Mecanismos de Compartilhamento de Recursos para Aplicações de Mídia Contínua na Internet

por

Carlo Kleber da Silva Rodrigues

UFRJ

Tese submetida para a obtenção do título de

Doutor em Ciências em Engenharia de Sistemas e Computação

ao Programa de Pós-Graduação de Engenharia de Sistemas e Computação da COPPE/UFRJ

por

Carlo Kleber da Silva Rodrigues

Abril 2006

MECANISMOS DE COMPARTILHAMENTO DE RECURSOS PARA APLICAÇÕES DE MÍDIA CONTÍNUA NA INTERNET

Carlo Kleber da Silva Rodrigues

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Rosa Maria Meri Leão, Dr.

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

Prof. Otto Carlos Muniz Bandeira Duarte, Dr. Ing.

Prof. Jussara Marques de Almeida, Ph.D.

Prof. Sérgio Vale Aguiar Campos, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2006

RODRIGUES, CARLO KLEBER DA SILVA

Mecanismos de Compartilhamento de Recursos para Aplicações de Mídia Contínua na Internet [Rio de Janeiro] 2006

XVIII, 152 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2006)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Mecanismos de Compartilhamento de Recursos
2. Técnicas de Compartilhamento de Banda
3. Vídeo sob Demanda
4. Transmissão de Mídia Contínua

I. COPPE/UFRJ II. Título (Série)

A Deus em primeiro lugar.

A minha esposa Mazinha e ao meu filho Carlo Dérik.

Agradecimentos

A Deus por tudo que sou e realizei.

A minha esposa Mazinha por ter assumido todas as tarefas pertinentes as nossas vidas que eram alheias à realização deste trabalho, por ter suprido o papel de pai perante nosso filho nas inúmeras ocasiões em que estive ausente, por sua compreensão em entender a importância que este trabalho teve para mim e assim ter aceitado com resignação os sacrifícios advindos, pela motivação que me deu diante dos obstáculos que tive de transpor, enfim por ser esta inestimável amiga e companheira ao longo de todos esses anos.

Ao meu filho Carlo Dérik. A sua simples existência já me faz feliz e renova minhas forças para continuar em busca da evolução.

Aos meus pais Carlos Walter (*in memorium*) e Ana Lúcia por terem me iniciado nos caminhos dos estudos. Ambos, cada um com seu jeito especial de ser, são para mim exemplos de trabalho, inteligência, confiança, perseverança, obstinação e compreensão.

Ao **Exército Brasileiro** e ao **IME** por terem me dado a oportunidade da realização deste trabalho. Ao Centro de Desenvolvimento de Sistemas **(CDS)** do Exército Brasileiro, onde atualmente trabalho, pelo apoio nos últimos onze meses para que eu pudesse concluir este trabalho. Neste contexto, agradeço também ao Programa de Engenharia de Sistemas e Computação da **UFRJ/COPPE** por toda a infra-estrutura acadêmica que me permitiu a realização deste trabalho.

À professora Rosa Maria Meri Leão **(UFRJ)** por ter sido, além de minha orientadora, uma grande incentivadora dos meus anseios acadêmicos. Com sua deter-

minação, conhecimento, paciência e inextinguível dedicação, me ensinou e iluminou os caminhos que tive de percorrer para a realização deste trabalho. Muito cresci intelectualmente e humanamente durante o tempo em que estive sob sua orientação.

A todos os meus professores pelos valiosos ensinamentos e, em especial, ao professor Edmundo Albuquerque de Sousa e Silva **(UFRJ)**, pelo exemplo de dedicação e comprometimento com a educação do nosso país. Ao meu supervisor acadêmico no **IME**, TC Paulo César Salgado Vidal, pela confiança que depositou em mim e irrestrito apoio que me deu ao longo desses anos de curso. E a professora Jussara Almeida **(UFMG)** por algumas importantes discussões e sugestões sobre serviço de vídeo sob demanda com interatividade.

A todos os colegas do grupo de trabalho LAND **(UFRJ)** pela amizade, apoio e paciência para me ajudar com os obstáculos e dificuldades com que me deparei. Muito pude aprender com todos eles. Em especial, sem desmerecimento dos demais colegas, agradeço ao Bernardo, Melba, Daniel Sadok, Ana, Guto, Guilherme (GD), Allyson, Flávio, Bruno, José Carlos, Fernando, Xiang, Hugo, Edmundo, Carolzinha, Boechat, Sidney e Diana. Também agradeço ao colega Marcus Vinícius **(UFMG)** com quem realizei discussões bem proveitosas e por ter cedido cargas sintéticas obtidas a partir de servidores multimídia reais que foram utilizadas neste trabalho.

Agradeço à secretária Carolina Vieira do LAND **(UFRJ)** que, além de seu primoroso trabalho profissional, se mostrou uma inestimável amiga, jamais deixando que quaisquer empecilhos fossem suficientemente grandes para impedir a sua ajuda. Por meio de suas ações, todos que a cercam recebem permanentes exemplos de solidariedade e caridade humana. Por último, mas não menos importante, também agradeço às bibliotecárias Selma Mendes e Raquel Porto da Biblioteca do **NCE (UFRJ)** pelo apoio na localização de várias referências utilizadas neste trabalho.

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

MECANISMOS DE COMPARTILHAMENTO DE RECURSOS PARA APLICAÇÕES DE MÍDIA CONTÍNUA NA INTERNET

Carlo Kleber da Silva Rodrigues

Abril/2006

Orientadora: Rosa Maria Meri Leão
Programa: Engenharia de Sistemas e Computação

Os mecanismos de compartilhamento de recursos são essenciais para tornar os sistemas de vídeo sob demanda escaláveis e, portanto, economicamente viáveis para amplo emprego em diversas atividades de nossa sociedade como, por exemplo, ensino, pesquisa e entretenimento. Neste escopo, como contribuições principais, esta tese apresenta um modelo analítico preciso e simples para calcular a distribuição de uso da banda do servidor e faz a proposta de três novas técnicas de compartilhamento de banda considerando a interatividade do cliente.

O modelo analítico desenvolvido considera o serviço de atendimento realizado pela clássica técnica *Patching* e aborda tanto o caso da transmissão de um único objeto como de múltiplos objetos. No caso do servidor de único objeto, obtém-se a distribuição modelada por meio de uma variável aleatória binomial, parametrizada pela taxa de chegada de requisições e duração do objeto. Já no caso do servidor de múltiplos objetos, mostra-se que a distribuição é dada pela soma de variáveis aleatórias binomiais independentes. Os experimentos conduzidos validam o modelo proposto e ilustram a importância do uso desta distribuição em projetos reais, enfatizando seu uso para avaliação e determinação da qualidade de serviço do sistema.

Em relação às três novas técnicas propostas, as duas primeiras são baseadas em *Patching* e a última no paradigma de HSM (*Hierachical Stream Merging*). Através de simulações e análises competitivas, evidencia-se o satisfatório nível de otimização de banda. Ainda, no escopo de interatividade, apresenta-se neste trabalho uma aproximação analítica para um importante limiar de tempo usado em decisões de união e abertura de fluxos e discute-se sobre outros limiares correlacionados. Por fim, também é investigado a importância do uso de *buffer* local no lado do cliente e o impacto de pequenas descontinuidades do serviço visando a otimização da banda.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

RESOURCE SHARING MECHANISMS FOR CONTINUOUS MEDIA APPLICATIONS OVER INTERNET

Carlo Kleber da Silva Rodrigues

April/2006

Advisor: Rosa Maria Meri Leão
Department: Computer and System Engineering

Resource sharing mechanisms are essential to make video on demand servers more scalable and therefore economically deployable for several modern society activities such as teaching, research and entertainment. In this scope, this thesis mainly presents a precise and simple analytical model to calculate the bandwidth usage server distribution and proposes three novel bandwidth sharing techniques that consider client interactivity.

The analytical model assumes the server operating with the classical Patching technique and studies the single object case as well as the multiple object case. In the former, it is shown that the bandwidth distribuition may be satisfactorily modeled as a binomial random variable; and in the latter, as a sum of independent binomial random variables. The experiments have validated the analytical model and outlined how important this distribution is for instance in real video on demand system designs.

As for the three novel techniques, the two first ones are based on the Patching scheme and the last on HSM (Hierarchical Stream Merging) protocols. Considerable bandwidth savings are observed through simulation and competitive analysis in several different real-case based scenarios. Still in the scope of interactivity, this thesis presents an analytical approximation for an important time threshold used for merging and opening new streams in the system and discusses over other related time thresholds. At last, the use of local buffer in the client side and the impact of service descontinuities on bandwidth optimization are both investigated in this work as well.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Visão Geral

UM serviço de vídeo sob demanda (VoD - *Video on Demand*) idealmente deve permitir que clientes, remotamente localizados, possam assistir no momento desejado a qualquer um dos objetos de um grande conjunto de objetos armazenados em um servidor central ou em um conjunto de servidores. Para tanto, os objetos são distribuídos para os clientes através de redes de comunicação. Este tipo de serviço tem uma grande diversidade de aplicações como, por exemplo, ensino a distância, filmes sob demanda, bibliotecas virtuais e *tele-shopping*. Usualmente, a caracterização deste serviço é feita com base nos aspectos mencionados a seguir.

1. Duração da sessão: um sistema de vídeo sob demanda deve ser capaz de atender igualmente sessões de longa e curta duração. Por exemplo, um filme típico tem duração de 90-120 min, enquanto que uma visita a uma biblioteca virtual tem uma duração imprevisível, podendo ser bem mais longa ou bem mais curta.

