). Em seguida, discute-se como os custos e a riqueza impactam na otimização da *Coverage* (Subseção 5.1.2).

### 5.1.1 Minimizar Custo

A avaliação do modelo inicia pelas funções de objetivos de minimização dos custos - *MinCost*. Como cenários, foram considerados diferentes quantidades de enxames (*Coverage*) e diferentes objetivos mínimos de *Richness*. A amostra foi fixada em 100% ($\beta_{\%} = 1$).

Para apresentar os resultados, foi gerado um par de gráficos para cada recurso. Eles diferem quanto à apresentação da *Coverage*, no eixo horizontal: no primeiro ela é apresentada em termos de quantidade de pares (Figuras 5.1 e 5.3) e no segundo em quantidade de enxames (Figuras 5.2 e 5.4). Já o eixo vertical representa o custo (em termos de *download* ou de conexão), em escala logarítmica. Cada curva é uma função da quantidade de recursos a serem gastos com determinada combinação em relação à cobertura. Como o objetivo é minimizar os custos, quanto mais baixa a curva, melhor. Por sua vez, cada conjunto de pontos representa resultados gerados pelo modelo para um determinado *Richness* mínimo.

A discussão se concentra no monitoramento de três atributos: Infohash, Peer address e Peer connected. Conforme discutido anteriormente, cada combinação de unidades

implica uma riqueza correspondente. Nesse sentido, os gráficos contêm apenas as curvas referentes às combinações consideradas relevantes. As combinações avaliadas e suas riquezas são apresentadas na Tabela 5.1. A seguir, os 4 gráficos são analisados em detalhe.

Tabela 5.1: Riqueza das combinações avaliadas

| Combinação | Richness | | |
|---|---|---|---|
| | Infohash | Peer adress | Peer connected |
| Torrent | 1 | 0 | 0 |
| Torrent Announce | 1 | 1 | 0 |
| Torrent Announce Handshake | 1 | 1 | 1 |
| Scrape | 1 | 0 | 0 |
| Scrape Announce | 1 | 1 | 0 |
| Scrape Announce Handshake | 1 | 1 | 1 |

O primeiro gráfico (Figura 5.1) mostra os resultados da minimização de *download* como função da cobertura em termos de quantidade de pares. Em geral, observa-se que os pontos são próximos às curvas. Isso significa que as quantidades de unidades calculadas pelo modelo refletem combinações viáveis e portanto podem ser consideradas alternativas realísticas. Por exemplo, para monitorar apenas o atributo Infohash, na maioria dos casos o menor custo de *download* é 50 MB, o que reflete a escolha de se obter uma instância da unidade Scrape, já que esse valor corresponde ao custo da unidade ($R_{Scrape,Download} = 50$ MB). Essa combinação é ótima porque ela monitora o atributo em questão (Infohash), conforme Tabela 5.1, e possui custo mais baixo (curva mais próxima ao eixo horizontal).

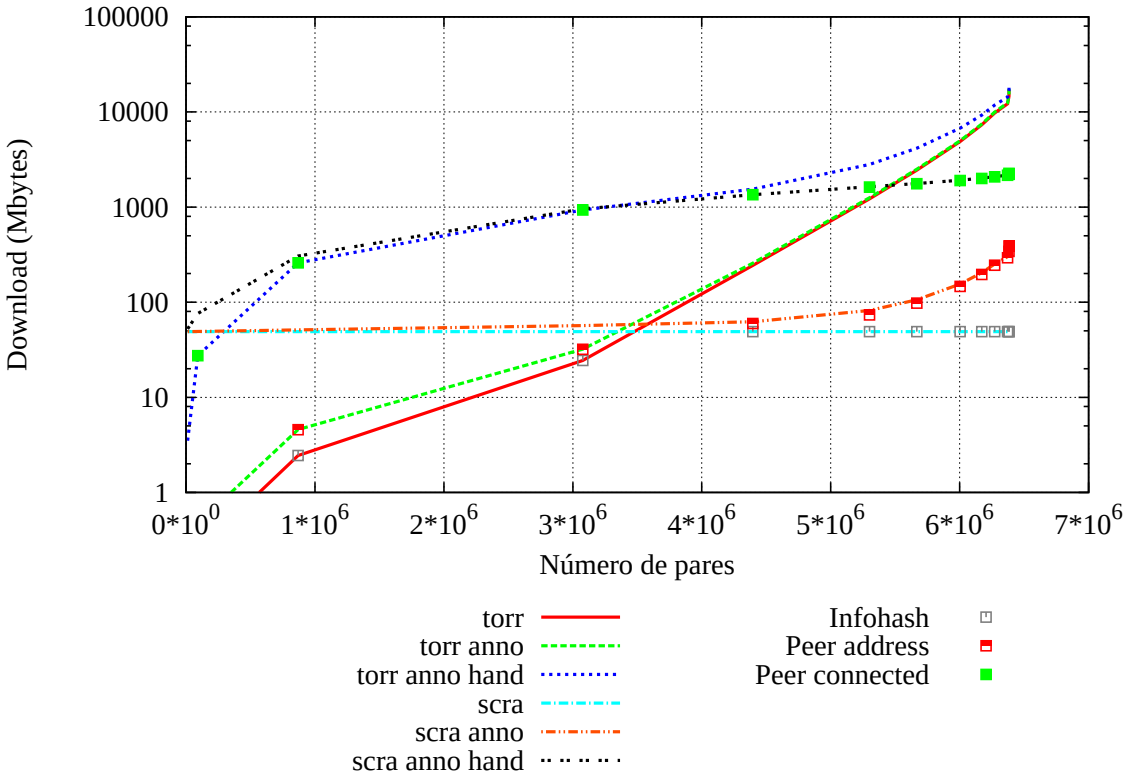



Figura 5.1: Minimização do download em relação à cobertura de pares

Na Figura 5.1 também é possível observar que a escolha da combinação (e consequentemente o custo) também é influenciada pela *Coverage*. Por exemplo, para monitorar o atributo Peer connected, inicialmente (3 primeiros pontos) o modelo indica a combinação composta pelas unidades Torrent, Announce e Handshake. Nos pontos seguintes, quando o custo da unidade Scrape torna-se menor que o somatório do custo da unidade Torrent, o plano é alterado para as unidades Scrape, Announce e Handshake.

No segundo gráfico (Figura 5.2) é possível observar que a escolha da combinação (e consequentemente o custo) é influenciada pelo *Richness* mínimo, conforme explicado a seguir. Primeiro, quando o *Richness* mínimo corresponde apenas ao atributo Infohash, a melhor opção é obter uma instância da unidade Scrape, na maioria dos cenários considerados. Segundo, quando o *Richness* mínimo corresponde ao atributo Peer address, a melhor opção em geral é obter uma instância da unidade *Scrape* e múltiplas instâncias da unidade *Announce*. Por fim, quando o *Richness* mínimo corresponde ao atributo Peer connected, além das duas unidades do caso anterior, é necessário estabelecer conexão com o par (unidade Handshake).

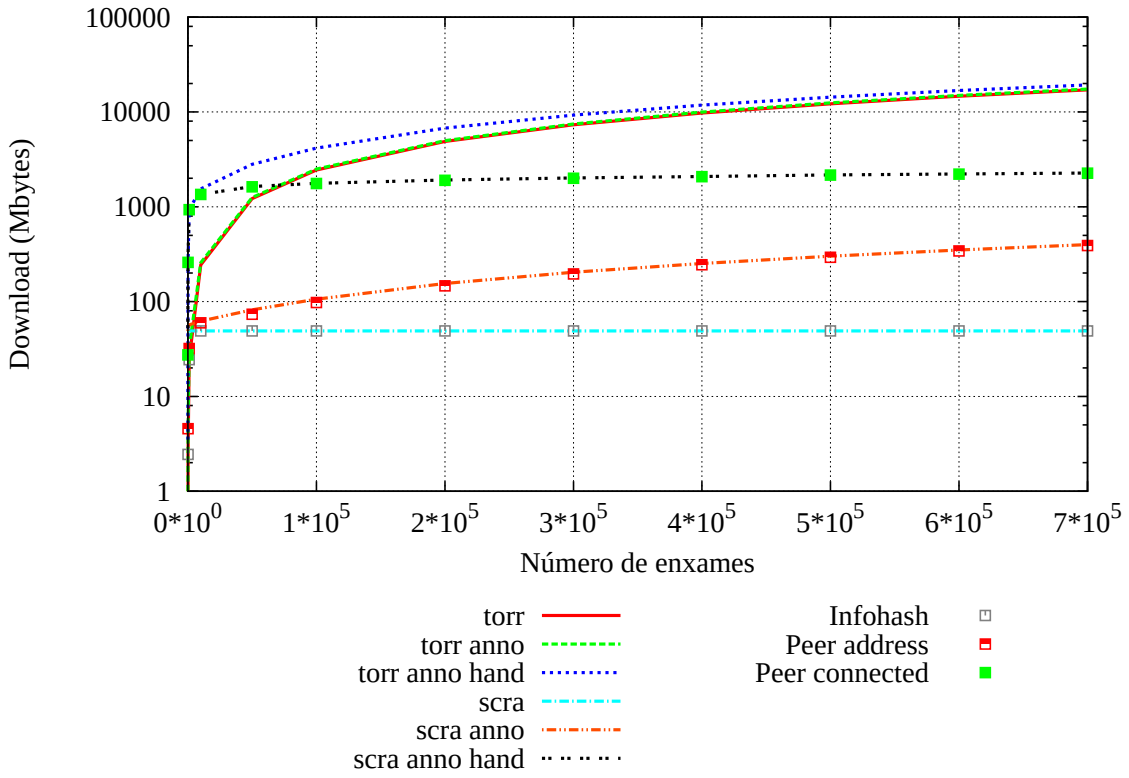



Figura 5.2: Minimização do download em relação à cobertura de enxames

Nas Figuras 5.3 e 5.4, correspondentes à minimização do custo de conexão, novamente as proximidades entre pontos e curvas indicam que o modelo gera resultados realísticos, ou seja, que refletem custos de combinações viáveis. Adicionalmente, observa-se que a combinação composta pela unidade Scrape é considerada vantajosa em relação à unidade "Torrent" desde o princípio; em contraste com a minimização de *download*, onde essa opção tornou-se vantajosa apenas a partir de uma determinada quantidade de enxames. Isso é explicado pela diferença entre as razões dos custos: 1/1 conexão no primeiro caso e 25 KB/50 MB no segundo.

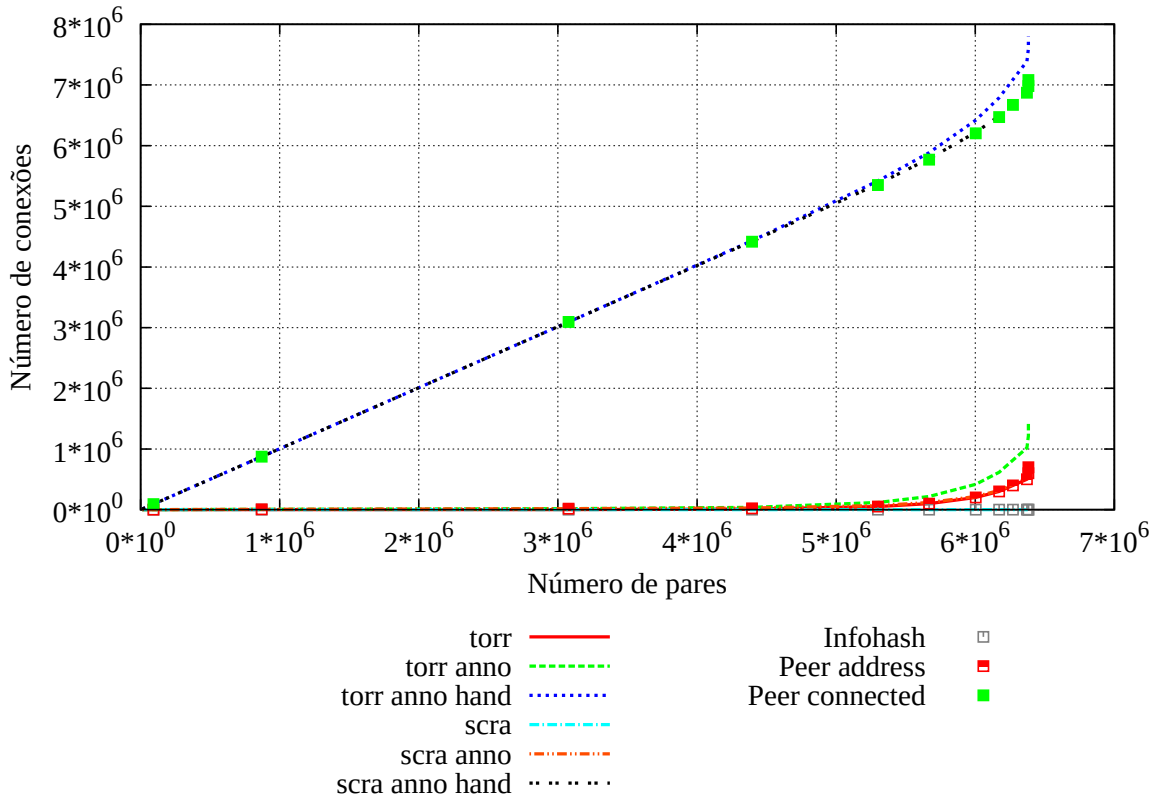



Figura 5.3: Minimização de conexões em relação à cobertura de pares

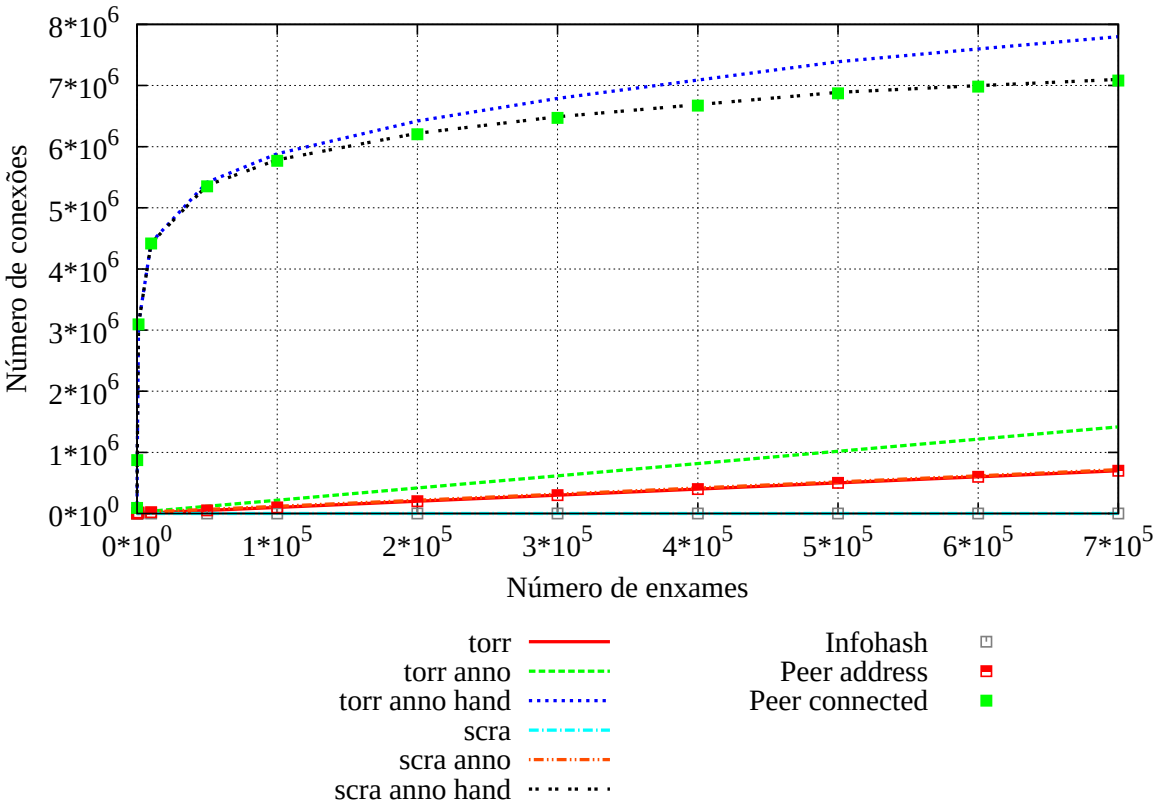



Figura 5.4: Minimização de conexões em relação à cobertura de enxames

### 5.1.2 Maximizar a Cobertura

A avaliação do modelo termina com a função objetivo *MaxCoverage*, através dos seguintes cenários. Em termos de custo, variou-se as quantidades disponíveis. Em termos de *Richness*, foram considerados dois casos: o atributo Infohash no primeiro caso e o atributo Peer connected no segundo.

