SOBRE DOIS FENÔMENOS EM REDES P2P DO TIPO BITTORRENT

Fabrício Murai Ferreira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Daniel Ratton Figueiredo
Edmundo Albuquerque de Souza e Silva

Rio de Janeiro
Abril de 2011

SOBRE DOIS FENÔMENOS EM REDES P2P DO TIPO BITTORRENT

Fabrício Murai Ferreira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Edmundo Albuquerque de Souza e Silva, Ph.D.

Prof. Daniel Ratton Figueiredo, Ph.D.

Prof. Artur Ziviani, Dr.

Prof. José Ferreira de Rezende, Dr.

Prof. Virgílio Augusto Fernandes de Almeida, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2011

Ferreira, Fabrício Murai

Sobre dois fenômenos em redes P2P do tipo BitTorrent/Fabrício Murai Ferreira. – Rio de Janeiro: UFRJ/COPPE, 2011.

XI, 74 p.: il.; 29, 7cm.

Orientadores: Daniel Ratton Figueiredo
Edmundo Albuquerque de Souza e Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 71 – 74.

1. P2P. 2. BitTorrent. 3. Assortatividade. 4. Análise Transiente. 5. Modelo de desempenho. I. Figueiredo, Daniel Ratton *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Primeiramente, gostaria de agradecer a minha família pelo apoio e compreensão. Em particular, a minha esposa Gabriela e a minha mãe Emília que tiveram bastante paciência nos momentos em que estive mais atribulado.

Agradeço também aos meus orientadores, Daniel e Edmundo, que foram essenciais para a realização desse trabalho. Mais do que isso, contribuíram de forma fundamental na minha formação acadêmica e pessoal. Ambos são pessoas que trabalham com bastante seriedade. Daniel, em particular, nunca me deixou desmotivar, nem mesmo quando as coisas não davam certo. Edmundo, mais experiente, sempre me impediu de perder o foco do trabalho. Ter sido orientado por estes dois professores certamente foi uma combinação muito proveitosa.

Carol foi também uma pessoa muito importante ao longo desta jornada. Na sua sala, eu e muitos outros alunos, tivemos a oportunidade de conversar com alguém cujo coração é de valor inestimável, capaz de tranquilizar nos momentos certos e dar ânimo nos momentos necessários. Ela certamente será lembrada por todos, mesmo quando não estiver mais no LAND.

Gostaria de agradecer ao Guto, que esteve trabalhando ativamente comigo nos últimos meses. Obrigado também ao Rafael e ao GD pelas discussões interessantes que tivemos sobre este trabalho. Foi muito importante ouvir as críticas e opiniões de vocês. Ao Bernardo, Allyson e Gabriel agradeço pelas coisas que me ensinaram durante nosso convívio. Aprendi bastante com eles sobre como desenvolver atividades com outras pessoas. Sou grato ao Gaspare, Jefferson e Vicente, pelas discussões interessantes acerca de outros assuntos que tivemos no LAND e nos ônibus da linha 663. É certo que algumas delas foram mais acaloradas, mas tudo sempre terminou bem. Nestes últimos anos, tive a oportunidade de trabalhar com muitas pessoas diferentes. Em particular, foi muito bom ter trabalhado com Watanabe, Luiz, Alejandra, Thothu, Guilherme Domingues e Rodrigo Paim.

Agradeço ao Érico que me trouxe ao LAND em 2005, a todas as outras pessoas que de alguma forma me deram apoio ao longo dessa jornada e àquelas que me ajudaram a desenvolver um olhar mais crítico sobre as coisas. Em especial, agradeço ao Paulo, Yanko, Flávio, Rafael e André. Por fim, agradeço o suporte financeiro oferecido pelo CNPq, Faperj, Cederj e daqueles que pegaram carona comigo.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SOBRE DOIS FENÔMENOS EM REDES P2P DO TIPO BITTORRENT

Fabrício Murai Ferreira

Abril/2011

Orientadores: Daniel Ratton Figueiredo
Edmundo Albuquerque de Souza e Silva

Programa: Engenharia de Sistemas e Computação

O BitTorrent (BT) é uma aplicação par-a-par (P2P) para compartilhamento de arquivos bastante popular nos dias de hoje, principalmente devido ao seu bom desempenho. Desta forma, modelar e entender a dinâmica desse sistema se tornou um recorrente tópico de pesquisa. Ao longo dos anos, diferentes estudos e modelos desvendaram fenômenos que emergem nesse sistema, muitos com impacto direto no seu desempenho. Nesse trabalho, investigamos dois desses fenômenos.

O primeiro é a clusterização de peers (usuários) por largura de banda, isto é, usuários com capacidade de acesso similar tendem a trocar mais dados entre si. Esse comportamento surge mesmo sem que haja preferência explícita por pares de largura de banda similar nos algoritmos empregados. Estudamos a dinâmica do BT para entender quais mecanismos são responsáveis por levar o sistema a esse estado. Entre outras observações, nossos resultados indicam que uma característica fundamental do BT (*unchoke* otimista) é essencial para que o sistema atinja alta clusterização.

O segundo fenômeno estudado trata da ocorrência de tempos de downloads heterogêneos em redes de peers homogêneas com relação à capacidade de *upload*. Esse fenômeno têm diversas implicações para o sistema, tais como alta variabilidade nos tempos de *download*, injustiça com respeito à ordem de chegada dos peers, saídas em rajada e sincronização das partes do conteúdo detidas pelos peers. A fim de caracterizar esse comportamento, utilizamos um modelo de simulação bastante detalhado em conjunto com a realização de experimentos reais. Finalmente, propomos um modelo que explica esse fenômeno e suas consequências, o qual prevê taxas de download heterogêneas para peers homogêneos, como função do conteúdo de cada um.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TWO PHENOMENA IN BITTORRENT-LIKE P2P NETWORKS

Fabrício Murai Ferreira

April/2011

Advisors: Daniel Ratton Figueiredo
Edmundo Albuquerque de Souza e Silva

Department: Systems Engineering and Computer Science

BitTorrent (BT) is a very popular peer-to-peer (P2P) file sharing application mainly due to its good performance. Thus, modeling and understanding BT dynamics has become a recurrent research topic. Over the years, different models have uncovered various phenomena exhibited by the system, many of which have direct impact on its performance. In this work, we investigate two of such phenomena.

The first phenomenum is the clusterization of users (peers) with respect to their bandwidth capacity, i.e. users with similar capacities tend to exchange more data with each other. This behavior arises even though there is no explicit preference for similar bandwidth peers in the algorithms employed. We study the BT dynamics and develop a simultation model to understand which mechanisms are responsible for driving the system into this state. Among other observations, our results indicate that a key aspect of BT (namely, the optimistic unchoke) is essential for the system to achieve high clusterization.

The second phenomenum is the occurrence of heterogeneous download times in homogeneous swarms, where peers have identical upload capacity. This behavior has many consequences to the system, such as high variability of download times, unfairness with respect to peer arrival order, bursty departures and content synchronization. Detailed packet-level simulations and prototype-based experiments on the Internet were performed to characterize this phenomenon. We also develop a mathematical model that explains this phenomenum and its consequences, accurately predicting the heterogeneous download rates of the homogeneous peers as a function of their content.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

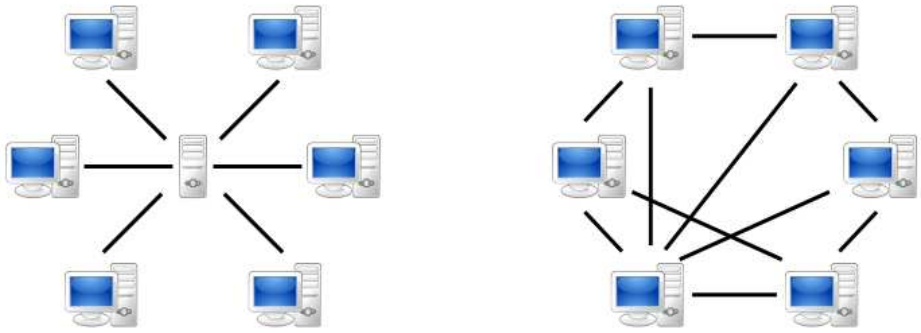
Introdução

A Internet vem se expandindo e evoluindo de forma acentuada em muitos aspectos. Um exemplo dessa transformação é a evolução da arquitetura dos aplicativos. Por exemplo, a maioria das aplicações tradicionais adotam a arquitetura cliente-servidor. Entretanto, atualmente novas aplicações têm como modelo a arquitetura conhecida como par-a-par (peer-to-peer ou P2P), que oferece diversas vantagens, sendo uma das principais, a alta escalabilidade com o número de clientes.

Na arquitetura cliente-servidor, existem um ou mais computadores que oferecem um determinado serviço a um conjunto de usuários (ver Figura 1.1a). Neste modelo, usuários (clientes) demandam a execução de tarefas pelo servidor. Para que esse serviço seja escalável, ou seja, para que o serviço atenda de forma satisfatória um crescente número de clientes, é necessário ampliar a infraestrutura que o serviço utiliza. No entanto, isso pode ter um custo muito elevado. Como exemplo de sistema que utiliza essa arquitetura, mencionamos o YouTube. O YouTube é um site que armazena uma enorme quantidade de vídeos, e transmite mais de 2 bilhões de exibições por dia [1]. Para que isso seja possível, são necessários milhares de servidores espalhados pelo mundo.

Na arquitetura P2P, os usuários interessados em um determinado serviço, denominados peers, cedem parte dos seus recursos, como poder de processamento, armazenamento e capacidade de transmissão para auxiliar outros usuários (ver Figura 1.1b). Assim, a escalabilidade advém do uso dos recursos dos próprios usuários para o funcionamento do sistema. Alguns dos serviços oferecidos por sistemas P2P são: videoconferência (ex.: FreeMeeting [2]), compartilhamento de arquivos (ex.: BitTorrent [3]), backup (ex.: CFS [4]), streaming de vídeo (ex.: PPLive [5]).

O BitTorrent (BT) é uma das aplicações P2P mais populares da Internet. Ele é usado por milhões de pessoas diariamente para baixar milhões de arquivos (filmes, séries de TV, música etc) e é responsável por grande parte do tráfego de dados que atravessa a Internet atualmente [6]. Enquanto um usuário do BT faz download de um arquivo, ele auxilia o processo de disseminação transmitindo as partes já



(a) Cliente-servidor (b) Par-a-par

Figura 1.1: Comparação ente arquiteturas.

obtidas a outros peers interessados no mesmo conteúdo. Assim, quanto maior o número de usuários, maior é a capacidade de disseminação que o sistema possui. Isso permite, por exemplo, que um provedor de conteúdo disponibilize um arquivo que será solicitado por milhares de usuários simultaneamente usando apenas um servidor.

O sucesso do BitTorrent está intimamente ligado ao seu desempenho: usuários do BT fazem download de arquivos em tempo relativamente pequeno. Esta característica somada à alta complexidade da dinâmica do sistema, muito embora seja ela definida por alguns mecanismos simples, tem atraído a atenção de muitos pesquisadores.

Ao longo dos anos, diferentes trabalhos na literatura mostraram diversas características que emergem nesse sistema, muitas das quais têm impacto direto no seu desempenho e usabilidade e, por conseguinte, é fundamental entendê-las.

1.1 Contribuição

Uma característica que foi observada em redes BitTorrent e também em outras redes P2P para compartilhamento de arquivos é conhecida como *Assortative Mixing* (AM). O AM é uma característica topológica, definida sobre uma rede, que expressa uma tendência na criação de relacionamentos entre vértices similares (ou diferentes), sob algum aspecto de similaridade, seja essa rede de informação, social, ou mesmo de outro tipo. No caso do BitTorrent, os peers que compõem uma rede possuem enlaces de acesso à Internet com diferentes larguras de banda. Recentemente, observou-se a ocorrência do AM entre os peers de uma rede BitTorrent quando a largura de banda é a métrica de similaridade usada [7], ou seja, peers tendem a se conectar e trocar

informação com peers que possuem largura de banda semelhante a sua própria.

Assim como outras propriedades topológicas, AM também tem um impacto fundamental na funcionalidade do sistema. No caso de sistemas P2P, como o BitTorrent, a presença de AM em relação à largura de banda está diretamente relacionada à escalabilidade e robustez do sistema. Em particular, um sistema P2P com alto grau de AM (i.e., alta clusterização) é mais eficiente e oferece melhor serviço a seus usuários [8]. É então fundamental entender os motivos que levam a atingir alto grau de AM.

A aplicação BitTorrent para compartilhamento de arquivos é relativamente complexa, sendo formada por diversos mecanismos que operam simultaneamente. Em essência, o seu funcionamento consiste em um processo dinâmico de troca de peers que tenta maximizar a taxa de download dando em troca taxas de upload. No entanto, não há preferência explícita nos algoritmos empregados por pares com largura de banda similar. A característica observada é decorrência dos mecanismos que definem a aplicação. Mas que mecanismos são esses? Qual é a influência dos parâmetros do sistema, neste processo, que leva ao AM?

A primeira parte deste trabalho consiste no estudo da dinâmica do BitTorrent para entender quais mecanismos são responsáveis por induzir o sistema a um estado com alto grau de AM. Em particular, propomos um modelo de simulação que captura a essência da dinâmica do BitTorrent. Através de simulações, mostramos que o processo dinâmico, que se baseia apenas em regras locais, leva o sistema de um estado inicial aleatório a um estado com alto grau de AM. Estudamos ainda o impacto dos parâmetros do modelo neste processo dinâmico. Entre outras observações, nossos resultados indicam que uma característica fundamental do BT (*unchoke otimista*) é essencial para que o sistema atinja altas taxas de AM [9, 10]. Como complemento ao modelo de simulação, propomos um modelo simplificado que pode ser resolvido analiticamente. Este modelo pode ser usado para obter soluções para o regime estacionário e transiente do sistema.

Uma outra característica inerente às aplicações P2P é a grande desigualdade na popularidade dos arquivos disponíveis para download. Em particular, poucos arquivos atraem o interesse de milhares de usuários enquanto a grande maioria dos arquivos tem popularidade muito baixa.

A segunda parte deste trabalho se refere à identificação e ao estudo de fenômenos, até então inexplorados na literatura, que ocorrem predominantemente em arquivos impopulares. Como exemplo, observa-se que os usuários do sistema podem obter taxas de download diferentes mesmo quando a população é homogênea com relação à banda. Essa característica tem diversas implicações para o sistema como alta variabilidade dos tempos de download, injustiça em relação à ordem de chegada dos peers e a sincronização das partes do conteúdo detidas pelos peers [11].

Para estudar a heterogeneidade das taxas de download em populações do BT onde todos os usuários possuem a mesma largura de banda, desenvolvemos um outro modelo de simulação, significativamente mais detalhado que o primeiro. Em paralelo, realizamos experimentos reais. Finalmente, propomos um modelo que auxilia na explicação desse fenômeno e suas consequências. O modelo prevê taxas de download heterogêneas para peers homogêneos, como função do conteúdo de cada um.

É importante enfatizar que as características aqui estudadas estão fortemente correlacionados com o desempenho do sistema. O estudo dessas características nos permite identificar cenários em que a aplicação não possui desempenho satisfatório e nos dá as intuições necessárias para melhorá-la, seja do ponto de vista do usuário ou mesmo da rede.

1.2 Organização do texto

O restante deste texto está organizado da seguinte forma. O Capítulo 2 traz uma visão geral de sistemas P2P, da aplicação BitTorrent e também apresenta uma revisão bibliográfica dos trabalhos relacionados. O Capítulo 3 trata da primeira contribuição deste trabalho, que é um estudo dos mecanismos do BitTorrent que levam ao surgimento do AM. O Capítulo 4 traz a segunda e principal contribuição deste trabalho: a identificação e modelagem de diversos fenômenos que surgem predominantemente com arquivos pouco populares. Por fim, o Capítulo 5 apresenta uma discussão a respeito das conclusões obtidas e das principais contribuições deste trabalho.

Capítulo 2

BitTorrent e Trabalhos Relacionados

Este capítulo discorre sobre sistemas P2P e explica de forma detalhada o funcionamento do BitTorrent, um dos mais populares aplicativos para compartilhamento de arquivos. Além disso, traz um resumo dos principais trabalhos sobre análise de desempenho e sobre a identificação e estudo de características interessantes no BitTorrent e em outras aplicações similares.

2.1 Sistemas P2P

Os sistemas P2P ganharam popularidade a partir do final da década de 90, com a criação do Napster. O Napster era um programa que permitia aos usuários compartilhar arquivos MP3. Através desta aplicação, um usuário podia procurar por uma música disponibilizada por outro e fazer o seu download diretamente, isto é, sem intermédio de um servidor. Apesar de ter sido descontinuado devido a questões legais, ele atraiu a atenção para o potencial do paradigma P2P.

Tanto na arquitetura P2P quanto na arquitetura cliente-servidor, os usuários que entram no sistema aumentam a demanda por recursos. Entretanto, na arquitetura P2P os usuários trazem consigo recursos que são agregados ao sistema, podendo ser usados para servir outros usuários. Isso faz com que a capacidade do sistema aumente com a chegada de novos peers, o que possibilita implementar sistemas de larga escala a baixo custo.

Dentre os exemplos de sistemas P2P que possuem um grande número de usuários, podemos citar o PPLive e o BitTorrent. O PPLive é uma aplicação IPTV, ou seja, um software cuja finalidade é transmitir canais de *streaming* de vídeo através da Internet. Apesar de bastante popular, pouco se sabe sobre a sua arquitetura além do fato de ser baseada em redes P2P, pois é uma aplicação proprietária. Entretanto,

Hei et al. [12] descobriram alguns aspectos do funcionamento do PPLive que descrevemos a seguir. Inicialmente o usuário obtém a lista de canais disponíveis a partir de um servidor de canais. Depois que um canal é escolhido, a aplicação solicita aos servidores responsáveis pelo gerenciamento de peers listas contendo os usuários sintonizados naquele mesmo canal. O PPLive então contacta alguns desses peers, que passam a ser seus vizinhos. Os vizinhos transmitem ao usuário pedaços do conteúdo desejado. Além disso, transmitem também listas contendo outros peers que estão nesse canal. Por fim, os pedaços obtidos ficam temporariamente armazenados para que possam ser retransmitidos aos vizinhos que ainda não os possuem.

O BitTorrent é o foco do nosso estudo e seu funcionamento é explicado com detalhes na seção a seguir.

2.2 BitTorrent

O BitTorrent é uma aplicação P2P para compartilhamento de arquivos que implementa o protocolo de mesmo nome. Ele é baseado em *swarms*, que são conjuntos de usuários interessados em baixar ou compartilhar um determinado conteúdo, que pode ser um único arquivo ou um conjunto de arquivos. Nesse sistema, um usuário não precisa baixar o conteúdo por completo antes de participar da sua disseminação. Para que isso seja possível, o conteúdo é inicialmente dividido em pedaços e, cada pedaço, subdividido em blocos (ver Figura 2.1). Esses pedaços são trocados entre os *peers* (i.e., usuários) que participam do swarm enquanto o conteúdo é baixado, aumentando a capacidade de disseminação do swarm.

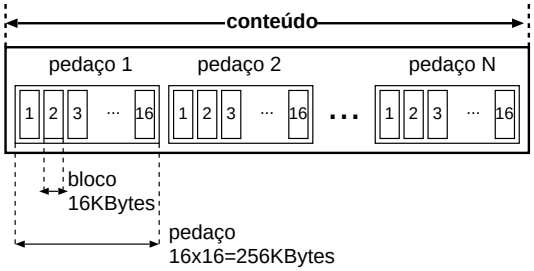


Figura 2.1: No BitTorrent, o conteúdo é dividido em pedaços de 256KBytes, que são por sua vez subdivididos em 16 blocos de 16KBytes.

As entidades que participam de um swarm podem ser de três tipos:

- *Seeds:* são peers que possuem uma cópia completa do conteúdo, i.e., todos os pedaços que o compõem e por isto fazem apenas o upload de pedaços para outros usuários.

- *Leechers:* são peers que ainda não recuperaram o conteúdo por completo e continuam fazendo download e upload de pedaços.

- *Tracker:* é uma espécie de coordenador do swarm, responsável por manter uma lista com todos os peers que participam do swarm.

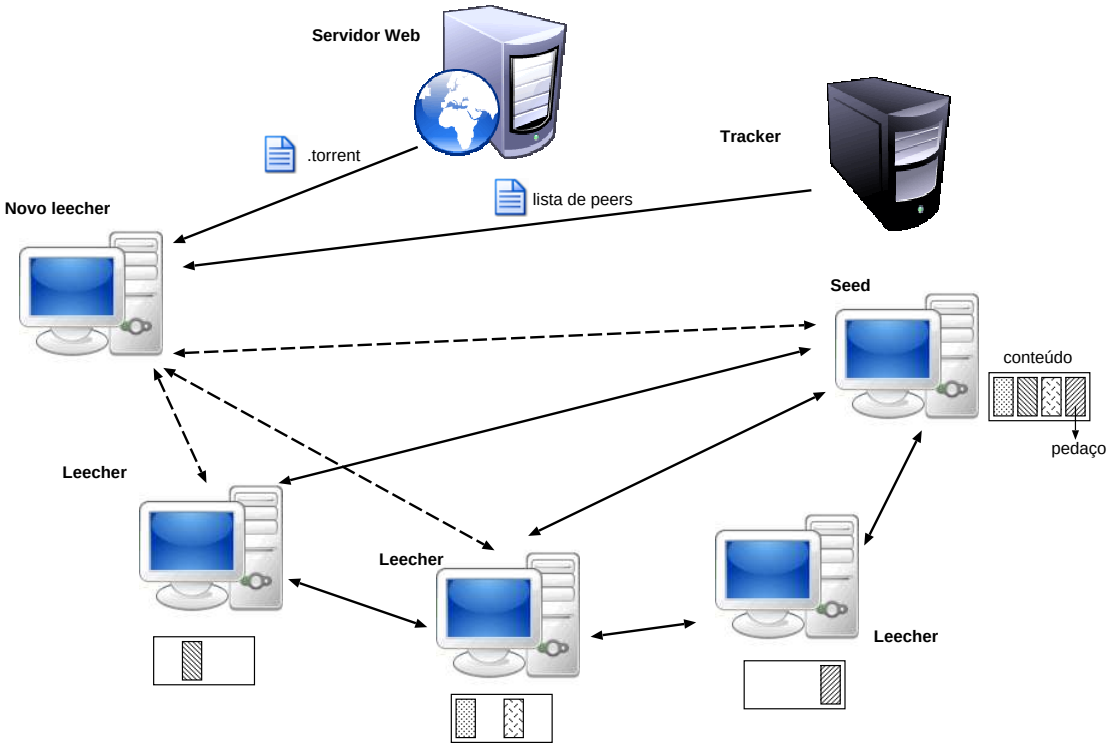


Figura 2.2: Um novo leecher entra no swarm.

Inicialmente, um seed interessado em compartilhar um certo conteúdo, dispara um tracker e disponibiliza (geralmente em uma página Web) um arquivo .torrent, que possui metadados sobre o conteúdo e o endereço do tracker. Usuários interessados em baixar este conteúdo utilizam ferramentas de busca para encontrar o arquivo .torrent e então se conectam ao tracker, passando então a desempenhar o papel de leechers (ver Figura 2.2). O tracker envia para cada leecher uma lista com L outros peers selecionados aleatoriamente entre os seeds e leechers conectados ao swarm. Os usuários então tentam se conectar a $A < L$ outros peers e, em caso de sucesso, eles se tornam vizinhos e trocam seus *bitmaps*. O *bitmap* é um mapa que indica quais pedaços um leecher possui (veja a Figura 2.1). Conforme um leecher vai baixando novos pedaços, ele envia atualizações do seu *bitmap* para seus vizinhos. Ao longo do tempo, alguns vizinhos podem se desconectar por terem terminado o download, desistido de baixar o conteúdo ou simplesmente interrompido o download para continuá-lo mais tarde. Caso o número de vizinhos de um leecher fique abaixo do limite m, ele passa a requisitar novos peers ao tracker a cada T_L segundos, enquanto a quantidade de peers que ele tem não ultrapassar este limite. Existe ainda um número máximo de conexões $M > m$ que um peer mantém com os outros peers. Caso um peer conectado a M vizinhos receba novos pedidos de conexão, ele irá recusá-los.

Para receber novos pedaços, um leecher envia mensagens comunicando seu interesse aos vizinhos que possuem pedaços que ele ainda não tem. Entretanto, antes que o leecher em questão comece a solicitar estes pedaços, ele precisa aguardar o recebimento de uma autorização, concedida através da mensagem de *unchoke*. O número de *unchokes* que um peer concede simultaneamente (ou equivalentemente, o número de uploads simultâneos) é limitado. É necessário, portanto, que cada peer escolha, dentre os vizinhos que estão interessados (i.e., que comunicaram interesse), aqueles para os quais irá fazer upload. Essa escolha se dá segundo o **algoritmo de seleção de pares**. Ademais, os pedaços não são pedidos de forma sequencial, mas sim priorizados conforme o **algoritmo de seleção de pedaços**. No que segue apresentamos os detalhes destes algoritmos, que são responsáveis por reger a dinâmica do sistema BitTorrent.