2. Requisitos de banda: os requisitos para as taxas de transmissão dos objetos estão usualmente entre 1.5 Mbps (MPEG-1) e 3-10 Mbps (MPEG-2) por fluxo iniciado. Assim, a banda total utilizada em operação contínua para transmissão dos vários objetos simultaneamente tende a ser significativa.

3. Interatividade: idealmente o serviço de VoD deveria ter condições de emular todas as ações disponíveis de um aparelho videocassete doméstico (i.e., ações VCR - *Video Cassette Recorder*) como, por exemplo, as ações de *Play* , *Stop*, *Fast Forwards*, *Rewind* e *Pause/Resume*. Entretanto, as propostas atuais têm mostrado que o custo/benefício, em termos da banda a ser utilizada, nem sempre se apresenta perfeitamente satisfatório para o emprego comercial. Daí, a solução adotada é, em sua maioria, a emulação parcial dessas ações.

4. Qualidade de serviço (QoS): o julgamento da qualidade de serviço reside principalmente nos pontos comentados a seguir. Primeiro, a latência do serviço, ou seja, a demora até o início da recepção do objeto. Segundo, a taxa de desistência, ou seja, a taxa com que os usuários abandonam o sistema por impaciência devido a demora para início de recepção do objeto. Terceiro, a qualidade percebida pelo cliente durante a visualização do objeto. E, por fim, a abrangência da emulação de interatividade.

A forma mais simples de atender os clientes que utilizam um serviço de VoD é através do escalonamento de um fluxo *unicast* para cada requisição dos clientes. O resultado é então um crescimento linear da banda do servidor. Como esta banda, contudo, é um recurso limitado, o uso de mecanismos de compartilhamento de recursos, mais especialmente, de técnicas de compartilhamento de banda, passa a ser notadamente indispensável para consecução de maior escalabilidade do sistema.

De maneira geral, as técnicas de compartilhamento de banda podem ser do tipo orientada a requisições de clientes ou técnicas de difusão (*broadcast*) periódica. As técnicas orientadas a requisições (p. ex., [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) são reativas no sentido de que transmitem dados em resposta às solicitações dos clientes. Os dados são transmitidos em *multicast* (implementado na camada IP ou de aplicação) e compartilhados por um grupo de clientes. Elas, em sua maioria, provêem serviço imediato e economizam o uso de banda por evitar transmissões redundantes de dados. As técnicas de difusão periódica (p. ex., [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]) garantem uma latência de serviço dentro de um intervalo de tempo estabelecido e são proativas no sentido de que realizam a transmissão periódica de segmentos do objeto independentemente da existência de solicitações de clientes. Mais precisamente, a

idéia é dividir o objeto em segmentos e transmití-los de forma cíclica em canais distintos. Enquanto o cliente exibe um determinado segmento, existe a garantia de que o próximo segmento é recebido em tempo hábil para exibição contínua.

As técnicas de compartilhamento de banda orientadas a requisições são usualmente avaliadas de acordo com os valores médios de requisitos de banda associados. Pesquisas anteriores [22, 23, 4, 5, 24, 6, 25] incluem discussões sobre a métrica banda média que é definida como a banda necessária para o servidor atender as requisições de clientes para um dado objeto. No entanto, a banda média pode não fornecer uma estimativa precisa para dimensionar o servidor. Isto é evidenciado em cenários onde a banda real requerida possui significativa variância. Considerando o modelo de acesso seqüencial dos clientes, um limite (piso) teórico para a banda do servidor foi apresentado em [5, 26] para as técnicas orientadas a requisições: crescimento logarítmico com a taxa de chegadas de requisições.

Já as técnicas de difusão periódica são comumente avaliadas de acordo com o número de canais que são utilizados para transmitir periodicamente os segmentos de um mesmo objeto e a latência de serviço exigida [13, 27]. Naturalmente, quanto menor for o número de canais necessários e menor o tempo de latência, mais eficiente é a técnica. No entanto, diminuir o número de canais e também simultaneamente a latência ocasionada não é algo trivial de se conseguir. Os pesquisadores buscam então estabelecer um compromisso entre a satisfação do cliente (máxima latência tolerada) e a performance absoluta (mínimos requisitos de banda) [13]. Considerando o modelo de acesso seqüencial dos clientes, um limite (piso) teórico para a banda do servidor foi apresentado em [26, 27, 13], onde mostrou-se que existe um crescimento logarítmico com o inverso da latência de serviço.

Estudos anteriores ainda mostram que, para objetos de popularidade alta, ou seja, objetos que são acessados muito freqüentemente, as técnicas de compartilhamento do tipo difusão periódica são mais eficientes, em termos de economia de banda, que as técnicas orientadas a requisições de clientes e, quando a popularidade é baixa ou média, as técnicas orientadas a requisições de clientes é que são mais eficientes [13, 22].

Independentemente de serem do tipo orientada a requisições ou de difusão periódica, as técnicas que consideram a interatividade do cliente, ou seja, o acesso não-seqüencial do cliente (p. ex., [28, 29, 30]), evoluíram quase que em paralelo com as técnicas que admitiam apenas o acesso seqüencial do cliente (p. ex., [2, 7, 13]). Mais precisamente, no ano de 1994, houve a proposta formal da primeira técnica sem interatividade, a qual era do tipo orientada a requisições e recebeu o nome de *Batching* [31, 32, 33]. Nessa mesma época, também houve a primeira proposta formal de técnica com interatividade por meio do trabalho de Banker et al. [34], a qual era baseada em difusão periódica e introduziu o conceito de *sintonia* nos servidores de VoD então existentes. A partir de então, o desenvolvimento das técnicas de acesso seqüencial e não-seqüencial terminou tornando-se um só na realidade e, inevitavelmente, os conceitos e soluções que se mostravam eficientes no desenvolvimento de uma dada técnica passavam, quando possível, a serem utilizados em outra, independentemente do modelo de acesso do cliente originalmente considerado.

1.2 Motivações e Contribuições

Esta tese tem as seguintes principais contribuições:

- **1)** Propomos [35, 36, 37] um modelo analítico simples para calcular a distribuição de uso da banda do servidor para a técnica *Patching* [2]. Consideramos tanto o caso do servidor de único objeto como o de múltiplos objetos.

A técnica *Patching* é um esquema orientado a requisições e para o acesso seqüencial do cliente. Trabalhos recentes (p. ex., [38, 39, 40, 41, 42, 43, 44, 45, 46]) sobre esta técnica têm sido apresentados na literatura. Por exemplo, os autores de [39] empregam a técnica *Patching* com serviço *peer-to-peer*. Já os autores de [38] fazem a integração de *Patching* com esquemas de *proxy caching*. A simplicidade extrema e a eficiência competitiva, em termos de otimização de banda do servidor, são os principais atrativos desta técnica. A simplicidade é atestada principalmente em [47], onde é feita uma análise de complexidade de técnicas orientadas a requisições. O estudo é baseado na informação que o servidor precisa armazenar e no trabalho que

o mesmo desempenha para o tratamento de cada requisição feita pelo cliente. De forma simples, quando uma requisição chega ao servidor pode ser necessário alterar a duração de um fluxo *multicast* já em andamento no sistema e/ou os clientes que escutam um fluxo *multicast*. Como conseqüência, o servidor precisa então atualizar o estado de um ou mais fluxos *multicast* e enviar mensagens para os clientes afetados. Um dos resultados obtidos é, pois, a verificação da significativa simplicidade da técnica *Patching* em relação a outros esquemas orientados a requisições. Já a eficiência competitiva é verificada, por exemplo, em [6, 8, 40], onde observa-se que os requisitos de banda média do servidor são bem semelhantes àqueles de técnicas mais complexas, considerando cenários envolvendo objetos de popularidade moderada e média.

Como mencionamos antes, valores médios de banda nem sempre provêem estimativas reais dos requisitos de banda do servidor e, portanto, não devem ser utilizados isoladamente para assegurar um determinado nível de QoS para os clientes. Com isto em mente, Bar-Noy et al. [47] foram os primeiros a medir a performance de técnicas orientadas a requisições usando outras métricas diferentes de valores médios. Eles utilizaram a banda máxima e a distribuição de uso da banda para analisar competitivamente as técnicas *Fibonacci Tree* [8, 48], Diádica [9] e ERMT (*Earliest Reachable Merge Target*) [6]. Nesse trabalho fica evidenciado a satisfatória performance da técnica Diádica em contraposição a uma melhor performance do esquema ERMT à custa de uma maior complexidade do sistema. Contudo, nenhum resultado analítico é derivado em relação as métricas consideradas e todas as conclusões se baseiam em resultados de simulações.

A qualidade de serviço de técnicas orientadas a requisições também foi avaliada em [46, 25]. De Souza e Silva et al. [46] propuseram um algoritmo para computar a distribuição da banda requerida pela técnica *Patching*. O modelo analítico proposto estima a distribuição da banda requisitada em uma janela de tempo ótima. É mostrado que os fluxos de dados iniciados em uma janela são uma boa estimativa dos fluxos de dados efetivamente transmitidos em uma janela (i.e., tudo que é iniciado em uma janela é transmitido dentro da própria janela) quando na estacionariedade do sistema (i.e., tempo $\rightarrow \infty$). O custo computacional do algoritmo é $O(P^2)$, onde

P é uma função do erro estimado. Os resultados indicam que a probabilidade de se exceder o valor médio da banda do servidor é bem significativa. Nenhuma avaliação, contudo, é feita para o caso da transmissão de múltiplos objetos.