A Figura 5.5 apresenta os resultados da maximização da cobertura quando a riqueza mínima é o atributo Infohash. O eixo vertical é apresentado em termos de número de enxames porque Infohash é um atributo da fonte Swarm. Já o eixo horizontal é apresentado em termos de *download* porque em termos de restrição, este é o recurso que gera mais impacto no monitoramento de enxames. A exemplo dos gráficos anteriores, cada curva foi gerada a partir da Equação 5.1 para uma determinada combinação de unidades. Como o objetivo é maximizar a cobertura, quanto mais alta a curva, melhor. Por sua vez, cada ponto representa o resultado gerado pelo modelo para uma determinada quantidade de recursos. O número máximo de enxames foi limitado em 3000 para destacar a região do gráfico onde é possível observar que o modelo retorna diferentes combinações de unidades dependendo da capacidade disponível, como discutido a seguir.

Como esperado, para capacidades "menores" (2 MB e 25MB) o modelo indica a combinação "Torrent" como solução. Com essa combinação, o custo cresce linearmente conforme a quantidade de enxames observados, até encontrar o limiar de 50 MB, que equivale ao custo de uma instância da unidade Scrape. A partir desse ponto, a combinação "Scrape" torna-se a opção mais vantajosa, permitindo monitorar todos os enxames considerados no cenário.

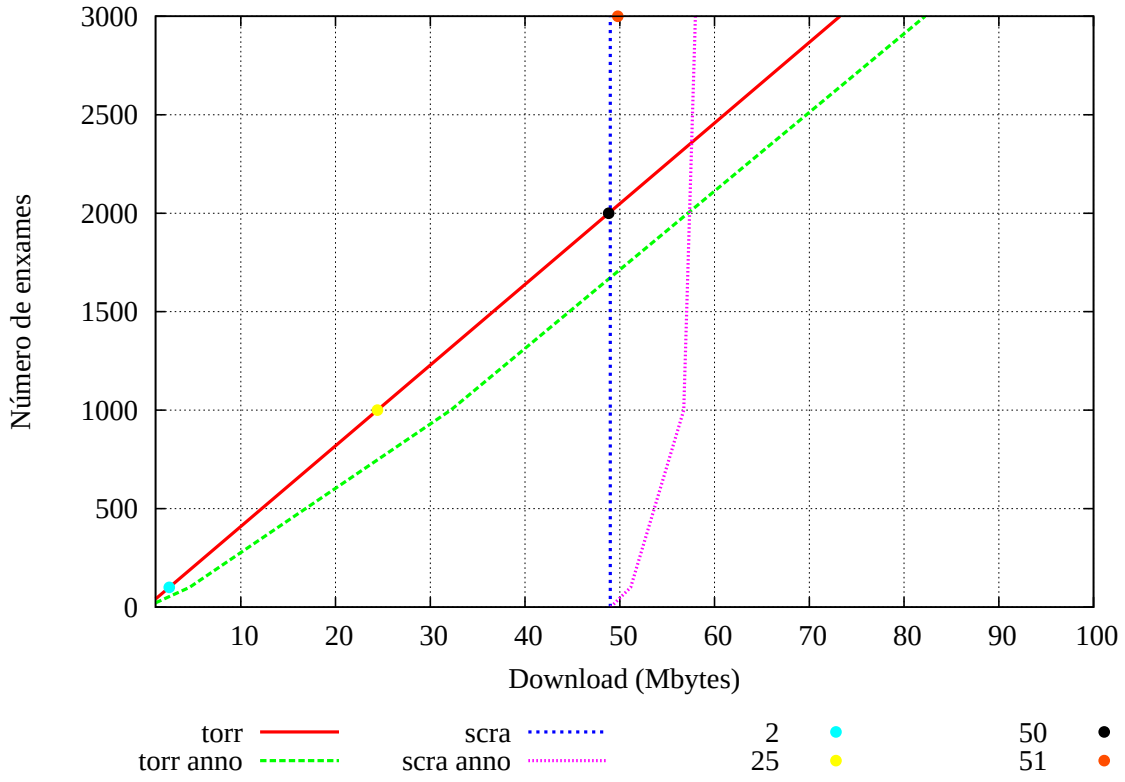



Figura 5.5: Maximização da cobertura em função da capacidade de *download*

Por fim, a Figura 5.6 apresenta os resultados da maximização da cobertura quando a riqueza é composta pelo atributo Peer connected. Como o atributo de interesse se refere à fonte Peer, o eixo vertical é apresentado em termos de número de pares (em escala logarítmica). O eixo horizontal, por sua vez, é apresentado em termos de número conexões porque esse recurso é proporcionalmente mais utilizado quando se monitora pares. Como no gráfico anterior, cada curva foi gerada a partir da Equação 5.1 para uma determinada combinação de unidades, enquanto que cada ponto representa o resultado gerado pelo modelo para uma determinada quantidade de recursos. Novamente, como o objetivo é maximizar a cobertura, quanto mais alta a curva, melhor. Note-se que o modelo sugere opções próximas das únicas combinações que monitoram a riqueza desejada (Peer connected), ou seja, aquelas que incluem as unidades Announce e Handshake.

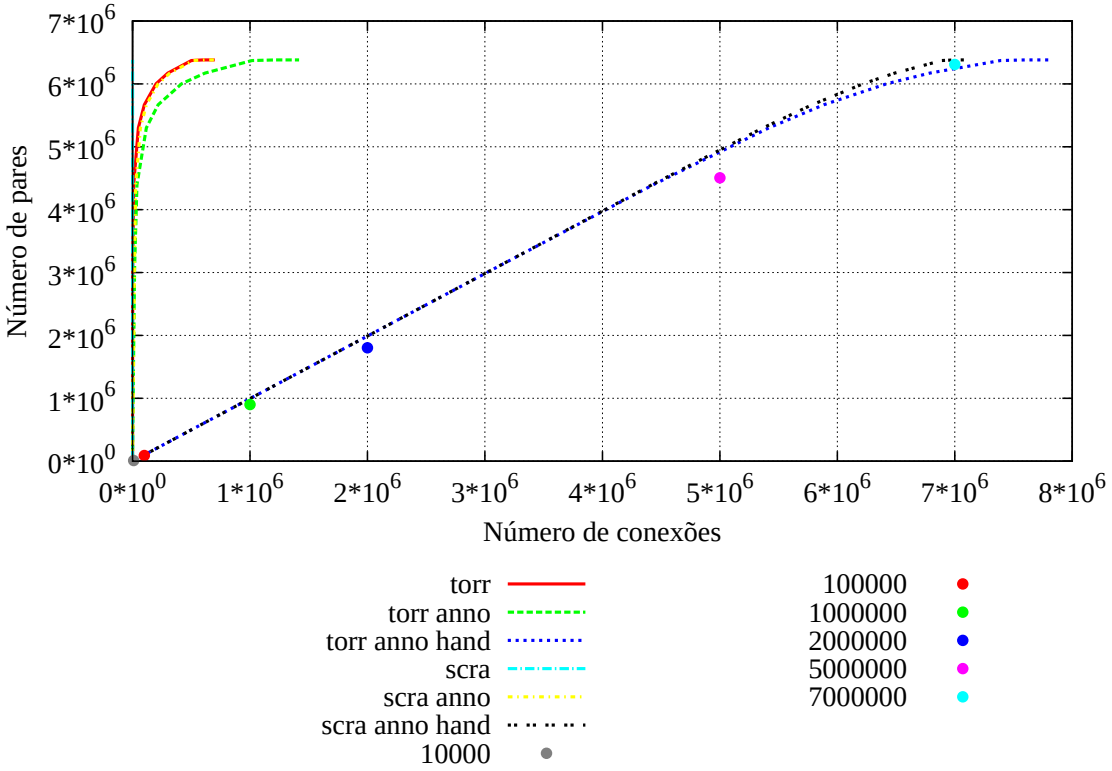



Figura 5.6: Maximização da cobertura em função da capacidade de conexão

## 5.2 Redes Simuladas

Após avaliar os resultados do modelo para uma rede conhecida, discute-se a instanciação do modelo para outras redes P2P. Pode-se afirmar que atualmente o BitTorrent é a rede P2P que apresenta o modelo mais complexo em termos de número de tipos de fontes, de unidades e de atributos. Para entender como o modelo poderia ser utilizado para redes mais complexas, que possam ser desenvolvidas futuramente, foram geradas instâncias aleatórias do modelo utilizando diferentes tamanhos de conjuntos, como explicado a seguir.

A Tabela 5.2 apresenta os parâmetros usados para gerar instâncias aleatórias e o tempo gasto para executar o modelo. As colunas $|A|$, $|U|$, $|O|$ e $|E|$ indicam os tamanhos dos respectivos conjuntos. Na falta de uma regra, optou-se por utilizar

o mesmo tamanho para todos os conjuntos, com exceção do conjunto de recursos. O número de tipos de recursos é mantido igual a 2 porque considerou-se que os mesmos recursos seriam modelados em instâncias futuras. O tempo apresentado refere-se a 10 execuções; cada uma correspondente a uma instância gerada com semente aleatória diferente. Os experimentos foram executados no solver CPlex em uma máquina com processador Intel Core i5 2,4 GHz e 8 GB de memória RAM. Foi estabelecido como objetivo a minimização do primeiro recurso ($Min.Cost(e_1)$).

Note-se que para $|U|$ até 50, o tempo gasto é no máximo 10 segundos nos casos avaliados. O tempo sobe para aproximadamente 93 segundos para instâncias de tamanho 100. Para fins de comparação, a instância do BitTorrent tem tamanho $|O| = 3$, $|U| = 6$ e $|A| = 32$, e durante a avaliação o tempo gasto foi 0,01 segundo em média.

Tabela 5.2: Tempo para execução de instâncias do modelo

| $\|O\|$ | $\|U\|$ | $\|A\|$ | $\|E\|$ | Tempo (s) |
|---|---|---|---|---|
| 6 | 6 | 6 | 2 | 0,01 |
| 10 | 10 | 10 | 2 | 0,03 |
| 50 | 50 | 50 | 2 | 9,41 |
| 75 | 75 | 75 | 2 | 15,57 |
| 100 | 100 | 100 | 2 | 92,69 |

O tempo necessário para processar um modelo de otimização depende de outros fatores além do tamanho da instância. Esses fatores, que variam de modelo para modelo, são chamados na literatura de "dificuldade do problema". Como as instâncias foram geradas de forma aleatória, é possível que elas sejam consideradas problemas relativamente "fáceis" ou "difíceis". Por isso, não é possível afirmar que os tempos medidos nessa avaliação representem precisamente a realidade. Pretende-se, contudo, argumentar que é razoável esperar que o modelo proposto tenha potencial para ser instanciado futuramente para redes mais complexas que o BitTorrent. Por exemplo, se o tempo para executar o modelo é menor que 10 segundos, então a adaptação poderia ocorrer a cada 100 segundos ou mais, posto que se espera que o tempo de reavaliação seja bem menor do que o tempo que o plano fica ativo.

# 6 CONCLUSÃO E TRABALHOS FUTUROS

Em (MANSILHA et al., 2011) foi apresentada uma arquitetura de monitoramento do universo BitTorrent. Essa arquitetura demonstra a viabilidade técnica de se combinar diferentes estratégias de monitoramento da rede P2P alvo. O referido trabalho discute a flexibilidade e a escalabilidade da arquitetura proposta, mas não responde questões sobre como usá-la de forma otimizada, considerando perguntas a serem respondidas e recursos computacionais disponíveis.

Como solução para essas questões, este trabalho apresentou um controle adaptativo de monitoramento. Nele, a cada rodada um modelo de programação é executado para otimizar o monitoramento considerando o estado percebido da rede na rodada anterior. O controle adaptativo permite que a cada ciclo o melhor plano seja posto em prática para ajustar o monitoramento às modificações no estado percebido da rede. O modelo de otimização, por sua vez, é necessário porque elimina a necessidade de um especialista para escolher o plano a ser utilizado a cada rodada.

Os resultados da avaliação indicam que o modelo matemático gera resultados adequados para a rede BitTorrent e que sua complexidade o permite ser computado quando instanciado para outras redes P2P, mais complexas. Para demonstrar o primeiro aspecto, foram gerados resultados considerando diversas alternativas e comparados com os resultados gerados pelo modelo para uma série de cenários. Em todos os casos avaliados o modelo proposto gerou solução intuitivamente ótima.

Para demonstrar o segundo aspecto, foi medido o tempo de execução para uma série de instâncias de redes mais complexas, geradas aleatoriamente. Embora não sejam conclusivos, os resultados sugerem que o modelo têm potencial para ser aplicado em redes mais complexas que BitTorrent em termos de número de tipos de elementos.

A otimização de monitoramentos é um tema amplo porque seu estudo envolve questões relacionas à metodologia experimental e também à metodologia analítica. Enquanto o trabalho desenvolvido previamente seguiu uma abordagem experimental para o problema (MANSILHA et al., 2011), a presente dissertação dedicou-se a aspectos analíticos. Nesse sentido, a contribuição principal deste trabalho é o modelo de otimização de monitoramentos de redes P2P. Ao que se sabe, esta é a primeira abordagem analítica para o referido problema. Argumenta-se que, além da aplicação demonstrada nesta dissertação, o modelo tem potencial para ser aplicado para outros fins, como avaliação de técnicas de monitoramento e vulnerabilidades de privacidade de redes P2P a sistemas de monitoramento.

Como trabalhos futuros, são consideradas quatro iniciativas principais. Primeiro, investigar o impacto dos valores das propriedades *Required* ($R_{u,e}$) e *Instance* ($N_{u,a}$) nos resultados gerados pelo modelo. Nesse sentido, uma questão a ser respondida,

por exemplo, é como escolher valores representativos para os referidos parâmetros e o impacto dessa decisão na solução.

A segunda iniciativa é implementar o modelo de controle no protótipo do TorrentU para avaliá-lo em ambiente real. Vislumbra-se, por exemplo, um estudo sobre o impacto do tamanho das rodadas em diversos monitoramentos, variando-se os objetivos específicos.

A terceira iniciativa é investigar um modelo de otimização para a distribuição de carga entre conjuntos de Telescopes. O objetivo desse modelo é determinar direções e estratégias a serem usadas por cada telescópio para atingir um determinado objetivo de observação de forma ótima global.

Por fim, a terceira iniciativa é desenvolver um algoritmo para escalonar cargas de monitoramento (quantidades de instâncias de unidades) visando questões de ordem temporal. Por exemplo, podem ser tratadas questões relacionadas ao intervalo mínimo entre requisições, que é estipulado por alguns tipos de fontes para alguns tipos de unidades, ou ainda a minimização do tempo necessário para executar um plano de monitoramento.

# REFERÊNCIAS

ADAR, E.; HUBERMAN, B. Free Riding on Gnutella. **First Monday (Peer-Reviewed Journal of the Internet)**, [S.l.], v.5, n.10, Oct 2000.

ANDRADE, N.; MOWBRAY, M.; LIMA, A.; WAGNER, G.; RIPEANU, M. Influences on cooperation in BitTorrent communities. In: ACM SIGCOMM WORKSHOP ON ECONOMICS OF PEER-TO-PEER SYSTEMS, 2005., New York, NY, USA. **Proceedings** ACM, 2005. p.111–115. (P2PECON'05).

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, Washington, DC, USA. **Proceedings** IEEE Computer Society, 2007. p.2–. (ICIMP'07).

COHEN, B. **The BitTorrent Protocol Specification**. Disponível em: <http://www.bittorrent.org/beps/bep_0003.html >. Acessado em: 01/06/2011.

DALE, C.; LIU, J. A measurement study of piece population in BitTorrent. In: IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, Washington, DC. **Proceedings** IEEE Computer Society, 2007. p.405–410. (GLOBECOM'07).

DOUCEUR, J. R. The Sybil Attack. In: INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS, 1., Cambridge, MA, USA. **Proceedings** Springer-Verlag, 2002. p.251–260. (IPTPS'02, v.2429 / 2002).

DOYEN, G.; FESTOR, O.; EMMANUELNATAF. A CIM Extension for Peer-to-Peer Network and Service Management. In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 11., New York, NY, USA. **Proceedings** Springer, 2004. p.801–810. (ICT 2004, v.3124).

FAN, B.; LUI, J. C. S.; CHIU, D.-M. The design trade-offs of BitTorrent-like file sharing protocols. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.17, n.2, p.365–376, Apr. 2009.

GLPK (GNU Linear Programming Kit). Disponível em: <http://www.gnu.org/software/glpk/ >. Acessado em: 01/08/2011.

GT-UNIT. Disponível em: <http://labcom.inf.ufrgs.br/ gtunit/ >. Acessado em: 12/02/2012.

GUO, L.; CHEN, S.; XIAO, Z.; TAN, E.; DING, X.; ZHANG, X. A performance study of BitTorrent-like peer-to-peer systems. **IEEE Journal on Selected Areas in Communications**, Piscataway, NJ, USA, v.25, n.1, p.155 –169, jan. 2007.

HORNG, M.-F.; CHEN, C.-W.; CHUANG, C.-S.; LIN, C.-Y. Identification and Analysis of P2P Traffic - An Example of BitTorrent. In: FIRST INTERNATIONAL CONFERENCE ON INNOVATIVE COMPUTING, INFORMATION AND CONTROL, Washington, DC, USA. **Proceedings** IEEE Computer Society, 2006. p.266–269. (ICICIC'06, v.2).