2.2.1 Algoritmo de seleção de pares

A ideia chave do algoritmo de seleção de pares é priorizar os vizinhos mais importantes, ou seja, aqueles dos quais o usuário recebe mais informação. No entanto, o conjunto dos vizinhos mais importantes pode mudar com bastante frequência ao longo do tempo. Por isso, o algoritmo de seleção de vizinhos é executado periodicamente. Ao final de sua execução, ele envia mensagens de *unchoke* (desbloqueio) aos vizinhos que receberam o direito de fazer pedidos ao usuário e mensagens de *choke* (bloqueio) aos vizinhos que passaram a estar bloqueados.

O algoritmo é executado periodicamente a cada T_P segundos por cada peer e resulta no envio das mensagens de *unchoke* e *choke*. Seja x o número máximo de *unchokes* simultâneos que um peer pode conceder. A seleção de pares depende do tipo de peer que o executa, se é um leecher ou um seed. No caso do leecher, o número de *unchokes* simultâneos é composto de 2 parcelas: *unchokes* regulares x_r e *unchokes* otimistas x_o, tal que $x = x_r + x_o$. Os *unchokes* regulares são concedidos aos peers interessados que mais contribuíram com download para o leecher na última janela de observação w. Por este motivo, diz-se que o BitTorrent implementa a política *tit-for-tat*. Note que quanto mais upload um peer fizer, maior é a chance de ele ser selecionado pelos seus vizinhos. Portanto, o algoritmo de seleção de vizinhos funciona também como um mecanismo de incentivo para que os peers contribuam uns com os outros.

Na literatura, a descrição do funcionamento do *unchoke* otimista é confusa, pois é feita de forma diferente em diferentes trabalhos. Apresentaremos aqui uma descrição obtida a partir do estudo do código do cliente que usaremos como referência: o BitTorrent 4.0.0 [3]. Considere que cada leecher mantenha uma lista de vizinhos. Os *unchokes* otimistas são concedidos durante a seleção de pares aos peers interessados

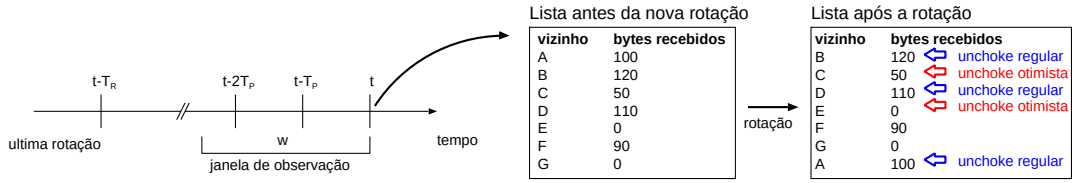


Figura 2.3: Execução do algoritmo de seleção de pares com rotação, para $x_r = 3$ e $x_o = 2$.

que aparecem no início na lista do leecher e que não receberam *unchokes* regulares. A cada T_R segundos, a lista do leecher é rotacionada em uma posição, alterando, muitas vezes, os vizinhos que receberão o *unchoke* otimista. A Figura 2.3 ilustra uma execução do algoritmo de seleção de pares, mostrando os vizinhos que são selecionados para receberem *unchokes* regulares e *unchokes* otimistas.

No caso do seed, a cada T_P segundos, os vizinhos que receberam *unchoke* há mais tempo são bloqueados e dão lugar àqueles que estão bloqueados há mais tempo. A quantidade de vizinhos substituída em cada execução é escolhida de modo que cada vizinho permaneça desbloqueado por T_S segundos, onde T_S é múltiplo de T_P.

2.2.2 Algoritmo de seleção de pedaços

Essencialmente, o algoritmo de seleção de pedaços (veja a Figura 2.1) prioriza aqueles mais raros para que sejam rapidamente disseminados pelo swarm. Assim, torna-se mais fácil recuperar estes pedaços e, ao mesmo tempo, o swarm fica mais resiliente à saída de peers. Note que como os pedaços mais raros estão sendo replicados mais rapidamente, é menos provável que um peer saia do swarm levando consigo a única cópia de um determinado pedaço.

Os pedidos dos leechers são feitos por bloco, uma subunidade do pedaço. O algoritmo de seleção de pedaços, na realidade, determina de qual pedaço o bloco deve ser pedido, dado um certo vizinho. Ele é composto por quatro modos de operação:

- *Strict priority.* O strict priority prioriza os pedaços incompletos, isto é, aqueles que o leecher já começou a baixar, mas ainda não possui todos os blocos. Apesar de não ser comentado na literatura, o cliente de referência mantém listas separadas para os pedaços incompletos que começaram a ser baixados de outro leecher e os pedaços que começaram a ser baixados de um seed. Se o vizinho em questão não possui nenhum dos pedaços que o leecher considera como sendo incompletos, o strict priority dá lugar a uma das outras políticas.

- *Random first.* Caso o número de pedaços completos que o leecher já possui seja menor que o valor limitante r, o random first é empregado: um pedaço é

Tabela 2.1: Parâmetros do cliente de referência.

Parâmetro	Valor
tamanho da lista de peers	$L = 50$ peers
tentativas de conexão	$A = 35$ peers
número mínimo de vizinhos	$m = 20$ peers
número máximo de vizinhos	$M = 80$ peers
intervalo de pedido de lista	$T_L = 25$ segundos
intervalo de seleção de pares	$T_P = 10$ segundos
período de *unchoke* do seed	$T_S = 30$ segundos
intervalo de rotação da lista de vizinhos do leecher	$T_R = 30$ segundos
número de *unchokes* simultâneos	$x = 4$ *unchokes*
número de *unchokes* regulares	$x_r = 3$ *unchokes* regulares
número de *unchokes* otimistas	$x_o = 1$ *unchoke* otimista
tamanho da janela de observação	$w = 20$ segundos
limite do random first	$r = 4$ pedaços

escolhido aleatoriamente entre aqueles que o vizinho tem e que interessam ao leecher.

- *Rarest first*. Caso o leecher tenha pelo menos r pedaços completos, o rarest first determina que o mais raro dentre aqueles disponíveis na vizinhança seja escolhido. Para isso, o leecher verifica quantas cópias de cada pedaço do conteúdo estão disponíveis na sua vizinhança, verificando o bitfield de cada um de seus peers. Se houver empate, o pedaço a ser pedido é escolhido aleatoriamente.

  Finalmente, quando um leecher detecta que todos os blocos do conteúdo já foram pedidos e que está apenas aguardando o seu recebimento para terminar o download, o modo *end game* entra em ação.

- Modo *end game*. Nesse modo, o leecher passa a pedir os mesmos blocos para outros vizinhos e, assim que cada bloco é recebido, ele cancela os pedidos duplicados.

Utilizaremos o cliente BitTorrent 4.0.0 como referência para todos os modelos desenvolvidos neste trabalho. Os valores dos parâmetros deste cliente são encontrados na Tabela 2.1. É importante ressaltar que o BitTorrent é um protocolo aberto e que existem diversas outras implementações, como o Vuze, o BitComet, o μTorrent, e o Transmission, que possuem muitas variações nos detalhes de seu funcionamento.

2.3 Trabalhos Relacionados

Diversos modelos foram propostos na literatura para estudar o comportamento e desempenho do BitTorrent e outras redes P2P similares. Entre os mais importantes,

está o modelo de fluido proposto por Qiu e Srikant [13] para estudar a escalabilidade, desempenho e eficiência do BitTorrent para uma população homogênea. Dois anos depois, um modelo baseado em equações diferenciais estocásticas foi proposto por Fan et al. [14], capaz de obter resultados mais precisos que o anterior. Além disso, este último trabalho traz uma série de soluções fechadas para medidas de desempenho do sistema em estado estacionário em conjunto com uma análise de sensibilidade do tempo médio de download para os parâmetros do sistema. Por fim, ele apresenta um modelo para estudar a disponibilidade de um arquivo e, para aumentá-la, propõe uma nova política de seleção de pedaços.

Yang e Veciana [15] estudaram o regime transiente de um swarm através de um processo de ramificação. Eles mostraram que durante este regime a capacidade do sistema cresce exponencialmente com a taxa de chegada dos peers, o que tornaria o sistema escalável.

Em alguns clientes mais recentes do protocolo BitTorrent é possível configurar o número máximo de uploads. Neglia et al. [16] definem um jogo em que um usuário do BitTorrent pode escolher o número de uploads simultâneos e estudam a perda de eficiência do sistema em função da falta de coordenação dos peers.

Para caracterizar a influência dos algoritmos de seleção de pedaços e de seleção de pares, Legout et al. [17] realizaram uma série de experimentos reais. Os autores concluem que a substituição do *rarest-first* e do algoritmo de seleção de pares por outros algoritmos não pode trazer ganhos substanciais ao desempenho do sistema e, portanto, não se justificam.

Existem ainda alguns trabalhos que calculam limites relacionados ao desempenho do sistema. O tempo de disseminação de um arquivo, por exemplo, é o tempo necessário para que todo um conjunto de usuários recuperem o arquivo por completo. Kumar e Ross [18] derivaram o tempo mínimo para disseminação de um arquivo e comparam com o tempo obtido pelo BitTorrent, concluindo que há pouca perda de eficiência. Note que isso é diferente de se minimizar o tempo médio de download, um problema que é abordado por Ezovski e Andrew [19]. Eles encontraram uma fórmula para o valor mínimo do tempo médio de download e propuseram um escalonamento para alcançá-lo, mas que é baseado em uma solução centralizada.

O primeiro trabalho a considerar swarms heterogêneos com relação à largura de banda foi proposto por Piccolo e Neglia[20]. Neste trabalho, os autores estudam o impacto da heterogeneidade e mostram que esta pode melhorar a disseminação dos blocos.

Um modelo matemático para o tempo médio de download em populações heterogêneas foi descrito por Liao et al. [21]. No entanto, este modelo assume uma população inicial de tamanho fixo. Já o modelo proposto por Chow et al. [22] considera que o sistema está em estado estacionário e que novas chegadas e saídas

continuam acontecendo.

Um segundo conjunto de trabalhos está voltado para a identificação e caracterização de fenômenos em sistemas P2P para compartilhamento de arquivos. Como exemplo, podemos citar a clusterização dos peers segundo sua largura de banda, isto é, peers que têm capacidade similar tendem a trocar mais dados entre si. A primeira validação experimental desse fenômeno foi realizada por Legout et al. [7]. Experimentos executados por Bharambe et al. [8] corroboram este resultado e mostram que se a clusterização fosse explicitamente implementada pelo protocolo, a utilização média do *uplink* aumentaria significativamente, melhorando o desempenho do sistema como um todo.

Uma outra característica interessante é a capacidade de um swarm de recuperar um conteúdo sem que haja nenhum seed no sistema. Esta capacidade é conhecida no contexto de aplicações P2P como autossustentabilidade. Menasche et al. [23] investigam a dependência dos peers em relação a um seed e propõem um modelo capaz de estimar a autossustentabilidade em função da popularidade do conteúdo.

Por fim, Hajek e Zhu [24] identificam um problema que pode ocorrer quando a capacidade do seed não é grande o suficiente em relação à taxa de chegada de novos usuários: um pedaço pode se tornar muito raro e o número de leechers no sistema pode crescer indefinidamente. Esse problema é conhecido como a síndrome do pedaço faltante.

Capítulo 3

Assortative Mixing

A formação de relacionamentos entre vértices com características similares é uma propriedade que vem sendo observada em diversas redes reais. Esta característica, conhecida como *assortative mixing* (AM), está presente não somente em redes sociais, mas também vem sendo observada em redes par-a-par (P2P) para compartilhamento de arquivos, onde a similaridade dos vértices é dada pela largura de banda de acesso dos pares. Esta "clusterização" em redes P2P, apesar de ser responsável pelo bom desempenho do sistema, não está explicitamente refletida no funcionamento dos protocolos destas aplicações, como o BitTorrent (BT). Neste capítulo, propomos um modelo simples da dinâmica das conexões entre os peers do BT para entender quais mecanismos do protocolo BitTorrent levam à ocorrência do AM. Além disso, estudamos a influência dos parâmetros do modelo na taxa de AM atingida pelo sistema. Finalmente, apresentamos um modelo analítico simplificado da dinâmica de conexões do sistema e estudamos seu regime estacionário.

3.1 Introdução

Muitas redes que representam sistemas reais possuem características topológicas comuns, mesmo aquelas de origem distintas, tais como redes sociais, tecnológicas, biológicas ou de informação. Dentre as características mais frequentemente observadas, podemos citar a baixa distância média entre os vértices, a alta clusterização e a distribuição do grau dos vértices em cauda longa [25, 26].

Uma outra característica topológica que vem sendo observada é o relacionamento entre vértices similares (ou diferentes) sob algum aspecto de similaridade, ou seja, ao rotularmos os vértices com seus respectivos tipos, existe uma tendência maior para relacionamentos entre vértices do mesmo tipo (ou de tipos bem diferentes) [27]. Em redes sociais, onde a rede representa algum relacionamento entre um conjunto de pessoas, é bem estudado e conhecido o fato de que pessoas tendem a se relacionar com pessoas similares, seja o tipo dado pela idade, sexo, etnia, nacionalidade, nível

de escolaridade, prestígio, classe social, posses e ocupação [28, 29]. Este fenômeno de relacionamento entre pares similares é conhecido por *assortative mixing* (AM) ou *diadicity* [30].

O AM também vem sendo observado em redes distintas das redes sociais. Em redes par-a-par (P2P) para compartilhamento de arquivos, tais como BitTorrent e Gnutella, *peers* (vértices) que formam a rede possuem conexões com diferentes larguras de banda, variando de conexões discadas de 56 Kbps àquelas que utilizam outras tecnologias, alcançando velocidades superiores à 100 Mbps. Recentemente, observou-se AM entre os peers de uma rede BitTorrent quando o tipo do peer é dado pela sua largura de banda [7], ou seja, peers tendem a trocar informação com peers que possuem largura de banda parecida com a sua própria.

Assim como outras propriedades topológicas, o AM também possui implicações fundamentais na funcionalidade do sistema. No caso de sistemas P2P, como o BitTorrent, a presença de AM com relação à largura de banda está diretamente relacionada à escalabilidade e robustez do sistema. Em particular, um sistema P2P com alta taxa de AM é mais eficiente e oferece melhor serviço a seus usuários. É então fundamental entender por que tais sistemas atingem altas taxas de AM.

Conforme vimos no Capítulo 2, o protocolo BitTorrent para compartilhamento de arquivos é relativamente complexo, sendo formado por diversos mecanismos que operam simultaneamente. Em essência, o protocolo consiste em um processo dinâmico de troca de peers que tenta maximizar a taxa de download dando em troca taxas de upload. No entanto, não há preferência explícita no protocolo por pares com largura de banda similar. O fenômeno observado é decorrência dos mecanismos que definem o protocolo. Mas que mecanismos são estes? Qual é a influência dos parâmetros do sistema, neste processo, que leva a altas taxas de AM?

Neste capítulo, estamos interessados em estudar o processo dinâmico do protocolo BitTorrent para entender quais mecanismos do protocolo são responsáveis por induzir o sistema a um estado com alta taxa de AM. Com esse objetivo, iremos propor dois modelos para a dinâmica do BT. O primeiro se trata de um modelo detalhado que captura a essência da dinâmica do protocolo BitTorrent, onde todos os peers estão representados no modelo. Através de simulações, mostramos que o processo dinâmico, que se baseia apenas em regras locais, leva o sistema de um estado inicial aleatório a um estado com alta taxa de AM. Estudamos ainda o impacto dos parâmetros do modelo neste processo dinâmico e observamos que, se a parametrização original do BitTorrent fosse modificada, poderíamos induzir o surgimento do AM mais rapidamente ou alcançar valores mais elevados.

Contudo, é impraticável calcular as probabilidades de estado em regime estacionário do sistema representado por este primeiro modelo devido ao seu imenso espaço de estados. Por essa razão, vamos propor ainda um segundo modelo que irá

nos permitir analisar a rede de conexões do BT nesse regime. A fim de que o modelo fosse tratável e, consequentemente, que fosse possível obter soluções analíticas para o mesmo, consideramos em detalhe um único peer, enquanto o restante do sistema é modelado de forma agregada.

3.2 Modelo detalhado da dinâmica das conexões

Na Seção 2.2, vimos que quando um leecher contacta o tracker para iniciar o download de um dado conteúdo, recebe uma lista contendo peers escolhidos de forma uniformemente aleatória dentre todos os do swarm. Dentre estes, são selecionados alguns peers de maneira uniforme para abrir uma conexão, podendo esta ser estabelecida ou não. Note que o conhecimento que cada peer possui define uma rede que indica o conjunto dos peers conhecidos por cada um. O conhecimento é um relacionamento bidirecional: se i conhece j, então j conhece i.

Uma forma natural de representar essa rede é através de um grafo não direcionado. Por simplicidade, vamos supor que a rede é formada por n peers e que cada um deles conhece k outros peers e, portanto, esta rede pode ser bem representada por um grafo aleatório k-regular (todo vértice tem grau k). Vamos chamar este grafo de **"grafo de conhecimento"**, que será representado por G_c. Vamos assumir que nenhum peer entra ou sai da rede e, por conseguinte, que esse grafo é estático, isto é, o conhecimento se mantém imutável ao longo do tempo.

No protocolo BitTorrent, o algoritmo de seleção de pares é executado por todos os peers. O envio de dados é assimétrico, ou seja, caso i esteja enviando dados para j não implica que j esteja enviando dados para i. Além disso, um peer só pode enviar dados para os peers que ele conhece. Assim, o relacionamento "envio de dados" será codificado no nosso modelo através de uma aresta direcionada, cuja existência está condicionada ao fato dos vértices origem e destino serem vizinhos em G_c. Além disso, um peer só pode enviar dados para os vizinhos que neles estão interessados. Contudo, estamos assumindo que, a todo instante de tempo, todo peer possui pelo menos um pedaço que interessa a cada um de seus vizinhos. Esta suposição é razoável quando o seed é bem provisionado e é essencial para que ocorra clusterização segundo Legout et al. [7].

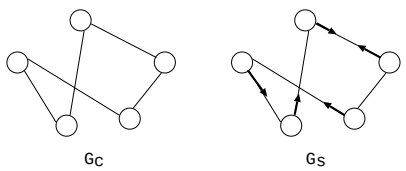


Figura 3.1: Exemplo de grafo de conhecimento e grafo de serviço para os parâmetros n=5, k=2, x=1.

Seja x o número de *unchokes* simultâneos que um peer mantém, ou seja, x arestas direcionadas representam o fluxo de dados de upload de cada vértice. As conexões ativas no nosso modelo serão representadas por um grafo direcionado G_s, definido sobre o grafo de conhecimento, conforme mostrado na Figura 3.1. Este grafo será denominado **"grafo de serviço"**.

A ideia do algoritmo de seleção de pares é dar prioridade aos relacionamentos que trazem mais benefícios para um peer. No contexto de compartilhamento de arquivos, o relacionamento mais valioso é aquele que é capaz de fornecer mais informação por unidade de tempo.

Assim, quando um peer i executa esse algoritmo, ele irá enviar mensagens de *unchoke* regular aos peers dos quais conseguiu extrair mais dados, levando em consideração as arestas que incidem nele em G_s. Lembre que a existência de uma aresta (j, i) em G_s indica que j está transmitindo para i e, analogamente, a inexistência indica que não está ocorrendo transmissão nesse sentido. Ao final do algoritmo, o peer estará retribuindo através de arestas que saem de i em G_s aos vizinhos que para ele são mais vantajosos.

A quantidade de dados fornecida é função, dentre outras coisas, da largura de banda do peer que envia os dados e da largura de banda do peer que os recebe. Mais especificamente, a taxa de transmissão de i para j é limitada superiormente por $\min\{\text{uplink}(i), \text{downlink}(j)\}$.

Por simplicidade, vamos supor no modelo que o *uplink* e o *downlink* de um usuário i são simétricos e serão indicados pela tag $c_i \in \{1, \ldots, v\}$, onde v é o maior valor possível de c_i. No entanto, essa suposição pode ser eliminada de forma a representar melhor usuários que fazem uso de tecnologias assimétricas, como o *Cable* e o ADSL.

Assumindo que a largura de banda de um peer e de seu vizinho sejam os únicos fatores que influenciam a transmissão de dados entre eles, podemos afirmar que a taxa máxima de transmissão de i para j é igual ao mínimo entre as suas capacidades. Se, por exemplo, $c_i > c_j$, o peer i não poderá enviar dados para j a uma taxa maior que c_j, pois j não pode receber a uma taxa maior que sua própria capacidade. Analogamente, se $c_i < c_j$, a taxa de transmissão é limitada por c_i.

Portanto, o benefício que um relacionamento indicado pela aresta (j, i) traz ao vértice j, pode ser mensurado pela taxa máxima de transmissão de j para i. Assim sendo, podemos definir uma função utilidade f que captura a prioridade de um vizinho j para receber o *unchoke* de um vértice i, onde valores mais altos indicam maior prioridade:

$$f(i,j) = \begin{cases} 0 & \text{se } \nexists (j,i) \in G_s \\ \min\{c_i, c_j\} & \text{caso contrário} \end{cases}$$

No BitTorrent, a cada vez que o algoritmo de seleção de pares é executado em

um nó, x_r peers recebem o *unchoke* regular e x_o peers recebem o *unchoke* otimista. Note que todos os peers que haviam recebido *unchoke* antes da execução do algoritmo podem ser substituídos. Nosso modelo irá incorporar este mecanismo de troca de arestas em G_s de maneira simplificada, permitindo que um vértice troque apenas uma aresta de saída a cada execução do algoritmo local, ou seja, deixe de fazer upload para um de seus vizinhos e passe a fazer para outro.

A dinâmica do modelo será a seguinte: a cada passo, um vértice i é aleatoriamente escolhido para realizar a troca de aresta. A conexão com o vértice j menos valioso dentre os que estão sendo servidos atualmente será fechada e será estabelecida uma conexão com o vértice k mais valioso dentre os que não estão sendo servidos por i, desde que $f(i,k) > f(i,j)$. Isto é, vamos substituir a aresta (i,j) pela aresta (i,k) em G_s. Formalmente, seja $j_{min} = \arg\min_j f(i,j) \quad \forall (i,j) \in G_s$ e $k_{max} = \arg\max_k f(i,k) \quad \forall (i,k) \in G_c \quad e \quad (i,k) \notin G_s$. Se $f(i,j_{min}) < f(i,k_{max})$, então a aresta direcionada (i,j_{min}) é substituída pela aresta direcionada (i,k_{max}). Essa substituição será chamada de *troca regular* e equivale ao *unchoke* regular do protocolo BT.

A Figura 3.2 ilustra o funcionamento desse mecanismo. O número ao lado de um vértice indica a sua tag. Na primeira parte da figura, o vértice A é escolhido para realizar a troca regular. Ele possui dois vizinhos, B e D, cujas prioridades são dadas por $f(A,B) = 0$ e $f(A,D) = 1$, respectivamente. Note que o vizinho B não possui aresta direcionada para A e, por isso, possui menor prioridade. Logo, o vértice A substitui a aresta (A,B) pela aresta (A,D), estando essa mudança refletida no segundo grafo da figura. Nesse grafo, C é escolhido para realizar uma troca regular. Apesar de ambos os vizinhos E e B do vértice C possuírem arestas de saída para C em G_s, $f(C,B) > f(C,E)$. Isso significa que B pode oferecer um serviço de melhor qualidade ao vértice C, recebendo então maior prioridade. Finalmente, o vértice C substitui a aresta (C,E) pela aresta (C,B), resultando no terceiro grafo da figura.

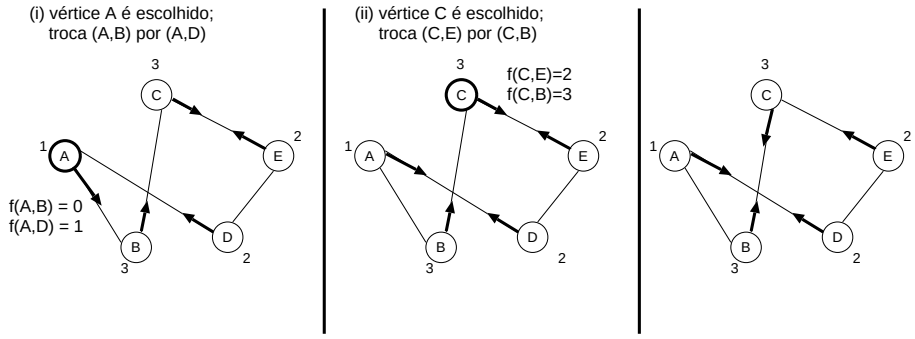


Figura 3.2: Exemplos de troca regular. Nota-se a preferência pela reciprocidade e pela qualidade de serviço.