Em [25], modelos analíticos aproximados foram desenvolvidos para analisar a qualidade de serviço das seguintes três concepções orientadas a requisições: *Patching*, *Bandwidth Skimming* [7] e HSM (*Hierarchical Stream Merging*) [6]. Os autores utilizaram um modelo de rede de filas fechado para estimar as métricas tempo médio de espera e fração de clientes que abandonam o sistema sem receber atendimento, pelo fato do servidor estar temporariamente sobrecarregado (i.e., taxa de desistência ou *balking rate*). Os resultados são obtidos por meio de solução iterativa de um conjunto de equações. Como um dos principais resultados, é mostrado que configurações de servidores, baseadas na banda média, podem ter associadas taxas de desistência bem significativas e não toleráveis.

Conforme vemos, a análise da técnica *Patching* feita até o presente é baseada na resolução de uma equação [46] ou em conjunto de equações [25]. Além disso, em [46] apenas o caso de único objeto é considerado, e em [25] apenas o caso de múltiplos objetos é analisado. A nossa primeira contribuição é, pois, a proposta de um modelo analítico simples para calcular a distribuição do uso da banda do servidor para técnica *Patching*, considerando tanto o caso de único objeto como de múltiplos objetos. Mais precisamente, fazemos a derivação de uma expressão analítica fechada para representação da distribuição de uso da banda do servidor no caso de único objeto e, a partir daí, obtemos a distribuição para o caso de múltiplo objetos por meio da Transformada Rápida de Fourier [49]. Através de experimentos em diversos cenários, também mostramos como utilizar esta distribuição para assegurar níveis de QoS desejados.

- **2)** Propomos [35, 50, 51] três novas técnicas de compartilhamento de banda para servidores de vídeo com interatividade: *Patching* Interativo Eficiente - PIE, *Patching* Interativo Completo - PIC e *Merge* Interativo - MI.

Devido aos resultados já alcançados e apresentados na literatura (Capítulo 2), aliados ao desenvolvimento tecnológico dos sistemas de VoD [52, 53] e ao desejo es-

pontâneo do cliente em ter disponível serviços interativos, acreditamos que o modelo de acesso não-seqüencial do cliente se estabelece, a partir de agora, como aquele a ser mais preferencialmente utilizado em novas propostas. Temos então, como nossa segunda contribuição, a apresentação de três novas técnicas para o serviço com interatividade: *Patching* Interativo Eficiente - PIE, *Patching* Interativo Completo - PIC e *Merge* Interativo - MI. Para as duas primeiras, baseamo-nos na concepção do clássico *Patching* [2, 54, 22, 55] e, para a última, nos esquemas de HSM (*Hierarchical Stream Merging*) [5, 6, 7].

O emprego do paradigma de *Patching* para a implementação das duas primeiras técnicas deu-se pela sua simplicidade e performance competitiva, conforme comentamos anteriormente. Estas duas técnicas utilizam-se parcial ou integralmente de uma abordagem inédita de seguintes critérios: (1) a união de fluxos *multicast* com um fluxo recém iniciado que esteja transmitindo uma unidade (bloco) de dados posterior do objeto; (2) a localização de um novo fluxo *multicast* para os clientes que por alguma razão não puderam se unir ao fluxo alvo do escalonamento anterior; (3) a ação *espontânea* do servidor para realizar uniões de fluxos *multicast* que estejam em andamento no sistema.

Já no caso da técnica MI, o principal estímulo para usarmos o paradigma de HSM foi a sua eficiência em potencial. Esta eficiência se dá pela característica de pesquisa *exaustiva* e possivelmente *irrestrita* visando o compartilhamento de dados e, ainda, pelo fato de todos os fluxos abertos no sistema serem *multicast* e, assim, poderem ser fluxos alvos. Entretanto, diferentemente de *Patching* que possui estruturas de união de fluxos representadas por árvores de no máximo dois níveis, em esquemas de HSM temos árvores sem limites de profundidade. Isto pode ocasionar um alto número de mensagens trocadas entre clientes e servidor e, por conseqüência, implica um esquema de maior complexidade de sistema. Neste caso a discussão se volta não somente para a performance em termos de otimização de banda conseguida, mas também para a avaliação do nível de complexidade introduzido no sistema.

- **3)** Obtemos [35, 50, 51] uma aproximação analítica para um limiar de tempo usado em decisões de união e abertura de fluxos para as técnicas PIE e PIC.

  Muito embora estudos analíticos para derivação de limiares de tempo já te-

nham sido realizados para acesso seqüencial (p. ex., em [22, 55, 5]), esta é a primeira vez, salvo engano, que uma derivação analítica formal é realizada para o acesso não-seqüencial. Também, experimentalmente, determinamos patamares de valores a serem atribuídos para outros limiares de tempo correlacionados em diferentes cenários de interatividade.

- **4)** Analisamos [56, 51] a necessidade do emprego de um limiar de tempo para decisões de união e abertura de fluxos na técnica MI.

  A motivação para esta análise reside no fato de que várias técnicas de acesso seqüencial, baseadas em HSM, se mostram afeitas à utilização de um limiar de tempo para realizar abertura de novos fluxos e seleção de fluxos alvos (p. ex., [8, 9, 57, 10, 58, 11, 59]). Com este limiar, busca-se a otimização em termos de banda do servidor. Daí, a necessidade de verificação da possibilidade de uma otimização, por meio de um limiar de tempo, também decorre naturalmente em um cenário com interatividade.

- **5)** Investigamos [50, 56, 51] a importância do uso de *buffer* local no lado do cliente para fins de evitar que dados já armazenados necessitem ser solicitados novamente.

  Fazemos análises tanto para a banda do servidor quanto para a banda do cliente. Diferentemente de trabalhos anteriores que conjugam a proposta da técnica de compartilhamento de banda com o uso do *buffer* (Capítulo 2), nossa contribuição é um estudo que explicita vantagens/desvantagens advindas do emprego do *buffer*, independentemente da técnica que esteja sendo utilizada.

- **6)** Estudamos [56, 51] o impacto da introdução de pequenas descontinuidades no serviço de VoD oferecido para fins de otimização da banda do servidor.

  A descontinuidade está atrelada ao grau de tolerância do cliente. Naturalmente esta tolerância varia de acordo com o perfil psicológico do cliente e do tipo de aplicação em si. Tendo ciência disto, realizamos estudos experimentais com diversos valores de descontinuidades para quantificar a otimização possível de ser conseguida.

- **7)** Por meio das propostas feitas e das análises e estudos realizados neste trabalho, estabelecemos [56, 51] por decorrência um comparativo detalhado em relação à eficiência que pode ser alcançada pelo emprego das concepções dos paradigmas de *Patching* e de HSM na implementação de novas técnicas para aplicações de VoD com interatividade.

1.3 Organização da Tese

Cinco capítulos constituem o texto desta tese, incluindo este primeiro de introdução. O objetivo deste capítulo inicial é dar uma visão geral do assunto, evidenciar sua importância no cenário atual de pesquisa, destacar as motivações, mencionar as contribuições e, por fim, apresentar a organização do texto de forma global, conforme fazemos agora.

O Capítulo 2 realiza uma revisão bibliográfica sobre as principais técnicas orientadas a requisições do cliente com acesso seqüencial e sobre as técnicas desenvolvidas para interatividade. O objetivo é informar sobre a evolução ocorrida ao longo dos anos, permitindo um conhecimento do estado da arte destas áreas de conhecimento, além também de introduzir aspectos teóricos básicos necessários para compreensão do texto.

No Capítulo 3 apresentamos a derivação da distribuição de uso da banda do servidor de VoD considerando o emprego da técnica *Patching*. Para tanto, inicialmente introduzimos conceitos e derivações analíticas de resultados conhecidos da clássica técnica *Patching* para, em seguida, realizarmos a derivação da distribuição proposta. Também neste capítulo incluímos resultados de simulações que validam o modelo que desenvolvemos, além de resultados de experimentos que realizamos em diversos cenários para exemplificar o uso da distribuição para obtenção de diferentes níveis de qualidade de serviço.

O Capítulo 4 inicia com a apresentação da forma mais comum de atendimento de requisições em cenários com interatividade. Em seguida, faz-se uma revisão dos algoritmos de operação de duas técnicas relativamente recentes da literatura. Por fim,

as novas técnicas propostas são apresentadas e faz-se a derivação analítica de um dos limiares de tempo empregados para união e abertura de fluxos no sistema. Neste capítulo, incluímos ainda os estudos experimentais conduzidos por meio de simulações em diferentes cenários de interatividade. Estes estudos objetivam, principalmente, a avaliação de performance das técnicas propostas, a obtenção de conclusões sobre o emprego de *buffer* local no cliente, a avaliação da expressão analítica do limiar de tempo que realizamos, a determinação de valores ideais para os demais limiares de tempo empregados pelas técnicas e, por último, a quantificação da influência de descontinuidades do serviço na banda do servidor.

O Capítulo 5 conclui esta tese, apresentando uma breve síntese, a qual relembra as motivações, as contribuições e as constatações então advindas. Também existe ainda, ao final desse capítulo, uma menção sobre os possíveis direcionamentos para trabalhos futuros. Por fim, os Apêndices A e B trazem algumas figuras e tabelas de experimentos realizados.

Capítulo 2

Revisão Bibliográfica

2.1 Introdução

ESTE capítulo traz uma visão geral sobre a evolução ocorrida ao longo dos anos com as principais técnicas orientadas a requisições, considerando acesso seqüencial, e também com as principais técnicas desenvolvidas para interatividade, i.e., acesso não-seqüencial. Isto permite-nos um conhecimento do estado da arte dessas áreas de conhecimento, além também de introduzir os aspectos teóricos básicos necessários para compreensão do texto.