HUGHES, D.; COULSON, G.; WALKERDINE, J. Free Riding on Gnutella Revisited: the bell tolls? **IEEE Distributed Systems Online**, Piscataway, NJ, USA, v.6, n.6, p.1–, June 2005.

IBM ILOG CPLEX Optimizer. Disponível em: <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ >. Acessado em: 25/12/2011.

IOSUP, A.; GARBACKI, P.; POUWELSE, J.; EPEMA, D. Correlating Topology and Path Characteristics of Overlay Networks and the Internet. In: IEEE INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID WORKSHOPS, 6., Washington, DC, USA. **Proceedings** IEEE Computer Society, 2006. p.10–10. (CCGrid 2006, v.2).

ISOHUNT. Disponível em: <http://www.isohunt.com/ >. Acessado em: 02/01/2012.

IZAL, M.; URVOY-KELLER, G.; BIERSACK, E.; FELBER, P. A.; AL-HAMRA, A.; GARCES-ERICE, L. Dissecting BitTorrent: five months in a torrent's lifetime. In: PASSIVE AND ACTIVE MEASUREMENT WORKSHOP, 5., New York, NY, USA. **Proceedings** Springer, 2004. p.1–11. (PAM 2004, v.3015 / 2004).

KARAKAYA, M.; KORPEOGLU, I.; ULUSOY, O. Free Riding in Peer-to-Peer Networks. **IEEE Internet Computing**, Washington, DC, USA, v.13, n.2, p.92 – 98, Mar 2009.

KRYCZKA, M.; CUEVAS, R.; GUERRERO, C.; AZCORRA, A. Unrevealing the structure of live BitTorrent swarms: methodology and analysis. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2011., Washington, DC, USA. **Proceedings** IEEE Computer Society, 2011. p.230–239. (P2P'11).

LE BLOND, S.; LEGOUT, A.; LEFESSANT, F.; DABBOUS, W.; KAAFAR, M. A. Spying the world from your laptop: identifying and profiling content providers and big downloaders in bittorrent. In: USENIX CONFERENCE ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS: BOTNETS, SPYWARE, WORMS, AND MORE, 3., Berkeley, CA, USA. **Proceedings** USENIX Association, 2010. p.4–4. (LEET'10).

LEGOUT, A.; LIOGKAS, N.; KOHLER, E.; ZHANG, L. Clustering and sharing incentives in BitTorrent systems. In: ACM SIGMETRICS INTERNATIONAL

CONFERENCE ON MEASUREMENT AND MODELING OF COMPUTER SYSTEMS, 2007., New York, NY, USA. **Proceedings** ACM, 2007. p.301–312. (SIGMETRICS'07).

LEHMANN, M. B.; MANSILHA, R. B.; BARCELLOS, M. P.; SANTOS, F. R. Swarming: como bittorrent revolucionou a internet. In: SOUZA, A. F. de; MEIRA JR., W. (Ed.). **Atualizações em Informática 2011**. Rio de Janeiro, RJ: PUC-Rio, 2011. p.209 – 258. (JAI 2011).

MANSILHA, R. B. **Avaliando uma Arquitetura Flexível para Obtenção de Informação sobre o Universo de Redes BitTorrent**. 2009. Trabalho Individual — Universidade Federal do Rio Grande do Sul - UFRGS, Porto Alegre - RS.

MANSILHA, R. B.; BAYS, L. R.; LEHMANN, M. B.; MEZZOMO, A.; FACCHINI, G.; GASPARY, L. P.; BARCELLOS, M. P. Observing the BitTorrent Universe Through Telescopes. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2011., Washington, DC, USA. **Proceedings** IEEE Computer Society, 2011. (IM 2011).

MANSILHA, R. B.; MEZZOMO, A.; FACCHINI, G.; GASPARY, L. P.; BARCELLOS, M. P. Observando o Universo BitTorrent Através de Telescópios. In: XXVIII SIMPóSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUíDOS, Porto Alegre, RS. **Anais** SBC, 2010. (SBRC 2010).

POUWELSE, J.; GARBACKI, P.; EPEMA, D.; SIPS, H. The bittorrent p2p file-sharing system: measurements and analysis. In: PEER-TO-PEER SYSTEMS, 4., Berlin, Heidelberg. **Proceedings** Springer-Verlag, 2005. p.205–216. (IPTPS'05).

ROBINSON, D.; COAR, K. **The Common Gateway Interface (CGI) Version 1.1**. [S.l.]: Internet Engineering Task Force, 2004. RFC. (3875). Disponível em: <http://www.rfc-editor.org/rfc/rfc3875.txt >.

SALVADOR, P.; NOGUEIRA, A. Study on geographical distribution and availability of BitTorrent peers sharing video files. In: IEEE INTERNATIONAL SYMPOSIUM ON CONSUMER ELECTRONICS, 2008., Washington, DC, USA. **Proceedings** IEEE Computer Society, 2008. p.1–4. (ISCE 2008).

SAROIU, S.; GUMMADI, K. P.; DUNN, R. J.; GRIBBLE, S. D.; LEVY, H. M. An analysis of Internet content delivery systems. **SIGOPS Operating Systems Review**, New York, NY, USA, v.36, n.SI, p.315–327, Dec. 2002.

SAROIU, S.; GUMMADI, K. P.; GRIBBLE, S. D. A Measurement Study of Peer-to-Peer File Sharing Systems. In: MULTIMEDIA COMPUTING AND NETWORKING, 2002., Bellingham, WA, USA. **Proceedings** SPIE, 2002. n.1, p.156–170. (MMCN'02, v.1).

SCHMIDT, A.; ANTUNES, R.; BARCELLOS, M.; GASPARY, L. P. Characterizing Dissemination of Illegal Copies of Content through Monitoring of BitTorrent Networks. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2012., Washington, DC, USA. **Proceedings** IEEE Computer Society, 2012. (NOMS 2012).

SCHMIDT, A.; BARCELLOS, M.; GASPARY, L. P. Rumo à Caracterização de Disseminação Ilegal de Filmes em Redes BitTorrent. In: XI SIMPóSIO BRASILEIRO EM SEGURANçA DA INFORMAçãO E SISTEMAS COMPUTACIONAIS, Porto Alegre, RS. **Anais** SBC, 2011. (SBSEG 2011).

SCHULZE, H.; MOCHALSKI, K. **Internet Study 2008/2009**. [S.l.]: Ipoque, 2009. Disponível em: <https://portal.ipoque.com/ >.

SILVERSTON, T.; FOURMAUX, O.; CROWCROFT, J. Towards an Incentive Mechanism for Peer-to-Peer Multimedia Live Streaming Systems. In: INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 8., Washington, DC, USA. **Proceedings** IEEE Computer Society, 2008. p.125 – 128. (P2P'08).

STUTZBACH, D.; REJAIE, R. Understanding churn in peer-to-peer networks. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT, 6., New York, NY, USA. **Proceedings** ACM, 2006. (IMC'06).

STUTZBACH, D.; REJAIE, R.; DUFFIELD, N.; SEN, S.; WILLINGER, W. On Unbiased Sampling for Unstructured Peer-to-Peer Networks. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.17, n.2, p.377 – 390, Apr 2009.

STUTZBACH, D.; REJAIE, R.; SEN, S. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.16, n.2, p.267 – 280, Apr 2008.

YOSHIDA, M.; NAKAO, A. Measuring BitTorrent swarms beyond reach. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2011., Washington, DC, USA. **Proceedings** IEEE Computer Society, 2011. p.220–229. (P2P'11).

ZHANG, C.; DHUNGEL, P.; WU, D.; ROSS, K. Unraveling the BitTorrent Ecosystem. **IEEE Transactions on Parallel and Distributed Systems**, Washington, DC, USA, v.PP, n.99, p.1, 2011.

ZHAO, S.; STUTZBACH, D.; REJAIE, R. Characterizing files in the modern gnutella network: a measurement study. In: MULTIMEDIA COMPUTING AND NETWORKING, 2006., Bellingham, WA, USA. **Proceedings** SPIE, 2006. n.1, p.35–50. (MMCN'06, v.13).

# APÊNDICE A - MODELO DE PROGRAMAÇÃO

```
###  Conjuntos
set SOURCE;     # Conjunto de fontes.
set UNIT ;      # Conjunto de unidades.
set ATTRIBUTE;  # Conjunto de ATTRIBUTE.
set RESOURCE;   # Conjunto de recursos.

###  Propriedades do tipo de rede
param R{UNIT, RESOURCE};                # Required
param N{ATTRIBUTE, UNIT} integer;       # Instance
param P{ATTRIBUTE, UNIT} binary;        # Prerequesite
param C{ATTRIBUTE, SOURCE} binary;      # Concern

### Propriedades do estado do monitoramento
param W{ATTRIBUTE}      # Crawled
param M{SOURCE};        # Monitored
param S{a in ATTRIBUTE} integer := sum{i in SOURCE} C[a,i]*M[i] ;        # State

 ### Métricas do monitoramento
param A_obj{ATTRIBUTE} binary;  # Richness
param B_obj;                    # Coverage
param H{RESOURCE};              # Cost


 ### Variáveis
var l{ u in UNIT, a in ATTRIBUTE} integer >= 0; # Instancias lidas
var b_plan >=0.0 <=1.0;                         # % monitorado
var o{ u in UNIT} integer >= 0;                 # Qtde de unidades
# neste caso, o conjunto de variáveis a seguir é parâmetro
param a_plan{a in ATTRIBUTE}  := A_obj[a];      # Riqueza

### Função objetivo ###
maximize Coverage:   b_plan;

###  Restrições  ###
 # Riqueza mínima
RichMin{a in ATTRIBUTE}:
        a_plan[a] >=  A_obj[a];

# Cobertura mínima
CovMin:
        b_plan >= B_obj;

# Custo máximo
 CostMax {e in RESOURCE}:
        sum{u in UNIT} o[u] * R[u,e] <= H[e] ;

# Cobertura proporcional
CovRich {a in ATTRIBUTE}:
        sum{u in UNIT} l[u,a] >=  a_plan[a] * b_plan * S[a];

# ATTRIBUTE pré-requisitos
PreReq{u in UNIT, a in ATTRIBUTE}:
     o[u]*P[a,u] <=  a_plan[a] * 1000000000;

# ATTRIBUTE lidos
Ob  {u in UNIT, a in ATTRIBUTE}:
```

```
        l[u,a] <=  o[u]*N[a,u];

# ATTRIBUTE existentes
CovMin{a in ATTRIBUTE}:
        sum{u in UNIT} l[u,a] <=    S[a];

solve;
end;
```

# APÊNDICE B - DADOS DO MODELO

```
param : RESOURCE :        H :=
                down    10000
                conn    10000
        ;

param : ATTRIBUTE :     A_obj :=
Infohash        0
Leechers        0
Seeders         0
Content         0
Votes           0
Encoding        0
Creation_date   0
Publication_date 0
Comment         0
Published_by    0
Created_by      0
File_names      0
File_sizes      0
File_md5sum     0
File_paths      0
Number_of_pieces 0
Piece_hash      0
Piece_lenght    0
Announce_Swarm  0
Peer_address    0
Peer_address_removed 0
Peer_connected          0
Peer_reachable  0
Peer_id         0
Peer_client     0
Extension       0
Bitfield        0
Tracker_address 0
Announce_interval 0
Failure_reason  0
Warning_message 0
Scrape_interval 0
        ;

param : SOURCE :        M :=
                swarm   100000
                tracker 1
                peer    5668009
        ;

param  B_obj := 1.000000 ;

param   R :  down       conn    :=
        ht              30       1
        tr              25       1
        an              0.5     1
        sc              50176   1
        ha              0.3     1
        pe              0.3     1
        ;
```

```
param C:        tracker swarm   peer :=
Infohash        0       1       0
Leechers        0       1       0
Seeders 0       1       0
Content 0       1       0
Votes   0       1       0
Encoding        0       1       0
Creation_date   0       1       0
Publication_date        0       1       0
Comment 0       1       0
Published_by    0       1       0
Created_by      0       1       0
File_names      0       1       0
File_sizes      0       1       0
File_md5sum     0       1       0
File_paths      0       1       0
Number_of_pieces        0       1       0
Piece_hash      0       1       0
Piece_lenght    0       1       0
Announce_Swarm  0       1       0
Peer_address    0       0       1
Peer_address_removed    0       0       1
Peer_connected  0       0       1
Peer_reachable  0       0       1
Peer_id 0       0       1
Peer_client     0       0       1
Extension       0       0       1
Bitfield        0       0       1
Tracker_address 1       0       0
Announce_interval       1       0       0
Failure_reason  1       0       0
Warning_message 1       0       0
Scrape_interval 1       0       0
        ;

 param N :  ht          tr  sc an ha  pe  :=
Infohash        1       1       730708  0       0       0
Leechers        1       0       730708  1       0       0
Seeders 1       0       730708  1       0       0
Content 1       1       0       0       0       0
Votes   1       0       0       0       0       0
Encoding        0       1       0       0       0       0
Creation_date   0       1       0       0       0       0
Publication_date        1       0       0       0       0       0
Comment 0       1       0       0       0       0
Published_by    1       0       0       0       0       0
Created_by      0       1       0       0       0       0
File_names      0       1       0       0       0       0
File_sizes      0       1       0       0       0       0
File_md5sum     0       1       0       0       0       0
File_paths      0       1       0       0       0       0
Number_of_pieces        0       1       0       0       0       0
Piece_hash      0       1       0       0       0       0
Piece_lenght    0       1       0       0       0       0
Announce_Swarm  0       0       0       1       0       0
Peer_address    0       0       0       200     0       25
Peer_address_removed    0       0       0       0       0       25
Peer_connected  0       0       0       0       1       0
Peer_reachable  0       0       0       0       1       0
Peer_id 0       0       0       0       1       0
Peer_client     0       0       0       0       1       0
Extension       0       0       0       0       1       0
Bitfield        0       0       0       0       1       0
Tracker_address 5       5       0       0       0       0
Announce_interval       0       0       0       1       0       0
Failure_reason  0       0       1       1       0       0
Warning_message 0       0       1       1       0       0
Scrape_interval 0       0       1       0       0       0
        ;
```

```
 param P :       ht     tr  sc an ha pe :=
Infohash         0       0       0       1       0       0
Leechers         0       0       0       0       0       0
Seeders          0       0       0       0       0       0
Content 0        0       0       0       0       0
Votes   0        0       0       0       0       0
Encoding         0       0       0       0       0       0
Creation_date    0       0       0       0       0       0
Publication_date         0       0       0       0       0       0
Comment 0        0       0       0       0       0
Published_by     0       0       0       0       0       0
Created_by       0       0       0       0       0       0
File_names       0       0       0       0       0       0
File_sizes       0       0       0       0       0       0
File_md5sum      0       0       0       0       0       0
File_paths       0       0       0       0       0       0
Number_of_pieces         0       0       0       0       0       0
Piece_hash       0       0       0       0       0       0
Piece_lenght     0       0       0       0       0       0
Announce_Swarm   0       0       0       0       0       0
Peer_address     0       0       0       0       1       1
Peer_address_removed     0       0       0       0       0       0
Peer_connected   0       0       0       0       0       1
Peer_reachable   0       0       0       0       0       0
Peer_id 0        0       0       0       0       0
Peer_client      0       0       0       0       0       0
Extension        0       0       0       0       0       0
Bitfield         0       0       0       0       0       0
Tracker_address  0       0       0       1       0       0
Announce_interval        0       0       0       0       0       0
Failure_reason   0       0       0       0       0       0
Warning_message  0       0       0       0       0       0
Scrape_interval  0       0       0       0       0       0
        ;
end;
```

# APÊNDICE C - ARTIGO PUBLICADO NO SBRC

Neste anexo é apresentado o artigo “Observando o Universo BitTorrent Através de Telescópios", desenvolvido durante o primeiro ano do mestrado. O artigo apresenta como contribuições um apanhado sobre estratégias de monitoramento, um modelo de gerenciamento de informações do universo BiTorrent e, principalmente, uma arquitetura de monitoramento desse universo.