Da forma como foi descrita até o momento, a dinâmica do modelo faria com que

Tabela 3.1: Parâmetros do modelo detalhado da dinâmica de conexões.

Parâmetro	Descrição
n	número total de peers
k	número de vizinhos de cada peer
x	número de *uploads* simultâneos por peer
p	probabilidade de executar a troca otimista
v	número de valores de tags diferentes

as arestas com origem em i fossem aos poucos sendo direcionadas aos vértices que possuem arestas para i. Isso limitaria a descoberta de vértices com a mesma tag, pois uma vez que i estiver retribuindo a todos os seus vizinhos, ele não irá mais fazer trocas de arestas. Essa troca de arestas cessa porque i não irá substituir um vizinho j que para ele faz upload, i.e. $f(i,j) > 0$, por um vizinho k que não faz, i.e. $f(i.k) = 0$. Como veremos na seção seguinte, a *troca otimista* (equivalente ao *unchoke* otimista) faz com que a troca de arestas prossiga e é fundamental para o surgimento de altas taxas de AM.

Se ao ser selecionado, um vértice não desejar realizar uma troca regular, ou seja, $f(i, j_{min}) \geq f(i, k_{max})$, ele irá fazer uma troca otimista com probabilidade p, onde p é um parâmetro do modelo. Nesse caso, ele irá substituir sua aresta (i, j_{min}) por uma aresta $(i,k) \in G_c$ e $(i,k) \notin G_s$ escolhida aleatoriamente. O parâmetro p e os outros parâmetros empregados nesse modelo estão listados na Tabela 3.1.

A troca otimista está ilustrada na Figura 3.3. O vértice D é escolhido para fazer uma troca regular, mas não deseja substituir nenhuma aresta, pois já possui aresta de saída para o vizinho de maior prioridade, isto é, para o vértice A. Nesse caso, a troca otimista irá ocorrer com probabilidade p e cada um dos vizinhos para os quais D não tem aresta pode ser selecionado com mesma probabilidade. Em particular, nessa figura, ocorrendo a troca otimista, o vértice E é o único que pode ser escolhido.

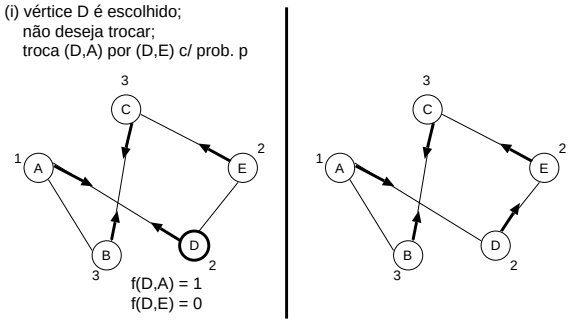


Figura 3.3: Exemplos de troca otimista. Se o vértice E for o próximo a realizar a troca, ele irá substituir (E,C) por (E,D), aumentando o grau de AM.

Resumidamente, as simplificações que empregamos no modelo são: a inexistência de novas chegadas ou partidas de peers no sistema, a representação do grafo de co-

nhecimento através de um grafo regular, a seleção aleatória do peer que irá executar a troca regular, a impossibilidade de trocar mais de uma aresta simultaneamente e a avaliação da prioridade dos vizinhos de um peer em função apenas das capacidades e da (in)existência de arestas.

Apesar das diversas simplificações, o modelo permite estudar ambientes mais reais e parâmetros mais diversificados. Podemos, por exemplo, levar em consideração a assimetria entre as bandas de download e upload dos peers definindo uma tag como sendo um par (*downlink*, *uplink*) e mudar a definição da função acima para representar o novo gargalo de transmissão. A função utilidade pode ainda ser descrita de forma arbitrária para outros tipos de rede, desde que caracterize corretamente a prioridade que os peers têm para estabelecer relacionamentos. Variações na estrutura do grafo do conhecimento e no grafo de serviço, além de distribuição não uniforme das tags também podem ser capturadas pelo modelo. Vale notar que o modelo apresentado é bem geral e poderia ser prontamente utilizado para o estudo de outras redes mudando-se a função que determina a prioridade de um peer ao estabelecer um relacionamento. Em uma rede de trocas comerciais, por exemplo, poderia-se priorizar o fornecedor que oferece o produto mais barato ou com menor tempo de entrega.

3.2.1 Experimentos e resultados

A dinâmica do modelo apresentado acima pode ser descrita por uma sequência de estados, onde cada estado é composto por variáveis que indicam os vértices aos quais incidem as arestas do grafo de serviço G_s em um dado instante de tempo. Note que esse modelo é Markoviano: a transição para o próximo estado só depende do estado atual. Em particular, ele pode ser representado por uma Cadeia de Markov de Tempo Discreto (CMTD). Entretanto, é difícil encontrar soluções analíticas para essa cadeia devido ao seu imenso espaço de estados. Note que cada vértice escolhe x entre k vizinhos para fazer upload. Então, mesmo para valores baixos de parâmetros, tais como $n = 10$, $k = 5$, $x = 3$, o espaço de estados é muito grande: $n^{\binom{k}{x}} = 10^{10}$. Por essa razão, fizemos o uso de simulações para obter resultados a partir deste modelo detalhado da dinâmica do BT. Mais precisamente, desenvolvemos um simulador em C++ específico para este modelo que permite simular sistemas bem grandes.

No conjunto de experimentos que segue, avaliaremos o grau de AM do grafo de serviço ao longo do tempo. Serão realizados experimentos variando individualmente os parâmetros do modelo listados na Tabela 3.3 a fim de estudar sua influência.

Para os experimentos a seguir, os seguintes parâmetros são utilizados a menos que o contrário seja dito: $n = 1000, k = 50, x = 10, v = 2, p = 0.40$ e tags uniformemente distribuídas (isto é, cada valor possível de tag será atribuído a n/v vértices). Repare

que o valor dos parâmetros escolhidos em si não é tão relevante, pois nosso objetivo é justamente variá-los de forma a caracterizar sua influência no grau de AM. Em particular, quando $x = 10$, estamos considerando um número de uploads simultâneos maior do que no cliente de referência (i.e., três).

Para quantificar o AM do grafo de serviço G_s em um dado instante, vamos utilizar como medida o *assortative coefficient* definido por Newman [27] da seguinte forma. Seja E_{ij} o número de arestas em uma rede que conectam vértices do tipo i com vértices do tipo j, para $i, j \in \{1, \ldots, v\}$. Seja $\mathbf{E}$ a matriz formada pelos elementos E_{ij}. Normalizando $\mathbf{E}$ pelo somatório de seus elementos, temos a matriz $\mathbf{e}$:

$$\mathbf{e} = \frac{\mathbf{E}}{\sum_{i,j} E_{ij}}$$

O *assortative coefficient* é então calculado da seguinte maneira:

$$r = \frac{\sum_i e_{ii} - \sum_i a_i b_i}{1 - \sum_i a_i b_i} \tag{3.1}$$

onde $a_i = \sum_j e_{ij}$ e $b_i = \sum_j e_{ji}$. Repare que quando os relacionamentos são estabelecidos de forma aleatória, ou seja, não existe nenhuma preferência por relacionamentos entre os tipos de vértices, a fração de arestas entre vértices do mesmo tipo (e_{ii}) é aproximadamente o valor esperado da fração de arestas entre eles (i.e., $a_i b_i$) para todo i e, portanto, os termos do numerador se anulam fazendo com que r fique próximo de zero. Por outro lado, quando os vértices somente se relacionam com outros do mesmo tipo, $\sum_i e_{ii} = 1$ e, consequentemente, r vale 1 também. Quando o *assortative coefficient* está próximo de 1, dizemos que o grafo possui alta taxa de AM.

Durante os experimentos, verificamos que diferentes execuções da simulação geravam curvas muito semelhantes e, por isso, o intervalo de confiança resultante é relativamente pequeno. Por simplicidade, iremos mostrar, para cada experimento, os resultados correspondentes à uma única execução, omitindo os intervalos de confiança.

As interações que um vértice realiza com os outros são locais, o que indica que o número de vértices no sistema não influencia no valor do *assortative coefficient*. No entanto, aumentando o número de vértices, aumentamos também o número médio de iterações até que um dado vértice seja selecionado durante a simulação. Por esse motivo, normalizamos o número de iterações usando o número de vértices, ou seja, o eixo das abcissas representa o número de iterações dividido por n. Podemos observar que na Figura 3.4 as curvas são quase indistinguíveis, o que está de acordo com nossa intuição.

Mais do que analisar os valores de x e k separadamente, estamos interessados na

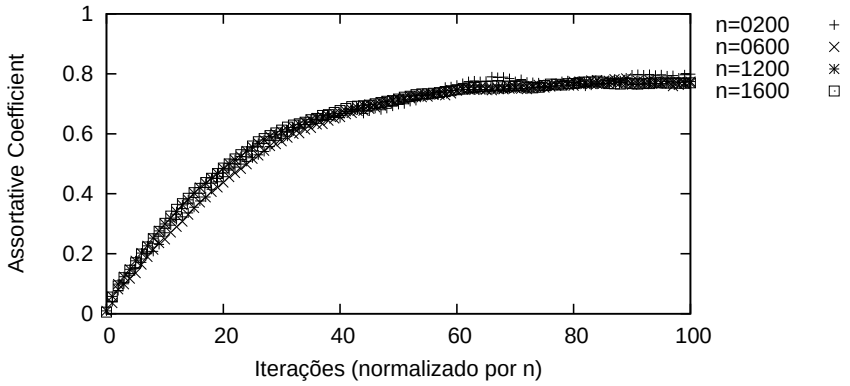


Figura 3.4: Variação de r em função do número de vértices n.

taxa de AM quando uma fração x/k dos peers está conectada. Para isso, faremos variar o grau de saída do vértice em G_s (x), mantendo fixo o número de vizinhos em G_c (k).

Intuitivamente, diminuindo a fração x/k de vizinhos para o qual um peer faz upload, a taxa de AM aumenta, pois é mais fácil encontrar grupos pequenos de vértices com a mesma tag. Da mesma forma, para valores muito altos de x/k, a fração de vizinhos com a mesma tag que um vértice passa a ser menor que x/k, limitando assim o valor do *assortative coefficient*. No entanto, devemos considerar os efeitos da troca otimista quando x é muito pequeno, pois neste caso, muitas vezes estamos trocando um vértice com a mesma tag por um vértice com tag diferente e, como o número de vértices que recebe upload é pequeno, essa troca irá reduzir significativamente o valor do *assortative coefficient*.

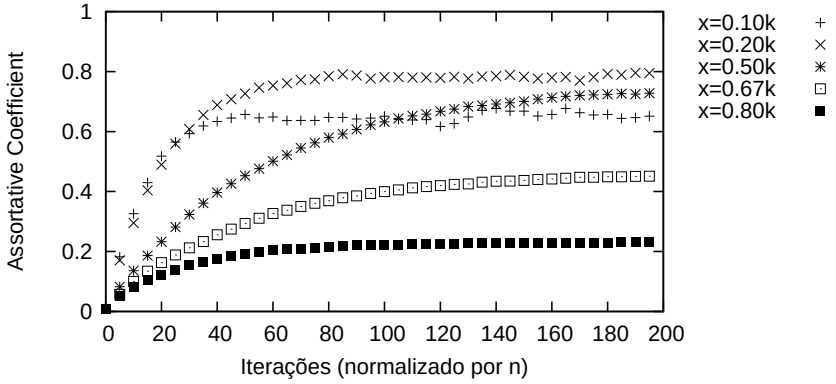


Figura 3.5: Variação de r em função do número de uploads simultâneos x.

A Figura 3.5 mostra a evolução do *assortative coefficient* r (ver Equação (3.1)) ao longo do tempo, para diferentes frações x/k. Note que para $x = 0.80k, 0.67k, 0.50k, 0.20k$ o valor do *assortative coefficient* alcançado é estritamente crescente na redução de x. Porém, para $x = 0.10k (x = 5)$, a alta probabilidade de troca otimista $(p = 0.40)$ faz com que: (i) o *assortative coefficient* alcançado seja

menor e (ii) a curva apresente mais variabilidade que as demais. Observe que como $v = 2$, é inevitável que para $x > 0.5k$ alguns peers deem *unchoke* em peers que possuem tags diferentes. Afinal, um peer terá em média $k/v = 0.5k$ vizinhos com a mesma tag que ele.

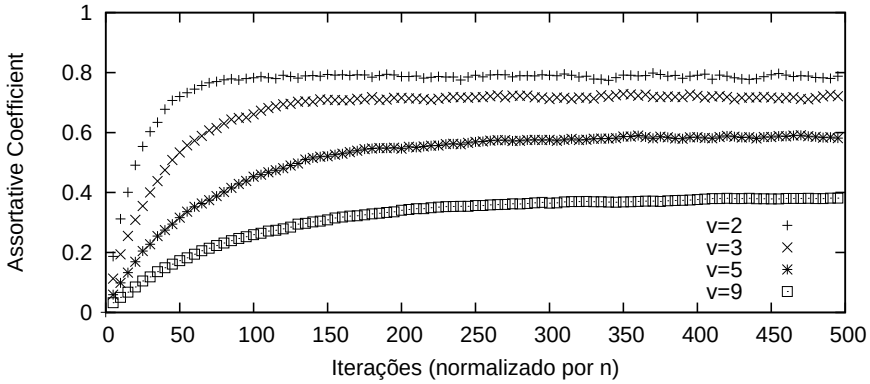


Figura 3.6: Variação de r em função do número de tags diferentes v.

Variar o número de tags tem um efeito bastante semelhante a variar o número de uploads simultâneos. Quando temos mais tags, a quantidade de vizinhos de um vértice com a mesma tag que a dele diminui, fazendo com que fique mais difícil encontrá-los e atingir um alto *assortative coefficient* (ver Figura 3.6). Dado o número de tags, podemos calcular um limite assintótico para o valor do *assortative coefficient*. Na Subseção 3.2.2, apresentamos uma análise que relaciona os parâmetros x, k e v ao valor máximo que r pode assumir.

Conforme discutimos anteriormente, a troca otimista é fundamental para que um vértice descubra os vizinhos que possuem a mesma tag que ele. Sem este recurso, o modelo se resumiria praticamente a "casar" arestas direcionadas. Inicialmente, um vértice i com $c_i = 2$ que executa a troca otimista substituindo sua aresta "menos valiosa", irá manter ou aumentar o *assortative coefficient*, pois quando a taxa de AM é baixa, sua aresta "menos valiosa" provavelmente incide em um vértice j com $c_j = 1$. Por isso, valores altos de p fazem com que o *assortative coefficient* cresça mais rapidamente. Por outro lado, depois que r atinge valores altos, a troca otimista faz com que os vértices substituam arestas incidentes a vértices de mesma tag, por arestas que incidem em vértices de tag diferente. Logo, valores altos de p fazem com que o valor final de r seja mais baixo. Para verificar esses fatos, observamos o *assortative coefficient* para diferentes valores de p.

Os resultados mostrados na Figura 3.7 devem ser analisados em conjunto com a Tabela 3.2. As curvas mostram a evolução de r ao longo do tempo, para diferentes valores de p, enquanto as linhas da tabela mostram uma aproximação para o valor de convergência de r. Observamos que aproximadamente até a iteração 10000 todas as curvas crescem de maneira similar, independentemente do valor de p. Isso indica que,

Tabela 3.2: Valor médio do *assortative coefficient* (r) para as últimas 100 de 4200 iterações (normalizado por n).

Valor de p	Valor de r
0.00	0.1624
0.01	0.8815
0.05	0.8508
0.20	0.7889
1.00	0.7694

no início, a clusterização é causada principalmente por trocas regulares. Note que a curva correspondente a $p = 0$ para de crescer por volta deste instante, atingindo uma baixa taxa de AM ($r = 0.1624$, segundo a tabela). A partir de então, o efeito causado pelas trocas otimistas passa a ser dominante. Considerando apenas as curvas em que $p > 0$, concluímos que quanto maior o valor de p, mais rápido ocorre a convergência de r. Entretanto, observamos que conforme o valor de p aumenta, menor é o valor atingido por r. A curva $p = 0.01$, por exemplo, tem média igual a 0.8815 no intervalo que vai de 4100 a 4200 iterações normalizadas. Este valor é significativamente mais alto do que aqueles alcançados com valores menores de p.

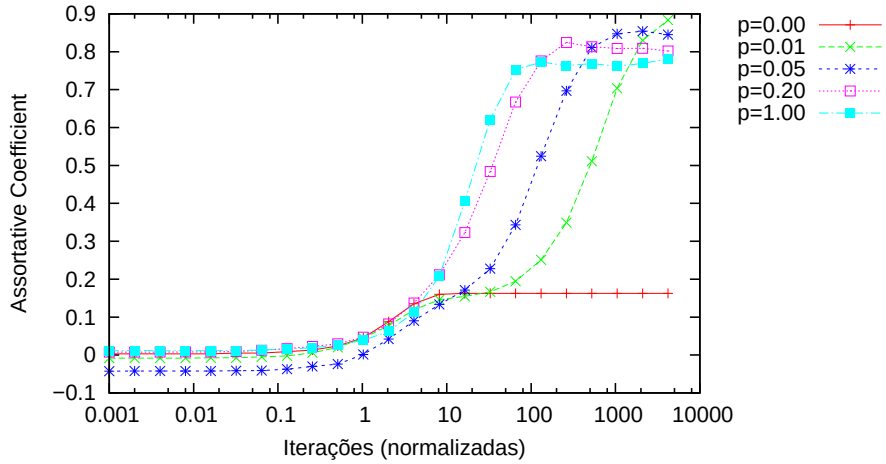


Figura 3.7: Variação de r em função da probabilidade de troca otimista p.

Por fim, repare que para um dado grafo de conhecimento, o valor máximo que o *assortative coefficient* pode atingir é um número fixo. Note ainda que este coeficiente seria facilmente obtido se as tags dos vizinhos fossem todas conhecidas e houvesse uma preferência explícita pelo fornecimento de serviço a vértices de mesma tag. Ainda assim, existe uma probabilidade não nula de alcançarmos este coeficiente com o modelo proposto. Sendo o modelo do grafo de conhecimento aleatório, o valor máximo do *assortative coefficient* é, portanto, uma variável aleatória R_{max}. A seguir iremos calcular a esperança dessa variável, isto é, $E[R_{max}]$.

3.2.2 Cálculo analítico de $E[R_{max}]$

A contribuição individual de um vértice para R_{max} é dada em função das tags dos vizinhos a quem ele está conectado. Vamos assumir que o número total de vértices de cada tipo é maior ou igual a $k+1$, fazendo com que seja sempre possível que um vértice tenha a mesma tag que todos seus vizinhos. Seja Z o número de vizinhos com a mesma tag que um certo vértice em G_c. Se $Z \geq x$, ou seja, maior que o número de uploads que ele faz, então R_{max} é obtido quando todos os x uploads desse vértice são feitos para outros de mesma tag (ou seja, todas as arestas que partem do vértice em G_s incidem em vértices de mesma tag). Por outro lado, se $Z < x$, R_{max} é obtido quando Z de um total de x arestas são dispostas para os vizinhos de mesma tag.

Pela definição do modelo, Z é uma v.a. com distribuição $binomial(k, 1/v)$, desde que o número de vértices de cada tipo seja maior que k. Assim, o valor esperado de e_{ii} para R_{max} é obtido a partir da distribuição de Z nos vértices com tag i:

$$\begin{aligned} E[e_{ii}] &= \frac{\#\text{ arestas (u,v) t.q. } c_u = i}{\#\text{ total de arestas}} \times \mathrm{E}[\#\text{ arestas (u,w) t.q. } c_w = i | c_u = i] \\ &= \frac{n/v \times x}{nx} \times \sum_{j=1}^{x-1} P(Z=j) \times \frac{j}{x} + P(Z \geq x) \times 1 \\ &= \frac{1}{v} \times \sum_{j=1}^{x-1} P(Z=j) \times \frac{j}{x} + P(Z \geq x) \times 1 \end{aligned}$$

Como a distribuição de Z é idêntica para todas as v tags, temos

$$E\Big[\sum_i e_{ii}\Big] = \sum_{j=1}^{x-1} P(Z=j) \times \frac{j}{x} + P(Z \geq x) \times 1 \tag{3.2}$$

Além disso, como o número de vértices de cada tag é igual a n/v e o grau de saída é igual em G_s para todo vértice, a fração de arestas que partem de vértices de tag i é $a_i = 1/v$ para todas as tags. Para calcular $E[R_{max}]$, vamos assumir que $b_i = 1/v$ para todo i, ou seja, o número de arestas que chegam em vértices de tag i é igual para todas as tags. Essa suposição é bem realista, pois a função utilidade empregada captura a noção de reciprocidade, fazendo com que o número de arestas incidentes a cada vértice esteja muito próximo de x. Logo,

$$\begin{aligned} E[R_{max}] &= \frac{E[\sum_i e_{ii}] - \sum_i a_i b_i}{1 - \sum_i a_i b_i} \\ &= \frac{E[\sum_i e_{ii}] - \frac{1}{v}}{1 - \frac{1}{v}} \\ &= \frac{E[\sum_i e_{ii}] \times v - 1}{v - 1} \end{aligned} \tag{3.3}$$

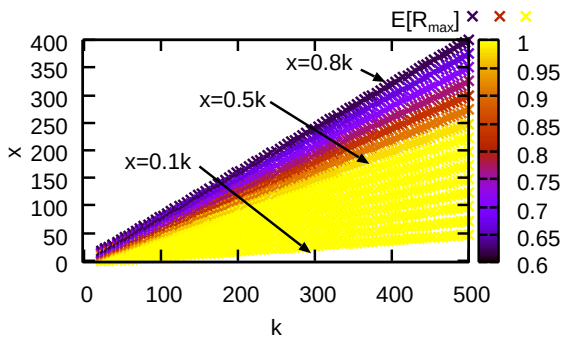


Figura 3.8: Valores de $E[R_{max}]$ para x=0.10k,0.15k,...,0.80k

A Figura 3.8 ilustra os resultados obtidos a partir da avaliação de $E[R_{max}]$ na Equação (3.3), para diversos valores de k e x. Note que $E[R_{max}]$ independe de n. A coordenada x de um ponto marcado nessa figura corresponde ao número de vizinhos k e a coordenada y, ao número de uploads simultâneos x. A cor, por sua vez, indica o valor de $E[R_{max}]$, variando de roxo ($E[R_{max}] = 0.6$) a amarelo ($E[R_{max}] = 1.0$). Note que todos os pontos em uma reta que parte de origem mantem a mesma proporção x/k. Em particular, as retas $x = \{0.1k, 0.5k, 0.8k\}$ estão marcadas no gráfico através de setas.

Observando pontos que estão sobre uma mesma reta, notamos que a variação de $E[R_{max}]$ é muito pequena. Isso indica que esse valor não depende dos valores absolutos de k ou x, mas sim da razão x/k, cujo impacto na dinâmica do sistema foi avaliada na Seção 3.2.1. Com respeito à essa fração, podemos observar que um aumento em x/k ocasiona uma redução de $E[R_{max}]$.

Para valores pequenos de x/k, essa relação é praticamente imperceptível no gráfico. Entretanto há uma mudança de comportamento quando $x > k/2$, conforme havíamos observado na Figura 3.5. Neste caso, $E[R_{max}]$ diminui rapidamente.

3.3 Modelo simplificado da dinâmica de conexões

O modelo de simulação discutido na seção anterior mostrou como os parâmetros do sistema impactam na velocidade e na intensidade com que o AM ocorre no BitTorrent. No entanto, devido ao fato do espaço de estados ser muito grande, não pudemos obter soluções analíticas para o modelo. Com esse intuito, propomos um modelo simplificado para a dinâmica de conexões do BT em uma rede heterogênea. As soluções desse modelo para o regime estacionário são passíveis de serem encontradas analiticamente.

Vamos modelar a dinâmica de conexões do BT, isto é, para quais vizinhos cada peer faz upload e de quem ele recebe download ao longo do tempo, considerando apenas um peer em detalhe, ao qual vamos nos referir como "peer modelado". Este

peer será denotado por ψ. Como um peer só interage com seus vizinhos, somente ψ e seus vizinhos precisariam ser considerados para capturar a dinâmica de conexões entre eles. Entretanto, a dinâmica de cada vizinho depende, por sua vez, da vizinhança dele, e assim por diante. Para fazer com que esse modelo seja matematicamente tratável, vamos considerar o comportamento dos vizinhos de ψ de forma agregada, sem modelar sua dinâmica em detalhe. Mais especificamente, vamos assumir que o restante do sistema está em estado estacionário quando o peer ψ entra na rede. Esta prática é comum na área de modelagem (ver [31], por exemplo).