Muito embora tenhamos mencionado que o desenvolvimento de técnicas para acesso seqüencial e não-seqüencial tornou-se praticamente um só, preferimos fazer esta exposição em duas seções independentes para efeito de maior clareza e facilidade de entendimento. Precisamente, na Seção 2.2 tratamos das técnicas orientadas a requisições com acesso seqüencial, e na Seção 2.3 discutimos sobre as técnicas para interatividade.

2.2 Técnicas Orientadas a Requisições

Conforme já comentamos, no ano de 1994, houve a primeira proposta formal de técnica orientada a requisições: *Batching* [31, 32, 33]. Nesta técnica, no momento da

chegada da primeira requisição para um dado objeto, uma janela de tempo é aberta. Todas as outras requisições que chegam no limite desta janela e para o mesmo objeto são agrupadas. Ao final da janela, todas as requisições são então servidas através de um único fluxo *multicast*. Na chegada da próxima requisição, após o limiar de tempo estabelecido pela janela, o processo é então reiniciado. A otimização de uso da banda é efetivamente conseguida. Mas, por outro lado, há a introdução de uma latência de serviço correspondente, no pior caso, ao tamanho da janela projetada.

Um pouco mais tarde, foi feita a apresentação da técnica denominada *Piggybacking* [1, 60, 61, 62]. É um técnica que pode ser implementada com serviço imediato, i.e., não apresenta latência de serviço como em *Batching*. Sua idéia base é alterar dinamicamente as taxas de exibição de fluxos já em andamento e relativos a um mesmo objeto, de tal sorte a permitir que um fluxo alcance o outro e então um deles possa ser extinto. Uma das principais limitações desta técnica é a exigência de um *hardware* adequado para suportar as alterações das taxas dinamicamente. Uma outra limitação é a variação máxima na taxa de exibição que deve estar dentro de aproximadamente 5% da taxa normal. Isto para evitar que ocorra uma deterioração perceptível na qualidade da exibição do objeto [53, 23]. Esta condição naturalmente limita o número de fluxos possíveis de serem unidos.

No ano de 1998 houve propostas de técnicas muito semelhantes entre si: *Stream Tapping* [54] e *Patching* [2]. Por proverem serviço imediato, estas técnicas também solucionam o problema da latência de serviço presente na técnica *Batching*. A versão otimizada da técnica *Patching* foi apresentada em [23, 22]. Descrevemos a seguir sua idéia base e deixamos implícito, salvo mencionado diferentemente, que, ao mencionarmos à técnica *Patching*, estamos nos referindo a sua versão otimizada.

Na chegada da primeira requisição (cliente) para um dado objeto, um fluxo *multicast* é aberto, provendo assim serviço imediato. A partir daí, novas chegadas para o mesmo objeto e dentro de um determinado limiar de tempo, denominado janela ótima, passam também a escutar o fluxo *multicast* em andamento (armazenando em *buffer* local para posterior exibição) e recebem, ao mesmo tempo, a parte perdida inicialmente, denominada *patch*, através de fluxos *unicast* (fazendo exibição imediata). Após o recebimento da parte perdida, os clientes escutam apenas o fluxo

multicast originado pela primeira requisição. A exibição passa a ser feita a partir da informação previamente armazenada em *buffer*, e os correspondentes fluxos *unicast* são extintos. Tanto o fluxo *multicast* como os fluxos *unicast* transmitem dados na mesma taxa de exibição do objeto. Note que o *buffer* local do cliente é usado para fins de sincronismo, garantindo uma exibição contínua dos dados recebidos simultaneamente a partir dos fluxos *multicast* e *unicast*, e ainda que a banda de cada cliente corresponde a duas vezes a taxa de exibição do objeto. Por fim, a primeira chegada (requisição), após a janela de tempo ótima, reinicia o processo.

A Figura 2.1(a) traz um exemplo da operação da técnica *Patching*. Neste exemplo, temos um objeto de tamanho total de 10 unidades de tempo e as chegadas (requisições) obedecendo a seguinte disposição: $(t_0, t_1, t_2, t_3) = (0, 1, 3, 4)$, i.e., a primeira chegada ocorre no tempo 0, e a segunda, terceira e quarta chegadas nos tempos $1, 3$ e 4, respectivamente. A primeira chegada inicia um fluxo *multicast* S_0, e as demais chegadas, dado que ocorrem na janela de tempo W, abrem individualmente fluxos *unicast* (P_1, P_2, P_3) para receber as partes perdidas inicialmente, ou seja, os *patches*. Note que a concepção da técnica em si também permite-nos visualizar a estrutura de união de fluxos como uma floresta de árvores de no máximo dois níveis. Por exemplo, a árvore de união de fluxos da Figura 2.1(b) é uma forma alternativa de observar o mesmo processo descrito na Figura 2.1(a). Os rótulos dos nós representam os tempos em que os fluxos são respectivamente iniciados, e a relação pai e filho entre os nós informa a hierarquia de escuta que os fluxos possuem. A partir da árvore mostrada, podemos afirmar que os fluxos iniciados em t_1, t_2, t_3 escutam o fluxo iniciado em t_0, e ainda que apenas o fluxo iniciado em t_0 (raiz) é *multicast*, enquanto que todos os outros são *unicast*. Como dito anteriormente, as árvores possuem no máximo dois níveis, pois todas as chegadas que ocorrem dentro da janela ótima sempre escutam o fluxo relacionado à raiz da árvore. A primeira chegada, após a janela ótima, reinicia o processo criando uma nova árvore. Na Seção 3.2 apresentamos formulações analíticas referentes ao cálculo da banda média e ao tamanho da janela ótima da técnica *Patching*.

Um pouco mais tarde, houve uma proposta de extensão da técnica *Patching*. Esta extensão ficou conhecida como *Transition Patching* [3] e tem como idéia base

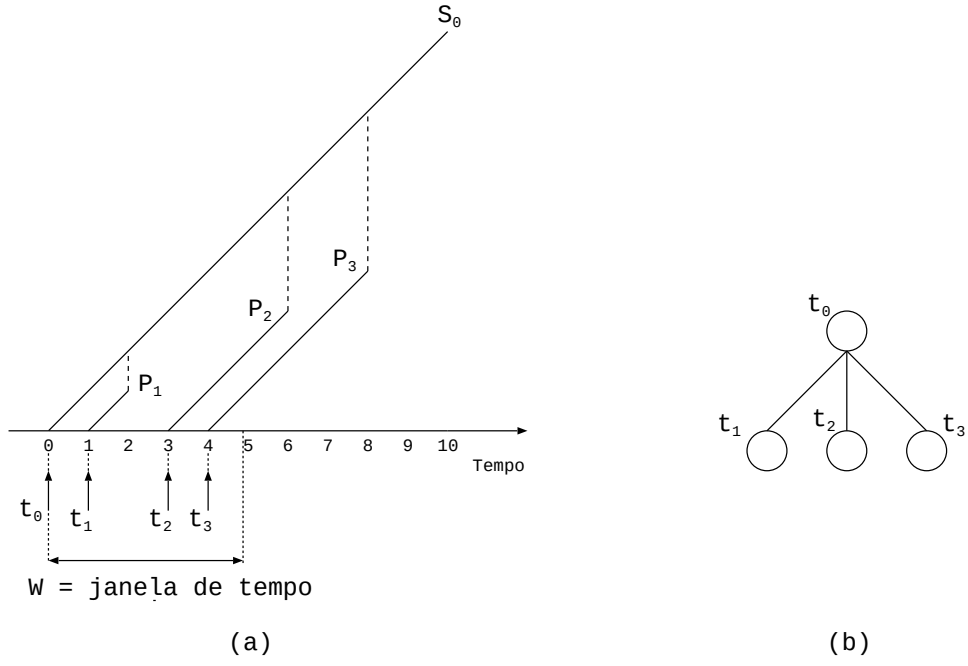


Figura 2.1: (a) Diagrama de união de fluxos; (b) Árvore de união de fluxos.

a possibilidade da árvore de união de fluxos aceitar até três níveis de profundidade. Na prática, isto significa dividir uma janela ótima W_r (semelhante a da técnica *Patching* original) em subjanelas ótimas W_t de tal sorte a permitir que, associado a cada uma destas subjanelas, excetuando-se apenas a primeira, exista um fluxo *multicast* que possa ser compartilhado por todos os clientes pertencentes a mesma. A determinação das subjanelas W_t e da própria janela ótima principal W_r não é, contudo, de solução imediata e baseia-se na atribuição experimental de diversos pares de valores (W_r, W_t) para minimizar a banda média resultante. Os resultados dos experimentos realizados de fato mostram uma melhor performance de *Transition Patching* em relação à técnica *Patching*, mas não confirmam contundentemente que a relação otimização da banda versus custo de implementação seja satisfatória.