- Título: Observando o Universo BitTorrent Através de Telescópios
- Conferência: XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2010)
- URL: http://sbrc2010.inf.ufrgs.br/
- Data: 24 a 28 de maio de 2010
- Local: Gramado, RS, Brasil

# Observando o Universo BitTorrent Através de Telescópios

**Rodrigo Brandão Mansilha, Alan Mezzomo, Giovani Facchini, Luciano Paschoal Gaspary, Marinho Pilla Barcellos**

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

***Resumo.*** *Trabalhos recentes na literatura indicam que o BitTorrent é o protocolo de compartilhamento de arquivos com maior popularidade, sendo responsável por mais de 45-78% de todo tráfego par-a-par, dependendo da localização geográfica. Apesar da importância dessas estatísticas e o crescente interesse por "redes baseadas em enxames", ainda pouco se sabe sobre a dinâmica do ecosistema BitTorrent, principalmente devido à natureza limitada – em termos de cobertura, riqueza de detalhes e exatidão – dos métodos de monitoramento empregados para este fim. Para atender a esta demanda, propõe-se um modelo de informações de gerenciamento e uma arquitetura de monitoramento que permite a combinação flexível de diferentes estratégias para observar o "Universo BitTorrent". Para avaliar a viabilidade conceitual e técnica, implementou-se um protótipo da arquitetura, e o mesmo foi usado para instanciar tarefas de monitoramento que combinam diferentes subconjuntos de estratégias.*

***Abstract.*** *Recent analysis of the latest peer-to-peer (P2P) trends worldwide shows that BitTorrent is the most popular file sharing protocol, responsible for more than 45-78% of all peer-to-peer P2P traffic, depending on geographical location. Despite the importance of these statistics and the growing interest for swarm-based networked systems, very little is known about the dynamics of the BitTorrent "ecosystem", mainly due to the limited nature – in terms of coverage, accuracy and richness of detail – of the monitoring methods designed to this end. To tackle this issue, we propose a management information model and a monitoring architecture that allows a flexible combination of strategies to observe the "BitTorrent Universe". To show concept and technical feasibility, we have implemented a prototypical implementation of the architecture, and as an example instantiated monitoring tasks combining a different subset of monitoring strategies.*

## 1. Introdução

Redes *Peer-to-Peer* (P2P) permitem que recursos sejam compartilhados entre usuários de forma eficiente e escalável. Dentre as classes de aplicações P2P, a de compartilhamento de arquivos é provavelmente a mais popular atualmente [Karakaya et al. 2009]. Incontestáveis exemplos de sucesso na Internet, entre eles o BitTorrent, têm sido acompanhados com grande interesse por parte da comunidade científica.

Apesar de nos últimos anos terem sido publicados muitos estudos sobre redes BitTorrent ([Izal et al. 2004, Pouwelse et al. 2005, Guo et al. 2007, Dale and Liu 2007, Zhang et al. 2009, Zhang et al. 2010]), ainda pouco se sabe sobre o real funcionamento delas e padrões de comportamento de seus usuários. Informações sobre compartilhamento de conteúdo em redes BitTorrent são úteis por uma série de razões. Por exemplo,

Este trabalho foi desenvolvido com suporte da Rede Nacional de Ensino e Pesquisa (RNP).

comercialmente, podem amparar campanhas de *marketing* baseadas em popularidade de conteúdo, estimar perdas financeiras com cópias ilegais [Chow et al. 2007] e auxiliar *Internet Service Providers* na investigação de métodos que minimizem o custo do tráfego BitTorrent. Também podem auxiliar instituições da justiça na detecção de certos tipos de ataques efetuados contra redes BitTorrent e no combate à pedofilia. Apesar desses benefícios potenciais, os métodos existentes para observação de redes BitTorrent deixam a desejar. As informações divulgadas frequentemente são pobres em detalhe, cobertura (quantidade de redes e usuários) e/ou exatidão. Além disso, a inexistência de um modelo de informações do BitTorrent dificulta intercâmbio de dados entre a comunidade científica.

Três fatores explicam a atual falta de conhecimento sobre a dinâmica do *universo BitTorrent*. Primeiro, sua escala planetária, onde milhões de usuários operam em nível de aplicação. A dimensão, complexidade, heterogeneidade e incerteza de tal universo provêem oportunidades limitadas de observação constituindo obstáculos para monitoração efetiva. Segundo, em contraste com outras redes P2P de compartilhamento de arquivos como Gnutella e Kazaa, no BitTorrent a rede é composta por milhões de redes sobrepostas (*overlays*) desconexas menores denominadas "enxames", sendo o primeiro desafio "chegar" a cada enxame de interesse. Por último, o protocolo BitTorrent nasceu de uma implementação (por Cohen) que foi sendo incrementada, estendida e testada por usuários, ao invés do uso de uma metodologia formal ou especificação. Isto levou à criação de diversas implementações distintas de agentes de usuário e extensões ao protocolo. Consequentemente, para monitorar o Universo BitTorrent efetivamente é necessário contemplar os diversos agentes e protocolos e acompanhar suas mudanças.

Portanto, antes que as informações sobre as redes BitTorrent possam ser extraídas consistente e confiavelmente, desafios de pesquisa devem ser superados. Neste contexto, este trabalho apresenta como contribuição principal a proposta de uma arquitetura flexível para monitoramento eficiente do "universo BitTorrent". Foram identificadas e acopladas à arquitetura diversas estratégias de observação, que flexivelmente podem ser combinadas de acordo com as informações desejadas, visando a obtenção eficiente de resultados.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta os elementos do universo BitTorrent e discute estratégias de monitoramento. As mesmas servem de base para o projeto de uma arquitetura, descrita na Seção 3. A Seção 4 resume aspectos práticos de monitoramento mais relevantes, enquanto a Seção 5 detalha a metodologia experimental usada (com enxames reais na Internet) e discute os resultados. A Seção 6 apresenta trabalhos relacionados a este, e a Seção 7 encerra o artigo com considerações finais e propostas para trabalhos futuros.

## 2. Estratégias para Monitoramento do Universo

Esta seção define o Universo BitTorrent (Subseção 2.1) e identifica estratégias para monitoramento dos seus componentes (Subseção 2.2). A seguir, discute os custos das estratégias (Subseção 2.3) e argumenta sobre vantagens e desvantagens de cada uma (Subseção 2.4).

### 2.1. Universo BitTorrent

O universo BitTorrent é composto por **enxames**, **pares**, **rastreadores** e **conteúdos**. Um par é um agente de usuário que executa o protocolo e participa de um ou mais enxames, de acordo com o conteúdo que deseja compartilhar. Um par é nomeado semeador, quando

possui conteúdo completo, ou sugador, caso contrário. Para ingressar em um enxame, o par tipicamente contata o rastreador e como resposta recebe uma lista de IPs de pares (*peer list*) que participam do enxame. Portanto, o rastreador atua como ponto de encontro. Alternativamente, existem duas extensões do protocolo que vêm sendo amplamente adotadas. A primeira, denominada *Peer Exchange* (PEX), permite que pares façam o intercâmbio de lista de pares, e a segunda, geralmente chamada de *"trackerless"*, permite que os pares se encontrem através de tabelas *hash* distribuídas (DHT). No âmbito deste trabalho, por "conteúdo"subentende-se "conteúdo digital", que é diferente de "conteúdo informacional". Enquanto o primeiro significa um conjunto específico de arquivos digitais, o segundo significa a informação disponibilizada nos arquivos. Por exemplo, dois torrents podem se referir à um mesmo conteúdo informacional, porém à conteúdos digitais diferentes, como no caso de torrents de uma mesma música codificada em duas taxas de compressão diferentes. O conteúdo compartilhado é organizado em peças. Uma descrição detalhada do protocolo pode ser encontrada em [Konrath et al. 2007].

Para participar de um enxame, um agente de usuário utiliza os metadados disponíveis no respectivo arquivo torrent. Esse arquivo contém informações sobre as peças (para cada peça, seu *hash* e tamanho) que formam o conteúdo e arquivos (nomes e tamanhos). Para distribuir um conteúdo, um par deve gerar um torrent e torná-lo público. Os torrents estão tipicamente disponíveis em sítios dedicados a promover o compartilhamento de arquivos em redes BitTorrent, os quais chamamos de "comunidades". Um exemplo dessas comunidades é a IsoHunt. Além de publicar torrents, algumas comunidades disponibilizam rastreadores. As comunidades podem ser do tipo "aberta" ou "fechada", sendo que neste último caso o acesso é restrito a membros cadastrados. Algumas comunidades atuam primariamente como agregadoras de torrents, indexando informações para permitir buscas e apontando para torrents em outras comunidades.

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

### 2.2. Estratégias

Há diferentes estratégias para extrair informações do universo BitTorrent. Pode-se dividi-las em três grupos de acordo com a fonte de informação.

**Comunidades**. De maneira geral, as comunidades BitTorrent existem para fornecer arquivos torrent e permitir a interação entre usuários com os mesmos interesses. Comunidades apresentam variados tipos de informação relacionados aos conteúdos, rastreadores e quantidade de pares. Trabalhos como [Andrade et al. 2005, Pouwelse et al. 2005] são exemplos que aplicam a estratégia de monitorar comunidades para obter uma visão ampla do universo.

**Rastreadores**. Uma outra estratégia é monitorar os rastreadores, que disponibilizam informações mais atualizadas e precisas que as comunidades, assim como in-

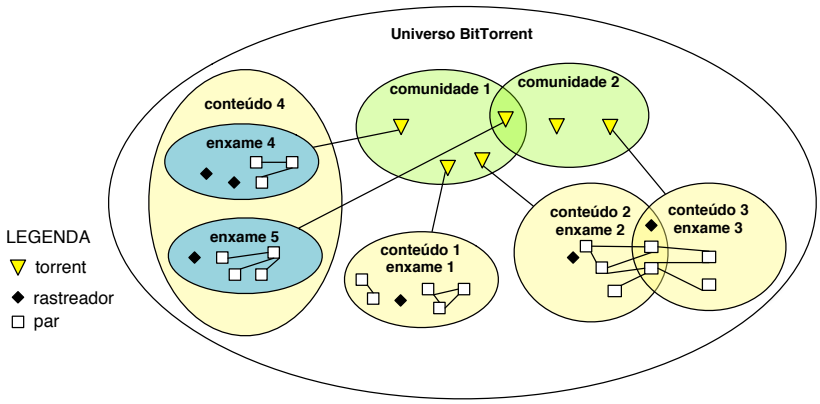



**Figura 1. Exemplo de componentes e relações no universo de redes BitTorrent**

formações sobre si e sobre os pares (potencialmente apenas um conjunto de endereços IP). Na literatura, são encontradas três variações desta estratégia. A primeira é baseada na análise de *logs* do rastreador, quando disponíveis; um exemplo é encontrado em [Izal et al. 2004]. A segunda variação da estratégia é observar continuamente o conteúdo das páginas Web dinamicamente geradas pelos rastreadores, quando disponíveis; um exemplo é [Andrade et al. 2005]. A terceira variação é registrar-se nos rastreadores usando agentes modificados; um exemplo é encontrado em [Pouwelse et al. 2005]. A quantidade de informação disponibilizada pelo rastreador de uma comunidade dependerá do tipo de conteúdo que está sendo compartilhado e se a comunidade é aberta ou fechada.

**Pares**. A terceira e última estratégia é monitorar os pares diretamente. Existem duas abordagens, uma ativa e outra passiva. A abordagem passiva para monitorar redes P2P é usada em [Saroiu et al. 2002, Horng et al. 2006], e consiste em capturar e processar pacotes que trafegam nos *enlaces* de rede utilizados pelos pares observados. Em contraste, a abordagem ativa se baseia em instrumentar agentes de usuário que se conectam com pares e extraem informações. Há três variações para a abordagem ativa, todas com o mesmo requisito para implementação e utilização: receber como entrada informações sobre os enxames, ou seja, o endereço IP dos pares do enxame ou, alternativamente, do rastreador, a partir do qual pode-se obter endereços de pares. As três variações diferenciam-se quanto ao nível de mensagens trocadas: na primeira ([Pouwelse et al. 2005, Iosup et al. 2006]), o agente conecta com pares remotos, efetua o *handshake*, troca o mapa de peças e encerra a conexão em seguida; na segunda, mantém conexão com os pares, mas não efetua troca de dados; e na terceira ([Legout et al. 2007]), um par participa ativamente do enxame, inclusive compartilhando dados.

## 2.3. Custo das Estratégias

As estratégias apresentam diferentes custos e possibilidades de configuração. O custo pode ser mensurado, por exemplo, em termos de utilização de memória, carga de CPU e recursos de rede.

O custo de cada combinação de estratégias é estimado como o produto do custo das escolhas feitas em relação a cobertura, riqueza de detalhe e exatidão. A **cobertura** refere-se ao número de elementos a serem observados, à localização geográfica (por exemplo, todos os pares da Alemanha), intervalo de tempo (por exemplo, todos os torrents publicados no mês) ou tipo de conteúdo (por exemplo, somente livros). A **riqueza de detalhe** corresponde ao número de atributos de interesse, isto é, às variáveis a serem

coletadas; alguns exemplos são o número de pares envolvidos, número e taxas de *downloads*, porção de pares que têm suporte a uma dada característica como criptografia ou PEX, etc. Por último, a **exatidão** representa a qualidade das informações extraídas das redes BitTorrent. A exatidão é afetada por dois parâmetros principais: frequência (isto é, período em que amostras estão sendo coletadas) e tamanho da amostra (relativo ao percentual da população consultada).

### 2.4. Escolha de Estratégias

Limitar o monitoramento às comunidades traz como vantagem mensurável a economia de recursos de rede, além da não intrusividade no enxame, pois o custo de rede associado a este monitoramento refere-se apenas ao *download* de um pequeno número de páginas html, resultando em poucos kilobytes. Entre as desvantagens está a ausência de informações específicas sobre os pares, que as demais estratégias são capazes de obter. Mesmo que a quantidade de pares que compartilham determinado torrent seja mostrada, nada pode ser presumido, visto que a frequência de atualização dessa informação não é publicada (está potencialmente desatualizada).

A vantagem de monitorar rastreadores em comparação com o monitoramento de comunidades é a possibilidade de identificar a população de pares, o que aumenta o nível de riqueza significativamente. Em comparação com monitoramento de pares, as vantagens são a menor intrusividade e o menor custo para obtenção de informações. Entre suas desvantagens, há a ausência de informações relativas ao conteúdo compartilhado e informações detalhadas sobre o estado atual dos pares e das peças disponíveis. A exatidão é relativamente baixa, pois o tempo com que o rastreador é periodicamente contatado por um par é tipicamente quinze minutos, podendo passar de uma hora.

A maior vantagem de se monitorar os pares em comparação com as outras estratégias é alcançar o maior nível de riqueza de detalhe possível. Entretanto, monitorar pares é a estratégia que implica o maior custo. Como existem duas variantes desta estratégia, monitoração passiva e ativa, elas são comparadas a seguir. A principal vantagem de se usar monitoramento passivo é a menor intrusividade no enxame; além disso, o único custo de rede vem da transferência de logs para a "estação de gerenciamento". As desvantagens são necessitar poder computacional suficiente para inspecionar todos os dados gravados, a impossibilidade virtual de identificação de tráfego criptografado e a dificuldade para se instanciar pontos de monitoramento passivos em todas as sub-redes desejáveis. Logo, o monitoramento passivo é mais adequado a ambientes menores e controlados, como dentro de um sistema autônomo. Para todas as outras situações, o uso de agentes de usuário modificados é provavelmente mais apropriado.

Dentre as estratégias, a primeira e a segunda demandam apenas acesso aos sítios e ao rastreador, o que tipicamente não apresenta dificuldades. Por outro lado, seus resultados podem deixar a desejar em riqueza de detalhes. A terceira estratégia, monitorar diretamente pares, tende a ser a mais rica em informações, porém é em princípio a menos escalável por demandar o estabelecimento de conexões com pares. Individualmente, elas representam uma parte do problema atacado neste trabalho, pois tipicamente limitam-se a partes específicas do universo. Se combinadas, as estratégias podem fazer parte de uma solução mais completa.

O TorrentU combina estratégias que possibilitam a extração das informações do universo BitTorrent. O ponto principal que difere este trabalho dos demais (mais detalhes na Seção 6, de trabalhos relacionados) é a flexibilidade com respeito à cobertura, riqueza

de detalhe e exatidão das informações, permitindo que peças de informação sejam obtidas através da combinação ideal de um conjunto de estratégias.

## 3. Observação do Universo com TorrentU

Esta seção oferece uma visão geral do TorrentU. A Subseção 3.1 apresenta o modelo de informações projetado para gerenciar todos os dados do BitTorrent dentro do universo, e a Subseção 3.2 descreve a arquitetura conceitual da solução proposta.

### 3.1. Modelo de Informações

Um modelo de informações do universo BitTorrent foi criado para estruturar, ligar e racionalizar as informações sobre comunidades, torrents, rastreadores, pares, enxames, etc. Ele é baseado em um subconjunto do Common Information Model (CIM), definido pelo Distributed Management Task Force (DMTF), e no modelo par-a-par, proposto por Doyen *et al.* [Doyen et al. 2004]. Até onde se tem conhecimento, o presente artigo é o primeiro trabalho com intuito de formalizar e organizar, através de um modelo de informações, os vários conceitos associados ao sistema BitTorrent, compreendendo uma contribuição adicional deste trabalho.

A Figura 2 ilustra uma visão simplificada do modelo. As classes em branco contornadas com linhas sólidas e trastejadas designam, respectivamente, classes originadas do CIM e do modelo de Doyen *et al.*. Por outro lado, as classes em cinza são aquelas propostas para expressar os conceitos de BitTorrent e seus relacionamentos. Devido a restrições de espaço, não é possível detalhar as relações ilustradas na figura.