Na grande maioria das tecnologias de acesso à Internet empregadas na atualidade, a capacidade do uplink é menor ou igual a do downlink, isto é, a velocidade do enlace no sentido de saída para à Internet é sempre menor ou igual à velocidade do enlace no sentido contrário. Essa diferença entre o downlink e o uplink é ainda mais notória para determinadas tecnologias, como ADSL e a cabo. Por essa razão, uma das suposições que usaremos daqui para frente é que o downlink nunca é o gargalo no recebimento de dados. Logo, apenas o uplink dos peers será considerado.

Tendo em vista que estamos lidando com populações heterogêneas, vamos particionar os vizinhos de ψ em três classes ou conjuntos:

- Conjunto L: peers com uplink inferior ao peer principal.
- Conjunto E: peers com uplink similar ao peer principal.
- Conjunto G: peers com uplink superior ao peer principal.

A definição de similar no contexto de capacidade de uplink pode ser um tanto quanto subjetiva e, de maneira geral, deve ser escolhida de maneira relativa e em função do intervalo de variação das capacidades dos peers. Neste trabalho, vamos definir como tendo capacidade similar os vizinhos com exatamente a mesma capacidade que o peer modelado. Vamos assumir que a capacidade média do uplink de cada um desses conjuntos é um parâmetro do modelo.

Sejam N_L, N_E e N_G as cardinalidades dos conjuntos L, E e G, respectivamente, tal que $N = N_L + N_E + N_G$ é o número de vizinhos do peer modelado. Cada um desses conjuntos, será subdividido em dois outros conjuntos, U_i e D_i, ($i \in \{L, E, G\}$). O subconjunto U_i contém os vizinhos para quem ψ está fazendo upload atualmente. De maneira similar, D_i contém aqueles vizinhos de quem ψ está fazendo download. Note que $|U_i \cup D_i| \leq N_i$, para todo i. Além disso, note que U_i é incrementado em uma unidade quando ψ decide fazer upload para alguém em i. Por outro lado, D_i é acrescido de uma unidade quando alguém em i passa a fazer upload para ψ. A Figura 3.9 ilustra os conjuntos L, E, G, seus subconjuntos e a intuição à respeito da simplificação descrita a seguir.

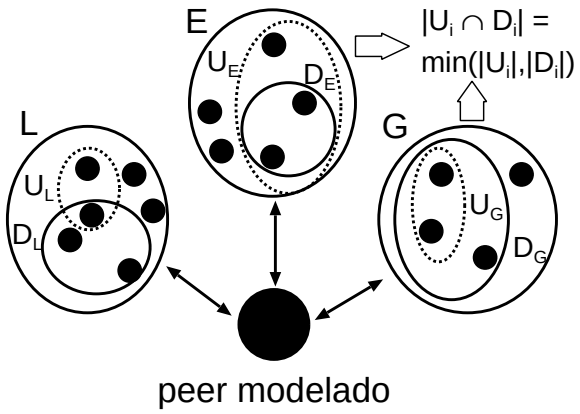


Figura 3.9: Subconjuntos de vizinhos do peer ψ.

Tabela 3.3: Variáveis de estado do modelo analítico.

Variável	Descrição
u_L	número de uploads que ψ faz para peers em L
u_E	número de uploads que ψ faz para peers em E
u_G	número de uploads que ψ faz para peers em G
d_L	número de downloads que ψ faz de peers em L
d_E	número de downloads que ψ faz de peers em E
d_G	número de downloads que ψ faz de peers em G
O	classe para o qual ψ está fazendo *unchoke* otimista

Para simplificar ainda mais o modelo, não iremos representar os conjuntos U_i e D_i, $(i \in \{L, E, G\})$, mas apenas sua cardinalidade, ou seja, $u_i = |U_i|$ e $d_i = |D_i|$. Estas seis variáveis inteiras serão as principais variáveis de estado do nosso modelo (ver Tabela 3.3). Além disso, vamos usar uma variável de estado auxiliar para indicar a classe do peer para quem ψ realizou o último *unchoke* otimista. A combinação dessas variáveis define o espaço de estados do modelo. Portanto, a dinâmica do sistema precisa ser definida em função dessas variáveis de estado. Um aspecto fundamental dessa dinâmica é determinar as transições entre os estados e suas respectivas taxas. Para isso, vamos fazer as seguintes suposições:

- Vamos assumir que quando o peer ψ escolhe fazer upload para um vizinho da classe L ou E por meio de um *unchoke* otimista, o vizinho escolhido irá começar a fazer upload para ψ em um futuro próximo. Entretanto, se o peer escolhido for da classe G, então ele não irá retribuir o upload de ψ. De maneira similar, quando um peer da classe i decide fazer upload para ψ em função de um *unchoke* otimista, o peer modelado pode passar a retribuir em um futuro próximo. Contudo, isso depende da classe daquele peer e dos vizinhos para quem ψ faz upload no instante considerado. Caso aquele peer seja da classe G, por exemplo, então o peer modelado sempre irá retribuir, passando a fazer upload para ele. Estas suposições são realistas, pois conforme foi discutido

antes, em um dado instante, os peers estarão trocando dados com outros de largura de banda similar com alta probabilidade. Por isso, quando um peer passa a receber dados de um vizinho de classe superior, é muito provável que ele deixe de fazer upload para um vizinho antigo e passe a fazer para o novo, que tem maior capacidade.

- Embora nós usemos u_i e d_i como variáveis de estado, a dinâmica do BT depende também do número de nós que estão simultaneamente fazendo download e upload para ψ, que é $|U_i \cap D_i|$. Como aproximação para esse valor, vamos assumir que $|U_i \cap D_i| = \min(u_i, d_i)$, o que significa que o número de downloads e uploads simultâneos para uma dada classe i é dado pelo menor dos dois valores. Esta suposição é realista pois os *unchoke*s regulares podem ser vistos como um mecanismo tit-for-tat, isto é, existe uma tendência a retribuir os downloads (e cessar os uploads que não estão sendo retribuídos). Realizamos estudos preliminares baseados em simulação que corroboram esta suposição.

- O algoritmo de seleção de vizinhos do peer modelado, por outro lado, é considerado em detalhe. Periodicamente o peer ψ passa a retribuir os vizinhos com maior capacidade que lhe deram *unchoke*. Além disso, se o vizinho que recebeu o *unchoke* otimista do peer modelado recebe um unchoke regular, um novo vizinho é imediatamente escolhido pelo *unchoke* otimista, assim como ocorre no BT. Quando isso ocorre, precisamos atualizar corretamente as variáveis de estado. Por isso, precisamos armazenar também como variável de estado a classe do último vizinho de ψ a receber um *unchoke* otimista (ver Tabela 3.3). Mais a frente descrevemos as taxas com que os eventos acontecem neste modelo.

- Embora os eventos no BT sejam periódicos e com um período determinístico em sua grande maioria, assumimos que os intervalos entre cada tipo de evento temporizado (*unchokes* regulares e otimistas) têm distribuição exponencial. Portanto, construímos uma cadeia de Markov para o sistema que pode ser avaliada numericamente no estado estacionário.

3.3.1 Eventos e taxas

Para facilitar o entendimento do modelo, vamos organizar os eventos do sistema segundo as condições que os causaram.

Conjunto L

- ***Unchoke* regular:** um peer χ do conjunto L que não está retribuindo o download que recebe de ψ passa a fazê-lo.

Visto que estamos assumindo que os peers em L tem outros peers do mesmo conjunto como seus *unchokes* regulares, sabemos que χ irá necessariamente substituir um desses peers por ψ, pois o último possui maior capacidade. Para que essa substituição ocorra, no momento em que χ executa o algoritmo de seleção de pares, ψ precisa ter enviado uma quantidade de dados maior ou igual que a quantidade que um peer em L consegue enviar em uma janela de observação w.

Um peer em L transmite a taxa c_L, logo, envia em uma janela de observação de tamanho w um total de wc_L unidades de dados. Como ψ tem capacidade c_E, ele pode transmitir a mesma quantidade de dados em um tempo $\frac{wc_L}{c_E}$. No entanto, os peers executam o algoritmo de seleção de pares em períodos de tamanho T_P. Por isso, o tempo médio até que o evento considerado ocorra deve ser arredondado até a próxima execução do algoritmo, o que pode ser feito aplicando-se as operações de divisão, teto ($\lceil . \rceil$) e multiplicação nessa ordem, conforme mostrado na equação

$$\Delta_{L,r} = \left\lceil \frac{\frac{wc_L}{c_E}}{T_P} \right\rceil \times T_P \tag{3.4}$$

onde w é o tamanho da janela de observação, c_i é a capacidade do uplink de um peer em i e T_P é o intervalo de seleção de pares. Portanto, a taxa com que esse evento ocorre é

$$\lambda_{L,r} = (u_L - d_L) \times \frac{1}{\Delta_{L,r}} \tag{3.5}$$

onde $u_L - d_L$ é a aproximação da quantidade de peers em L que recebem de ψ mas não retribuem.

Atualização de variáveis

$d_L \leftarrow d_L + 1$

- ***Unchoke* otimista:** um peer χ do conjunto L que não está fazendo upload para ψ passa a fazê-lo.

  Vamos assumir que os peers em L trocam o *unchoke* otimista a cada T_R segundos, onde T_R é o intervalo de rotação da lista de vizinhos de um leecher. Assumiremos também que o novo vizinho a receber o *unchoke* otimista é escolhido de maneira aleatória e uniforme entre os que não receberam *unchokes* regulares. Logo, a taxa com que esse evento ocorre é

$$\lambda_{L,o} = (N_L - d_L) \times \frac{1}{T_R} \times \frac{1}{N - x_r} \tag{3.6}$$

onde N_L é o tamanho do conjunto L, $N_L - d_L$ é o número de peers em L que

não fazem upload para ψ, N é o número médio de vizinhos que um peer possui e x_r é o número de *unchokes* regulares que um peer faz.

Atualização de variáveis

$d_L \leftarrow d_L + 1$

- ***Choke* otimista:** um peer χ do conjunto L que faz upload para ψ por meio de *unchoke* otimista e não é correspondido deixa de fazê-lo.

  A taxa com que os peers em L que deram *unchoke* otimista em ψ o substituem, depende da quantidade de peers nessa situação. Vamos assumir que todos os downloads oriundos de vizinhos em L que ψ não retribui são *unchokes* otimistas. Logo, a taxa com que esse evento ocorre é

$$\mu_{L,o} = (d_L - u_L) \times \frac{1}{T_R}. \tag{3.7}$$

  onde $d_L - u_L$ é uma aproximação para o número de vizinhos em L de quem ψ faz download, mas para os quais ψ não faz upload.

  Note que não iremos considerar a ocorrência de *choke* regular, pois: 1) estamos considerando que os downloads não correspondidos são provenientes de *unchokes* otimistas e 2) que aqueles que são correspondidos não terão incentivo para substituir ψ por outro vizinho.

  Atualização de variáveis

  $d_L \leftarrow d_L - 1$

Conjunto E

- ***Unchoke* regular:** um peer χ do conjunto E que não está retribuindo o download que recebe de ψ passa a fazê-lo.

  Após enviar dados durante uma janela de observação completa w, ψ passa a disputar com os vizinhos de χ a chance de ser selecionado entre os melhores vizinhos e receber o *unchoke* regular. Considerando que os *unchokes* regulares de χ são peers do conjunto E e que ele não recebe download de nenhum outro peer além desses e de ψ, a probabilidade de ψ ser escolhido na próxima seleção de pares é 3/4. Nesse caso, o tempo até que o evento considerado ocorra é w. A probabilidade de ψ ser escolhido apenas na próxima execução do algoritmo de seleção de pares é $1/4 \times 3/4$, porque ele não é escolhido na primeira execução (prob. 1/4), mas é escolhido na segunda (prob. 3/4). Nesse último caso, ele é selecionado após $w + T_P$ unidades de tempo. De maneira geral, ψ é selecionado na (i+1)-ésima execução com probabilidade $(1/4)^i \times 3/4$, após $w + iT_P$ unidades de tempo, conforme mostrado na Tabela 3.4.

Tabela 3.4: Distribuição do tempo que ψ leva até receber um *unchoke* regular de um peer em E.

Tempo	Probabilidade
w	$3/4$
$w + T_P$	$1/4 \times 3/4$
$w + 2T_P$	$(1/4)^2 \times 3/4$
$\vdots$	$\vdots$
$w + iT_P$	$(1/4)^i \times 3/4$

O tempo médio até que esse evento ocorra pode ser calculado usando-se a equação $\sum_{i=1}^{\infty} ix^i = \frac{x}{(x-1)^2}$, conforme vemos a seguir.

$$\begin{aligned}
\Delta_{E,r} &= \sum_{i=1}^{\infty}(w + iT_P)\left(\left(\frac{1}{4}\right)^i \times \frac{3}{4}\right) && (3.8)\\
&= w + \frac{3}{4}T_P \sum_{i=1}^{\infty} i\left(\frac{1}{4}\right)^i && (3.9)\\
&= w + \frac{3}{4}T_P \times \frac{4}{9} && (3.10)\\
&= w + \frac{1}{3}T_P && (3.11)
\end{aligned}$$

A taxa com que esse evento ocorre depende do número de peers em E que recebem download de ψ mas não retribuem. Portanto, ela é dada por

$$\lambda_{E,r} = (u_E - d_E) \times \frac{1}{\Delta_{E,r}}. \quad (3.12)$$

Atualização de variáveis

$d_E \leftarrow d_E + 1$

- ***Unchoke* otimista:** um peer χ do conjunto E que não está fazendo upload para ψ passa a fazê-lo.

  Este evento é análogo ao *unchoke* otimista executado pelos peers do conjunto L. Portanto, a taxa com que ele ocorre é dada por

$$\lambda_{E,o} = (N_E - d_E) \times \frac{1}{T_R} \times \frac{1}{N - x_r} \quad (3.13)$$

  onde N_E é o tamanho do conjunto E, $N_E - d_E$ é o número de peers em E que não fazem upload para ψ, N é o número médio de vizinhos que um peer possui e x_r é o número de *unchokes* regulares que um peer faz.

  Atualização de variáveis

  $d_E \leftarrow d_E + 1$

- ***Choke* regular:** um peer χ do conjunto E que faz upload para ψ por meio de *unchoke* regular e não é correspondido deixa de fazê-lo.

  Esse evento corresponde à interrupção do envio de dados a ψ, que havia recebido anteriormente um *unchoke* regular de χ. Como o unchoke regular é concedido com base na política tit-for-tat, sabemos que ψ estava fazendo upload para χ em um momento anterior. Iremos assumir que um peer em E não dá um *choke* regular em ψ a não ser que o último tenha parado de enviar dados ao primeiro (assim como no evento que estamos considerando). Além disso, vamos supor que χ possui pelo menos um vizinho ϕ em E do qual faz download, mas para o qual não faz upload. Assim, na próxima execução do algoritmo de seleção de pares, χ irá substituir ψ por ϕ. Logo, a taxa com que esse evento acontece é

  $$\mu_{E,r} = (d_E - u_E) \times \frac{1}{T_P} \tag{3.14}$$

  onde $d_E - u_E$ é uma aproximação para o número de vizinhos em E de quem ψ faz download, mas para os quais não faz upload.

  Atualização de variáveis

  $d_E \leftarrow d_E - 1$

Note que não iremos considerar a ocorrência de *choke* otimista, pois estamos assumindo que os downloads não correspondidos são *unchokes* regulares. A intuição por trás dessa simplificação é que um *unchoke* otimista oriundo de um peer em E tem alta probabilidade de ser retribuído por ψ e ser substituído por um *unchoke* regular para ψ.

Conjunto G

- ***Unchoke* otimista:** um peer χ do conjunto G que não está fazendo upload para ψ passa a fazê-lo.

  Este evento é análogo ao *unchoke* otimista executado pelos peers do conjunto L. Portanto, a taxa com que ele ocorre é dada por

  $$\lambda_{G,o} = (N_G - d_G) \times \frac{1}{T_R} \times \frac{1}{N - x_r} \tag{3.15}$$

  onde N_G é o tamanho do conjunto G, $N_G - d_G$ é o número de peers em G que não fazem upload para ψ, N é o número médio de vizinhos que um peer possui e x_r é o número de *unchokes* regulares que um peer faz.

  Atualização de variáveis

  $d_G \leftarrow d_G + 1$

- ***Choke* otimista:** um peer χ do conjunto G que faz upload para ψ por meio de *unchoke* otimista e não é correspondido deixa de fazê-lo.

  A taxa com que os peers em G que deram *unchoke* otimista em ψ o substituem depende da quantidade de peers nessa situação. Diferentemente do que fizemos para o conjunto L, vamos considerar que **todos** os downloads que ψ recebe de peers em G são oriundos de *unchokes* otimistas e não apenas os que não são retribuídos. Isto é razoável pois uma vez que um peer em G passa a trocar dados com vizinhos do mesmo conjunto, ele não possui incentivo para retribuir ψ. Logo, a taxa com que esse evento ocorre é

  $$\mu_{G,o} = d_G \times \frac{1}{T_R}. \tag{3.16}$$

  Atualização de variáveis

  $d_G \leftarrow d_G - 1$

Note que não iremos modelar a ocorrência de *unchoke* regular e, por conseguinte, de *choke* regular.

Peer modelado (ψ)

Neste modelo, ψ é considerado de forma detalhada em oposição aos conjuntos que vimos anteriormente. Em particular, iremos modelar a correlação existente entre o *unchoke* regular e o otimista.

Conforme explicado anteriormente, o algoritmo de seleção de vizinhos é executado a cada intervalo $T_P = 10s$. Em algumas das execuções, um novo peer é selecionado para receber o *unchoke* otimista obrigatoriamente. Mais precisamente, esse evento acontece em intervalos periódicos de tamanho $T_R = 30s$. Isto significa, que a cada 3 execuções do algoritmo, em uma delas um novo peer recebe o *unchoke* otimista.

Entretanto, se em uma execução do algoritmo em que um novo *unchoke* otimista não é obrigatório, o peer com o *unchoke* otimista passar a ser um *unchoke* regular, um novo peer é escolhido de maneira aleatória e uniforme para receber o *unchoke* otimista. Note que isso não afeta o temporizador T_R.

Iremos considerar então dois casos principais: 1) seleção de vizinhos com *unchoke* otimista obrigatório e 2) seleção de vizinhos comum. Dentro do último caso, há ainda duas subdivisões: 2.1) *unchoke* otimista passou a ser *unchoke* regular e 2.2) *unchoke* otimista continua sendo *unchoke* otimista.

1. **Seleção de vizinhos com *unchoke* otimista obrigatório**: os vizinhos com capacidade mais alta dentre aqueles de quem ψ faz download são selecionados

como *unchokes* regulares. Dentre os vizinhos não selecionados, um é escolhido de maneira aleatória e uniforme como *unchoke* otimista.

Note que não estamos modelando a quantidade de pedaços que foram trocados entre os peers e nem mesmo o momento em que os *unchokes* acontecem. Por isso, o único referencial que ψ possui para fazer a seleção de vizinhos é a capacidade deles. Portanto, a seleção de vizinhos é feita conforme mostrado no Algoritmo 1.

Algoritmo 1: Seleção de vizinhos com *unchoke* otimista obrigatório, realizada pelo peer no modelo analítico.

```
1  u_G ← 0;
2  u_E ← 0;
3  u_L ← 0;
4  Unchokes ← x_r;
5  para i ∈ {G, E, L} faça
6      se Unchokes > 0 então
7          se Unchokes < d_i então
8              u_i ← Unchokes;
9          senão
10             u_i ← d_i;
11         fim
12         Unchokes ← Unchokes − u_i;
13     fim
14 fim
15 r ← U(0,1) ;                  // variável aleatória uniforme entre 0 e 1
16 se r ≤ (N_L − u_L) / (N − (u_L + u_E + u_G)) então
17     i ← L;
18 senão
19     se r ≤ ((N_L − u_L) + (N_E − u_E)) / (N − (u_L + u_E + u_G)) então
20         i ← E;
21     senão
22         i ← G;
23     fim
24 fim
25 O ← i;
26 u_i ← u_i + 1;
```

Nas linhas 1 a 3, as variáveis de estado que contam o número de uploads para cada classe são zeradas. A variável *Unchokes* é inicializada com o número máximo de *unchokes* regulares x_r e indica qual o número de *unchokes* restantes. No laço que vai da linha 5 à 14, os conjuntos de vizinhos são percorridos em ordem, daquele de maior capacidade para o de menor, a fim de encontrar os peers que mais contribuem com ψ e selecioná-los como *unchokes* regulares. Depois disso, nas linhas 15 a 22, um vizinho dentre os que não receberam

unchoke é escolhido de maneira aleatória e uniforme como *unchoke* otimista. Na linha 23 a classe do *unchoke* otimista é armazenada e na linha 24 a variável de estado correspondente é atualizada.

Neste modelo, o intervalo T_R é múltiplo do intervalo de seleção de vizinhos T_P. Assim, a taxa com que esse evento ocorre é simplesmente

$$\lambda_{\psi,1} = \frac{1}{T_R} \tag{3.17}$$

2. **Seleção de vizinhos comum**: os vizinhos com capacidade mais alta dentre aqueles de quem ψ faz download são selecionados como *unchokes* regulares. Se o *unchoke* otimista passou a ser um *unchoke* regular, outro peer precisa ser escolhido como *unchoke* otimista (item 2(a)). Caso contrário, ele continuará como *unchoke* otimista por mais tempo (item 2(b)).

   (a) **Seleção de vizinhos comum, trocando *unchoke* otimista.**

   Conforme dito anteriormente, os peers em G não irão retribuir ψ. Portanto, só iremos verificar se um vizinho que era *unchoke* otimista passou a retribuir ψ e foi selecionado para receber o *unchoke* regular caso ele pertença aos conjuntos L ou E ($O \in \{L, E\}$).

   Para que um vizinho χ seja selecionado para receber o *unchoke* regular, o número de vizinhos de maior capacidade que χ do qual ψ faz download precisa ser menor que x_r. Isto significa que se $O = E$, então temos que ter $d_G < x_r$. Por outro lado, se $O = L$, a condição necessária é

   Condição 1 $d_G + d_E < x_r$.

   Uma segunda condição necessária é que o *unchoke* otimista esteja retribuindo ψ. Vamos assumir que se o número de uploads que ψ faz para classe O é menor que o número de downloads que ele faz dessa classe (isto é, $u_O > d_O$), um dos vizinhos que não retribui ψ é o *unchoke* otimista. Logo, a condição necessária é que

   Condição 2 $u_O \leq d_O$.

   Finalmente, se o número de *unchokes* restantes (indicado pela variável $Unchokes$ no Algoritmo 1) para a classe O for menor que o número de peers em O do qual ψ faz download, o *unchoke* otimista pode ou não ser trocado. A probabilidade dele ser trocado é dada por

$$p_t = \begin{cases} \dfrac{x_r - d_G}{d_E}, & \text{para } O = E \quad (3.18) \\ \dfrac{x_r - (d_G + d_E)}{d_L}, & \text{para } O = L. \quad (3.19) \end{cases}$$

Note que o numerador de p_t indica o número de *unchokes* restantes e o denominador, o número de peers na classe O que podem ser escolhidos.

O novo *unchoke* otimista é escolhido de maneira aleatória e uniforme entre os vizinhos que não receberam *unchoke* regular.

A taxa com que esse evento acontece, dado que as duas condições acima são satisfeitas é

$$\lambda_{\psi,2(a)} = \left(\frac{1}{T_P} - \frac{1}{T_R}\right) \times p_t \tag{3.20}$$

Quando esse evento acontece, a atualização de variáveis é feita exatamente como no Algoritmo 1. A seguir, veremos o que acontece quando o *unchoke* otimista é mantido.

(b) **Seleção de vizinhos comum, mantendo *unchoke* otimista.**

Mesmo quando as Condições 1 e 2 acima são satisfeitas, existe uma chance de o *unchoke* otimista não ser escolhido para receber o *unchoke* regular. Mais precisamente, essa probabilidade é

$$p_m = \begin{cases} 1 - \dfrac{x_r - d_G}{d_E}, & \text{para } O = E \qquad (3.21) \\ 1 - \dfrac{x_r - (d_G + d_E)}{d_L}, & \text{para } O = L. \qquad (3.22) \end{cases}$$

Logo, a taxa com que esse evento ocorre dado que as duas condições são satisfeitas é dada por

$$\lambda_{\psi,2(b)} = \left(\frac{1}{T_P} - \frac{1}{T_R}\right) \times p_m \tag{3.23}$$

Por outro lado, se uma das condições não for satisfeita, o *unchoke* otimista é mantido sempre que ocorrer a seleção de vizinhos comum. Nesse caso, a taxa com que o evento ocorre é

$$\lambda_{\psi,2(b)} = \frac{1}{T_P} - \frac{1}{T_R} \tag{3.24}$$

Sejam as condições satisfeitas ou não, a seleção de vizinhos ocorre exatamente igual ao Algoritmo 1 até a linha 14. Depois disso, em vez de selecionar um novo *unchoke* otimista, a variável O é consultada para atualizar u_O corretamente, ou seja, $u_O \leftarrow u_O + 1$.