Também por volta de 1998, houve a proposta de uma técnica que consistia na adaptação do esquema de difusão periódica *Skyscraper Broadcast* [15] para um esquema que tivesse a característica de reagir em face das requisições dos clientes. Esta nova proposta foi denominada *Dynamic Skyscraper* [63] e teve, como uma das motivações básicas, a condição de que a popularidade dos objetos, ou seja, o número médio de requisições para os objetos, pode variar dinamicamente e atingir valores relativamente baixos e moderados. Daí, a alocação estática de canais para os

objetos, característica dos esquemas de difusão periódica, conforme comentamos no Capítulo 1, pode não ter a eficiência esperada em aplicações reais. De forma sucinta, o serviço imediato nesta técnica é conseguido por meio de um método denominado *channel stealing*, que consiste em dinamicamente realocar um grupo de canais, que não tenha clientes fazendo escuta, para atender as requisições que chegam ao sistema e não encontram um grupo de canais já transmitindo o objeto requisitado. Se não houver um grupo de canais disponível, as requisições comuns (que não encontram um grupo de canais transmitindo o objeto requisitado) podem então entrar em uma fila para atendimento posterior. Uma otimização desta técnica foi sugerida em [5]. Em síntese, esta otimização consiste em uma alteração do tamanho original dos segmentos do objeto, transmitidos nos diferentes canais, e em uma implementação específica do particionamento proposto em [24]. Os resultados dos experimentos realizados com esta técnica mostram uma eficiência, em termos de otimização de banda, inferior a da técnica *Patching* original para objetos de popularidade baixa-média, e superior para objetos de popularidade média-alta [5].

Ainda em 1998, houve o início das propostas que ficaram conhecidas como técnicas baseadas em HSM ou simplesmente técnicas HSM (*Hierarchical Stream Merging*). Assim como na concepção de *Patching*, estas técnicas também provêem serviço imediato e baseiam-se no compartilhamento de fluxos já em andamento para atendimento das requisições que chegam ao sistema. A principal diferença para a técnica *Patching* é que todos os fluxos iniciados são *multicast* e a árvore de união de fluxos não está mais limitada a apenas dois níveis. Admitindo que os fluxos abertos transmitam na mesma taxa de exibição do objeto requisitado e que a banda do cliente corresponda a duas vezes esta taxa, explicamos o princípio básico de sua operação a seguir.

O atendimento de uma requisição do cliente consiste em abrir um fluxo *multicast* para este cliente e simultaneamente fazê-lo escutar (armazenando em *buffer* local para posterior exibição) um outro fluxo *multicast* iniciado anteriormente a sua chegada no sistema. Como o passar do tempo, é possível que o fluxo originalmente aberto pelo cliente não seja mais necessário pelo fato dele (o cliente) já ter em *buffer* a informação que seria ainda transmitida por este fluxo. Neste caso, o fluxo origi-

nalmente aberto pelo cliente é extinto e o cliente pode escutar apenas o fluxo que já estava em andamento no sistema quando se deu a sua chegada. Este processo de escutar fluxos anteriormente iniciados, ou seja, que estão transmitindo informações à frente da desejada, pode ser repetido indefinidamente, resultando em uma união hierárquica de fluxos em contraposição ao esquema de *Patching*, onde a união de fluxos só ocorre uma única vez para cada cliente, conforme explicamos anteriormente.

Na Figura 2.2(a) temos um exemplo de operação de uma técnica baseada em HSM. O objeto tem tamanho total de 10 unidades de tempo e as chegadas (requisições) obedecem à seguinte disposição: $(t_0, t_1, t_2, t_3) = (0, 1, 3, 4)$, i.e., a primeira chegada ocorre no tempo 0, e a segunda, terceira e quarta chegadas nos tempos $1, 3$ e 4, respectivamente. Cada uma das chegadas inicia um fluxo *multicast* independente e recebe os dados conforme explicado a seguir. A chegada em t_0 recebe o objeto completo através do fluxo S_0, que é o fluxo associado à raiz da árvore. A chegada em t_1 recebe dados tanto de S_1 como de S_0 simultaneamente por uma unidade, e então passa a receber apenas de S_0 por oito unidades. A chegada em t_2 recebe dados de S_2 por duas unidades, e então passa a receber simultaneamente dos fluxos S_2 e S_0 por três unidades, e finalmente passa a receber dados apenas de S_0 por mais duas unidades. A chegada em t_3 recebe dados dos fluxos S_2 e S_3 simultaneamente por uma unidade, então passa a receber dados de S_2 e S_0 simultaneamente por três unidades, e então, finalmente, recebe dados apenas de S_0 por mais duas unidades. Observe que a árvore possui três níveis. É importante mencionar que a decisão de que fluxo escutar e por quanto tempo escutar depende da técnica em si. Ou seja, para uma mesma disposição de chegadas, a árvore final de união de fluxos não é necessariamente a mesma se considerarmos o emprego de diferentes técnicas HSM.

A idéia geral das técnicas HSM consiste, portanto, em definir uma seqüência de árvores de união de fluxos para atender as requisições que chegam ao sistema. Daí, existem dois importantes aspectos a considerar. Primeiro, a decisão de que fluxo escutar e por quanto tempo escutar. Segundo, a definição da janela de tempo que compreende o início do fluxo raiz até o instante a partir do qual uma chegada necessariamente inicia uma nova árvore. Estes dois aspectos distinguem as várias propostas.

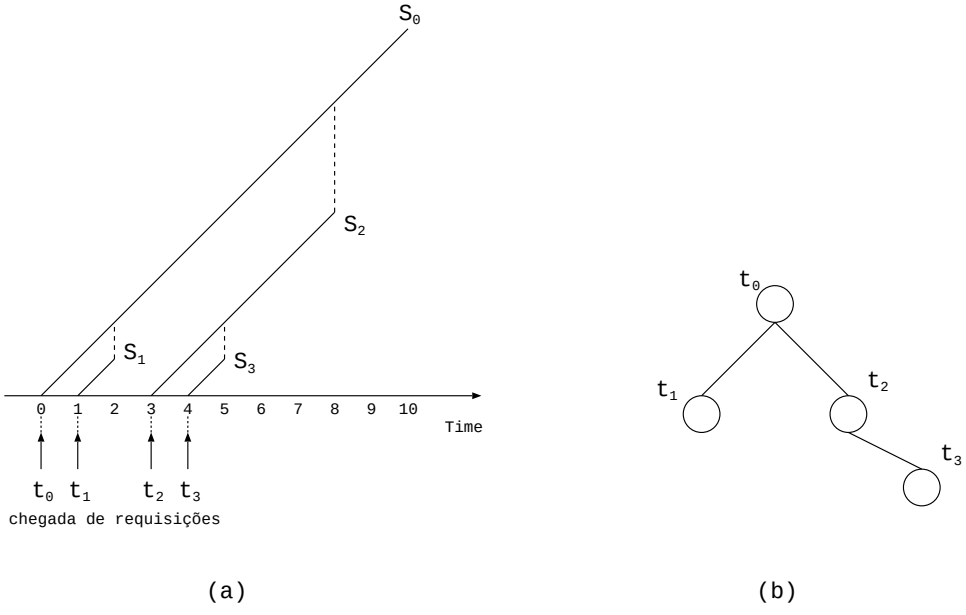


Figura 2.2: (a) Diagrama de união de fluxos; (b) Árvore de união de fluxos.

As técnicas HSM originaram-se com os trabalhos de Eager, Vernon and Zahorjan [5, 6, 7]. Destas primeiras propostas, a de mais simples operação chama-se *Closest Target* [6]. Nesta técnica um fluxo *multicast* é aberto para atender a solicitação do cliente, e o fluxo a ser simultaneamente escutado (armazenado em *buffer*) é aquele que está transmitindo a unidade de dados mais próxima daquela desejada pela requisição, mesmo que não exista tempo suficiente para que ele seja alcançado, ou seja, mesmo que o fluxo termine antes de ser alcançado. Não há, portanto, imposição de qualquer limiar de tempo. Os experimentos realizados mostram que a técnica *Closest Target* possui uma eficiência satisfatória, em termos de economia de banda, quando comparada com outras técnicas orientadas a requisições e, inclusive, com um escalonamento ótimo *off-line* [6]. A Seção 3.2 discute mais detalhes desta técnica.

Com o passar dos anos, seguiram-se outras propostas de técnicas baseadas em HSM (p. ex., [8, 9, 57, 10, 58, 11, 59]). Destas últimas, podemos destacar a técnica Diádica (*Dyadic Technique*) [9, 64] pela eficiência competitiva e, principalmente, pela forma de definição da árvore de união de fluxos, a qual permite uma avaliação analítica mais rigorosa que as já feitas até então para as técnicas HSM. A eficiência competitiva desta técnica é inclusive ratificada no trabalho posterior de Wong et al [11]. Os autores usam a noção de razão competitiva [65] para as métricas banda

máxima e banda total, respectivamente. A banda máxima refere-se ao número máximo de fluxos necessários para transmissão do objeto, e a banda total refere-se à duração total dos fluxos para transmissão do objeto [58, 59]. A razão competitiva é a razão entre o valor da métrica obtido experimentalmente (ou analiticamente) e aquele conseguido por meio de um escalonamento ótimo *off-line*.

A janela de tempo da técnica Diádica, a qual compreende o início do fluxo raiz até o instante a partir do qual uma nova chegada necessariamente inicia uma nova árvore, é igual à metade do tamanho total do objeto. Esta janela é então dividida em intervalos diádicos da forma explicada a seguir. Seja x o instante em que o fluxo da raíz é iniciado e T o tamanho total do objeto. A técnica é aplicada no intervalo $(x, y]$, onde $y = x + \frac{T}{2}$. O intervalo diádico i é denotado por I_i. O fluxo mais recente de cada intervalo diádico torna-se filho do fluxo relativo à raiz da árvore, e o processo é então aplicado recursivamente para cada filho com a finalidade de obter as subárvores enraizadas no mesmo. A Figura 2.3 ilustra os intervalos diádicos, onde $\alpha = 2$ é o fator base numérico utilizado para subdivisão destes intervalos. Tam et al. [47] propuseram uma variante deste esquema original que divide os intervalos pela razão dourada (*golden ratio*) $\phi = \frac{1+\sqrt{5}}{2}$. Em outras palavras, faz-se $\alpha = \phi$. O motivo para utilizar a razão dourada está no fato de ser possível demonstrar que este valor maximiza a eficiência do algoritmo (i.e., valor ótimo) [47].