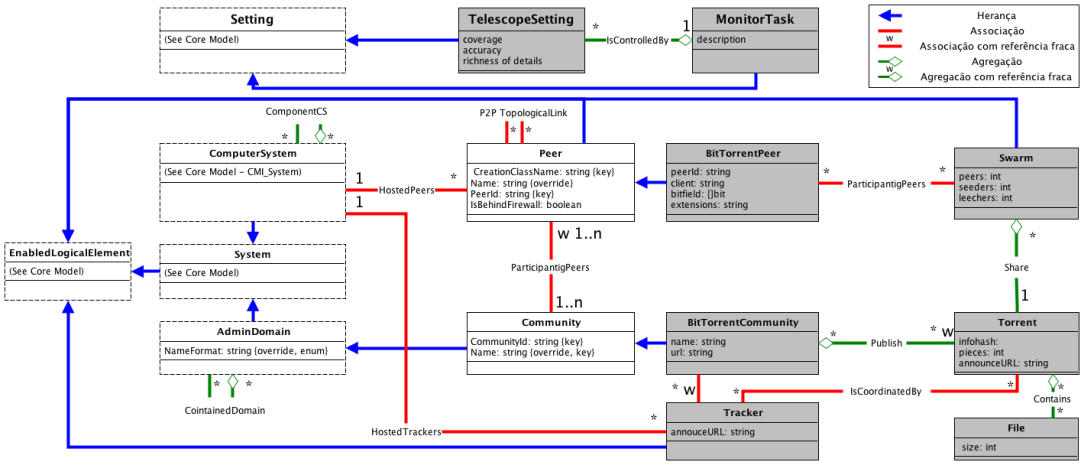



**Figura 2. Modelo de Informações do BitTorrent**

### 3.2. Arquitetura

A arquitetura é composta por dois componentes principais, denominados TorrentU Observer e TorrentU Telescope, que interagem entre si, com o usuário e com o universo, conforme a Figura 3.

**TorrentU Observer**. Este componente faz o papel de *front-end*, isto é, gerente da aplicação, permitindo que o operador configure o sistema e observe os dados coletados em tempo real (assim como o histórico dos dados). Além disso, o Observer deve determinar um conjunto de estratégias e parâmetros a serem utilizados em função de um conjunto objetivo de dados a serem coletados e um conjunto restritivo de recursos disponíveis.

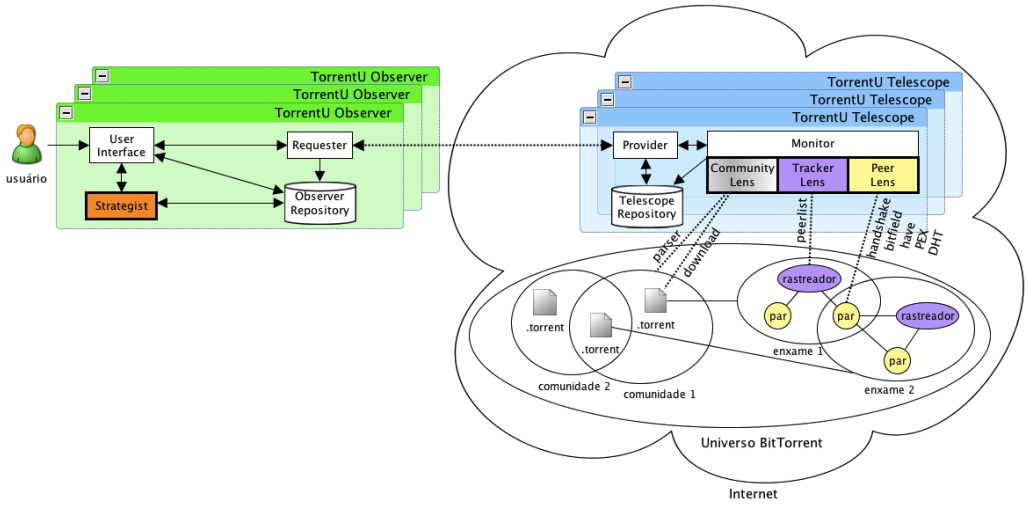



**Figura 3. Arquitetura *TorrentU***

O Observer é composto por quatro subcomponentes: User Interface, Requester, Strategist e Observer Repository. O User Interface é o subcomponente responsável pela interação com o usuário. O Requester é a interface de comunicação com o Telescope. O Strategist ajuda a determinar o melhor conjunto de estratégias e atributos, dada uma certa tarefa de monitoramento. Por fim, o Observer Repository armazena, seguindo o modelo de informações apresentado na subseção anterior, os resultados obtidos pelo Telescope, mantendo-os disponíveis para o User Interface. Esse repositório é particularmente importante para dados de séries temporais, pois "fotografias" são periodicamente criadas e precisam ser armazenadas para posterior recuperação e análise dos dados.

**TorrentU Telescope**. Este componente é responsável pelo monitoramento do universo BitTorrent e pelo retorno de resultados de acordo com as requisições enviadas pelo Observer. Ele é composto por três subcomponentes, denominados Provider, Telescope Repository e Monitor. O Provider é a interface de comunicação com o Observer. O Telescope Repository é responsável pelo armazenamento dos dados coletados no Telescope. A exemplo do que ocorre com o Observer Repository, ele também adota o modelo de informações apresentado na Subseção 3.1 como forma de persistir, de forma estruturada, os dados coletados. Por fim, o Monitor é o subcomponente que, de fato, contata os elementos do BitTorrent (portais de comunidades, rastreadores e pares). O Monitor é subdividido em três partes, denominadas "lentes" (*lens*), sendo cada uma responsável por monitorar um grupo diferente de elementos do universo: *Community Lens*, *Tracker Lens* e *Peer Lens*. Essa modularização permite que as lentes existentes possam ser substituídas, assim como novas possam ser facilmente incorporadas na arquitetura (sem modificação de seus componentes essenciais).

Uma grande desafio por trás do monitoramento do universo BitTorrent é criar planos de monitoramento que gerenciem o compromisso entre os objetivos de observação e recursos computacionais disponíveis, balanceando adequadamente riqueza de detalhe, cobertura e exatidão. Este desafio é tratado pelo Observer. Devido à natureza complexa do problema e restrição de espaço, este trabalho é focado na operação do Telescope e suas lentes, as quais são detalhadas na próxima seção.

## 4. Lentes do Telescópio

Conforme discutido na seção anterior, o Telescope é formado por três componentes. O Telescope Repository e o Provider são implementações típicas de repositórios e interfaces de serviços web, respectivamente (portanto, maiores detalhes são omitidos). Em contraste, é importante descrever o Monitor, pois ele é específico deste trabalho e implementa as estratégias discutidas na Seção 2. Nas próximas subseções são apresentadas as lentes do Monitor.

### 4.1. Community Lens

O Community Lens monitora a publicação de arquivos torrents em um sítio da web. Conforme comentado anteriormente, informações de interesse podem ser extraídas de páginas da Internet ou de arquivos torrent. Para monitorar páginas é necessário o desenvolvimento de *parsers* específicos para cada comunidade, como por exemplo IsoHunt. Definiu-se uma estrutura bem clara de maneira a diminuir o custo de elaboração de novos *parsers*.

Um *crawler* focado é utilizado para se obter arquivos torrent de comunidades e sítios que possuem mecanismo de busca de torrents. Normalmente, todos os torrents de um sítio da web seriam gradualmente transferidos, começando pelos publicados mais recentemente até um certo ponto nos mais antigos. Entretanto, para reduzir a quantidade total de torrents a serem carregados de um dado sítio, filtros dessas comunidades são usados para reduzir o conjunto de torrents a serem recuperados. O processo de varredura e transferência dos arquivos torrent ocorre em rodadas. A cada rodada, os dados coletados são adicionados ou atualizados no Telescope Repository.

O Community Lens é capaz de limitar a cobertura utilizando três classes de filtros: conteúdo (tipo), quantidade (número de torrents) e localidade (país da comunidade). Em termos de riqueza de detalhe, o Community Lens é o nível mais superficial em termos de riqueza de detalhes e subdivide-se em dois: no primeiro sub-nível, os sítios são apenas percorridos e listas de torrents, obtidas; no segundo sub-nível, os torrents são de fato carregados.

### 4.2. Tracker Lens

Os rastreadores são monitorados pelo Tracker Lens. Ele é usado para extrair a quantidade de pares (sugadores e semeadores), endereços IP e portas dos pares, além de informações sobre os próprios rastreadores.

O Tracker Lens se anuncia ao rastreador, que responde com um subconjunto aleatório de sua lista completa de pares. Devido à escalabilidade, os rastreadores impõem um limite à frequência dos pedidos feitos por um mesmo par. Por isso, se a frequência pretendida é maior que a permitida pelo rastreador, é necessário instanciar múltiplas identidades para realizar o anúncio.

O contato com o rastreador pode falhar devido à indisponibilidade do mesmo ou problemas na rede. Além disso, o enxame pode estar sem novos pares ou com zero pares. Portanto, são estabelecidas regras para determinar a frequência dos contatos em função das respostas do rastreador (verificando sua disponibilidade) e do conteúdo retornado (estimando a existência de pares desconhecidos). Dessa forma, o Tracker Lens pode direcionar seus recursos para os rastreadores que podem confiavelmente informar mais pares novos.

### 4.3. Peer Lens

Como o nome indica, o Peer Lens observa pares. Contatando-os, é possível extrair uma série de informações importantes, como o nível de popularidade de determinado agente de usuário, assim como inferir valores como taxas de *download*.

O Peer Lens é um agente de usuário BitTorrent modificado. Para aumentar sua escalabilidade, o Peer Lens usa um esquema *round-robin*: a cada rodada, a lente tenta estabelecer conexões com $p$ pares; uma vez abertas, as conexões são mantidas por um tempo $t$, aguardando mensagens HAVE[1]. Ao término do período, as conexões são encerradas, e o processo reinicia com os próximos $p$ pares da fila circular. Logo que uma conexão é estabelecida, coleta-se informações sobre o par correspondente, como por exemplo versão do agente e disponibilidade e quantidade de peças concluídas.

Pares que não aceitam o estabelecimento de uma conexão remota, devido a *firewall* ou outra forma de proteção, são marcados como inalcançáveis. Para estes casos, o monitor precisa aguardar até que os inalcançáveis iniciem uma conexão com o Peer Lens. Para aumentar a probabilidade disso acontecer, utiliza-se uma estratégia inspirada em *Sybils* [Douceur 2002]. A lente cria e controla múltiplas identidades lógicas, anunciando-as ao rastreador. Dessa forma, aumenta-se a chance de que os inalcançáveis recebam pelo menos um dos IPs correspondentes ao Peer Lens, portanto aumentando a chance de conexão. Entretanto, essa técnica possui um custo associado: ao anunciar falsas identidades para os rastreadores, é possível que um ou mais pares tentem abrir múltiplas conexões com a lente. Essas conexões seriam rejeitadas, mas ainda assim passariam pelos processos de abertura e fechamento.

A cobertura é ajustada no Peer Lens filtrando-se os pares ou limitando-se a quantidade de pares ($p$). Além da cobertura, o Peer Lens é flexível em termos de exatidão, onde o valor de $t$ utilizado na janela deveria ser ajustado para aumentar ou diminuir a qualidade das informações. No limite inferior, $t = 0$ significa que a conexão é imediatamente fechada depois do *handshake*. No limite superior, $t = \infty$, a conexão é mantida aberta até o par remoto desconectar (ou falhar) ou o Peer Lens ser finalizado. O estabelecimento de conexão em nível de BitTorrent (mensagem HANDSHAKE) permite coletar uma série de informações, como por exemplo o agente sendo utilizado e a atual disponibilidade das peças. Após esta etapa, em geral, quanto maior o tempo de conexão, maior a exatidão sobre a disponibilidade de peças nos pares e mais precisa é a estimativa de velocidade de *download* do par. Além disso, se o par remoto implementa a extensão PEX, é possível trocar listas de pares e estimar a conectividade entre eles.

## 5. Avaliação

A arquitetura descrita na seção anterior foi materializada na forma de um protótipo, implementado em Java 1.6. A biblioteca Java NIO foi empregada para permitir um grande número de conexões em paralelo, assim como comunicação não-bloqueante. Os dados coletados foram armazenados em um SGBD Mysql através de um conjunto de tabelas que mapeiam o modelo de gerenciamento de informações.

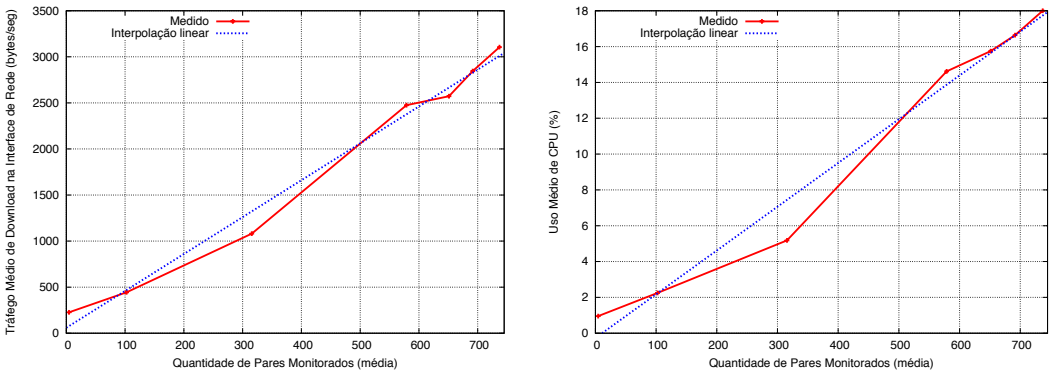
Esta seção oferece uma avaliação do TorrentU e está organizada como segue. A Subseção 5.1 apresenta evidências em favor do TorrentU, considerando o atendimento aos requisitos propostos. A Subseção 5.2 discute os resultados obtidos para um estudo de caso.

[1] um par envia uma mensagem HAVE para todos os pares vizinhos conectados quando uma peça é completada.

### 5.1. Atendimento aos Requisitos

Foram realizados experimentos com o objetivo de avaliar a **escalabilidade**, através da demanda de recursos em função da quantidade de componentes observados. Em particular, nas Figuras 4(a) e 4(b) apresenta-se, respectivamente, o consumo médio de tráfego (*download*) e carga média de processamento na CPU em função do número médio de pares conectados.

Observa-se que as quantidades de largura de banda de *download* e de uso de CPU podem ser interpoladas por uma função linear do número de pares conectados. Embora omitidos neste trabalho, resultados similares foram obtidos para taxa de *upload* e uso de memória. A partir desses resultados, conclui-se que a quantidade de recursos necessários aumenta linearmente em função do número médio de pares conectados. Portanto, TorrentU é escalável verticalmente mesmo quando a estratégia mais custosa é empregada. Além disso, o TorrentU é escalável horizontalmente, pois é possível dividir a carga entre múltiplos Telescopes executados em paralelo, em máquinas espalhadas pela rede.




(a) Download (b) CPU

**Figura 4. Ocupação de recursos**

Em um cenário onde os recursos não podem ser incrementados, uma alternativa para aumentar a cobertura do monitoramento é melhorar sua eficiência diminuindo (até mesmo anulando) o coeficiente angular da função de custo, que é linear. Conforme será visto a seguir, no TorrentU esta possibilidade é explorada ao máximo através da sua flexibilidade.

O TorrentU é **flexível** porque oferece uma gama de parâmetros para riqueza de detalhes, cobertura e exatidão. Em termos de riqueza, existem 4 níveis de custo. Em termos de cobertura, foram definidos 3 filtros de comunidade, 2 filtros de rastreador, 3 filtros de pares, e há a opção de definir novos. Em termos de exatidão, existem 2 parâmetros para cada nível de riqueza: periodicidade (com que os elementos são consultados) e tamanho da amostra (percentual do conjunto observado). Portanto, são $3 \times 2\ (comunidade) + 3 \times 2\ (torrent) + 2 \times 2\ (tracker) + 3 \times 2\ (peer) = 22$ parâmetros que, se multiplicados pelos valores possíveis, resultam em uma infinidade de opções.

Para um determinado objetivo de monitoramento, é possível usar estratégias que sejam mais **eficientes**. Para demonstrar essas possibilidades, foi analisada a quantidade total de *download* necessária para um cenário. Através de medições, foram obtidos os seguintes valores: arquivo torrent, 20 KB; lista com 50 pares a partir do rastreador, 467 bytes; mensagem HAVE de um par, 7 bytes; mensagem HANDSHAKE de um par, 136 bytes, e;

mensagem BITFIELD de um par, 180 bytes. Considere o seguinte caso de monitoramento hipotético: quantidade de pares, 100.000, quantidade de torrents, 100 e quantidade de peças por torrent, 500. Neste cenário, caso a pergunta seja "quais os conteúdos compartilhados?", para a qual o monitoramento das comunidades é suficiente, o custo possível varia entre 2 MB (realizando *download* de torrents apenas) e 384 MB (considerando também todas as mensagens do protocolo). O TorrentU é eficiente porque é flexível no uso de diferentes estratégias, e por isso pode minimizar o custo nas mais diversas situações. No caso citado, apenas o Community Lens seria empregado para realizar o *download* dos torrents, evitando 382 MB de *download* desnecessário. Se por exemplo o monitoramento acima, com 100.000 pares, fosse realizado no intervalo de 2 dias, então as taxas de sobrecarga seriam apenas, respectivamente, 0,09 Kbps e 17,68 Kbps.