3.3.2 Experimentos e Resultados

Iremos avaliar o modelo analítico apresentado quanto à caracterização do estado estacionário da dinâmica de conexões entre os peers de diferentes classes. Para a

validação desses resultados, iremos utilizar um modelo de simulação do BT bastante detalhado, não apresentado até o momento, que é descrito a seguir.

Modelo detalhado de simulação do BT

A primeira versão do modelo de simulação detalhado do BT foi desenvolvida por outro aluno do Laboratório LAND (ver [32]) para o ambiente de modelagem e experimentação Tangram-II [33, 34]. Este simulador implementa fielmente o protocolo do cliente BitTorrent 4.0.0, com todos seus mecanismos e mensagens de controle. As entidades que participam do swarm foram implementadas como objetos separados que se comunicam através de mensagens no ambiente de modelagem do Tangram-II. Isso facilita a depuração e a implementação de possíveis variantes do protocolo BT.

Para estudar o AM, o modelo de simulação foi reescrito (uma parte do trabalho atual) para que fosse possível instanciar peers com diferentes larguras de banda. Diferentemente da versão anterior, novos peers podem chegar no sistema após o início da simulação, ou seja, não é necessário instanciar todos os peers logo no começo. Além disso, foram implementadas outras funcionalidades não utilizadas neste trabalho, como a possibilidade de um peer permanecer no sistema por um tempo após o final do download. Nessa nova versão, existe um objeto responsável pelo processo de chegada de novos peers, capaz de instanciá-los passando através de mensagens parâmetros de funcionamento como os descritos acima.

O cenário que iremos considerar é conhecido como *flash crowd*, isto é, uma grande quantidade de leechers entra no sistema praticamente ao mesmo tempo e, depois disso, não ocorrem mais chegadas. Iremos assumir que os leechers deixam o sistema logo após concluírem o download (e não saem por nenhum outro motivo) e que um único seed permanece conectado ao swarm o tempo todo.

Vamos simular $N = 50$ leechers que chegam ao sistema segundo um Processo de Poisson, com taxa $\lambda = 1$ leecher/s. Por simplicidade, iremos considerar que cada peer conhece todos os outros peers do sistema. Outros parâmetros utilizados na simulação foram: capacidade de upload do seed $c_S = 1000$ KBps, capacidade de upload de das classes de leecher iguais a $c_L = 16$ KBps, $c_E = 32$ KBps e $c_G = 64$ KBps, e tamanho do arquivo igual a 600 MB.

No modelo analítico, os únicos parâmetros são os números de vizinhos de ψ em cada classe (N_L, N_E e N_G, onde $N_L + N_E + N_G = 49$). As taxas com que os eventos ocorrem são funções das capacidades dos peers, dos parâmetros do BitTorrent e das variáveis de estado. Quanto à distribuição dos vizinhos, iremos considerar quatro casos:

- vizinhos quase igualmente distribuídos ($N_L = 16$, $N_E = 17$, $N_G = 16$);
- maior número de vizinhos em L ($N_L = 25$, $N_E = 12$, $N_G = 12$);
- maior número de vizinhos em E ($N_L = 12$, $N_E = 25$, $N_G = 12$);

- maior número de vizinhos em G ($N_L = 12$, $N_E = 12$, $N_G = 25$).

As métricas que utilizamos nesse estudo são calculadas em estado estacionário. Mais precisamente, estamos interessados no número médio de downloads e número médio de uploads que um peer de capacidade intermediária faz para cada classe. Note que a métrica que utilizamos para avaliar o modelo de simulação anterior (*assortative coefficient*), é uma métrica global e por isso não pôde ser utilizada para avaliação desse modelo analítico, que provê estimativas individuais de um peer.

Para obtermos uma estimativa de quando as simulações atingiam o estado estacionário, analisamos o intervalo entre a chegada do último peer e a primeira saída do sistema (que corresponde ao momento em que o primeiro leecher termina o download). Calculamos então o valor médio instantâneo de cada métrica, considerando apenas os peers com capacidade c_E. Por inspeção visual, concluímos que nos últimos 1000 segundos do período analisado (que tinha aproximadamente 7000s), o sistema se encontrava em estado estacionário.

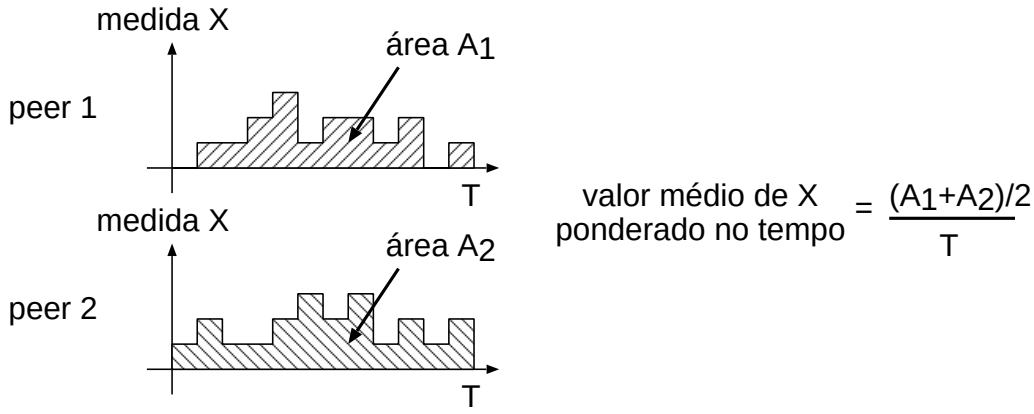


Figura 3.10: Figura que ilustra como calculamos o valor médio de X ponderado no tempo.

No entanto, não iremos utilizar diretamente o valor médio instantâneo das métricas calculado a partir das simulações. Iremos considerar o valor médio ponderado pelo tempo nos últimos 1000 segundos. A Figura 3.10 ilustra como esse valor é calculado a partir da simulação de uma medida de interesse X sobre o intervalo $[0, T]$, tomando-se 2 peers. Para N peers, o valor médio ponderado no tempo é

$$X_S = \frac{1}{NT} \sum_{i=1}^{N} A_i \tag{3.25}$$

onde A_i é a área do gráfico correspondente ao peer i.

O modelo analítico permite um particionamento do espaço de estados que dá origem a uma estrutura QBD (Quasi Birth-Death). A matriz de transição de estados é dita QBD quando existe um particionamento ordenado do espaço de estados

$\{\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_N}\}$, tal que as transições ocorrem apenas para estados na partição imediatamente anterior, na mesma partição, ou na partição imediatamente posterior. Esse tipo de estrutura permite métodos de solução eficientes, como o block-GTH [35].

A matriz a seguir ilustra a estrutura QBD, onde $Q_{i,j}$ representa o bloco que contém as transições de estados da partição $\mathbf{P_i}$ para estados da partição $\mathbf{P_j}$:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q_{1,1}} & \mathbf{Q_{1,2}} & 0 & 0 & \ldots & 0 & 0 \\ \mathbf{Q_{2,1}} & \mathbf{Q_{2,2}} & \mathbf{Q_{2,3}} & 0 & \ldots & 0 & 0 \\ 0 & \mathbf{Q_{3,2}} & \mathbf{Q_{3,3}} & \mathbf{Q_{3,4}} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & \mathbf{Q_{N,N-1}} & \mathbf{Q_{N,N}} \end{pmatrix}$$

Para calcular o valor das métricas a partir do modelo analítico, precisamos resolvê-lo, isto é, encontrar a solução no estado estacionário. Seja X_i o valor da medida X quando o sistema se encontra no estado $i \in S$, onde S é o espaço de estados do modelo. Seja também $\pi = (\pi_1, \pi_2, \ldots, \pi_S)$ a distribuição de estado estacionário do modelo. Logo, o valor da métrica em questão obtido analítico do modelo é dado por

$$X_A = \sum_{i=1}^{S} \pi_i \times X_i. \tag{3.26}$$

Cada um dos quatro cenários será simulado 10 vezes. A partir desses resultados, vamos obter a média e o intervalo de confiança de X_S para cada medida de interesse X, e por fim comparar aos valores de X_A.

Todos os resultados do modelo analítico foram obtidos em menos de um minuto de tempo de CPU, apesar do espaço de estados ser composto por aproximadamente 81 mil estados. Entretanto, apesar de poderem ser executadas em paralelo, as simulações requerem dezenas de horas de tempo de CPU.

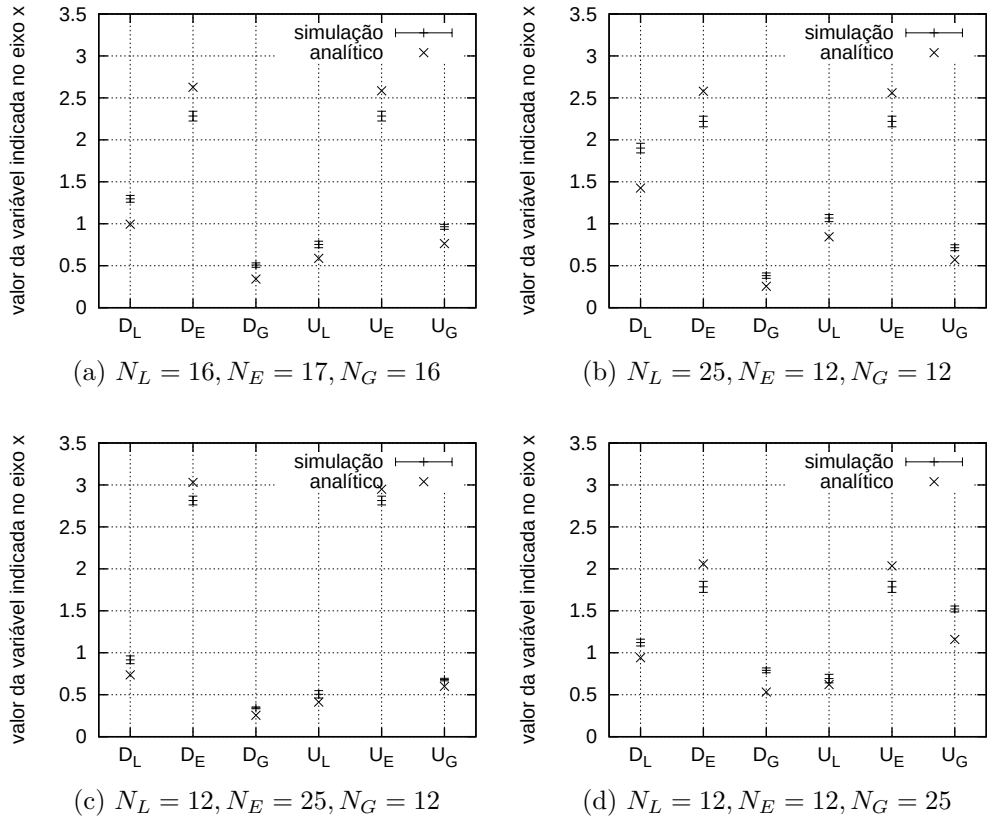
Resultados

A Figura 3.11 mostra a comparação dos resultados obtidos através da simulação e do modelo analítico nos quatro cenários. Cada cenário representa um número diferente de vizinhos em cada conjunto L, E e G. O eixo x indica a métrica sendo avaliada e o eixo y, o valor da métrica para os dois casos (simulação e analítico). No caso das simulações, mostramos também os intervalos de confiança. Como exemplo, na Figura 3.11(a), o peer modelado tem 16 vizinhos em L, 17 vizinhos em E e 16 vizinhos em G. Na simulação, a média de u_E obtida foi 2.28, enquanto no modelo analítico foi 2.59.

As Figuras 3.11(b-d) representam populações menos uniformes, respectivamente com muitos peers do tipo L, E e G. Em todos os quatro cenários, o número médio

de conexões (downloads e uploads) para peers da mesma classe foi significativamente maior do que o número médio de conexões para peers de outras classes. Além disso, observamos que $d_L > u_L$, $d_E \approx u_E$ e $d_G < u_G$, independentemente da distribuição da população. Isso se deve ao fato de que, por exemplo, os peers em L estão mais propensos a retribuir os peers de capacidade intermediária. Analogamente, os peers em G estão menos propensos a retribuir peers em E. O efeito da distribuição da população sobre o estado estacionário da dinâmica das conexões aparece nos diferentes valores obtidos em cada gráfico: o aumento na proporção da população de uma determinada classe, leva a um aumento no número médio de downloads e uploads para aquela classe. Todas essas tendências são capturadas corretamente pelo modelo analítico.

Concluímos que o modelo analítico consegue caracterizar bem o estado estacionário da dinâmica de conexões entre os peers. Os erros relativos são todos inferiores à 35%, sendo mais da metade inferior à 20%. Mais do que isso, o modelo consegue capturar bem a relação entre u_i e d_i para $i \in \{L, E, G\}$ em todos os cenários e a mudança na vizinhança do peer modelado é refletida nos resultados de maneira correta.



(a) $N_L = 16, N_E = 17, N_G = 16$ (b) $N_L = 25, N_E = 12, N_G = 12$

(c) $N_L = 12, N_E = 25, N_G = 12$ (d) $N_L = 12, N_E = 12, N_G = 25$

Figura 3.11: Comparação entre o modelo proposto e a simulação detalhada.

Em parte, as diferenças entre o modelo analítico e de simulação estão relacio-

nados ao fato de assumirmos que os peers em G não irão retribuir ψ. Isso faz com que os valores de d_G e u_G sejam subestimados. Além disso, fazendo menos uploads para peers em G, o modelo direciona mais *unchokes* para os vizinhos em E, superestimando u_E e, por consequência, d_E. Este comportamento pode ser observado nos resultados da Figura 3.11d. Nesse caso, como o número de peers do tipo G é grande, é mais provável que esses peers venham a retribuir peers do tipo E, fazendo com que a diferença relativa entre os resultados do modelo analítico e da simulação seja mais significativa.

3.4 Conclusões

Na primeira parte deste capítulo, propusemos um modelo simplificado da dinâmica de troca de conexões entre os peers no protocolo BitTorrent capaz de capturar o fenômeno *assortative mixing*. O modelo, assim como o protocolo, não incorpora a preferência dos vértices por outros do mesmo tipo de forma explícita e leva em conta que um vértice só pode manter conexões com um subconjunto dos vértices do grafo, assim como acontece no BitTorrent e em outras redes reais.

Utilizamos o *assortative coefficient* para quantificar o AM ao longo do tempo. Analisamos as curvas para a variação individual de alguns parâmetros do modelo. Observamos que o valor máximo desse coeficiente depende apenas do número de vértices que cada nó conhece (k), do número de vértices a quem ele oferece upload (x) e do número de tags diferentes na rede (v).

Pudemos também observar o papel fundamental que o *unchoke* otimista tem na ocorrência desse fenômeno, conhecido também como *Assortative Mixing* (AM). O *unchoke* otimista é responsável por promover o encontro entre os peers, que pouco sabem uns sobre os outros. Sem ele, a dinâmica das conexões entre os peers ficaria travada assim que todos os uploads estivessem sendo retribuídos. Isso poderia levar o sistema a estados de operação muito ruins, tanto do ponto de vista global quanto do individual.

A princípio, o *unchoke* otimista, quando combinado aos *unchokes* regulares, poderia fazer com que os peers fossem encontrando os vizinhos de maior capacidade dentro do sistema e passassem a trocar dados apenas com eles. De fato, isso pode ocorrer inicialmente, ou seja, usuários com as mais variadas capacidades podem trocar dados com aqueles de alta capacidade. Entretanto, estes últimos também estão realizando o *unchoke* otimista, que lhes permitirá encontrar outros peers com capacidade similar a deles. Assim, é inevitável que eles deixem de servir os vizinhos com menor capacidade. Por fim, não resta outra opção aos peers de capacidade inferior senão trocar dados entre si.

Dentre outras, vimos também que a taxa com que o *unchoke* otimista é executado

pode afetar o grau e a velocidade de clusterização por banda do sistema. Mais precisamente, existe um compromisso entre o grau e o tempo de convergência da clusterização: executar o *unchoke* otimista em intervalos menores induz o sistema a uma rápida clusterização, mas que não atinge valores altos de AM. Por outro lado, uma menor taxa de *unchoke* otimista faz com que o grau de AM alcançado seja maior, mas leva mais tempo para convergir.

Conforme observado em trabalhos anteriores [17], o *unchoke* otimista é essencial para o bom desempenho do BitTorrent. A principal contribuição deste modelo é mostrar que, além disso, o *unchoke* otimista é também fundamental para a formação de clusters por largura de banda, o que por sua vez contribui para a eficiente disseminação de dados nesse sistema.

Na segunda parte do capítulo, apresentamos um modelo analítico Markoviano simplificado para o estudo do estado estacionário da dinâmica de conexões em redes BitTorrent heterogêneas. Mostramos através de simulações detalhadas do protocolo BT que o modelo captura corretamente as tendências dessa rede para diferentes distribuições das capacidades dos vizinhos.

Dentre as tendências comuns observadas, vimos que, apesar de sempre retribuir as conexões oriundas de vizinhos de maior capacidade, o peer modelado termina por estabelecer um número maior de conexões com vizinhos de capacidade similar. A intuição por trás disso está no fato de que os vizinhos de maior capacidade param de fazer upload para o peer modelado depois de um tempo. Ele, por sua vez, irá escolher outros vizinhos para retribuir dentre os de capacidade similar e os de capacidade inferior. Sendo os vizinhos similares a melhor opção dentre as disponíveis, há então clusterização.

O modelo é inerentemente dinâmico e poderia ser usado como base para estudar o regime transiente da rede de conexões. Esse regime pode desempenhar um papel importante no sistema, podendo se estender por longos períodos de tempo sobre os quais o desempenho do sistema é afetado. De fato, este tipo de comportamento pode ser observado nos resultados do modelo detalhado apresentado na Seção 3.2. Por exemplo, quando a probabilidade de realizar um *unchoke* otimista é pequena, o transiente é longo (ver Figura 3.7).

Todos os modelos propostos neste trabalho são baseados no cliente BitTorrent 4.0.0. Outras implementações podem empregar variações do algoritmo de seleção de pares aqui apresentado, mas que, em essência, funcionam da mesma forma. Assim, de maneira geral, as tendências observadas nos nossos resultados também são válidas para outras implementações.

Capítulo 4

Tempos de download heterogêneos em swarms homogêneos do BitTorrent

No capítulo anterior vimos que o BT possui uma tendência que leva o sistema à uma alta clusterização de peers com largura de banda semelhante, apesar do protocolo não implementar nenhum mecanismo explícito de agrupamento de usuários com largura de banda semelhante. Essa clusterização faz com que peers com diferentes capacidades tenham desempenho bastante diferente. Entretanto, veremos que os peers podem ter desempenho muito desigual mesmo quando possuem capacidades semelhantes.

Neste capítulo, identificamos uma característica que não havia sido observada anteriormente: peers homogêneos (com relação às suas capacidades de upload) observam taxas de downloads heterogêneas. As consequências desse fenômeno, tais como uma alta variabilidade dos tempos de download, injustiça com respeito à ordem de chegada, saídas em rajada e sincronização de conteúdo, afetam os peers e o desempenho do sistema como um todo. Inicialmente, utilizamos simulações e experimentos na Internet baseados em um protótipo do BT para estudar essa característica. Além disso, desenvolvemos um modelo matemático que prevê com acurácia as taxas de download heterogêneas dos peers homogêneos em função do seu conteúdo. Este fenômeno é mais prevalecente em swarms pouco populares (com muito poucos peers). Embora este tipo de swarm não tenha sido objeto de estudo de muitas pesquisas até hoje, ele representa o tipo mais comum em redes P2P para compartilhamento de arquivos.

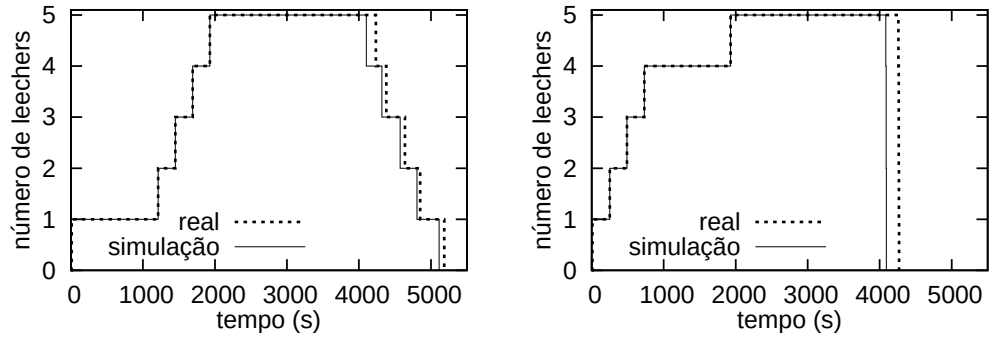
4.1 Introdução

É importante entender e caracterizar o desempenho do BT através de modelos matemáticos[36]. Diversos estudos mostraram aspectos peculiares da dinâmica do BT, muitos dos quais têm impacto direto no desempenho do sistema. Em particular, modelos que capturam o desempenho do usuário e do sistema para populações homogêneas e heterogêneas (com relação à capacidade de upload) foram propostos para vários cenários, incluindo aqueles em [15],[13],[21], [22] e até mesmo o modelo proposto na Seção 3.3. Entretanto, a maioria dos modelos propostos foca em sistemas de larga escala, considerando ou uma grande população inicial fixa ou altas taxas de chegada. Uma exceção é o modelo de [23] que trata de swarms pequenos.

Considere um swarm do BT onde todos os peers têm capacidade de upload idênticas, mas têm capacidade de download ilimitada (ou muito grande). Nesse contexto, nós identificamos uma característica que não havia sido observada: peers homogêneos recebem dados a taxas de download heterogêneas. A princípio esse comportamento não é intuitivo porque os peers são idênticos e deveriam exibir desempenho médio similar. Esta observação não foi capturada por nenhum modelo anterior (do nosso conhecimento). Uma das implicações importantes deste fenômeno é a sincronização de conteúdo entre os peers. Dois peers são ditos "sincronizados" depois que o conteúdo deles se torna idêntico em algum instante de tempo. Esta consequência é particularmente crítica visto que está fortemente relacionada à síndrome do pedaço faltante [24]. Esta síndrome se caracteriza pela formação de um conjunto numeroso de peers que possuem todos exceto um único mesmo pedaço. Quando vários peers estão sincronizados, estão interessados no mesmo subconjunto de pedaços, o que é uma generalização do caso em que este subconjunto é unitário.

Para caracterizar o fato de que peers homogêneos recebem download a taxas heterogêneas e suas consequências, usamos simulações detalhadas à nível de pacote e experimentos na Internet baseados em protótipo. Para descobrirmos os parâmetros críticos para a ocorrência desta característica, consideramos vários cenários. Além disso, desenvolvemos um modelo matemático que captura o fenômeno e prevê as taxas de download dos peers homogêneos em função do seu conteúdo. A comparação das previsões do modelo com os resultados de simulação indica que o modelo é bem acurado. O modelo também permite entender os motivos básicos que levam a esse comportamento: a alocação da capacidade de upload dos peers no BT depende fundamentalmente da relação de interesse nos pedaços detidos por cada um, que pode ser bastante assimétrica para swarms pouco populares.

Esse problema prevalece em swarms que têm uma população muito pequena de peers e um único seed (peer com o conteúdo completo) com largura de banda limitada. Contudo, este é o tipo de swarm mais comum no BT [37]. Realizamos



(a) Intervalos entre saídas: 10s, 10min, 4min, 4min, 4min.

(b) Intervalos entre chegada: 10s, 4min, 4min, 4min, 10min.

Figura 4.1: Evolução do número de leechers no swarm.

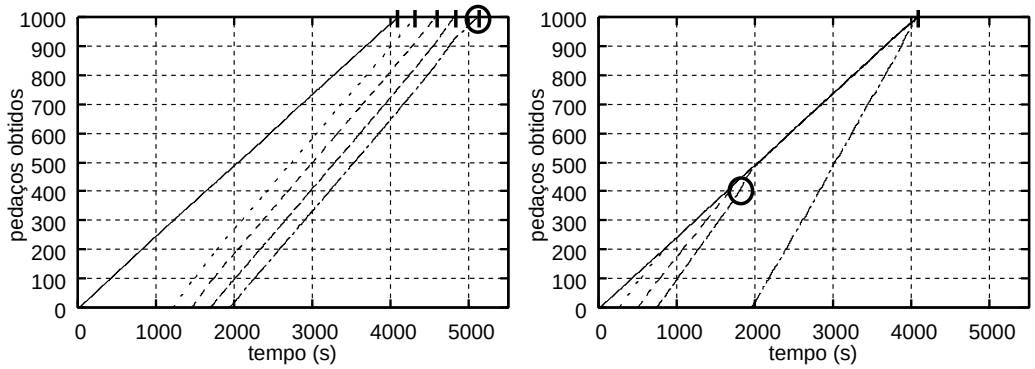
algumas medições que indicam que mais de 35% dos swarms têm menos de 5 peers em qualquer instante de tempo. Portanto, focamos nossa atenção em swarms pouco populares.