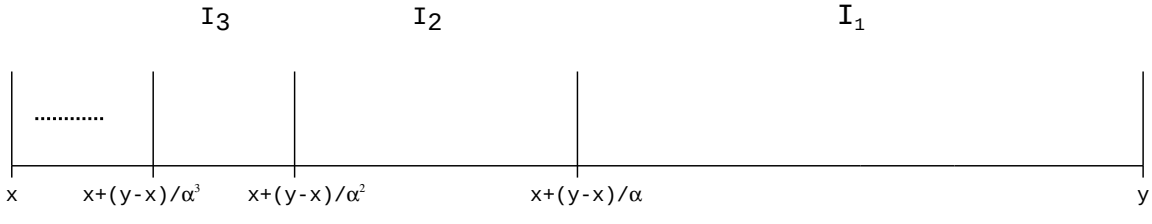


Figura 2.3: Intervalos diádicos.

Do ponto de vista de economia de banda do servidor, os resultados da literatura mostram que as técnicas HSM têm melhor desempenho que as técnicas baseadas nas concepções de *Patching*, *Batching*, *Piggybacking* e *Dynamic Skyscraper*. O ponto de discussão, contudo, volta-se para a complexidade de implementação e de gerência do próprio sistema. Isto porque a estrutura de união de fluxos das técnicas baseadas em HSM é potencialmente uma floresta de árvores sem limite de profundidade, impac-

tando diretamente na complexidade do número de mensagens trocadas entre clientes e servidor [41]. Daí, a depender da aplicação e do cenário de atendimento considerado, a eficiência absoluta então atingida pode não justificar o nível de complexidade associado. Uma discussão detalhada sobre técnicas HSM pode ser encontrada em Tam et al [47].

Por último, por volta de 2003, houve mais duas propostas de extensão da técnica *Patching*: *Two-Level Patching* [41] e *Recursive Patching* [43]. A idéia geral destas propostas é permitir que a árvore de união de fluxos possa atingir uma maior profundidade. No caso da técnica *Two-Level Patching*, a árvore de união de fluxos pode ter até três níveis de profundidade, assemelhando-se bastante à técnica *Transition Patching*, explicada há pouco. De forma sucinta, a diferença é que na técnica *Two-Level Patching* as subjanelas definidas, dentro de uma janela ótima, não são necessariamente de mesmo tamanho, e para cada uma destas subjanelas, inclusive a primeira, existe um fluxo *multicast* associado. A determinação das subjanelas e da janela ótima na técnica *Two-Level Patching* também não é de solução imediata e baseia-se em equações recursivas. Ademais, os resultados apresentados nos experimentos ainda não mostram de forma clara que a relação entre a economia da banda e o custo de implementação seja suficientemente satisfatória. Isto porque, apesar da técnica *Two-Level Patching* ser mais eficiente que a técnica *Patching* original e que a técnica *Dynamic Skyscraper* [24, 5] (para objetos de popularidade baixa-média e um pouco alta), sua performance ainda é inferior com respeito a esquemas baseados em HSM [41].

Já a técnica *Recursive Patching* [43] pode ser vista como uma generalização da técnica *Transition Patching*. Sua concepção é bem mais flexível, pois permite que a árvore de união de fluxos possa ter tantos níveis quanto for desejado. Na prática, isto significa subdividir recursivamente as subdivisões da janela ótima inicial de tal sorte a atingir o número de níveis desejado. Como possível de se imaginar, a determinação da janela ótima inicial e das subjanelas e suas eventuais subdivisões não é imediata. A complexidade computacional para esta solução cresce exponencialmente com o número de níveis e, além disso, os resultados apresentados nos experimentos também ainda não mostram uma relação entre otimização de banda e custo de implementação

suficientemente atrativa. Isto porque é feito um comparativo apenas sobre latência de serviço em relação à técnica *Transition Patching* e, sobretudo, os resultados só são atrativos – indicando reduções significativas – em cenários onde a popularidade dos objetos é relativamente alta, o que contraria a recomendação de uso preferencial de técnicas de difusão [13, 26], conforme explicamos anteriormente.

2.3 Técnicas para Interatividade

Banker et al. [34] em 1994, conforme já mencionamos, foram os precursores formais desta área de pesquisa. Eles introduziram o conceito de *sintonia* nos servidores de vídeo sob demanda então existentes, os quais simplesmente realizavam a difusão (*broadcast*) de cópias do mesmo objeto em canais distintos e defasados no tempo entre si. A sintonia consistia em eficientemente determinar qual dos canais existentes estava transmitindo a unidade de dados desejada (ou mais próxima) pelo cliente. Ou seja, as ações VCR eram conseguidas por meio de saltos entre os canais disponíveis para aquele objeto. Naturalmente nem todas as ações VCR eram possíveis de serem realizadas e o cliente não tinha garantia em relação à qualidade de serviço.

Um pouco mais tarde, houve então as propostas de Dan et al. [66] e Almeroth e Ammar [12]. Dan et al. [66] tiveram especial atenção com as operações VCR do tipo *Pause/Resume*, mas não discutiram claramente outros tipos de interatividade e união de fluxos no sistema. De forma simples, a idéia era dividir os canais disponíveis do servidor em três tipos: *dedicados*, *sob demanda* e *de contingência*. Nos canais do primeiro tipo eram disponibilizados os objetos (por inteiro) mais populares em intervalos fixos de tempo. Ao acontecer uma requisição para um destes objetos era então identificado o canal que tivesse condições de mais rapidamente possível atender a esta requisição. O agrupamento de clientes acontecia apenas por meio de *Batching* [31, 32, 33]. Os canais do segundo tipo eram usados para atender requisições para objetos menos populares. Neste caso, a alocação de canais acontecia em função da ocorrência de requisições. E, por último, os canais *de contingência* destinavam-se a minimizar o tempo de resposta para um cliente que realizasse a ação de *Resume*. Uma maior prioridade era sempre dada para ações de *Resume* sobre

ações de clientes que estivessem iniciando a exibição do objeto. Caso não houvesse canais *de contingência* disponíveis, os canais do tipo *sob demanda* poderiam inclusive ser utilizados para atender as ações de *Resume*.

Já Almeroth e Ammar [12] tiveram, como principal idéia, a utilização do *buffer* local do cliente para garantir um serviço contínuo. A idéia era, antes de atender a requisição do cliente através da identificação de um canal – que estivesse transmitindo a unidade de dados (bloco do objeto) desejada – ou da alocação de um novo canal, explorar ao máximo as informações já armazenadas em *buffer*. O acúmulo de informações em *buffer* local era possível porque quando o cliente, por exemplo, realizava uma ação de *Pause*, a recepção de unidades do objeto, a partir do fluxo de dados original, não era interrompida, ou seja, o cliente permanecia escutando o fluxo de dados sendo transmitido, mesmo não fazendo sua exibição imediata. Nessa proposta era admitido a separação de uma parte dos canais disponíveis do servidor para serem canais *de emergência*. Estes canais eram utilizados exclusivamente para evitar interrupções do serviço de atendimento devido à impossibilidade de localização de um grupo que estivesse transmitindo a unidade desejada pelo cliente e, também, devido à indisponibilidade de canais do servidor.

Liao e Li [28] em 1997 propuseram uma técnica chamada *Split and Merge protocol* ou simplesmente *SAM protocol*. Esta proposta implementava todos os tipos de ações VCR e foi a primeira a formalmente discutir a possibilidade de união de fluxos no sistema. A idéia básica era alocar um fluxo exclusivo de interação I tão logo o cliente executasse uma ação VCR. Após a ação ter sido completada, o servidor então tentava identificar um fluxo em andamento com o qual o cliente poderia ser unido e extinguia o fluxo I anteriormente criado. Como característica principal desta técnica, tem-se o fato de que o *buffer* utilizado para prover a união de fluxos localiza-se nos nós de acesso da rede e é compartilhado pelos outros clientes. A principal desvantagem desta técnica é a necessidade instantânea da alocação de fluxos de interação I sempre que houver uma ação VCR, diferentemente da proposta de Almeroth e Ammar [12] que, por possuir *buffer* local, primeiro verifica se a requisição não pode ser atendida pela informação ali armazenada. No caso de não existir banda disponível para alocação do fluxo de interação I, a ação VCR no *SAM protocol* é simplesmente

bloqueada. Abram-Profeta e Shin [30] melhoram a proposta do *SAM protocol* com a idéia de utilizar um *buffer* local e exclusivo no cliente em vez de um *buffer* remoto e compartilhado localizado no nó de acesso da rede.

Por volta de 1999, Poon e Lo [67] propuseram a técnica conhecida por *Single-Rate Multicast Double-Rate Unicast* ou simplesmente *SRMDRU*. Esta técnica também admite todos os tipos de ações VCR. A idéia base é a utilização de fluxos exclusivos de interação I, como na proposta de Liao e Li [28], só que após a ação VCR ter sido completada, o cliente passa a receber em uma taxa duas vezes maior que a taxa de exibição do objeto, com o intuito de o mais rapidamente possível alcançar um fluxo em andamento e então ser unido a ele. Esta concepção lembra aquela da técnica de acesso seqüencial conhecida como *Piggybacking* [1, 60, 61, 62].