### 5.2. Estudo de Caso

Para realizar o estudo de caso, acompanhou-se a agenda de lançamento de seriados norte americanos, muito populares em comunidades BitTorrent. Os monitoramentos iniciaram com quatro horas de antecedência ao lançamento do conteúdo original. Os resultados apresentados a seguir são referentes ao nono episódio da oitava temporada do seriado denominado "Family Guy" nas comunidades BTJunkie, IsoHunt e TorrentDownloads. A observação foi feita em um período pouco maior que 22 horas.

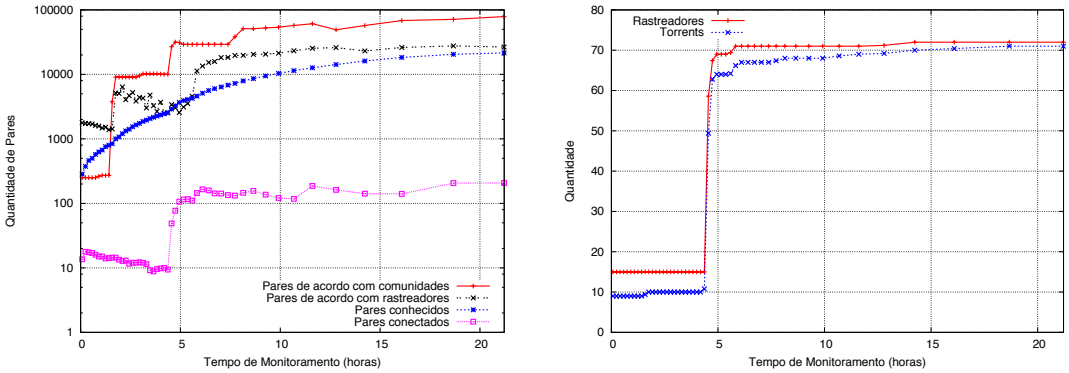
A Figura 5(a) apresenta uma série temporal da evolução da quantidade de pares no enxame segundo as três lentes, apresentada em escala logarítmica no eixo y. Como esperado, observe que há divergências entre os valores observados pelas lentes, conforme explicado a seguir. Primeiro, o número de pares conectados mostra-se muito inferior, o que se deve à parametrização do Peer Lens, configurado com uma amostragem reduzida de pares. Segundo, a lista de pares conhecidos cresce de forma constante, mas consome um certo tempo até convergir para um valor mais próximo àqueles informados pelas outras lentes. Por sua vez, a soma das quantidades de pares retornadas pelos rastreadores possui a limitação de contar múltiplas vezes pares que estão conectados em mais de um rastreador. Além disso, as informações divulgadas nas comunidades podem estar desatualizadas.

A Figura 5(b) apresenta uma série temporal da quantidade de torrents de acordo com duas lentes: Community Lens e Tracker Lens. Observe a quinta hora de monitoramento, aproximadamente 30min após o término da estréia nos E.U.A.: um grande número de torrents é adicionado ao universo. Observe que mesmo 4 horas antes do lançamento do conteúdo, quando o monitoramento foi iniciado, já existiam torrents nas comunidades. Uma possibilidade é que esses torrents estejam associados a tentativas de desestimular a distribuição ilegal de conteúdo, pois verificamos nos sítios das comunidades que 78% das avaliações destes torrents denunciavam conteúdo falso, e que todas as outras, favoráveis, partiram do mesmo usuário.

De forma complementar, a Tabela 1 resume os dados que respondem uma lista de perguntas, de acordo com determinada lente, em três momentos diferentes da história do monitoramento. O MaxMind GeoIP [GeoIP 2010] foi utilizado para mapear endereços IP para países.

## 6. Trabalhos Relacionados

Em [Stutzbach et al. 2008] é apresentado Cruiser, um *crawler* para rede Gnutella que captura *snapshots* da rede. Cruiser é um típico *crawler* P2P para redes não-estruturadas, que usa técnicas para aumentar a velocidade de captura dos *snapshots*. TorrentU, diferentemente, tem como alvo redes BitTorrent.




(a) Quantidade de pares (b) Quantidade de torrents e rastreadores

**Figura 5. Exemplo de observação do Universo Torrent**

**Tabela 1. Amostras do monitoramento em diversos horários**

| Pergunta | Lente | 1ª hora | 10ª hora | 20ª hora |
|---|---|---|---|---|
| Quantos torrents? | Community | 9 | 68 | 71 |
| Quantos arquivos (distintos/total)? | Community | 21 / 21 | 249 / 769 | 258 / 783 |
| Quantos rastreadores ativos? | Tracker | 15 | 71 | 72 |
| Qual a distribuição geográfica dos pares? (3 países mais populares) | Tracker | 23% Espanha<br>22% Irlanda<br>14% Brasil | 17% EUA<br>11% Espanha<br>9% Itália | 13% EUA<br>9% Itália<br>8% Espanha |
| Quais agentes são utilizados? (2 agentes mais populares) | Peer | 99% Azureus<br>0,62% UTorrent | 66% UTorrent<br>22% Azureus | 68% UTorrent<br>21% Azureus |
| Quantos pares são alcançáveis? | Peer | 59 sim / 225 não | 479 s / 876 n | 935 s / 1827 n |
| Quantos usuários estão compartilhando este conteúdo? | Community<br>Tracker<br>Peer | 271<br>1561<br>663 | 53029<br>20628<br>10580 | 78153<br>26629<br>21354 |

Em [Chow et al. 2007] é apresentado BTM, um sistema de monitoramento com foco em detecção de pirataria em BitTorrent. Ele possui um módulo de pesquisa de torrents e outro para análise dos mesmos. Este segundo módulo contata rastreadores e pares para obter as informações relevantes com o objetivo de detectar pares que distribuem conteúdo pirata.

BitProbes, proposto em [Isdal et al. 2007], é um sistema genérico de medição da Internet que usa pares do BitTorrent para monitorar as condições de uma rede. BitProbes também pode ser usado para monitorar enxames do BitTorrent. O sistema instanciado é composto por agentes BitTorrent modificados, que contam com um elemento denominado *shadow tracker* para compartilhamento de *peer list* e divisão da tarefa de monitoramento. Os torrents de entrada são escolhidos em um processo prévio em função do número de pares.

A principal vantagem do TorrentU sobre propostas anteriores é sua flexibilidade na escolha de um conjunto de estratégias que será empregado para extrair a informação desejada. Um conjunto de Telescopes é usado para acompanhar os elementos principais das redes BitTorrent. Cada Telescope têm três lentes, uma para cada tipo de elemento do BitTorrent. Um conjunto distribuído de Telescopes tem melhores chances de observar o universo sem sobrecarregar uma única parte da rede por causa de um monitor ou exceder

os limites de uso de recursos impostos pelas comunidades, rastreadores e pares. Um usuário pode especificar partes de interesse do universo a fim de que o TorrentU "aponte seus Telescópios" para aquela região. A arquitetura é totalmente configurável, permitindo que exatidão e tamanho da amostra sejam escolhidos para que peças do universo possam ser monitoradas de acordo com os recursos disponíveis.

## 7. Conclusão e Trabalhos Futuros

Este trabalho definiu o conceito de "universo BitTorrent" e seus elementos. Estratégias para extrair informações das redes BitTorrent foram identificadas (de acordo com a literatura e implementações). Essas estratégias diferem-se em custo, requisitos e peças de informação que elas são capazes de extrair; além disso, informações são obtidas em diferentes graus de exatidão.

Essas estratégias formaram a base para o projeto de uma arquitetura de monitoramento *flexível* que utiliza "Telescópios" para observar partes do universo BitTorrent. Dado o tamanho deste universo, é impraticável observá-lo inteiramente. Nem mesmo é pragmático acompanhar um grande conjunto de pares com muita riqueza de detalhe. Portanto, o TorrentU permite que um conjunto de estratégias seja escolhido e personalizado a fim de que os elementos de interesse no universo possam ser inspecionados considerando os recursos disponíveis.

Como trabalhos futuros, são consideradas três iniciativas principais. A primeira, projetar e implementar o BitTorrent Observer levando em consideração estratégias de otimização ao determinar a quantidade de Telescopes, para onde os mesmos estarão apontando, e que estratégias serão usadas, para atingir um determinado objetivo de observação. A segunda consiste em estender os experimentos, implementando múltiplas instâncias do protótipo do Telescope e efetuando experimentos de observação de longo prazo na Internet (PlanetLab). A terceira iniciativa é estender a presente arquitetura do TorrentU para que um conjunto de Observers compartilhe um conjunto maior de Telescopes e informações coletadas dos mesmos, seguindo uma filosofia P2P de monitoramento.

## Agradecimentos

À Rede Nacional de Ensino e Pesquisa (RNP) pelo suporte e ao PoP-RS/Leandro Márcio Bertholdo pela disponibilização do hardware utilizado na avaliação experimental apresentada neste artigo.

## Referências

Andrade, N., Mowbray, M., Lima, A., Wagner, G., and Ripeanu, M. (2005). Influences on cooperation in bittorrent communities. In *ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*, pages 111–115.

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., and Yiu, S. M. (2007). Btm - an automated rule-based bt monitoring system for piracy detection. *Second International Conference on Internet Monitoring and Protection 2007*, page 2.

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

Douceur, J. R. (2002). The sybil attack. In *1st International Workshop on Peer-to-Peer Systems*, volume 2429 / 2002, pages 251–260.

Doyen, G., Festor, O., and EmmanuelNataf (2004). A cim extension for peer-to-peer network and service management. *11th International Conference on Telecommunications*, pages 801–810.

GeoIP (2010). Maxmind geoip, http://www.maxmind.com/app/geoip_country.

Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X. (2007). A performance study of bittorrent-like peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 25(1):155–169.

Horng, M.-F., Chen, C.-W., Chuang, C.-S., and Lin, C.-Y. (2006). Identification and analysis of p2p traffic - an example of bittorrent. In *First International Conference on Innovative Computing, Information and Control (ICICIC)*, volume 2, pages 266–269.

Iosup, A., Garbacki, P., Pouwelse, J., and Epema, D. (2006). Correlating topology and path characteristics of overlay networks and the internet. *Cluster Computing and the Grid Workshops, 2006. Sixth IEEE International Symposium on*, 2:10–10.

Isdal, T., Piatek, M., Krishnamurthy, A., and Anderson, T. (2007). Leveraging bittorrent for end host measurements. *Lecture Notes in Computer Science*, 4427 / 2007:32–41.

Izal, M., Urvoy-Keller, G., Biersack, E., Felber, P. A., Al-Hamra, A., and Garces-Erice, L. (2004). Dissecting bittorrent: Five months in a torrent's lifetime. In *5th International Workshop, PAM 2004*, volume 3015 / 2004, pages 1–11.

Karakaya, M., Korpeoglu, I., and Ulusoy, O. (2009). Free riding in peer-to-peer networks. *Internet Computing, IEEE*, 13(2):92 – 98.

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

Legout, A., Liogkas, N., Kohler, E., and Zhang, L. (2007). Clustering and sharing incentives in bittorrent systems. *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 301–312.

Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. (2005). The bittorrent p2p file-sharing system: Measurements and analysis. *Lecture Notes in Computer Science*, 3640 / 2005:205–216.

Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. (2002). An analysis of internet content delivery systems. *Symposium on Operating Systems Design and Implementation (OSDI)*, pages 315–327.

Stutzbach, D., Rejaie, R., and Sen, S. (2008). Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *Networking, IEEE/ACM Transactions on*, 16(2):267 – 280.

Zhang, C., Dhungel, P., Wu, D., Liu, Z., and Ross, K. (2010). Bittorrent darknets. *IEEE INFOCOM*.

Zhang, C., Dhungel, P., Wu, D., and Ross, K. W. (2009). Unraveling the bittorrent ecosystem. pages 1–13.

# APÊNDICE D - ARTIGO PUBLICADO NO IM

Neste anexo é apresentado o artigo “Observing BitTorrent Universe Throught Telescopes", desenvolvido durante o segundo ano do mestrado. O artigo é uma versão aperfeiçoada do artigo apresentado no anexo anterior. Como diferencial, ele apresenta uma avaliação mais completa realizada a partir de um protótipo com maior maturidade.

- Título: Observing BitTorrent Universe Throught Telescopes
- Conferência: 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011)
- URL: http://www.ieee-im.org/
- Data: 23 a 27 de maio de 2011
- Local: Dublin, Irlanda

# Observing the BitTorrent Universe Through Telescopes

Rodrigo B Mansilha, Leonardo R Bays, Matheus B Lehmann, Alan Mezzomo,
Giovani Facchini, Luciano P Gaspary, Marinho P Barcellos
Institute of Informatics
Federal University of Rio Grande do Sul, Brazil
Email: marinho@inf.ufrgs.br

***Abstract*—Recent analysis of the latest peer-to-peer trends worldwide indicates that BitTorrent is the most popular file sharing protocol, taking more than half of the P2P traffic in some geographical locations. Despite several studies about the dynamics of the "BitTorrent universe", there exists no methodology to *systematically* observe it. This is mainly due to the challenges that need to be faced in order to observe the BitTorrent universe, the specificity of existing studies, and the *ad hoc* nature of the monitoring methods employed so far. In this paper, we propose a novel monitoring architecture (called TorrentU) that allows the systematic observation of large numbers of BitTorrent networks. Complementary monitoring strategies are flexibly combined to allow varying degrees of network/geographic coverage, information accuracy and richness of detail. To show the concept and technical feasibility of TorrentU, we implemented a prototype with the key parts of the architecture, and evaluated it through a case study with a rich set of monitoring campaigns running on PlanetLab nodes.**

## I. Introduction

Despite the fact that in recent years many studies about the "BitTorrent universe" have been carried out (e.g. [1], [2], [3], [4], [5], [6]), there exists no systematic methodology to observe it. Observing the real dynamics of BitTorrent networks and how current implementations interact with each other would be useful for several reasons. For instance, it may be useful in marketing campaigns relying on content popularity, or to the estimation of financial losses due to piracy. Such a monitoring methodology may be useful for ISPs to keep track of and better understand P2P activity, and to help design schemes aimed at reducing the costs associated with their BitTorrent users.

Devising a methodology to observe the BitTorrent universe leads to three challenges. First, the dimension, complexity, heterogeneity and uncertainty of BitTorrent networks provide limited opportunities of observation and constitute major obstacles for effective monitoring. Second, in contrast to other P2P file sharing networks, users are spread over millions of smaller networks, called *swarms*. Last, the BitTorrent protocol is open and does not have a corresponding standard or formal specification, which led to several distinct implementations and protocol extensions; to effectively monitor the universe of BitTorrent networks, popular implementations and extensions need to be taken into account.

In this paper we address those challenges and introduce TorrentU – a flexible, extensible BitTorrent monitoring system that can systematically observe parts of the BitTorrent universe. Different observation strategies are combined in a flexible manner according to the desired information, in order to obtain results in an efficient way.

The remainder of this paper is organized as follows. Section II presents the elements of the BitTorrent universe and discusses monitoring strategies. These strategies are the foundation to the design of an architecture, which is introduced in Section III. Section IV discusses the implementation of such strategies. Section V presents a case study that illustrates the experimental methodology employed to observe the BitTorrent universe in different scales. Section VI compares our approach with related work. Finally, Section VII presents concluding remarks and perspectives for future work.

## II. Strategies for Universe Observation

We begin by defining more precisely our concept of the BitTorrent universe (Subsection II-A). Then, we identify strategies for extracting information from its components (Subsection II-B), define observation campaigns (Subsection II-C) and the rationale for choosing strategies (Subsection II-D).

### A. BitTorrent Universe

The BitTorrent universe is comprised of *swarms*, *peers*, *trackers*, *contents* and *communities*. Unlike other P2P file sharing systems, BitTorrent does not have a global network through which all available content is shared. Each shared content forms a BitTorrent overlay network (a swarm), which is composed of trackers and peers.

A peer is a user agent that runs an implementation of the protocol and participates in one or more swarms, according to the content it wishes to share. For a given content, a peer is a *seeder* when it has that entire content, and a *leecher* otherwise. To join a swarm, the peer typically contacts the tracker and obtains a "peer list", with IP addresses and ports of peers participating in the swarm. There are two other ways in which a peer may learn about other peers: Peer Exchange (PEX), which enables a direct interchange of peer lists with other peers, and through a distributed tracker, which is implemented as a distributed hash table (DHT).

Content means "digital content", that is, a specific set of digital files, instead of the information itself available in those files. The content is divided into pieces, and pieces, in their

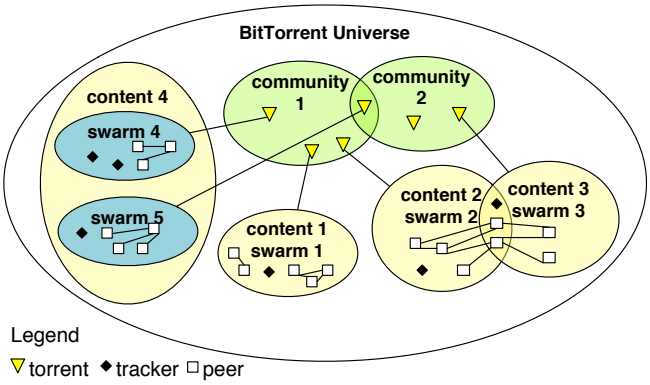



Fig. 1. Example of components and relationships in the BitTorrent universe

turn, divided into blocks. A peer keeps track of pieces its neighbor peers have and informs them about the pieces it has.