O restante do capítulo está organizado da seguinte forma. Na Seção 4.2 identificamos o problema. Em seguida, nas Seções 4.3 e 4.4 caracterizamos o problema e suas consequências usando simulações e experimentos com um cliente do BT, respectivamente. Na Seção 4.5 apresentamos nosso modelo matemático e sua validação. Uma possível aplicação para o modelo, prever saídas em rajada, é abordada na Seção 4.6. Finalmente, elucidamos alguns tópicos em aberto e apresentamos a conclusão deste capítulo na Seção 4.8.

4.2 Download heterogêneo em swarms homogêneos

Primeiramente ilustraremos o problema das taxas de downloads heterogêneas e suas consequências com dois exemplos simples. Considere um swarm formado por um seed e cinco leechers. Todos os peers, incluindo o seed, têm capacidades de upload idênticas (64 KBps), porém as capacidades de download são ilimitadas (ou muito grandes em relação às capacidades de upload). Os leechers fazem download de um arquivo dividido em 1000 pedaços (256 MB) e saem do swarm imediatamente após completar o download. O seed, por outro lado, nunca deixa o swarm. Esse sistema foi avaliado usando o simulador detalhado do BT descrito na Seção 3.3.2 e também uma versão instrumentada de um cliente BT executando no PlanetLab [38].

As Figuras 4.1a e 4.1b mostram a evolução do tamanho do swarm em função do tempo, tanto para os resultados de simulação quanto para os de experimentação. Na Figura 4.1a, os peers deixam o swarm na mesma ordem em que eles chegaram



(a) Corresponde a Figura 4.1a. (b) Corresponde a Figura 4.1b.

Figura 4.2: Evolução do número de pedaços baixados.

(respeitando a propriedade FIFO) e têm um tempo de download relativamente similar. Portanto, o tempo de download é relativamente indiferente à ordem de chegada (com exceção do primeiro peer).

A Figura 4.1b mostra a mesma métrica, sendo os tempos de chegada a única diferença (na verdade, a maioria dos intervalos entre chegadas de peers são preservados). É interessante observar que essa pequena diferença no padrão de chegadas faz com que a dinâmica do sistema exiba um comportamento bem diferente do anterior: apesar de entrarem no swarm em instantes relativamente distantes uns dos outros, todos os cinco leechers completaram seus respectivos downloads aproximadamente no mesmo instante. Mais precisamente, o tempo entre partidas é bem pequeno se comparado ao tempo entre chegadas, o que caracteriza aquilo que chamaremos de "partidas em rajada". Isto significa que os peers que entram mais tarde no swarm fazem download em menor tempo. De fato, o quinto peer completou o download em aproximadamente metade do tempo levado pelo primeiro leecher. Portanto, o sistema é bastante injusto no que diz respeito à ordem de chegada dos leechers, sendo os peers que chegam mais tarde significativamente favorecidos. O que está acontecendo? Por que a dinâmica do BT deu origem a esse comportamento? Iremos responder essas questões nas próximas seções.

4.3 Heterogeneidade em swarms homogêneos do BT

Para entender o comportamento apresentado pelo BT na Figura 4.1b, analisamos o número total de pedaços baixados por cada leecher ao longo do tempo, de um total de 1000 pedaços. Considere as Figuras 4.2a e 4.2b, onde cada curva indica o número total de pedaços baixados por um dado peer, correspondentes às Figuras

4.1a e 4.1b, respectivamente. Note que a inclinação de cada curva corresponde à taxa de download do leecher respectivo.

Inicialmente considere a Figura 4.2a. Apesar da inclinação do primeiro leecher ser menor que à dos outros peers, as curvas nunca se encontram. Em particular, cada leecher termina o download (e deixa o swarm) antes que o próximo leecher o alcance em número de blocos. Note também que as curvas de todos os outros leechers têm inclinações bem similares. Além disso, observamos um comportamento peculiar: a inclinação do quinto leecher decresce quando ele se torna o único leecher no sistema (o que ocorre no instante 4800).

Os resultados mostrados na Figura 4.2b que correspondem ao cenário considerado na Figura 4.1b mostram um comportamento bem diferente. Muitas observações interessantes podem ser obtidas dessa figura. A inclinação do primeiro peer é praticamente constante, permanecendo imutável com a chegada dos outros peers. A inclinação da curva dos outros peers é maior que aquela do primeiro, indicando que as curvas podem acabar se encontrando. Quando duas curvas se encontram, os leechers correspondentes têm o mesmo número de blocos e possivelmente o mesmo conteúdo (iremos explicar esse fato na seção seguinte). A figura também mostra que um peer mais novo não ultrapassa o primeiro peer, mas sim mantém o mesmo número de pedaços baixados após o ponto de junção, possivelmente com seus conteúdos sincronizados. Finalmente, a inclinação do segundo, terceiro e quarto peers são aproximadamente idênticas. Entretanto, a inclinação do quinto peer é ligeiramente maior que a dos outros, indicando uma maior taxa de download e consequentemente, menor tempo de download.

Em resumo, podemos fazer as seguintes observações:

- O primeiro leecher obtém o conteúdo a taxa aproximadamente constante.
- Os leechers subsequentes tem taxas de download superiores à do primeiro.
- Uma vez que um leecher atinge o número total de pedaços obtidos pelo primeiro leecher, as suas taxas de download se tornam idênticas.
- Uma vez que um leecher atinge o número de pedaços obtidos pelo primeiro leecher, as taxas de download dos outros leechers aumentam (ver círculo na Figura 4.2b).
- Considerando apenas o download dos 100 primeiros pedaços na Figura 4.2b, observamos que as quatro primeiras curvas possuem as mesmas características que as quatro primeiras curvas da Figura 4.2a quando observada por completo. Analogamente, podemos dizer que se o número de pedaços considerado fosse maior na Figura 4.2a, ela esta seria muito semelhante à Figura 4.2b. Portanto,

a ocorrência da característica discutida depende fortemente dos parâmetros do cenário analisado.

Todas essas observações estão relacionadas à dinâmica do BT e serão discutidas e explicadas na Seção 4.5 usando um modelo matemático simples. No restante desta seção, vamos discutir as consequências do fato observado e ilustrar que ele acontece mesmo quando as chegadas são aleatórias (mais precisamente, quando o processo de chegada é Poissoniano).

4.3.1 Consequências da heterogeneidade em swarms homogêneos

As observações acima implicam essencialmente que o tempo de download dos peers são muito diferentes, apesar da sua capacidade de download ser homogênea. Resumidamente, as consequências são:

- **Variabilidade nos tempos de download.** Dado que os peers podem ter taxas de download consistentemente diferentes, os tempos de download também podem ser diferentes.

- **Injustiça com relação à ordem de chegada dos peers.** Dado que as taxas de download e, por consequência, os tempos de download podem depender da ordem de chegada, o sistema é inerentemente injusto, potencialmente beneficiando aqueles que chegam mais tarde no swarm. Veremos também que, dependendo da capacidade do seed, os leechers que chegam mais cedo podem ser aqueles que são beneficiados.

- **Sincronização de conteúdo.** Devido às diferentes taxas de download e aos mecanismos de seleção de pedaços (principalmente o *rarest first*), os leechers podem sincronizar com relação ao número de pedaços obtidos e, em determinados casos, no próprio conteúdo. Isto significa que os peers podem acabar tendo exatamente o mesmo conteúdo em algum instante de tempo, mesmo chegando em instante de tempo diferentes.

- **Partidas em rajada.** Uma consequência direta da sincronização de conteúdo são as partidas em rajada. Vamos definir como partidas em rajada as saídas que ocorrem dentro de um intervalo de tempo pequeno quando comparado ao intervalo médio entre chegadas.

Embora as figuras não mostrem explicitamente a sincronização de conteúdo, podemos concluir que ela ocorre a partir das observações a seguir. Na Figura 4.2b, utilizamos um seed com capacidade $c_s = 0.25$ pedaços/s. Note que o primeiro

leecher a entrar baixa o arquivo completo em aproximadamente 4000 segundos, logo sua taxa média de download é $d_1 \approx 1000/4000 = 0.25$ pedaços/s. Mais do que isso, a sua taxa de download é praticamente constante e pode ser aproximada por c_s. Portanto, todos os pedaços que o seed insere no swarm são rapidamente repassados a este leecher. Assim sendo, os outros leechers possuem subconjuntos do conjunto de pedaços detidos pelo primeiro. Quando o número de pedaços de um leecher atinge o número de pedaços do primeiro, eles possuem exatamente o mesmo conteúdo.

Claramente o surgimento dessa característica e suas consequências dependem diretamente dos parâmetros do swarm. Em particular, o instante de chegada dos peers é certamente o mais determinante. Entretanto, parâmetros tais como a capacidade de upload do seed e dos leechers e o número de pedaços do conteúdo também são de fundamental importância. Intuitivamente, um arquivo com um grande número de pedaços ou um seed com uma baixa capacidade de upload aumentam a probabilidade de que as consequências acima ocorram. De fato, para qualquer ordem de chegada de um pequeno conjunto de peers, é sempre possível encontrar os parâmetros do sistema para os quais este comportamento e suas consequências ocorram.

4.3.2 Heterogeneidade sob chegadas Poisson

O comportamento descrito acima não requer chegadas determinísticas ou qualquer padrão de chegadas fixo. Ele surge mesmo quando os padrões de chegada são aleatórios e bem comportados. Nesta seção, estudamos as consequências das taxas de download heterogêneas quando as chegadas são Poisson.

Realizamos um grande número de avaliações usando simulações detalhadas do BT a nível de pacote. Em particular, consideramos um swarm do BT onde um único seed está presente durante todo o tempo, enquanto os leechers chegam segundo um processo de Poisson e deixam o swarm tão logo completam o download. Na avaliação a seguir, todos os leechers têm capacidade de upload igual a 64 KBps (e grande capacidade de download) e baixam um arquivo dividido em 1000 pedaços de 256 kB. A capacidade de upload do seed (c_s) varia entre 48 KBps, 64 KBps e 96 KBps, e a taxa de chegada de leechers λ é 1/1000 segundos. Estes cenários dão origem a um swarm que tem em média 3.7, 3.4 e 3.0 leechers respectivamente.

Primeiramente vamos estudar a variabilidade dos tempos de download e a injustiça com relação à ordem de chegada. A Figura 4.3 ilustra o tempo médio de download dos leechers em função da ordem de chegada no *busy period*[1]. Então a i-ésima chegada em um *busy period* é mapeada no índice i. As diferentes curvas correspondem à capacidades de upload diferentes do seed. Os resultados indicam claramente que o tempo de download depende da ordem de chegada dos leechers.

[1]Um *busy period* é definido como um intervalo de tempo que se inicia quando um leecher entra no swarm vazio e que termina assim que o swarm fica vazio novamente.

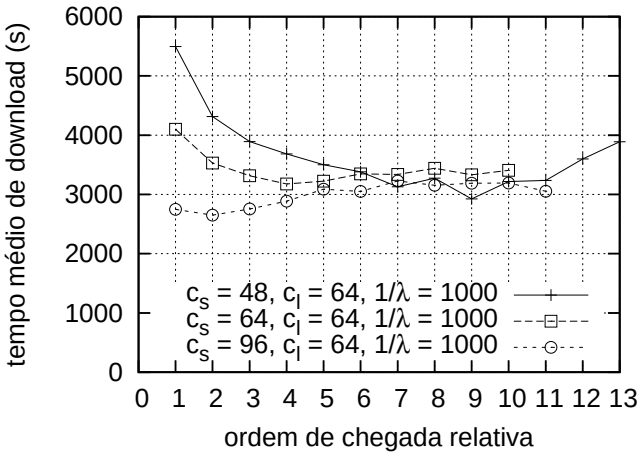


Figura 4.3: Tempo médio de download em função da ordem de chegada em um *busy period*.

Em particular, para o caso $c_s = 64$ KBps, o tempo médio de download tende a diminuir conforme aumenta a posição na ordem de chegada, e então os primeiros a chegar têm o tempo médio de download mais alto. Além disso, as diferenças nos tempos de download são significativas, podendo alcançar 30% (comparando a primeira e a quarta chegada).

A Figura 4.3 também indica que a variabilidade no tempo de download depende fortemente da capacidade de upload do seed. Em particular, um seed mais rápido leva ao efeito contrário: os tempos de download dos leechers tendem a aumentar com a posição na ordem de chegada. Intuitivamente, quando um seed lento está presente, os peers que chegaram mais tarde em um *busy period* obtêm altas taxas de download dos outros leechers, e por conseguinte, um menor tempo de download. Contudo, quando um seed rápido está presente, o primeiro leecher se beneficia da alta capacidade do seed até que ocorra a segunda chegada, resultando em menor tempo de download. Os resultados também ilustram efeitos de segunda ordem. Por exemplo, um peer que chegou muito tarde pode obter um tempo médio de download ligeiramente maior (ou menor) que um peer que chegou tarde (ex.: os leechers que chegaram em oitavo têm um tempo médio de download maior do que os que chegaram em quarto, para $c_s = 64$ KBps). Intuitivamente, isto acontece porque é provável que um peer que chegou muito tarde fique sozinho no *busy period*, tendo que recorrer ao seed para terminar o download. Como a capacidade de upload do seed pode ser menor (maior) que a taxa de download agregado que ele receberia dos outros leechers, seu tempo de download pode aumentar (diminuir). Este comportamento e suas consequências serão capturados e explicados pelo modelo matemático apresentado na seção seguinte.

A seguir, iremos estudar a intensidade de rajadas que ocorrem no processo de partida. A Figura 4.4 mostra a CCDF (função de distribuição cumulativa comple-

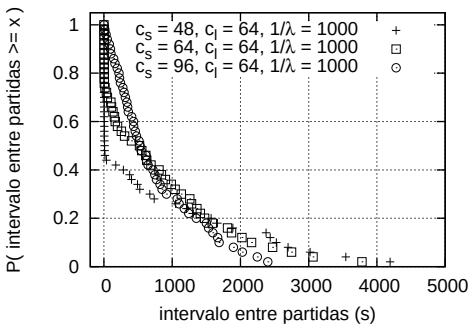


Figura 4.4: CCDF empírica do tempo entre partidas condicionado em um *busy period*.

Tabela 4.1: Estatísticas do tempo entre partidas dentro de um *busy period*.

c_s	Média amostral	Desvio padrão amostral
48 KBps	686.61	1107.15
64 KBps	800.16	948.57
96 KBps	741.33	638.42

mentar) empírica do tempo entre partidas condicionado no *busy period* (ou seja, não incluindo ao tempo entre partidas do último leecher em um *busy period* e o primeiro do próximo). Lembremos que o tempo entre chegadas dos peers segue uma distribuição exponencial com taxa 1/1000. Entretanto, os resultados indicam um processo de partida bastante distinto. Enquanto o intervalo médio entre chegadas é ≈ 1000 e o desvio padrão é ≈ 1000 nos três casos, as estatísticas do processo de partida em um *busy period* são significativamente diferentes, conforme mostra a Tabela 4.1.

Em particular, muitos peers tendem a deixar o swarm aproximadamente ao mesmo tempo: condicionando em um *busy period*, quase 30% dos peers deixam o swarm dentro de um intervalo de poucos segundos entre um e outro (quando $c_s = 64$ KBps). Além disso, o processo de partida também tem alta variabilidade: alguns peers levam dez vezes mais tempo para deixar o sistema após uma partida do que a média (quando $c_s = 64$ KBps). A figura também mostra de maneira clara que esse comportamento depende fortemente da capacidade de upload do seed, e é muito mais evidente quando o seed é lento. Intuitivamente, um seed mais lento eleva o tempo médio de download, aumentando portanto as chances que os leechers têm de sincronizar seus conteúdos durante o download e partir quase ao mesmo tempo. Notamos também que um seed rápido leva a um processo de saída com menos rajadas, embora ainda favoreça a ocorrência de curtos intervalos entre partidas (para $c_s = 96$ KBps, temos média igual a 741.33 e desvio padrão igual a 638.42). Finalmente, podemos ver o processo de saída como uma transformação que o BT aplica sobre o processo de chegada, modificando sua distribuição, média e desvio padrão.

Uma das consequências das taxas de download heterogêneas que está fortemente

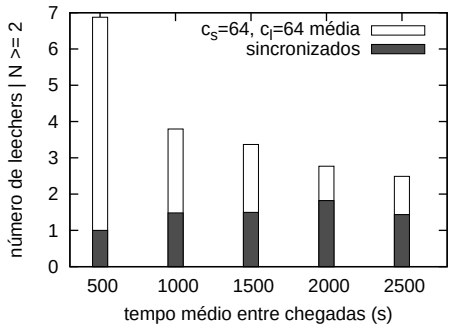


Figura 4.5: Número médio de leechers e número médio de leechers sincronizados, dado que existem pelo menos 2 leechers no swarm.

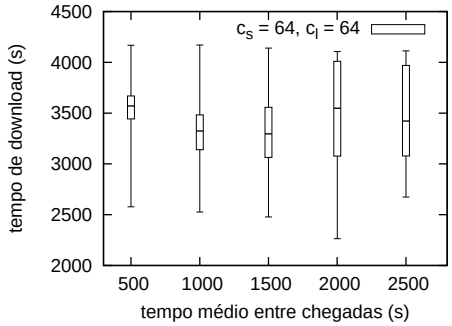


Figura 4.6: Box-plot do tempo de download dos leechers para diferentes tempos médios entre chegadas.

relacionada às saídas em rajada é a sincronização de conteúdo. A Figura 4.5 ilustra a intensidade de tal sincronização para diferentes taxas de chegada. Ela mostra o número médio de leechers no sistema e o número médio destes que estão sincronizados. Aqui iremos nos referir como sendo "leechers sincronizados" aqueles que não estão interessados em mais do que 50 pedaços (5% do arquivo) de qualquer outro. Tendo em vista que a sincronização só está definida quando existem 2 ou mais usuários, condicionamos os resultados aos intervalos em que $N \geq 2$. Observe que o número de leechers sincronizados permanece praticamente o mesmo conforme aumentamos o tempo médio entre chegadas, indicando que uma maior fração dos peers têm conteúdo similar quando a popularidade do swarm decresce.

A seguir iremos estudar a influência da taxa de chegada dos leechers nos tempos de download. A Figura 4.6 mostra um box-plot dos tempos de download dos peers em função do tempo médio entre chegadas (ou seja, o inverso da taxa de chegada), para $c_s = 64$ KBps. Para cada cenário, a curva box-plot indica o mínimo, o 25-percentil, mediana, 75-percentil e máximo do tempo de download. Note que quando o intervalo médio entre chegadas é grande (2000 ou 2500), o 75-percentil está muito próximo do tempo máximo de download, indicando que muitos peers têm tempos de download similares. Conforme o tempo médio entre chegadas dimi-

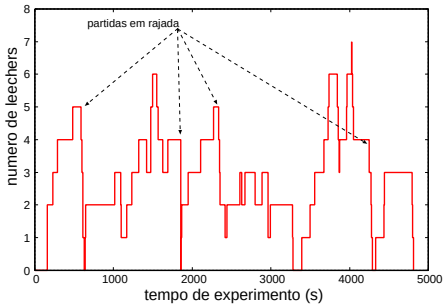


Figura 4.7: Tamanho do swarm para o experimento real.

nui, essa concentração próxima ao máximo diminui significativamente. De fato, a massa de probabilidade está concentrada em torno da média, indicando menor variabilidade ao reduzir o tempo entre chegadas. Contudo, a diferença entre o tempo mínimo e máximo de download não diminui com o tempo entre chegadas. Além disso, nós executamos simulações para diferentes valores de c_s e observamos que um seed rápido também tem uma forte influência neste comportamento, resultando em tempos de download muito menos concentrados: os percentis ficam mais bem distribuídos ao longo do intervalo entre o valor mínimo e o máximo.

4.4 Avaliação através de experimentos reais

Após apresentarmos os resultados obtidos através de simulações, descreveremos aqueles obtidos por experimentos baseados em protótipo implantados em cenários mais realísticos. Os experimentos reais foram realizados na Internet usando computadores do PlanetLab [38] para executar uma versão instrumentada de um cliente BT [3]. Embora um grande número de experimentos tenham sido conduzidos, iremos nos ater apenas a um conjunto restrito que é suficiente para o que queremos demonstrar. O objetivo aqui é validar o fenômeno da heterogeneidade em swarms BT homogêneos e suas consequências em uma aplicação real BT sendo executada na Internet.

Vamos considerar apenas swarms privados no experimento, no sentido de que somente peers controlados pelo experimento podem se conectar ao swarm para fazer upload e download do conteúdo. Cada swarm privado consiste de um único arquivo de S MB que é detido por um único seed que está sempre disponível e tem capacidade de upload c_s. Leechers interessados em baixar o conteúdo chegam no swarm segundo um processo de Poisson com taxa λ. Todos os leechers que chegam ao swarm são homogêneos e têm capacidade de upload c_l. Cada rodada do experimento é executada durante $t = 5000$ segundos.

Primeiramente vamos analisar a evolução do tamanho de um swarm pouco popular. A Figura 4.7 mostra o número de leechers no swarm ao longo do tempo

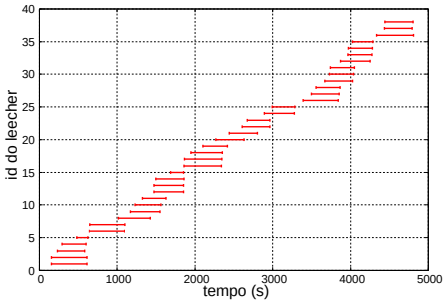
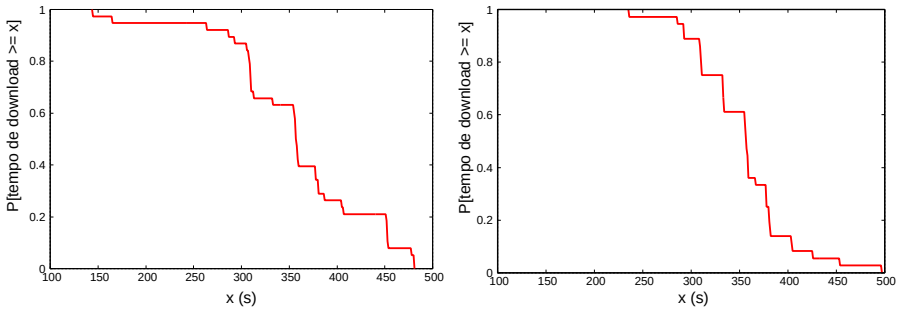


Figura 4.8: A dinâmica do swarm: chegadas e saídas de leechers.



(a) $c_s = 50$ KBps e $c_l = 50$ KBps. (b) $c_s = 60$ KBps e $c_l = 50$ KBps.

Figura 4.9: CCDF do tempo de download para os experimentos reais.

durante o período do experimento, com parâmetros $\lambda = 1/125$ peers/s, $S = 20$ MB, e $c_s = c_l = 50$ KBps. Podemos observar várias ocorrências de partidas em rajada, mesmo quando os leechers chegam segundo um processo de Poisson. Como discutido anteriormente, partidas em rajada são consequência da sincronização de conteúdo entre os leechers do swarm.

Através do mesmo experimento investigamos o impacto da ordem de chegada dos leechers nos seus tempos de download. A Figura 4.8 ilustra a dinâmica do swarm, onde cada linha horizontal corresponde ao tempo de vida de um leecher no swarm, iniciando quando o peer chega no swarm e terminando quando ele deixa o sistema. Note que os tempos de download dos peers (que correspondem ao seu tempo de vida no sistema) são significativamente diferentes. Em particular, em muitos casos leechers que chegam em diferentes instantes partem na mesma rajada. Por exemplo, o quinto leecher a chegar no swarm parte em uma rajada junto com os quatro anteriores. Portanto, o quinto leecher leva um tempo muito menor para completar o download, quando comparado ao primeiro leecher. Um comportamento semelhante acontece com o 15^o leecher e os três que chegaram imediatamente antes dele. Além de ilustrar a variabilidade dos tempos de download, esta observação também indica a injustiça com respeito à ordem de chegada dos leechers. Nesse caso, aqueles que chegam mais tarde em um *busy period* tendem a ter menores tempos de download.