Ma e Shin [29, 68] em 2001 apresentaram a técnica conhecida como *Best-Effort Patching* ou simplesmente *BEP*. Esta técnica segue a concepção da técnica original de acesso seqüencial *Patching* [2], com a diferença de que a estrutura de união de fluxos é uma floresta de árvores de até três níveis de profundidade. Sua operação é sucintamente descrita a seguir. Admita a ocorrência de uma requisição para a unidade u de um dado objeto. Inicialmente é identificado um fluxo em andamento que esteja transmitindo a unidade de dados (bloco, segmento, etc.) mais próxima possível de u. Seja S este fluxo. É então verificado se já existe um fluxo de *patch*, associado ao fluxo S, capaz de ser utilizado no atendimento da requisição ocorrida para u. Para tanto, é necessário que o fluxo de *patch* esteja transmitindo uma unidade de dados posterior à unidade u e que tenha uma duração suficientemente longa de tal sorte que possa ser alcançado por um outro fluxo a ser aberto para atender imediatamente a requisição por u. Caso não exista este fluxo de *patch*, a requisição para u é então atendida por meio da abertura de um novo fluxo de *patch* associado diretamente ao fluxo S. Note que nesta concepção o fluxo de *patch* também é *multicast*.

Além da técnica *BEP*, existem outras propostas para modificações do esquema original de *Patching* no intuito de admitir florestas de árvores de três ou mais níveis como, por exemplo, [3, 41, 43, 29, 68], conforme já mencionamos na subseção anterior. Entretanto, os resultados obtidos em todas essas propostas sugerem que

o nível de complexidade para implementação e de gerência do sistema, e daí a dificuldade de aplicabilidade prática, ainda não se justifica contundentemente em face dos resultados observados de otimização de banda. Ainda no ano de 2001, Chan e Chang [69] apresentaram uma proposta para interatividade baseada em um esquema híbrido que utilizava-se tanto da idéia de difusão (*broadcast*) como da concepção de *Patching*. De forma sucinta, a operação desse esquema consistia em transmitir o objeto inteiro em canais *multicast* defasados no tempo entre si, semelhantemente à proposta de Banker et al. [34]. Daí, para atender uma requisição, o sistema fazia o cliente escutar (isto é, armazenar em *buffer* local) o canal que estivesse transmitindo a unidade (segmento, bloco, etc.) do objeto mais próxima àquela de fato solicitada e, em paralelo, abria um canal *unicast* para transmitir a diferença, i.e., um *patch*, eventualmente existente entre a unidade disponibilizada no canal *multicast* e aquela de fato solicitada.

Bem recentemente, Netto [70] e Gorza [71] apresentaram a técnica chamada *Patching* Interativo ou simplesmente PI. Esta técnica consegue ter um significativo grau de simplicidade por preservar a estrutura de união em dois níveis e trazer consigo uma evolução natural das primeiras propostas baseadas na idéia de *Patching*. Isto principalmente em relação ao uso planejado do *buffer* local do cliente para as uniões de fluxos do sistema e decisões de abertura de novos fluxos. A idéia base de PI é atender a solicitação ocorrida por meio da identificação de um fluxo em andamento que esteja transmitindo uma unidade de dados próxima (i.e., dentro de um limiar de tempo previamente estabelecido) à unidade de dados efetivamente requisitada, e transmitir a diferença, i.e., um *patch*, usando um fluxo *unicast*. Uma outra característica interessante de PI é a possibilidade do atendimento de uma requisição por um fluxo que esteja transmitindo uma unidade de dados anterior àquela de fato solicitada, provocando colateralmente alguma descontinuidade no serviço oferecido. Os experimentos realizados mostram uma otimização de banda satisfatória da técnica PI em comparação com a técnica *Patching* original.

É importante ainda mencionar que os esquemas HSM (p. ex., [5, 9, 6, 11]), apesar de originalmente propostos para acesso seqüencial, conforme já comentado, podem em geral ser adaptados ou servir de base para o desenvolvimento de técnicas

para cenários com interatividade. Por exemplo, os trabalho apresentados em [72, 73] mostram que uma otimização de banda significativa pode ser obtida, em relação a um atendimento *unicast*. Já os autores de [74, 75] revelam que a introdução de otimizações específicas no esquema HSM denominado *Bandwidth Skimming* (BS) [7] pode levar a reduções atrativas da banda do servidor em cenários com interatividade. São propostas basicamente quatro tipos de otimizações: *Locality* (LOC), *Silent Prefetch* (SP), *Keep Merge Buffer* (KMB) e *Preserve Merge Tree* (PMT). Explicamos sucintamente as otimizações a seguir.

A idéia geral é utilizar o *buffer* local para armazenar as unidades previamente solicitadas e recebidas pelo cliente ou que estão sendo transmitidas para outros clientes. Requisições futuras para unidades já armazenadas podem então ser servidas localmente. Na otimização LOC todas as unidades de dados recebidas pelo cliente são armazenadas no *buffer* local. Esta otimização tem a mesma concepção de emprego de *buffer* local explorada em [70, 71, 50, 56] e, também, é abordada mais adiante no Capítulo 4. A otimização SP explora períodos de silêncio do cliente, i.e., períodos em que o cliente está em estado de *pause* ou lendo a partir de seu *buffer* local, para armazenar localmente unidades de dados sendo transmitidas para outros clientes. Já KMB armazena em *buffer* local as unidades de dados que seriam descartadas devido à impossibilidade de sucesso da união de fluxos no sistema. Finalmente, PMT busca evitar que tentativas de união de fluxos sejam interrompidas quando os clientes entram em estado de *pause*, conforme brevemente descrito a seguir. Seja S_i um fluxo que está sendo escutado por apenas um cliente C. Além disso, S_i é o fluxo alvo de um fluxo S_j (i.e., S_j está tentando se unir à S_i) e tem como alvo um outro fluxo S_k (i.e., S_i está tentando se unir à S_k). Se o cliente C entra em estado de *pause*, o servidor estende o fluxo S_i para permitir que as tentativas de uniões de fluxos possam continuar e, em caso de sucesso, economizem banda. O cliente C continua a escutar a extensão de S_i, armazenando as unidades de dados em seu *buffer* local. Se, ao sair do estado de *pause*, o cliente C solicita uma unidade de dados já contida em seu *buffer*, ele é então servido localmente. Os autores ainda avaliam uma estratégia híbrida que consiste em combinar as otimizações SP, LOC e KMB. Os resultados experimentais revelam reduções expressivas de 54%–29% na banda média do servidor, comparativamente com o esquema original BS.

Capítulo 3

Distribuição da Banda do Servidor

3.1 Introdução

ESTE capítulo apresenta a derivação da distribuição da banda do servidor de VoD, considerando o emprego da clássica técnica *Patching*. Sua organização é descrita a seguir. Na Seção 3.2 introduzimos alguns resultados analíticos e conceitos básicos da técnica *Patching*, além da terminologia empregada ao longo do capítulo. A Seção 3.3 é dedicada a apresentação do modelo analítico para obter a distribuição. Os resultados dos experimentos realizados estão na Seção 3.4. Por último, as conclusões constituem a Seção 3.5.

3.2 Conceitos Básicos e Terminologia

Considere um servidor de VoD e um grupo de clientes recebendo fluxos de dados relativos a objetos multimídia (filmes, *video clips*, etc.) através da Internet a partir deste servidor. Existe *multicast* na rede. O fato da implementação de *multicast* não ocorrer de forma irrestrita em toda rede pode ser minimizado por meio do uso de *proxies* que interconectam redes *unicast* a redes *multicast*. Uma das arquiteturas mais comuns consiste de um canal *unicast* do servidor ao *proxy* e de canais *multicast* do *proxy* para os clientes. Como, em geral, o *proxy* e os clientes estão na mesma rede local, a implementação de *multicast* torna-se mais simples. Ainda, soluções

como *multicast* na camada de aplicação, usando tunelamento (*tunelling*), também têm sido utilizadas na Internet [47, 52, 53].

Suponha ainda que: a técnica *Patching* é utilizada para prover serviço imediato neste servidor, os clientes sempre requisitam o início do objeto e assistem ao mesmo até o final sem interrupções (i.e., acesso seqüencial), a banda do cliente corresponde a duas vezes a taxa de exibição do objeto, e os fluxos *multicast* e *unicast* transmitem na mesma taxa de exibição do objeto. Estas suposições são todas consideradas no modelo original de *Patching* apresentado em [2].

Também considere que cada cliente possui um *buffer* local capaz de armazenar uma parte do objeto, sendo esta parte suficientemente grande para permitir a sincronização dos fluxos de dados simultaneamente recebidos a partir do servidor. Esta suposição é baseada nos trabalhos de [22, 23, 4]. É mostrado que, desde que a banda do cliente seja duas vezes a taxa de exibição do objeto, no pior caso, o cliente escuta simultaneamente dois fluxos durante um intervalo de tempo correspondente à metade do tamanho do objeto. Entretanto, como é possível calcular uma janela de tempo ótima [22, 55], aqui assumimos o tamanho do *buffer* local do cliente igual ao valor dessa janela. Lembramos que na Seção 2.2 já descrevemos a forma de operação de *Patching*. Por fim, na Tabela 3.1 temos uma síntese dos principais parâmetros utilizados neste capítulo.

Em [55, 22] são propostos modelos bastante semelhantes para estimativa da banda média para a técnica *Patching*. Nestes modelos é suposto que a chegada de requisições obedece a um processo de Poisson de taxa λ_j para o objeto O_j, onde $j = 1, ..., u$ e u é o total de objetos armazenados no servidor. A banda média do servidor para transmissão do objeto O_j, medida em unidades da taxa de exibição do objeto, é mostrada como sendo dada por:

$$B_j = \frac{D_j + \frac{\lambda_j W_j^2}{2}}{W_j + \frac{1}{\lambda_j}} \ , \tag{3.1}$$

onde D_j e W_j são a duração total e a janela do objeto O_j, respectivamente. D_j e W_j são expressos na mesma unidade de tempo.