To distribute a content, a user generates a torrent file and makes it public. Each torrent file is associated with an *infohash*, which identifies it uniquely. The torrent contains names and sizes of files shared, piece hashing information, and the address of one or more trackers (which may possibly be a DHT). Torrents are typically made available in sites dedicated to promote file sharing, which we refer to as "communities" and their portals. Examples are IsoHunt [7] and Mininova [8]. Besides publishing torrents, some communities maintain trackers as well. Communities may be "private", in which case access is restricted to registered members. Some communities, such as Torrentz [9], only provide aggregate information from other communities.

Figure 1 illustrates the BitTorrent universe, presenting three scenarios to demonstrate possible combinations of contents, trackers, and peers. First, *swarm1*, which shares *content1*, illustrates the scenario in which a given content is shared by a single swarm. Second, *swarm2* and *swarm3* illustrate the case where different swarms have peers in common. Note that these swarms are completely independent, because they refer to different content. Last, *swarm4* and *swarm5* exemplify the case when two independent swarms share the same content. This situation arises in two cases: (a) when the size of the pieces is different among swarms (leading to different *infohashes*); or (b) when the piece size is the same, but the sets of trackers are mutually exclusive.

### B. Strategies

The universe depicted in the figure offers at least three sources of information: torrent communities, trackers, and peers. The set of strategies that can be used to observe the universe will depend on which source is targeted, as discussed next.

**Communities**. Generally speaking, BitTorrent communities exist to provide torrent files and allow users with common interests to interact. Communities provide varying types of information related to each content available, such as its associated trackers, swarm size, "health level", user-provided comments and scores. Studies such as [10], [2] monitored communities as a strategy to infer more broadly about the universe.

**Trackers**. Targeting trackers is another strategy. We find in the literature four variations of this strategy. The first one is based on the analysis of tracker logs, when available; one example of this approach can be found in [1].

The second variation is to continuously observe the content of web pages dynamically generated by trackers, assuming that the pages are available; one example is [10]. The amount of information provided by the tracker of a community will depend on the kind of content being shared and if the community is public or private.

The third variation of the strategy is to announce oneself to the trackers following the protocol. This allows the observation of peer lists and amounts of peers, leechers and seeders that are registered with trackers. Examples of this approach are [2], [5].

The fourth variation of this strategy is to request a summary (scrape) of swarms from a tracker. This summary includes, for each entry, the amount of seeders, leechers and completed downloads so far. Although this summary is not strictly necessary for peers, it can be used to obtain statistics on swarms. In [11], this strategy is used for monitoring the trackers from The Pirate Bay community [12].

**Peers**. The third and last strategy is to monitor the peers directly. There are two variations, namely passive or active. The former requires privileged access to communication links and is based on packet capture and inspection, as in [13], [14]. In contrast, the latter is based on user agents instrumented to connect to peers and extract information from them. There are three variations of the active approach, all with the same deployment and operating requirements: they receive as input a list of peers in the swarm or identification of its tracker(s). The three variations differ with respect to the level of messages exchanged: in the first (e.g. [2], [15]) the user agent connects with remote peers, performs a *handshake*, exchanges piece maps, and then closes the connection. In the second, it keeps the connection with peers open but remains silent, avoiding any data exchange; in the third (e.g. [16]), a peer will actively participate in the swarm and share data.

### C. Observation Campaigns

Given the interest on some part of the universe, the above strategies may be combined in an "observation campaign". A campaign may be specified according to three key properties: network coverage, richness of detail, and information accuracy. **Coverage** indicates the scope of the monitoring. For example, it may refer to geographical location (e.g. all peers in Germany), time range (e.g. all torrents published this month), or content type (e.g. only books).

**Richness of detail** corresponds to the number of attributes of interest, that is, the variables to be collected; examples are number of peers involved, number of downloads, download

speeds, ratio of peers that support a given feature, such as cryptography or PEX, etc.

The third property, **accuracy**, represents the subjective quality of the information extracted from BitTorrent networks. It is affected by three key parameters: freshness (that is, the frequency in which samples are collected), sample size (number of samples that are obtained each time, considering a larger population), and the accuracy of the information source itself. In regards to the latter, the same piece of information can be obtained from different sources, with varying levels of quality. For example, community portals and trackers may report different numbers of peers in a swarm (see discussion in Subsection III-B).

The choice and tuning of monitoring strategies is also affected by the amount of resources available. The relation between strategies and associated costs (mainly CPU/memory consumption and network bandwidth utilization) is explored next.

### D. Choice of Strategies

The monitoring of communities alone has the advantage of using less network resources than the other strategies. Besides, it is much less intrusive to the swarm. One of its disadvantages is the absence of specific information about peers, which other strategies may be able to obtain. Even if the amount of peers sharing a given torrent is obtained from one or more community portals, this information is unreliable (might be inflated) and the frequency in which it is updated is likely to be low.

The main advantage of tracker monitoring, in comparison with community monitoring, is the ability to identify the population of peers in swarms. In comparison with peer monitoring, the advantages are lower intrusiveness and cost. Its relative disadvantages are the absence of information regarding shared content and lack of detailed information about individual peers and pieces. The accuracy of tracker monitoring is relatively low, since the period in which the tracker is periodically contacted by an agent is typically 10 minutes, but may be larger.

The first two strategies demand access only to the community portals and trackers, but provide less richness of detail. In contrast, the third one, peer monitoring, requires spending resources proportionally to the number of peers (usually much higher than the number of trackers and communities). Individually, the strategies discussed represent only a fraction of the problem addressed in this work. A combination of these strategies would lead to a better solution.

The *TorrentU* architecture employs monitors that combine the above strategies. One of its key distinguishing points (more details in Section VI) is the *flexibility* achieved in respect to the levels of coverage, richness of detail and accuracy in an observation campaign. Other important factors are *scalability* (ability to observe large numbers of swarms), *efficiency* (low cost to obtain information from a single element, community, tracker or peer), and *expansibility* (allowing the aggregation of new elements to address new protocol extensions).

## III. Observing the Universe

This section presents *TorrentU* and is organized as follows. Subsection III-A proposes the Management Information Model designed to represent all BitTorrent data within the universe, and Subsection III-B describes the conceptual architecture of the proposed solution.

### A. BitTorrent Management Information Model

A proper information model was needed to structure, link, and rationalize information about communities, torrents, trackers, peers, and swarms. To this end, and as a pre-requisite to our work, we propose the BitTorrent universe Management Information Model. It is based on a subset of the Common Information Model (CIM) defined by the Distributed Management Task Force (DMTF) and on the peer-to-peer model proposed by Doyen *et al.* [17]. To the best of our knowledge, there have not been prior attempts to formalize and organize, by means of an information model, the several (sometimes not well-understood) concepts associated to BitTorrent systems.

Figure 2 depicts a simplified view of the model, which we describe in brief next. Boxes outlined by dashed lines designate classes derived from CIM, while solid-line boxes indicate classes associated with the model of Doyen *et al.*. In turn, boxes highlighted in gray are those proposed to express BitTorrent concepts and their relationships. A BitTorrent community (class BitTorrentCommunity) is based on a centralized entity, a portal, which holds torrents (class Torrent). A torrent file contains, in addition to the list of published pieces and the hash code of each piece, one or more addresses of servers responsible for bootstrapping the peers into the network (modeled by the class Tracker). The set of peers (class BitTorrentPeer) downloading a specific content forms a swarm (class Swarm). In addition to the classes that model the BitTorrent system, MonitoringCampaign expresses configuration parameters.

The BitTorrent Management Information Model, in addition to formally specifying the relationships between BitTorrent components, represents the foundations for a data persistence model. More specifically, this model is employed by two *TorrentU* architectural components, which are introduced in the next subsection.

### B. Conceptual Architecture

The TorrentU architecture is comprised of two main components, namely *TorrentU Observer* and *TorrentU Telescope*, as illustrated in Figure 3. An Observer manages a set of Telescopes. A Telescope allows different "image resolutions": from a bird-eye-view of a constellation to a close look at one star only. In other words, it is possible to have a broad view by considering a large set of torrents discovered in the Universe, then choosing one and exploring it in more detail.

The architecture follows the classical manager–agent model. In practice, Observer and Telescope correspond, respectively, to a manager and to an agent of a typical monitoring system.

**TorrentU Observer**. This component coordinates a set of one or more Telescopes so that they are directed towards

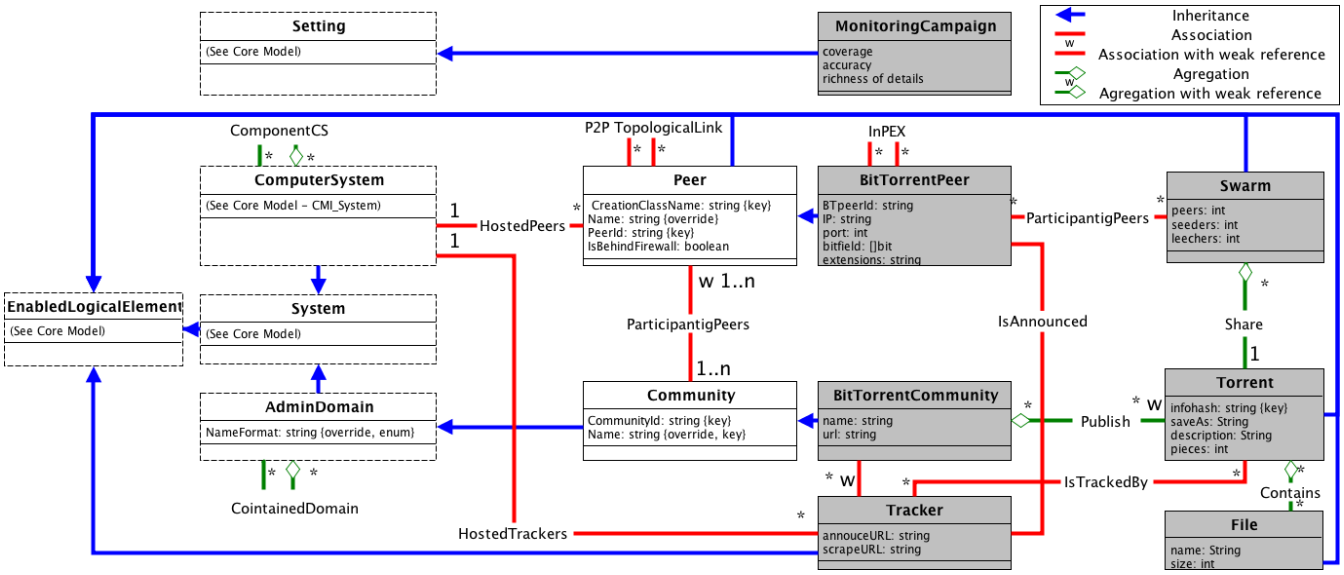



Fig. 2. Management Information Model of BitTorrent

points of interest in the universe. It decides on the choice of monitoring strategies and parameters to be employed according to the data to be collected and available Telescopes (and their resources). Observer also works as a front-end or management station, allowing the system to be configured, presenting (in human-readable form) data that is gathered in real-time (as well as trending and historical data). In addition to an user interface, it has three subcomponents: *Requester*, *Observer Repository*, and *Strategist*. They will be detailed in the following paragraphs.

Requester implements a client/server protocol to communicate with a Telescope. Similarly to FTP, separate connections are used for control and data. While the former is permanent during a "session", data connections are established on demand. The control protocol has three types of messages: Telescope Management, Monitoring Tasks and Data Management.

Telescope Management messages are used to set parameters or operations such as resetting a Telescope. The second type of message is employed to spawn *monitoring tasks*, as well as querying their respective states. Lastly, Data Management messages are used to obtain or remotely change data gathered by Telescopes. For the sake of flexibility, a Data Management message contains a SQL query, applicable to the tables stored in a Telescope. A query is typically used to either obtain data or to remove it from the tables at a Telescope. In the former case, the results will be downloaded through a data transfer connection.

The subcomponent Observer Repository stores information obtained from the Telescope. This repository is particularly important for time series data, in which case snapshots (e.g. of a set of attributes of the universe) are periodically taken and stored for later analysis. To this end, a persistence model, derived from the information model shown in Figure 2, plays an important role in organizing the potentially large volumes of retrieved data.

While coordinating the work of multiple Telescopes, the subcomponent Strategist considers the best choice of strategies to perform some given monitoring campaign, according to their cost and available resources. A piece of information, such as a file name, is obtained by retrieving *transfer units*, such as a torrent file, from the universe. Each information may appear in more than one transfer unit, and each unit is retrieved by one strategy (such as community monitoring). To illustrate, an "amount of peers" info may be obtained from a tracker (using ANNOUNCE or SCRAPE messages) or from another peer (using PEX). The mapping between strategies and transfer units is shown in Table I, along with examples.

TABLE I
MAPPING STRATEGY – TRANSFER UNIT – EXAMPLES OF INFORMATION

| Strategy | Transfer Unit | Examples of information |
|---|---|---|
| Community | html | # peers, infohash (magnet link) |
| Community | torrent | files, tracker's address |
| Tracker | announce | # peers, contact interval |
| Tracker | scrape (all) | infohash, # peers |
| Peer | handshake | NAT, protocol extension |
| Peer | bitfield | # pieces |
| Peer | have | # pieces |
| Peer | PEX | peer's address |

**TorrentU Telescope**. This component is responsible for looking at some specific part of the BitTorrent universe, as directed by an Observer. A Telescope is composed of three subcomponents, namely Provider, Telescope Repository, and Monitor. Provider is the interface for communicating with Observer. Telescope Repository organizes and persists the data gathered, also in accordance to the persistence model mentioned earlier. Monitor is the subcomponent that actu-

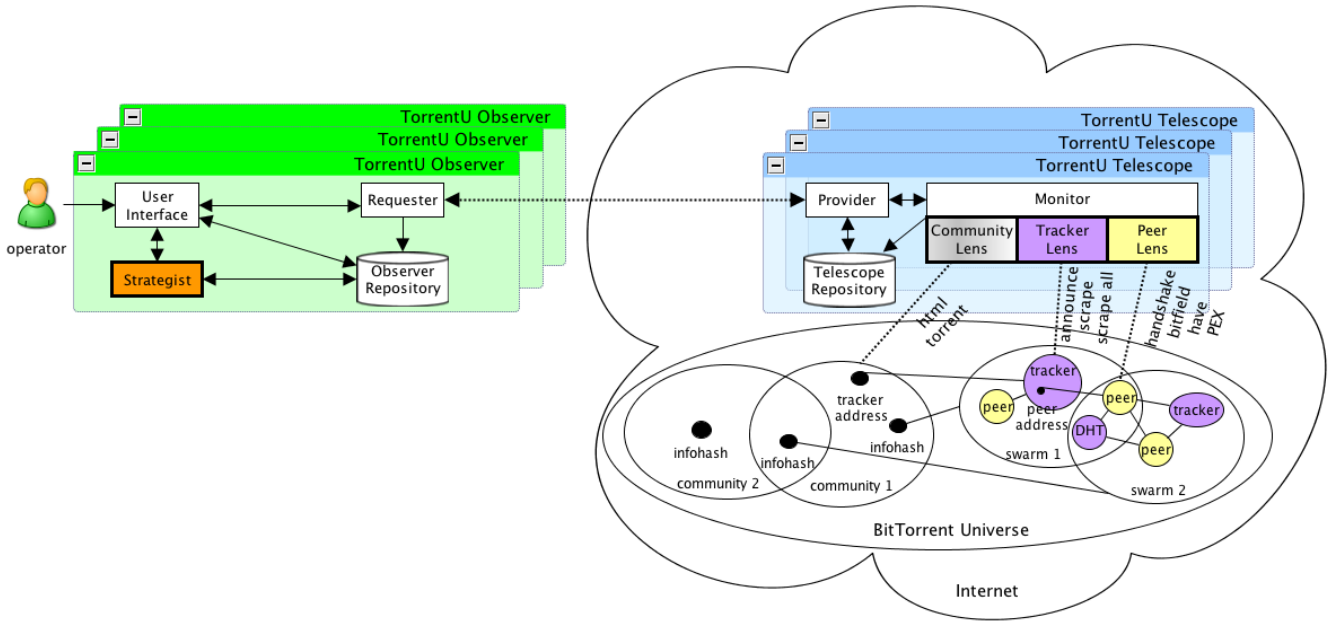



Fig. 3. Conceptual Architecture of *TorrentU*

ally contacts the BitTorrent elements (e.g. portals, trackers, and peers). Monitor is further divided in three parts, called "lenses", each responsible for monitoring a different group of elements in the universe. The lenses are *Community Lens*, *Tracker Lens*, and *Peer Lens*. This modularization allows existing lenses to be replaced, as well as new ones to be easily incorporated into the architecture (without modification of its core components).

## IV. Telescope Lenses

As introduced in the previous section and shown in Figure 3, the three existing lenses belong to subcomponent Monitor of a Telescope. They are key elements of our approach, and for this reason, will be detailed in the following subsections.