Iremos considerar a distribuição dos tempos de download dos leechers para ilus-

trar sua alta variabilidade relativa. As Figuras 4.9a e 4.9b mostram a função de distribuição complementar cumulativa (CCDF) do tempo de download calculado para dois experimentos onde o seed tem capacidades de upload distintas ($c_s = 50$ KBps e $c_s = 60$ KBps, respectivamente, e os outros parâmetros se mantêm inalterados). Em ambos os resultados, os tempos de download exibem uma alta variância, como pode ser observado nas figuras. No caso $c_s = 50$ KBps (Figura 4.9a), os valores mínimo e máximo são 145 e 480 segundos, respectivamente, sendo o máximo mais que três vezes maior que o mínimo. Quando a capacidade de upload do seed é mais alta que a dos leechers, observamos pela Figura 4.9b que a variância no tempo de download diminui, como esperado, visto que a capacidade do sistema aumenta. Por fim, notamos várias descontinuidades (isto é, quedas abruptas) em ambas as curvas CCDF que são causadas pelos conjuntos de leechers que levam aproximadamente o mesmo tempo para fazer o download.

4.5 Modelo

Desenvolvemos um modelo simples com o objetivo de entender a origem dos tempos de download heterogêneos e suas consequências. Nosso modelo fornece uma aproximação para as taxas médias de upload e download observadas por cada leecher em diferentes instantes de tempo.

Considere um swarm homogêneo de um conteúdo pouco popular com um único seed onde os leechers chegam sequencialmente e partem tão logo terminam seus downloads, assim como aquele mostrado na Figura 4.1a. Neste cenário, partidas em rajada podem acontecer apenas se os leechers mais novos obtiverem aproximadamente o mesmo número de pedaços que os mais velhos, e então deixarem o swarm quase ao mesmo tempo. Isso por sua vez implica que os leechers mais novos precisam ter taxas de download mais altas que os mais velhos, pelo menos durante alguns períodos de tempo. Como isso acontece? Em um instante de tempo, um leecher mais velho i pode possuir todos os pedaços baixados por um leecher mais novo j. Então, a capacidade de upload de j será usada para servir apenas outros leechers, até que j receba um pedaço que i não tem. Durante este período de tempo, j não pode servir i, mesmo que não tenha nenhum outro leecher para servir. Portanto, o conjunto de pedaços detidos por cada leecher é a causa fundamental das taxas de download heterogêneas e precisa ser levado em consideração no modelo.

A fim de capturar a observação acima, propomos a seguinte modelagem: cada peer (seja um seed ou um leecher) será representado por um sistema de filas com múltiplas filas (ver Figura 4.10), uma para cada vizinho, sob uma disciplina *processor sharing*. A fila j do peer i contém os pedaços que interessam ao peer j (isto é, todos os pedaços que i tem e j não tem). Quando o peer j baixa um desses pedaços,

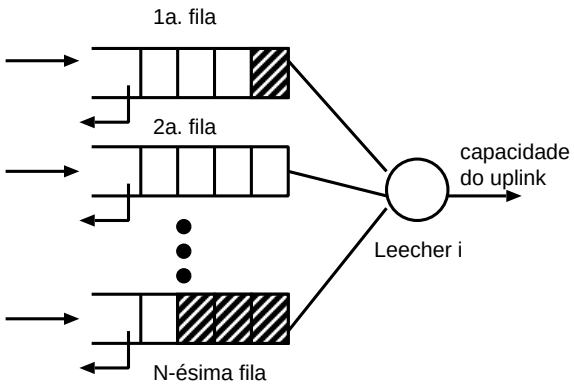


Figura 4.10: O leecher i pode ser representado por um sistema com múltiplas filas, uma para cada vizinho, contendo os pedaços que são interessantes para esses vizinhos.

seja a partir de i ou de algum outro peer, esse pedaço é removido da fila. Por outro lado, quando o peer i baixa um pedaço no qual seus vizinhos estão interessados, esse pedaço é colocado nas filas que correspondem à esses vizinhos, incrementando em uma unidade o tamanho das filas. Por fim, as filas do seed sempre têm todos os pedaços de que os vizinhos necessitam. Conforme um leecher vai baixando pedaços do seed e dos outros leechers, a fila correspondente no seed diminui, ficando finalmente vazia quando o leecher baixa o conteúdo por completo e deixa o swarm. Note que a ordem em que esses pedaços são servidos nessas filas depende da política de seleção de pedaços, mas isso não é relevante para a nossa discussão.

Sejam c_s e c_l a capacidade de upload do seed e dos leechers, respectivamente. Suponha que a capacidade de download dos leechers são muito maiores que c_s e c_l. Seja $N(t)$ o número de leechers no sistema no instante de tempo t. Tendo em vista que o seed sempre tem pedaços que interessam a todos os leechers, todas as $N(t)$ filas no seed têm pedaços acumulados. Então, todas as filas serão servidas à taxa $c_s/N(t)$. Note que, como o swarm é pouco popular, vamos supor que o tamanho do swarm é pequeno suficiente tal que cada leecher é vizinho de todos os outros peers, incluindo o seed.

Um leecher pode não ter nenhum pedaço interessante para alguns de seus vizinhos no tempo t. Considere que os leechers são identificados pela sua ordem de chegada, então o leecher i é o i-ésimo leecher a entrar no swarm. Além disso, seja $n_i(t) \leq N(t) - 1$ o número de leechers interessados nos pedaços que i possui. A taxa instantânea de upload de i para cada um desses leechers é $c_l/n_i(t)$.

Um leecher ter ou não pedaços que interessam um outro depende dos respectivos *bitmaps* desses leechers, isto é, dos subconjuntos de pedaços que cada um possui no instante t. O conjunto de bitmaps de todos leechers determina de forma única os pedaços que estão em cada fila do nosso modelo. Entretanto, a dinâmica dos bitmaps é muito complicada e capturar esta dinâmica seria desnecessariamente complicado

para modelar o fenômeno no qual estamos interessados. Em vez disso, vamos considerar o número de pedaços que cada leecher i possui, $b_i(t)$ $\forall i$, e inferir quando um leecher tem pedaços que interessam aos outros.

Por simplicidade, seja $b_i(t) = b_i$, $N(t) = N$ e $n_i(t) = n, \forall i$. Duas observações podem ser feitas com respeito à b_i, e a relação de interesse entre os leechers.

Observação 1 *Se $b_i > b_j$, então i tem pelo menos $b_i - b_j$ pedaços que interessam a j.*

Observação 2 *Se $0 < b_i \leq b_j$, é impossível determinar se i tem ou não pedaços que interessam a j sem informação adicional.*

A seguir iremos usar essas duas observações para derivar um modelo simples para capturar as taxas de upload e download entre os peers. Em relação à Observação 2, vamos considerar que nenhuma informação adicional está disponível. Por isso, vamos assumir que um peer não possui interesse em vizinhos que possuem menos pedaços que ele.

4.5.1 Um modelo de fluido simples

Vamos modelar o conteúdo como um fluido, ou equivalentemente, que seus pedaços podem ser subdivididos em partes infinitesimais que podem ser trocadas (através do upload/download) continuamente.

Para simplificar a explicação do modelo, assuma que $b_1 > b_2 > \cdots > b_N$, isto é, leechers mais velhos possuem necessariamente mais pedaços, onde o índice $1, 2, 3, \ldots$ representa a ordem de chegada dos leechers. Faremos as seguintes suposições:

- Mesmo que o leecher i tenha entrado no swarm depois de j, ou seja, $j < i$, i pode fazer upload de pedaços para j desde que i esteja baixando pedaços de qualquer peer k que tenha mais pedaços que j, isto é, $k < j$.

- Cada pedaço que um leecher baixa do seed é imediatamente interessante a todos os outros leechers, independentemente da idade deles. Embora isto não ocorra quando os vizinhos de um peer possuam, em conjunto, todos os pedaços que compõem o conteúdo, a política de seleção *rarest-first* usada no BT maximiza a probabilidade de que isto aconteça; A Figura 4.11 ilustra a ideia de que um peer mais novo pode enviar pedaços para um mais velho. Nesse caso, o peer 4 pode fazer upload para o peer 2, pois está recebendo do seed e do peer 1 pedaços que interessam ao peer 2.

- Como a capacidade de upload do seed é c_s, cada leecher baixa dele com taxa c_s/N. Agora seja g_{ij} a taxa a que o peer i pode potencialmente fazer upload

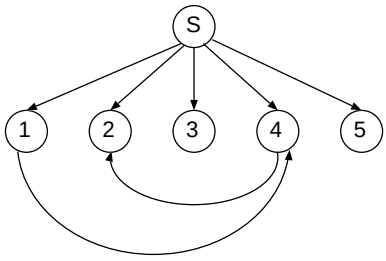


Figura 4.11: Exemplo de caso em que um peer recebe pedaços que interessam a um peer mais velho, podendo então transmiti-los ao último.

para j sem considerar restrições de capacidades (isto é, independentemente das capacidades de upload e download de i e j, respectivamente). Caso um leecher i seja mais velho que j, i possuirá pedaços que interessam a j. Portanto, da perspectiva do sistema de múltiplas filas, a fila j no leecher i tem pedaços acumulados e $g_{ij} = \infty$. Por outro lado, se i é mais novo que j, a taxa g_{ij} é dada pela taxa a que i recebe pedaços que interessam a j.

De acordo com as suposições anteriores, a taxa g_{ij} a que i recebe pedaços que interessam a j é igual a taxa a que os peers mais velhos que j fazem upload para o peer i somada a taxa a que i recebe do seed:

$$g_{ij} = c_s/N + \sum_{k<j} u_{ki}, i > j. \tag{4.1}$$

onde u_{ki} é a taxa a que o leecher k faz upload para i.

Por exemplo, na Figura 4.11, g_{42} é a soma da taxa a que o peer 2 recebe do seed ($c_s/5$) com a taxa a que ele recebe dos peers mais velhos que 2, i.e. a taxa a que ele recebe do peer 1 (u_{14}). Logo, $g_{24} = c_s/5 + u_{14}$.

Vamos fazer agora uma importante observação em relação à Equação (4.1). Considere o leecher i e um outro leecher j. Quanto mais velho j for em relação à i, menor é a taxa a que i poderá fazer upload para j, isto é, menor será g_{ij}. Se j é mais novo que i, então $g_{ij} = \infty$. Esta observação implica que $g_{i1} \leq g_{i2} \leq \cdots \leq g_{iN}$.

Como a capacidade de upload dos peers é finita, precisamos determinar como a capacidade de um dado peer i será dividida para servir cada um dos leechers. Em particular, lembre-se que u_{ij} é a taxa de upload do peer i para o peer j e note que $\sum_k u_{ik} \leq c_l$, onde c_l é a capacidade de upload de um leecher. Para determinar u_{ij}, iremos usar g_{ij} e um mecanismo de alocação de banda que segue um algoritmo de preenchimento progressivo, conforme ilustrado nas Figuras 4.12(a-e). A grosso modo, quantidades infinitesimais da capacidade são alocadas para cada leecher até que a capacidade de upload do peer seja atingida ou até que um ou mais leechers estejam satisfeitos com relação às restrições g_{ij} $\forall j$. Na Figura 4.12c, por exemplo, podemos ver que a capacidade de upload de i que foi alocada para o primeiro vizinho

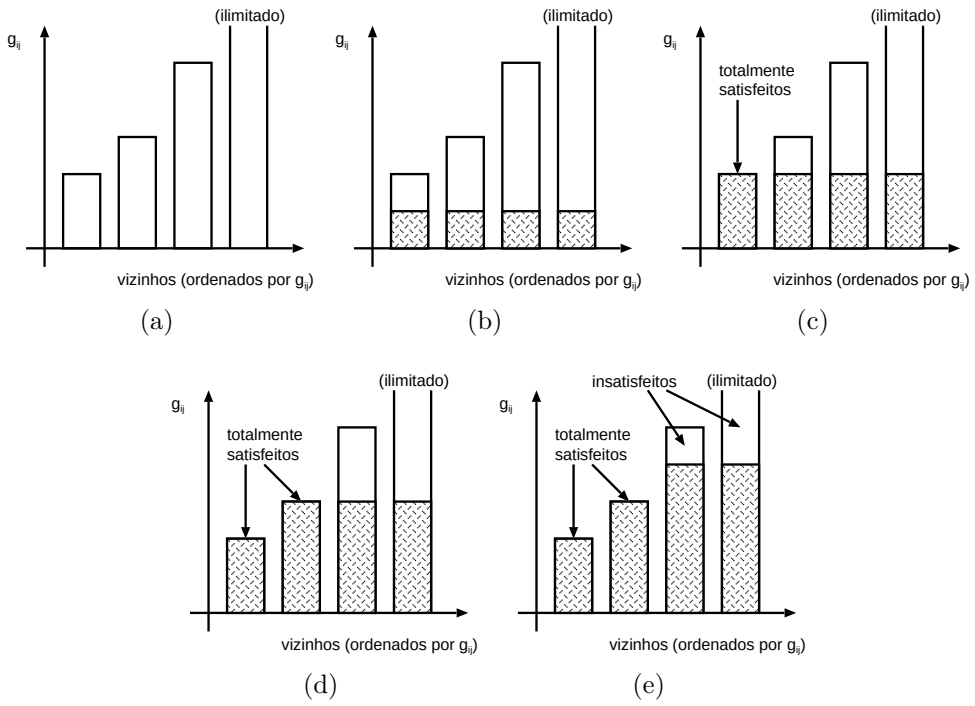


Figura 4.12: A alocação de banda do usuário i segue um algoritmo de preenchimento progressivo.

j (da esquerda para a direita) é igual à restrição definida por g_{ij}. Em seguida, o algoritmo continua a distribuir a capacidade restante entre os leechers cujas restrições ainda não estão satisfeitas. Na etapa indicada pela Figura 4.12d, vemos que a restrição correspondente ao segundo vizinho já foi satisfeita. Por outro lado, na alocação final mostrada na Figura 4.12e, observamos que a capacidade de upload destinada ao terceiro e quarto peers é menor que g_{ij}.

A alocação final da capacidade do leecher i pode ser obtida de forma eficiente calculando-se as variáveis u_{ij} na ordem $j = 1, \ldots, N$, através da seguinte equação:

$$u_{ij} = \min\left(g_{ij}, \frac{c_l - \sum_{k<j} u_{ik}}{n - |\{k|k < j, k \neq i\}|}\right) \tag{4.2}$$

onde $|A|$ é a cardinalidade do conjunto A. Lembre que na Equação (4.1), g_{ij} depende de $u_{1,i}, u_{2,i}, \ldots, u_{j-1,i}$, para $i > j$. Calculando u_{ij} na ordem $i = 1, \ldots, N$, nos asseguramos de que toda variável na Equação (4.2) foi previamente calculada.

Como exemplo, considere o cálculo da matriz $\mathbf{U} = (u_{ij})$, que determina as taxas de uploads entre os peers em um dado instante, para um pequeno swarm contendo um único seed e $N = 3$ leechers. Sejam as suas capacidades de upload iguais à $c_s = 60$ KBps e $c_l = 96$ KBps, respectivamente e assuma $b_1 > b_2 > b_3$. A matriz $\mathbf{U}$ e a ordem de cálculo dos elementos pode ser vista na Figura 4.13. A taxa de

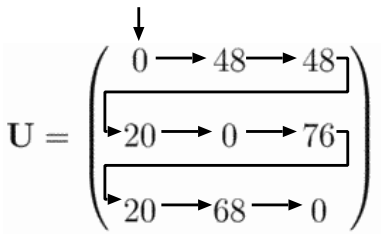


Figura 4.13: Exemplo da matriz $\mathbf{U} = (u_{ij})$ mostrando a ordem correta de cálculo.

download d_i do peer i é simplesmente c_s/N mais a soma dos elementos na coluna i:

$$d_i = c_s/N + \sum_{j=1}^{N} u_{ji} \tag{4.3}$$

Logo,

$$d_1 = 60/3 + 0 + 20 + 20 = 60 \tag{4.4}$$

$$d_2 = 60/3 + 48 + 0 + 68 = 136 \tag{4.5}$$

$$d_3 = 60/3 + 48 + 76 + 0 = 144 \tag{4.6}$$

As Equações (4.4)-(4.6) estão em consonância com a ideia de que peers homogêneos podem ter taxas de download heterogêneas dependendo do número de pedaços que cada leecher possui. Além disso, os leechers mais novos tendem a ter uma taxa de download maior, visto que eles obtêm uma maior taxa de upload dos outros leechers.

Em determinado instante, o número de pedaços que um leecher possui pode alcançar o número de pedaços de um leecher mais velho. De fato, isto irá ocorrer já que os leechers mais novos tendem a ter uma taxa de download mais alta. Quando isso acontece, esses dois leechers não terão mais pedaços interessantes um para o outro. Portanto, as Equações (4.1) e (4.2) precisam ser reescritas como função do número de pedaços que cada peer possui, $b_i, \forall i$:

$$g_{ij} = c_s/N + \sum_{b_k > b_j} u_{ki} \ , \ b_i \leq b_j. \tag{4.7}$$

$$u_{ij} = \min\left(g_{ij}, \frac{c_l - \sum_{k|b_k>b_j} u_{ik}}{n - |\{k|b_k > b_j, k \neq i\}|}\right) \tag{4.8}$$

Intuitivamente, a Equação (4.8) combina as duas restrições na taxa com que i faz upload para j. O primeiro termo diz respeito à taxa máxima instantânea sem considerar limitações de capacidade. O segundo termo reflete a fração da capacidade do upload de i que pode ser dedicada à j dado que parte da banda já foi alocada. Neste caso, $c_l - \sum_{k|b_k>b_j} u_{ik}$ é a capacidade restante de i e $n - |\{k|b_k > b_j, k \neq i\}|$ é

o número de peers que irão compartilhá-la (incluindo j).

Por fim, repare que estamos assumindo que um peer não possui interesse em vizinhos que têm menos pedaços que ele, pois segundo a Observação 2, não é possível determinar a existência desse tipo de interesse sem informação adicional. No entanto, o modelo proposto pode ser facilmente adaptado para o caso em que conhecemos o número exato de pedaços que o peer i possui que interessam ao peer j $\forall i, j$. Esse modelo adaptado é mais geral e contempla, por exemplo, o caso em que i e j possuem pedaços que interessam um ao outro, mesmo que tenham o mesmo número de pedaços. No entanto, a dinâmica da troca de pedaços é muito complicada e, portanto, a viabilidade de aplicar tal modelo é questionável. Além disso, veremos a seguir que o modelo original produz bons resultados.

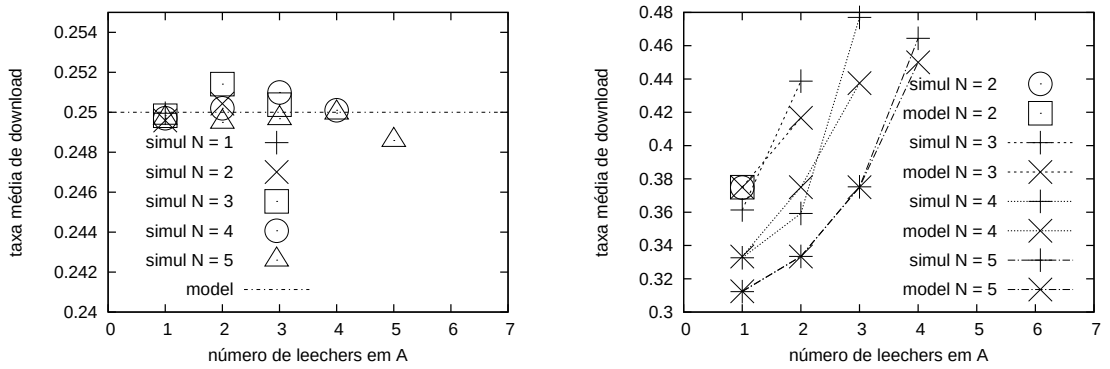
4.5.2 Validação do modelo

O modelo proposto na seção anterior fornece uma aproximação para a taxa média de download recebida por um leecher em um swarm a partir do relacionamento entre o número de pedaços que cada leecher possui (ver Equações (4.7) e (4.8)). Nesta seção, iremos validar o modelo comparando as previsões com resultados de simulação.

Considere um swarm homogêneo contendo N leechers, tal que $c_s = c_l$. Neste cenário, é razoável assumir que $b_1 \geq b_2 \geq \cdots \geq b_N$ considerando que o índice reflete a ordem de chegada de um peer. Vamos particionar o conjunto de leechers em dois subconjuntos: leechers com o mesmo número de pedaços que o leecher mais velho (subconjunto A), e aqueles com menos pedaços que o mais velho (subconjunto B). Para o cenário em questão, o modelo prevê que todos os leechers dentro de um mesmo subconjunto terão a mesma taxa de download. Além disso, um leecher em B terá uma taxa de download maior que um leecher em A e essa diferença depende do tamanho dos conjuntos. A seguir, iremos comparar a taxa média de download dos peers em cada um desses conjuntos calculada pelo modelo com os resultados de simulação.

Aqui usamos chegadas determinísticas para reproduzir os cenários exatos que pretendemos comparar. Para um swarm com N leechers, dos quais n_A pertencem ao conjunto A (isto é, têm b_1 pedaços), as chegadas ocorrem da seguinte forma: os primeiros n_A leechers entram próximos uns aos outros e depois que eles têm aproximadamente o mesmo número de pedaços, isto é, $|b_1 - b_i| < 3$, os outros $N - n_A$ leechers entram no swarm sequencialmente e distantes uns dos outros. Então calculamos a média das taxas de download recebidas pelos leechers em A e a média das taxas recebidas pelos leechers em B sobre um grande intervalo de tempo mas antes que ocorram partidas.

Para cada cenário, simulamos 5 rodadas. Os intervalos de confiança obtidos



(a) Taxas médias de download (leechers em A). (b) Taxas médias de download (leechers em B).

Figura 4.14: Resultados da simulação para $c_s = c_l = 0.25$.

são relativamente pequenos e serão omitidos. Os resultados para $1 \leq N \leq 5$ e $1 \leq n_A \leq N$ são mostrados nas Figuras 4.14(a,b). Nessas figuras, o eixo x indica o número de leechers no conjunto A e o eixo y indica a taxa média de download. Note que, se o número de leechers em A é x, o número de leechers em B é dado por $N - x$.

Primeiramente, iremos analisar o desempenho dos leechers em A com respeito à taxa de download. A taxa média de download de um leecher em A prevista pelo modelo para este cenário não depende de N ou n_A. Isso acontece porque os peers que fazem parte desse conjunto possuem todos os pedaços que o seed já inseriu no swarm. Logo, os vizinhos de um peer em A só podem enviar para ele o que recebem diretamente do seed, ou seja, estão limitados à taxa c_s/N. Como essa taxa de transmissão está aquém da capacidade que pode ser dedicada a um peer quando $c_s = c_l$, todo peer irá enviar dados para cada peer em A a uma taxa exatamente igual a c_s/N. Portanto, a taxa média de download prevista pelo modelo é $c_s/N + (N-1)c_s/N = c_s = 0.25$.

A Figura 4.14a mostra os resultados de simulação e do modelo para os leechers em A. A taxa média de download prevista pelo modelo está sendo representada pela linha pontilhada. O número total de peers considerado está representado por diferentes símbolos (o quadrado, por exemplo, indica 3 peers). Note que o mesmo símbolo pode aparecer várias vezes, uma para cada valor possível de n_A (o quadrado em $x = 2$, por exemplo, indica que existem 2 peers em A e 1 peer em B).

O modelo é bastante acurado, mesmo para diferentes valores de N e n_A. Em particular, o erro relativo é menor que 1% para todos os cenários.

Iremos agora analisar o desempenho dos leechers em B. Para um N fixo, conforme o número de leechers em A aumenta, a taxa média de download dos leechers em B aumenta. Por outro lado, para um n_A fixo, a taxa média de download dos leechers em B diminui com N. Essas variações na taxa média de download em B

acontecem porque quanto mais peers existem em A, maior é a capacidade que resta para transmitir dados para peers em B. Isso porque a taxa a que os leecher em A podem transmitir entre si é bastante restrita. Entretanto, se o número de leechers em B aumenta, o ganho individual oriundo da capacidade ociosa em A passa a ser menos significativo.

A Figura 4.14b mostra a taxa média de download para os leechers em B. Devido à numerosa quantidade de pontos mostrando resultados de simulação e do modelo, usamos '+' para identificar simulações e 'x' para identificar os resultados do modelo (exceto para $N = 2$, onde um círculo e um quadrado são usados, respectivamente). Além disso, para facilitar a comparação desses pontos, traçamos linhas conectando os resultados do mesmo tipo (simulação ou modelo) para o mesmo valor de N.

A curva pontilhada, por exemplo, representa $N = 3$. Nesse caso, n_A pode assumir valores de 1 a 3. Para essa curva, observamos que quando $n_A = 1$, os resultados do modelo e da simulação são praticamente idênticos. Aumentando n_A, temos um aumento na taxa média de download de peers em B, conforme a intuição apresentada anteriormente. Para $n_A = 2$, a taxa média de download obtida por simulação é um pouco maior. Por outro lado, para $n_A = 3$, a taxa média de download oriunda do modelo é maior, mas também não é significativa.