### A. Community Lens

Community Lens retrieves information from portals, which is achieved in two ways. First, it observes changes in webpages, downloads the corresponding html files, and parses them looking for relevant information. Examples of relevant information that can be extracted include description of new torrents, total number of torrents, number of seeders and leechers in each swarm, and number of active users. Second, Community Lens downloads and parses torrent files from the site, storing relevant information and following the model specified earlier. Although specific parsers need to be developed for each community, the modularity of design eases the task of adding new communities to the system.

A *directed crawler* is employed to retrieve torrent files from community and torrent searching websites. At any given moment, there is a set of torrent files being offered in a community. This set may be large, and downloading it entirely might require substantial resources from a Telescope (and might increase intrusiveness). Besides, we expect this set to be dynamic to some degree, that is, continually change with inactive torrents being removed and new ones being added (for example, we observed empirically that some communities "recycle" torrents). Finally, a community may impose some limit on the traffic generated by a user; the lens needs to respect this limit or otherwise it will be subject to banishment.

On the other hand, it is likely that an Observer will instruct a Telescope to download a subset of torrents only, since it is not feasible to observe the entire universe. Community Lens will define the subset of torrents to download from a given portal, in reverse chronological order, and then gradually attempt to download each torrent file (if still available). This process, which requires checking and downloading torrents, occurs in rounds: at each round, collected data is added to or updated in Telescope Repository.

A Community Lens can be instructed by an Observer to reduce its coverage using up to three classes of "filters": content (type), quantity (number of torrents) and locality (country hosting the portal). Information about content may be obtained from the website or inside the torrent file. Information about quantity, when available, may be obtained from portals. Locality filters may be employed considering the URL of the community or the identification of the default language employed in the website.

### B. Tracker Lens

Tracker Lens aggregates to the Telescope two of the variations of the tracker monitoring strategy. One is to request from

a tracker a scrape of all its tracked swarms, which includes the respective amounts of participating peers. It is also possible to request a scrape from a single specific swarm.

In the second variation, a Tracker Lens announces itself as a peer participating in a swarm to a tracker, which responds with a random subset of its complete peer list. To increase the scalability of BitTorrent, trackers typically restrict the frequency in which a peer may announce itself. Hence, if the frequency to be used by a Tracker Lens is higher than the one allowed by a targeted tracker, multiple instances of Tracker Lens (with different identities) would have to be used. Alternatively, a set of Telescopes is pointed towards the same tracker/swarm.

Trackers also limit the amount of swarms in which a peer may register itself simultaneously, as reported in [11]. Therefore, depending on the number of monitored swarms in a tracker, it may be necessary to announce the departure from a swarm. Leaving a swarm implies that no remote connections will be received from the peers on that swarm.

The coverage of a Tracker Lens in a Telescope may be narrowed through filters of quantity and locality. For example, less representative swarms can be filtered out by requiring a minimum number of peers, or trackers could be selected based on their location.

### C. Peer Lens

Peer Lens obtains information about a peer by establishing a TCP connection with it. Since the number of connections to be opened may exceed the available resources, Peer Lens employs a Round-Robin algorithm. At each round, it attempts to establish connections to $p$ peers; once open, it keeps them open (active) for a time $t$, awaiting HAVE messages[1]. When the interval is finished, the connections are closed, the window slides, and up to $p$ additional connections are established. As soon as a connection is established with a peer, Lens collects information about it, such as agent implementation version, availability, and number of completed pieces. If the connection is kept open, it is possible to keep track of the piece map evolution (as reported by that peer), and potentially infer download speeds.

Peers that do not take incoming connections (due to a firewall or another form of protection) are marked as unreachable. To learn about those peers, the monitor needs to wait until those remote peers start a connection to Lens. To improve the access to peers in such guarded hosts, it is possible to employ a variant of the Sybil attack [18]. This technique is used by Peer Lens to create and control multiple logical identities, announcing multiple ids to the trackers. Thus, it becomes more likely that those unreachable peers receive at least one IP associated with Lens, increasing the chance that a remote connection (to Lens) is initiated. Note that this bears a cost: by providing "fake" ids to the trackers, it is possible that one or more peers will attempt to open multiple connections to Lens. These connections would be then rejected by Lens, but this requires initiating the connection, handshaking, and then closing it.

The coverage is adjusted in Peer Lens by filtering out the peers or limiting their quantity. Besides coverage, Peer Lens is flexible in terms of accuracy, since the value of $t$ employed in the aforementioned window may be adjusted to increase or decrease the accuracy of the information. In the lower limit, $t = 0$, which means that the connection is immediately closed after the handshake. In the upper limit, $t = \infty$, the connection is kept open until either the remote peer disconnects (or fails) or Lens is switched off. The establishment of the connection at BitTorrent level (HANDSHAKE message) enables the collection of information such as the agent being used and the current piece availability. After this step, in general, the longer the connection time, the higher is the accuracy about the piece availability in peers and more precise the estimate of download speed of a peer. Besides, if the remote peer implements the PEX extension, it is possible to exchange peer lists and estimate the connectivity among peers.

## V. Evaluation

To show its technical feasibility, the architecture described in the previous section was implemented and evaluated. In the experiments, we employed sets of up to 50 PlanetLab nodes. We describe a case study in which parts of the BitTorrent universe were monitored, to demonstrate relevant observation capabilities of the proposed architecture. We emphasize that the focus of the evaluation are not the results themselves, as in typical studies on BitTorrent, but the flexibility in breadth and depth of information that can be achieved, and how efficiently.

The case study is comprised of three monitoring campaigns, each controlling a distinct set of Telescopes. While the first campaign continuously followed the publication of torrents in communities, the second one was used to enhance the richness of detail for contents of interest. The third campaign was used to capture bird-eye-view snapshots of the universe.

The first campaign required only one Telescope. The coverage was set to the four most popular communities (according to [19]): The Pirate Bay [12], ISOHunt [7], BTJunkie [20], and Mininova [8]. Further, it selected only new torrents, ignoring those already present when the experiment started. In regards to accuracy, the communities were revisited every 5 minutes and all new torrents, downloaded. The Telescope observed the communities for 140 hours, during which 15,071 torrents ($\approx$ 376 MB) were published, and of which 11,024 were unique. These unique torrents and their corresponding swarms were used to disseminate 363,251 files, or 5.98 TB of content, whose predominant piece size was 64 KB. (As an aside, we noticed an overlap between communities ISOHunt and The Pirate Bay, corresponding to 68% and 52% of published torrents, respectively.) Further, the Telescope discovered 748 unique tracker addresses. Such addresses were used by the third monitoring campaign, as described later in this section.

A second campaign directed a set of Telescopes to observe "closely" the dissemination of a given content. Telescopes

[1]A peer is supposed to send a HAVE message to all neighbor peers when it completes a piece.

followed for five days the sharing of episodes from a North American TV series (without downloading any content). With respect to richness of detail, the Telescopes were focused onto peers (in addition to trackers and communities) so that lenses captured specific information about peers as well. Coverage was set to select only torrents related to relevant episodes, including those published prior to the experiment. When the monitoring was run, 11 episodes had been released, which represented a total of 819 swarms. In this case, 11 Telescopes were used, one for each episode. To illustrate the kind of information that can be obtained and the relative lack of accuracy of each strategy, Figure 4 shows a time series with the number of peers sharing the season episodes as "seen" by each one of the three lenses ($y$ is in log scale).

In Figure 4, the divergence among values reported by different lenses confirms our intuition about the accuracy of the strategies, as explained next. First, the value obtained through community lenses may reflect outdated information available in community portals; for example, this information might have been obtained from trackers in a peak period. The amount of peers, i.e. an information returned by each tracker, ignores potential overlapping between trackers – when a peer is registered in more than one tracker in a multi-tracker torrent. In other words, the total number of peers in a swarm may not exactly correspond to the summation of peers reported by each tracker used in a torrent. Finally, the accounting of unique IP:port pairs is more accurate than the two previous strategies, but also limited by the fact that peers may silently depart and the tracker may report more peers than in fact are present.

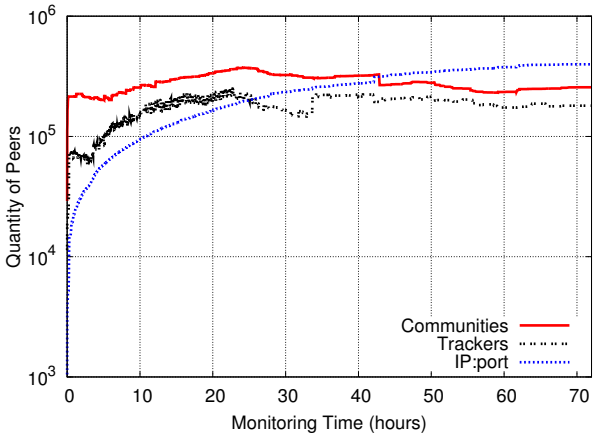



Fig. 4. Quantity of peers obtained through different lenses

The third campaign was used to cover a large spectrum of swarms in a shorter period of time, therefore obtaining a more comprehensive view of the universe. To be feasible, the Telescopes were zoomed out and less details were collected. We ran an experiment in which 50 Telescopes were pointed to the 748 trackers previously discovered (these trackers were collected from 15,071 new, unique torrents from the four most popular communities). Within 30 minutes, 380 responses were obtained (a large number of trackers have not replied); such responses accounted for 2.28 GB, allowing us to obtain information about 4 million swarms.

This campaign also demonstrates how an Observer can use Telescopes in different manners to consolidate results: to determine the most popular swarm, it was sufficient for the Observer to request the local information from each Telescope and then pick the highest value returned. In contrast, in order to establish the total number of unique torrents in the observed universe (in this case, 3,923,203), an Observer has to fetch all infohashes from Telescopes and then discard duplicates.

Finally, this campaign was used to demonstrate the capacity to collect IP addresses of peers. Within a 30 minute window, Telescopes found more than 1.1 million unique IP addresses. As an aside, we found that 38% of the peers were in Europe, 32% in Asia, 21% in North America, 6% in South America, 2% in Oceania, and only 1% in Africa. Within a second 30 minute window, a random sample of peers from this set was contacted, allowing us to collect more detailed information about peers, as shown in Table II.

TABLE II
PEER LENS SUMMARY

| | Quantity | % |
|---|---|---|
| Connection attempts | 65993 | 6% |
| Successful TCP connections | 24540 | 37% |
| Successful BitTorrent connections | 19622 | 80% |
| Seeders | 2075 | 11% |
| DHT enabled | 17332 | 88% |
| PEX enabled | 17027 | 87% |
| User Agent | 14027 $\mu$Torrent<br>1925 Mainline<br>1481 Azureus | 71%<br>10%<br>8% |

## VI. RELATED WORK

In this section we discuss related approaches for monitoring of P2P networks and, in particular, of BitTorrent.

The authors in [21] introduce Cruiser, a crawler that captures snapshots of Gnutella overlays. Cruiser is a typical P2P crawler for unstructured networks, which uses techniques to increase the velocity of snapshots obtention. TorrentU, differently, has BitTorrent networks as its target.

In [22], authors present BTM, a monitoring system with focus on piracy detection. BTM is organized in two modules; the first, called "Torrent Searcher", is responsible for finding torrents in sites specified by the user. Later, torrents found are fed to the second module, called "Torrent Analyzer". The latter module contacts trackers and peers to obtain relevant information, aiming at detecting peers that distribute copyright-protected materials.

BitProbes, proposed in [23], is a generic Internet measurement system which uses BitTorrent peers to monitor network conditions. BitProbes can be used to monitor BitTorrent swarms as well. It is comprised of a modified BitTorrent agent and a coordinating element, called *Shadow Tracker*, which shares peerlists and divides the monitoring tasks. The input

torrent files are selected previously according to the expected number of peers.

When compared to other approaches, TorrentU allows the systematic and efficient observation of the BitTorrent universe. One key advantage lies in its flexibility: a set of strategies can be combined to obtain the desired information. A distributed set of Telescopes can be deployed and controlled by an Observer to follow parts of the BitTorrent universe. Each Telescope has three lenses, adopting a different monitoring strategy. A distributed set of Telescopes has better chances of observation; it can look at the universe without overloading a single part of the network, or exceeding resource limits enforced by some communities, trackers and peers. The architecture is fully configurable, allowing the observation to be dimensioned according to available resources.

## VII. Conclusion and Future Work

This paper defined the concept of "BitTorrent universe" and its elements. Strategies to extract information from BitTorrent networks were identified in the literature and existing implementations. These strategies differ in terms of cost, requirements and pieces of information that they are capable of extracting. Besides, information is obtained with different degrees of coverage, accuracy and richness of detail.

The strategies formed the basis to the design of a *flexible* monitoring architecture which employs a distributed set of "Telescopes" to observe parts of the BitTorrent universe. Each Telescope has three different types of lenses, which reflect the aforementioned strategies and demand different costs. Information retrieved by a Telescope from universe constituents (such as portals and peers) is organized and stored in a structured fashion. To this end, we proposed a Management Information Model to keep data about BitTorrent networks. This model is able to formally express the relationship among components in the universe.

In a campaign, an Observer will follow some monitoring policy to direct a set of Telescopes towards points of interest in the universe. Two factors increase the scalability of TorrentU: the flexibility of a Telescope (through different lenses), and the ability to use a set of coordinated Telescopes. Therefore, not only more Telescopes can be used to increase monitoring capacity, but also the flexibility allows larger portions of the universe to be observed when less accurate or coarser (with less details) information is adequate.

As future work, we envisage three main initiatives. First, to improve the subcomponent Observer Strategist with multi-variable optimization strategies, increasing the efficiency of the monitoring process. Second, to extend our experiments in observing parts of the BitTorrent universe, with long-lived experiments (spanning several months). Third, to extend the present architecture of TorrentU so that networks of Telescopes can be shared by multiple Observers.

## Acknowledgements

This work has been supported by RNP Academic Network (GT-UniT WG) and CNPq Research Agency (Project P2P-SeC).

## References

[1] M. Izal, G. Urvoy-Keller, E. Biersack, P. A. Felber, A. Al-Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in a torrent's lifetime," in *5th International Workshop, PAM 2004*, vol. 3015 / 2004, Apr 2004, pp. 1–11.

[2] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," *Lecture Notes in Computer Science*, vol. 3640 / 2005, pp. 205–216, 2005.

[3] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of bittorrent-like peer-to-peer systems," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 1, pp. 155–169, 2007.

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

[5] C. Zhang, P. Dhungel, D. Wu, and K. W. Ross, "Unraveling the bittorrent ecosystem," pp. 1–13, Jul 2009.

[6] C. Zhang, P. Dhungel, D. Wu, Z. Liu, and K. Ross, "Bittorrent darknets," *IEEE INFOCOM*, March 2010.

[7] (2010, Feb) Isohunt. [Online]. Available: http://www.isohunt.com/

[8] (2010, Feb) Mininova. [Online]. Available: http://www.mininova.org/

[9] (2010, Feb) Torrentz. [Online]. Available: http://www.torrentz.com/

[10] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, "Influences on cooperation in bittorrent communities," in *ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*, 2005, pp. 111–115.

[11] S. L. Blond, A. Legout, F. Lefessant, W. Dabbous, and M. A. Kaafar, "Spying the world from your laptop," pp. 1–8, Apr 2010.

[12] (2010, Feb) The pirate bay. [Online]. Available: http://thepiratebay.org/

[13] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," *Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 315–327, Dec 2002.

[14] M.-F. Horng, C.-W. Chen, C.-S. Chuang, and C.-Y. Lin, "Identification and analysis of p2p traffic - an example of bittorrent," in *First International Conference on Innovative Computing, Information and Control (ICICIC)*, vol. 2, 2006, pp. 266–269.

[15] A. Iosup, P. Garbacki, J. Pouwelse, and D. Epema, "Correlating topology and path characteristics of overlay networks and the internet," *Cluster Computing and the Grid Workshops, 2006. Sixth IEEE International Symposium on*, vol. 2, pp. 10–10, 2006.

[16] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in bittorrent systems," *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 301–312, Jun 2007.

[17] G. Doyen, O. Festor, and EmmanuelNataf, "A cim extension for peer-to-peer network and service management," *11th International Conference on Telecommunications*, pp. 801–810, Jun 2004.

[18] J. R. Douceur, "The sybil attack," in *1st International Workshop on Peer-to-Peer Systems*, vol. 2429 / 2002, Mar 2002, pp. 251–260.

[19] (2010, Feb) Alexa. [Online]. Available: http://www.alexa.com/

[20] (2010, Feb) Btjunkie. [Online]. Available: http://btjunkie.org/

[21] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing unstructured overlay topologies in modern p2p file-sharing systems," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 2, pp. 267 – 280, Apr 2008.

[22] K. P. Chow, K. Y. Cheng, L. Y. Man, P. K. Y. Lai, L. C. K. Hui, C. F. Chong, K. H. Pun, W. W. Tsang, H. W. Chan, and S. M. Yiu, "Btm - an automated rule-based bt monitoring system for piracy detection," *Second International Conference on Internet Monitoring and Protection 2007*, p. 2, Jul 2007.

[23] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Leveraging bittorrent for end host measurements," *Lecture Notes in Computer Science*, vol. 4427 / 2007, pp. 32–41, Jan 2007.