Uma observação interessante é que um maior número de leechers no swarm implica em um maior intervalo para as possíveis taxas de download em B, dado que n_A pode variar de 1 a N. Essa tendência pode ser observada tanto na simulação quanto no modelo.

Por fim, notamos que o modelo é bastante acurado, sendo as diferenças menores que 10% em todos os casos e até mesmo imperceptíveis em alguns cenários. Mais importante ainda, o modelo captura bem o comportamento observado na simulação.

4.6 Previsão de partidas em rajada

O modelo apresentado na Seção 4.5 pode ser usado para estimar o número de partidas que ocorrem em uma rajada. Em particular, considere a chegada de um leecher que inicia um *busy period* (isto é, a primeira chegada após um momento em que o swarm está sem leechers). A seguir, iremos estimar o número médio de peers que deixam o swarm em uma rajada junto ao leecher que iniciou o *busy period*.

Na prática, as partidas em rajada não ocorrem exatamente ao mesmo tempo devido a variações inerentes à rede e a inexistência de sincronização entre os peers. Entretanto, nosso modelo não leva esses fatores em consideração e, por isso, estamos interessados nos leechers que deixam o swarm exatamente no mesmo instante em que o primeiro leecher.

Seja f o primeiro leecher de um *busy period* e assuma que o processo de chegada

de leechers segue uma distribuição Poisson com taxa λ. Assim como suposto pelo modelo, assuma que um seed está sempre presente e tem capacidade de upload c_s, enquanto os leechers tem capacidades idênticas e iguais a c_l. Finalmente, seja S o número de pedaços do conteúdo.

Cada leecher recebe pedaços do seed a uma taxa c_s/N, onde N é o número de peers no swarm. Esses pedaços são interessantes a todos os $N-1$ vizinhos e podem ser transmitidos para eles. Logo, se $c_l < c_s \times \frac{N-1}{N}$, podemos garantir que os leechers irão acumular pedaços que foram recebidos do seed e não foram repassados. Sendo assim, cada leecher irá possuir pedaços interessantes a cada um de seus vizinhos. Consequentemente, a taxa de upload entre dois leechers i e j quaisquer será igual a $u_{ij} = c_l/(N-1)$, visto que $g_{ij} = \infty$ (ver Equação (4.8)). Portanto, no regime em que $c_l < c_s \times \frac{N-1}{N}$, todos os leechers fazem download a mesma taxa, não podendo ocorrer partidas em rajada no momento em que f sai do swarm.

Por outro lado, quando $c_l \geq c_s \times \frac{N-1}{N}$, os vizinhos de f conseguem enviar para ele todos os pedaços que recebem do seed, sem que haja acúmulo. Como f faz download do seed a uma taxa c_s/N e de cada um de seus $N-1$ vizinhos a uma taxa c_s/N, o leecher f irá baixar o conteúdo a uma taxa fixa igual a c_s, independentemente do número de peers no swarm. Note que c_s é também o limite superior da taxa de download média, já que o seed não pode colocar novos pedaços na rede à uma taxa maior. Então, f irá levar $T = S/c_s$ para terminar o download.

Vamos agora mostrar como calcular os limites inferior e superior para o número de partidas em rajada no regime em que $c_l \geq c_s \times \frac{N-1}{N}$. Considere as chegadas que ocorrem enquanto o peer f está no swarm. Esse número de chegadas n é uma variável aleatória e segue a distribuição Poisson com parâmetros λ e T. As taxas de download desses leechers são função de n e dos seus instantes de chegada. Além disso, como discutido na Seção 4.5.2, valores mais altos de n implicam em uma intervalo maior para as taxas de download (ver Figura 4.14b). Para obter limites inferior e superior conservadores para essas taxas de download, consideramos um valor suficientemente grande para n. Em particular, usamos n_{99}, o 99-percentil de n, e portanto, $P[n \leq n_{99}] \leq 0.99$.

Dado que exatamente n_{99} leechers entraram no swarm antes da saída de f, podemos usar o modelo para obtermos as taxas mínima e máxima de download recebidas pelos peers, considerando quaisquer instantes de chegada. Sejam $d_{\min}$ e $d_{\max}$, respectivamente, as taxas mínima e máxima de download obtidas através do modelo dado que o swarm tem $n_{99}+1$ leechers.

Considere novamente os subconjuntos apresentados na seção anterior, A (leechers com o mesmo número de pedaços que f) e B (leechers com menos pedaços que f). A taxa mínima de download $d_{\min}$ é obtida por um leecher $m \in B$ quando o único

Tabela 4.2: Limites para o número esperado de leechers que partem junto com f em uma rajada, para $\lambda = 1/1000$ peers/s e $S = 1000$ pedaços.

c_s (KBps)	$E[n]$	B_{min}	B_{max}	$\frac{B_{min}}{E[n]}$	$\frac{B_{max}}{E[n]}$
48	5.333	1.667	4.378	0.312	0.821
64	4.000	0.400	1.895	0.100	0.474

leecher em A é f. Neste caso, essa taxa é dada por

$$d_{\min} = \frac{c_s}{n_{99}+1} + u_{f,m} + \sum_{i \in B} u_{i,m} \tag{4.9}$$

onde o termo $\sum_{i \in B} u_{i,m}$ corresponde a soma das taxas a que m recebe de peers em B.

Por outro lado, um leecher M em B alcança a taxa máxima de download $d_{\max}$ quando ele é o único elemento no conjunto B, ou seja, $|A| = n_{99}$. Neste outro caso, essa taxa é dada por

$$d_{\max} = \frac{c_s}{n_{99}+1} + \sum_{i \in A} u_{i,M} \tag{4.10}$$

Todos os leechers que chegam antes de $T - S/d_{\min}$ irão deixar o swarm junto com f, em uma rajada. O número esperado de peers, $B_{\min}$, que irá chegar dentro desse intervalo é dado simplesmente por

$$B_{min} = \lambda \left(T - \frac{S}{d_{\min}} \right) \tag{4.11}$$

Similarmente, no máximo todos os leechers que chegarem antes de $T - S/d_{\max}$ irão deixar o swarm junto com f, em uma rajada. O número esperado de peers, $B_{\max}$, que irão chegar dentro deste intervalo, é dado simplesmente por

$$B_{min} = \lambda \left(T - \frac{S}{d_{\max}} \right) \tag{4.12}$$

Por fim, o número médio de leechers que deixam o swarm junto com f, em uma rajada, está entre os limites $B_{\min}$ e $B_{\max}$.

A Tabela 4.2 mostra o número esperado de chegadas ao swarm antes que f deixe o sistema, $E[n]$, que é simplesmente λT, e ambos os limites inferior e superior $B_{\min}$ e $B_{\max}$, respectivamente. A tabela mostra resultados numéricos para diferentes valores de c_s, mas para $c_l = 64$ KBps, $\lambda = 1/1000$ peers/s e $S = 1000$ pedaços. Os resultados indicam que o número médio de peers que deixa o sistema em uma rajada pode ser significativo: entre 31% e 82% do número médio de chegadas quando o seed é mais lento que os leechers e entre 10% e 47% quando eles têm a mesma capacidade de upload. Também observamos que essas razões diminuem quando c_s aumenta,

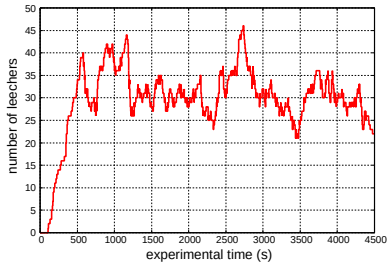


Figura 4.15: Evolução do número de leechers em um swarm popular ($\lambda = 1/12$, $c_s = 50$ KBps, $c_l = 50$ KBps)

indicando que as partidas em rajada são menos prováveis de acontecer com seeds rápidos.

4.7 Chegadas Poisson para swarms não tão pouco populares

O que acontece se considerarmos swarms muito populares, onde a taxa de chegada é muito alta, dando origem a swarms bem populosos? A Figura 4.15 mostra resultados experimentais da dinâmica de chegadas e partidas para este cenário (chegadas Poisson com taxa $\lambda = 1/12$ e capacidades de upload de $c_s = 50$ KBps e $c_l = 50$ KBps). Interessantemente, observamos várias das consequências de se ter taxas de download heterogêneas. Em particular, podemos observar partidas em rajada, sincronização de conteúdo e alta variabilidade dos tempos de download (peers que partem em uma grande rajada têm diferentes tempos de download, pois a chegada é bem-comportada), havendo saídas após 600s e após 1200s. De certa forma, o fenômeno é bem evidente durante a parte inicial do *busy period*, mas não tão forte depois disso, pois as partidas em rajada passam a ser compostas por um número menor de peers.

4.8 Conclusão

Neste capítulo identificamos, caracterizamos e modelamos um fenômeno interessante no BT: peers homogêneos (com respeito à capacidade de upload) têm taxas de download heterogêneas. Essa característica é mais evidente em swarms pouco populares (poucos leechers) e tem importantes consequências que impactam diretamente o desempenho dos peers e do sistema como um todo. Apesar da grande atenção que vem sendo destinada à área de avaliação e desempenho de redes P2P do tipo BT na última década, este é o primeiro trabalho a estudar esta característica.

Observamos que a razão pela qual as taxas heterogêneas ocorrem em swarms

homogêneos reside na distribuição dos pedaços. Em particular, vimos que esta pode ser bem diferente da distribuição uniforme de pedaços, que é assumida em grande parte dos trabalhos na literatura. De fato, em um estado em que os pedaços estão uniformemente distribuídos entre os peers, as taxas médias de download são iguais. É interessante notar que, ao contrário do que poderia se esperar, muitas vezes o sistema não converge para este estado.

O modelo matemático proposto neste trabalho utiliza a observação anterior para capturar as taxas de download heterogêneas. Ele nos permite prever com acurácia a alocação da capacidade do sistema (capacidade de upload agregada de todos os peers) em função da quantidade de pedaços que cada leecher possui.

Capítulo 5

Conclusões

O BitTorrent é uma aplicação P2P bastante popular nos dias de hoje, devendo grande parte desse sucesso ao seu bom desempenho no compartilhamento e recuperação de arquivos grandes, como filmes. Apesar de se basear em mecanismos distribuídos em que um peer só possui informação local (i.e., sobre sua vizinhança), o BitTorrent é muito eficiente. Entretanto, ele foi criado sem um estudo aprimorado de quais seriam os parâmetros ótimos para o seu funcionamento. Isso deixa margem para questionamentos, tais como "para qual ponto de operação o sistema é conduzido?" e "será que podemos atingir um ponto de operação melhor através de algumas modificações?".

No intuito de entender para onde o sistema é conduzido, investigamos duas características que emergem em redes formadas por peers do BT interessados em baixar um certo conteúdo. Essas características estão fortemente relacionadas ao desempenho do sistema.

Na primeira parte do Capítulo 3, estudamos o AM por largura de banda que ocorre em swarms do BT. Vimos como o AM está relacionado aos parâmetros e mecanismos desse sistema através da implementação um modelo de simulação da dinâmica do BT. Em particular, observamos que o valor máximo do *assortative coefficient* depende apenas do número de vértices que cada nó conhece (k), do número de vértices a quem ele oferece upload (x) e do número de tags diferentes na rede (v). No entanto, o tempo de convergência e o valor final do *assortative coefficient* dependem fortemente da probabilidade de troca otimista (p). Identificamos um compromisso no valor de p: se a troca otimista ocorre com alta probabilidade, o processo converge mais rápido, mas atinge taxas mais baixas de AM.

Conforme mostrado em [8], a clusterização pode ser benéfica ao sistema, melhorando o desempenho da disseminação de dados. Isto posto, podemos concluir que seria interessante que a taxa com que ele é executado fosse variável ao longo do tempo. Em particular, executar o *unchoke* otimista a uma taxa maior quando o usuário entra no sistema e diminuir essa taxa quando um grau elevado de AM

é atingido, poderia combinar as vantagens dos dois regimes: rápida convergência e altíssima clusterização.

No entanto, para desenvolver tal mecanismo de controle, seria necessário um estudo sobre quando e como a taxa de *unchoke* otimista deveria ser alterada. Esse mecanismo deve ser definido em função da vizinhança do peer e dos conjuntos para quem ele faz upload e de quem faz download. Por fim, note que sendo o sistema bastante dinâmico onde novas chegadas e partidas ocorrem a todo momento, seria natural que esse mecanismo utilizasse algoritmos adaptativos em oposição aos algoritmos estáticos que são empregados no BitTorrent atual. Deixamos então o desenvolvimento de tal mecanismo como possível trabalho futuro.

Ainda no Capítulo 3, propusemos um modelo analítico para estudar o estado estacionário da clusterização por largura de banda. Para que o modelo fosse analiticamente tratável, analisamos o comportamento de um peer de capacidade intermediária em detalhe, enquanto o restante do sistema era considerado de forma agregada. Devido a essa simplificação, não pudemos empregar o *assortative coefficient* como métrica, pois este é uma medida global. Alternativamente, obtivemos o número médio de downloads e uploads que um peer intermediário faz para cada classe de vizinhos. Entretanto, o modelo poderia ser adaptado para representar um peer de capacidade alta ou baixa e através dos resultados obtidos a partir dos três tipos de peer, permitir o cálculo de métricas definidas sobre todo o sistema. Para representar um usuário com link de baixa capacidade, por exemplo, nenhum vizinho deveria ser considerado no conjunto L, que continha os peers com capacidade inferior àquele sendo modelado.

No Capítulo 4, estudamos a ocorrência de tempos de downloads heterogêneos em swarms do BitTorrent homogêneos com relação à capacidade de upload. Do nosso conhecimento, esta é a primeira vez que este problema foi descrito na literatura. Neste trabalho, identificamos e caracterizamos este comportamento, que tem muitas implicações importantes, tais como alta variabilidade nos tempos de download, injustiça com respeito à ordem de chegada, partidas em rajada e sincronização de conteúdo entre os peers.

Para estudar este problema, propusemos um modelo analítico para as taxas de troca de dados entre os peers em swarms pouco populares. Entretanto, é interessante considerar a ocorrência do problema em cenários mais gerais. Embora nós tenhamos mostrado o seu surgimento sob um processo de chegada de peers fixo e sob chegadas Poisson, acreditamos que peers homogêneos podem ter taxas de download heterogêneas sob padrões de chegada bastante gerais. Em particular, dado um padrão de chegada de peers no swarm, é possível escolher os parâmetros do sistema (isto é, capacidades de upload do seed e dos leechers, tamanho do arquivo) tais que os efeitos descritos aqui sejam bem evidentes. Intuitivamente, escolhendo um seed rápido o

suficiente, os peers não serão capazes de disseminar os pedaços antigos antes que os novos sejam colocados no swarm, fazendo com que sempre tenham um grande número de pedaços diferentes. Por outro lado, escolher um arquivo grande o bastante faz com que muitos peers sincronizem antes de terminar o download. De certa forma, o comportamento observado e descrito é bastante geral, muito embora a caracterização e a modelagem desse fenômeno em swarms populares seja muito mais difícil, devido à complexa dinâmica da troca de pedaços no BT e consequentemente à relação de interesse entre os peers. Deixaremos a investigação desses cenários (swarms populares, outros processos de chegada) como outro possível trabalho futuro.

Vale salientar que a injustiça em relação à ordem de chegada (que é consequência direta da ocorrência de taxas de download heterogêneas) abre espaço para o estudo de estratégias que determinem o momento ótimo para um leecher entrar no swarm. Tais estratégias seriam fundamentadas na Teoria dos Jogos. Seria interessante estudar também a existência de equilíbrio e o desempenho do sistema nesse equilíbrio em pelo menos dois cenários: quando um único peer está agindo de forma estratégica e quando todos estão agindo assim.

A síndrome do pedaço faltante [24] está fortemente relacionada às nossas descobertas. O aspecto chave dessa síndrome é a sincronização de conteúdo, onde uma grande fração de peers tem todos, exceto um mesmo pedaço. Essa situação é particularmente ruim para o desempenho do swarm, já que a taxa de saída do swarm será limitada pela capacidade de upload do seed (supondo que os peers partem tão logo adquiram o último pedaço). Nosso trabalho mostrou que os peers podem sincronizar seus conteúdos muito antes do último pedaço. De certa forma, isso generaliza a síndrome para uma síndrome da sincronização de pedaços, que é inerente à dinâmica do BT, devido às taxas de download heterogêneas. Uma vez que os peers sincronizaram seus conteúdos, eles só podem adquirir novos pedaços a partir do seed, estando então limitados à capacidade de upload do seed. Nessa situação, a síndrome do pedaço faltante está fadada a ocorrer.

Os problemas do BT que identificamos neste trabalho levantam a importante necessidade de se modificar os mecanismos desse protocolo a fim de fazer com que os swarms operem em estados mais eficientes. Em particular, uma ideia que poderia ser investigada futuramente para minimizar a ineficiência causada pela má distribuição dos pedaços seria a implementação de mecanismos que fizessem com que os leechers permanecessem um tempo no sistema após o término do download. Essa política faria com que o número de seeds no sistema aumentasse, o que é equivalente a ter um único seed com capacidade aumentada. Conforme vimos anteriormente, quanto maior a capacidade do seed, maior é a probabilidade de que os leechers mais novos tenham pedaços a oferecer para os mais velhos. Isto melhoraria o desempenho do sistema como um todo e minimizaria a ocorrência de downloads heterogêneos.

Referências Bibliográficas

[1] "YouTube Facts & Figures". http://www.website-monitoring.com/blog/2010/05/17/youtube-facts-and-figures-history-statistics/, 2010.

[2] "Web site do FreeMeeting". 2011. http://land.ufrj.br/tools/fm.

[3] COHEN, B. "Mainline BitTorrent". 2011. http://www.bittorrent.com/.

[4] "Cooperative File System (CFS)". 2011. http://pdos.csail.mit.edu/chord/cfs.html.

[5] "PPLive". 2011. http://www.pptv.com/en.

[6] "IPOQUE Internet Study". 2007. http://www.ipoque.com/news_&_events/internet_studies/internet_study_2007.

[7] LEGOUT, A., LIOGKAS, N., KOHLER, E., et al. "Clustering and sharing incentives in BitTorrent systems", *SIGMETRICS Perform. Eval. Rev.*, v. 35, n. 1, pp. 301–312, 2007. ISSN: 0163-5999.

[8] BHARAMBE, A. R., HERLEY, C., PADMANABHAN, V. N. "Analyzing and Improving a BitTorrent Networks Performance Mechanisms". In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1 –12, apr. 2006.

[9] MURAI, F., FIGUEIREDO, D. R. "Formação de clusters em redes P2P por similaridade entre os nós". In: *XXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 729–740, Recife, PE, Maio 2009.

[10] MURAI, F., FIGUEIREDO, D. R. "Assortative mixing in BitTorrent-like networks". In: *Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, pp. 325–326, Piscataway, NJ, USA, 2009. IEEE Press. ISBN: 978-1-4244-3968-3. Extended Abstract.

[11] MURAI, F., ROCHA, A., FIGUEIREDO, D. R., et al. "Can identical BitTorrent peers experience different download times?" In: *IFIP WG 7.3 Performance 2010 Poster Abstracts*, pp. 29–30, Namur, Belgium, November 2010. Extended Abstract.

[12] HEI, X., LIANG, C., LIANG, J., et al. "A Measurement Study of a Large-Scale P2P IPTV System", *Multimedia, IEEE Transactions on*, v. 9, n. 8, pp. 1672 –1687, dez. 2007. ISSN: 1520-9210.

[13] QIU, D., SRIKANT, R. "Modeling and performance analysis of BitTorrent-like peer-to-peer networks". In: *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 367–378, New York, NY, USA, 2004. ACM. ISBN: 1-58113-862-8.

[14] FAN, B., CHIU, D.-M., LUI, J. "Stochastic Analysis and File Availability Enhancement for BT-like File Sharing Systems". In: *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pp. 30 –39, jun. 2006.

[15] YANG, X., DE VECIANA, G. "Service capacity of peer to peer networks". In: *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, v. 4, pp. 2242 – 2252 vol.4, mar. 2004.

[16] NEGLIA, G., PRESTI, G., ZHANG, H., et al. "A Network Formation Game Approach to Study BitTorrent Tit-for-Tat". In: Chahed, T., Tuffin, B. (Eds.), *Network Control and Optimization*, v. 4465, *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 13–22, 2007.

[17] LEGOUT, A., URVOY-KELLER, G., MICHIARDI, P. "Rarest first and choke algorithms are enough". In: *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 203–216, New York, NY, USA, 2006. ACM. ISBN: 1-59593-561-4.

[18] KUMAR, R., ROSS, K. "Peer-Assisted File Distribution: The Minimum Distribution Time". In: *Hot Topics in Web Systems and Technologies, 2006. HOTWEB '06. 1st IEEE Workshop on*, pp. 1 –11, nov. 2006.

[19] EZOVSKI, G., TANG, A., ANDREW, L. "Minimizing Average Finish Time in P2P Networks". In: *INFOCOM 2009, IEEE*, pp. 594 –602, apr. 2009.

[20] LO PICCOLO, F., NEGLIA, G. "The effect of heterogeneous link capacities in BitTorrent-like file sharing systems". In: *Hot-P2P*, pp. 40 – 47, oct. 2004.

[21] LIAO, W.-C., PAPADOPOULOS, F., PSOUNIS, K. "Performance analysis of BitTorrent-like systems with heterogeneous users", *Performance Evaluation*, v. 64, n. 9-12, pp. 876 – 891, 2007. ISSN: 0166-5316. Performance 2007, 26th International Symposium on Computer Performance, Modeling, Measurements, and Evaluation.

[22] CHOW, A., GOLUBCHIK, L., MISRA, V. "BitTorrent: An Extensible Heterogeneous Model". In: *INFOCOM 2009, IEEE*, pp. 585 –593, apr. 2009.

[23] MENASCHÉ, D. S., ROCHA, A., DE SOUZA E SILVA, E., et al. "Estimating self-sustainability in peer-to-peer swarming systems", *Performance Evaluation*, v. 67, n. 11, pp. 1243 – 1258, 2010. ISSN: 0166-5316.

[24] HAJEK, B., ZHU, J. "The missing piece syndrome in peer-to-peer communication". In: *IEEE ISIT*, pp. 1748 –1752, jun. 2010.

[25] ALBERT, R., BARABASI, A. L. "Statistical mechanics of complex networks", *Reviews of Modern Physics*, v. 74, n. 1, pp. 47–97, 2002. doi: http://dx.doi.org/10.1103/RevModPhys.74.47.

[26] NEWMAN, M. E. J. "The Structure and Function of Complex Networks", *SIAM Review*, v. 45, n. 2, pp. pp. 167–256, 2003. ISSN: 00361445.

[27] NEWMAN, M. E. J. "Mixing patterns in networks", *Phys. Rev. E*, v. 67, n. 2, pp. 026126, February 2003.

[28] J. A. CATANIA, T. J. COATES, S. K., FULLILOVE. "The population-based AMEN (AIDS in Multi-Ethnic Neighborhood) study", *Am. J. Public Health*, v. 82, pp. 284–287, 1992.

[29] MONGE, P. R., CONTRACTOR, N. "Theories of Communication Networks". pp. 223–224, Oxford University Press, USA, March 2003. ISBN: 0195160371.

[30] PARK, J., BARABÁSI, A.-L. "Distribution of node characteristics in complex networks", *Proceedings of the National Academy of Sciences*, v. 104, n. 46, pp. 17916–17920, 2007.

[31] BIANCHI, G. "Performance analysis of the IEEE 802.11 distributed coordination function", *IEEE Journal on Selected Areas in Communications*, v. 18, n. 3, pp. 535–547, mar. 2000. ISSN: 07338716.

[32] FILHO, L. J. H. *Algoritmos para Acesso Interativo em Aplicações de Vídeo P2P*. Tese de Mestrado, Universidade Federal do Rio de Janeiro/COPPE, set. 2009.

[33] "Web site do Tangram-II". `http://land.ufrj.br/tools/tangram2/tangram2.html`, jun 2011. [Último acesso: 20/04/2011].

[34] ROCHA, A., JAIME, G., MURAI, F., et al. "Novas evoluções integradas à ferramenta Tangram-II v3.1". In: *Salão de Ferramentas / XXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 33–40, Recife, PE, Maio 2009.

[35] E. DE SOUZA E SILVA, R. R. MUNTZ. "Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação". In: *VIII Escola de Computação*, 1992.

[36] XIA, R. L., MUPPALA, J. "A Survey of BitTorrent Performance", *IEEE Communications Surveys & Tutorials*, 2010. ISSN: 1553-877X.

[37] GUO, L., CHEN, S., XIAO, Z., et al. "A Performance Study of BitTorrent-like Peer-to-Peer Systems", *IEEE JSAC*, v. 25(1), pp. 155–169, 2007.

[38] PETERSON, L., BAVIER, A., FIUCZYNSKI, M. E., et al. "Experiences Building PlanetLab". In: *USENIX OSDI*, 2006.