Note que o denominador da Equação 3.1 é o tempo médio entre as transmissões *multicast* consecutivas, i.e., a duração da janela mais o tempo médio até a chegada

Símbolo	Definição
λ_j	taxa de chegada de requisições para o objeto O_j
D_j	duração total do objeto O_j, medida em unidades de tempo
W_j	janela para o objeto O_j, medida em unidades de tempo
N_j	número médio de requisições para o objeto O_j que chegam no intervalo D_j. Usualmente denotado de popularidade e computado por $N_j = \lambda_j D_j$
B_j	banda média do servidor para transmitir o objeto O_j, medida em unidades da taxa de exibição do objeto O_j
d_j	unidade de tempo em que o objeto O_j é dividido para análise
T_j	duração total do objeto O_j, medida em unidades d_j
w_j	janela para o objeto O_j, medida em unidades d_j
u	número total de objetos no servidor

Tabela 3.1: Principais parâmetros

de uma nova requisição. O numerador corresponde a soma da duração média dos *patches* com a duração do objeto inteiro.

Diferenciando-se a Equação 3.1 em relação a W_j e igualando o resultado a zero, obtemos o valor ótimo da janela:

$$W_{j,otimo} = \frac{\sqrt{2N_j+1}-1}{\lambda_j} \tag{3.2}$$

Substituindo este último resultado na Equação 3.1, obtemos a banda média do servidor para a técnica *Patching* quando é usada a janela ótima:

$$B_{j,otimizada} = \sqrt{2N_j + 1} - 1 \ , \tag{3.3}$$

onde $N_j = \lambda_j D_j$ é o número médio de requisições de clientes que chegam durante o período de tempo D_j. O parâmetro N_j é comumente chamado de popularidade do objeto. Por simplicidade, chamamos $W_{j,otimo}$ e $B_{j,otimizada}$ de W_j e B_j, respectivamente.

Conforme mencionado no Capítulo 1, quando a popularidade N_j é alta, as técnicas de compartilhamento do tipo difusão periódica são mais eficientes, em termos

de economia de banda, que as técnicas orientadas a requisições de clientes; e quando a popularidade N_j é baixa ou média, as técnicas orientadas a requisições de clientes é que são mais eficientes [13]. É razoável, portanto, termos uma forma de classificar os objetos de acordo com sua popularidade N_j para podermos escolher a técnica de compartilhamento mais eficiente para a transmissão do objeto [22, 13]. Esta classificação é feita usualmente da seguinte forma [7]: para objetos de popularidade alta (i.e., $N \geq 100$), usamos técnicas do tipo difusão periódica; para objetos de popularidade baixa (i.e., $N < 10$) e, mais especialmente, para objetos de popularidade média (i.e., $10 \leq N < 100$), usamos técnicas do tipo orientada a requisições. Como *Patching* é uma técnica orientada a requisições, nossas análises neste capítulo se restringem, principalmente, a valores de N_j no intervalo de 10 a 100.

3.3 Distribuição de Uso da Banda do Servidor

Nesta seção apresentamos o modelo para o cálculo da distribuição do número de fluxos concorrentes gerados pela técnica de *Patching*, ou equivalentemente, para a distribuição de uso da banda do servidor. Inicialmente consideramos o caso do servidor de um único objeto e, em seguida, mostramos como o resultado pode ser estendido para um servidor de múltiplos objetos.

3.3.1 Distribuição para Único Objeto

Considerando o emprego da técnica *Patching* pelo servidor de VoD e a chegada de requisições para o objeto O_j obedecendo a um processo de Poisson de taxa λ_j, seja $C(t)$ o processo estocástico que representa o número de janelas de *Patching* iniciadas no intervalo $(0, t)$. Segue diretamente então que $C(t)$ é um processo de renovação cujos pontos de renovação são os instantes em que o servidor escalona um fluxo *multicast* do objeto completo (ou seja, início de uma janela). Um ciclo de $C(t)$ tem duração média de $W_j + \frac{1}{\lambda_j} = W_j + \frac{D_j}{N_j}$ e, portanto, é este intervalo que utilizamos como base na análise para obtenção da distribuição almejada.

Agora assumimos que o objeto O_j é dividido em unidades de tempo de duração

d_j. Por exemplo, um objeto de duas horas pode ser dividido em 20 unidades de seis minutos cada, ou dividido em 100 unidades de 72 segundos cada. Definimos então $T_j = \frac{D_j}{d_j}$, ou seja, T_j é igual ao número de unidades de duração d_j em que o objeto O_j é dividido. De forma similar, a janela w_j é definida também em função de d_j, ou seja, $w_j = \frac{W_j}{d_j}$.

Ainda assumimos que se há pelo menos uma chegada de requisição para o objeto em uma unidade de tempo, então o servidor inicia um fluxo de dados, i.e., um fluxo *multicast* do objeto inteiro ou um fluxo de parte do objeto (*patch*), no início da próxima unidade para atender a esta requisição. A duração d_j pode ser feita tão pequena quanto desejarmos para fins de obter uma probabilidade desprezível de haver mais de uma chegada em uma mesma unidade de tempo e, também, uma garantia de serviço imediato.

Das considerações acima, decorre então que o intervalo base de análise $W_j + \frac{D_j}{N_j}$ pode ser reescrito como $w_j + \frac{T_j}{N_j}$, e a Equação 3.2 reescrita conforme mostrado a seguir.

$$w_j = \frac{T_j(\sqrt{2N_j + 1} - 1)}{N_j} \tag{3.4}$$

Alternativamente, o intervalo base de análise pode ser visto então como constituído por um total de $w_j + \frac{T_j}{N_j}$ unidades de duração d_j. Daí, este intervalo pode ser expresso no seguinte formato: $[1, w_j + \frac{T_j}{N_j}]$. Examinamos, portanto, cada uma das $w_j + \frac{T_j}{N_j}$ unidades deste intervalo para determinarmos a distribuição do número de fluxos concorrentes em cada uma delas. Uma vez que tenhamos este resultado, determinamos em seguida uma distribuição para representar o número de fluxos concorrentes em todo o intervalo. Como O_j pode representar qualquer objeto do servidor, daqui para frente omitimos o índice j para simplificar a notação.

A Figura 3.1 apresenta um cenário de transmissão de um objeto completo no caso em que T é três vezes maior que $w + \frac{T}{N}$. Apenas os fluxos *multicast* são mostrados para simplificar a figura. Em regime estacionário, este cenário pode ser visto como uma sucessão de intervalos $[1, w + \frac{T}{N}]$. Note que os fluxos iniciados em um intervalo podem influenciar o número de fluxos concorrentes no intervalo seguinte.

Isto ocorre porque os fluxos *multicast*, i.e., os fluxos que transmitem o objeto inteiro, não se extinguem no intervalo em que são gerados, assim como também não necessariamente os fluxos de *patch*.

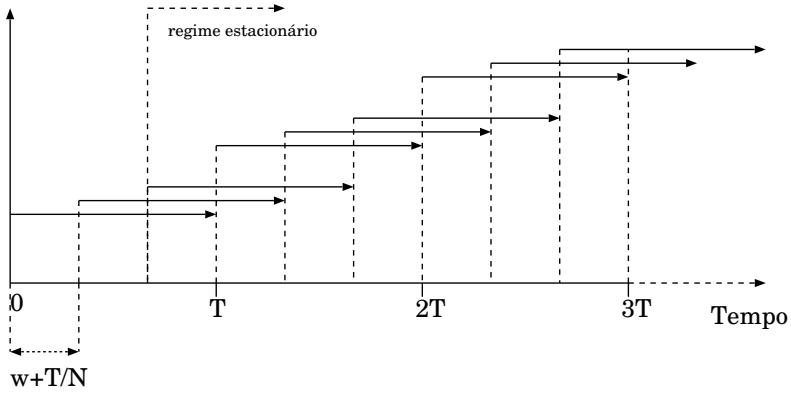


Figura 3.1: Fluxos *multicast* para transmissão do objeto completo

De forma mais detalhada, considere agora o k-ésimo intervalo $[1,\ w + \frac{T}{N}]$ do processo $C(t)$ em estado estacionário (veja Figura 3.2). Como explicamos, neste intervalo podemos ter transmissões devido a fluxos *multicast* e também devido a fluxos de *patch*. Em outras palavras, o número de fluxos concorrentes em cada unidade de tamanho d deste intervalo é dado por uma componente *multicast* mais uma componente de *patch*.

Seja M_C a componente *multicast*. Consideramos M_C como um valor constante e fazemos sua estimativa por meio da formulação de Little [76]: $M_C = \frac{T}{w+\frac{T}{N}} = \frac{N}{\sqrt{2N+1}}$ (i.e., o número médio de fluxos *multicast* é dado pela taxa de chegadas vezes a duração do objeto). Note que, no cenário ilustrado na Figura 3.1, esta constante é igual a três. Embora o número de fluxos *multicast* não seja de fato necessariamente constante devido à possível variação de taxa de chegada λ (e conseqüentemente de N), M_C pode ainda ser considerada uma boa aproximação, pois alterações na taxa de chegada λ, dentro de certos limites, não impactam severamente no valor de M_C. Isto porque para $N >> 1$, que se refere aos cenários de maior interesse, é possível simplificar a expressão de M_C da seguinte forma: $M_C = N/\sqrt{2N} = 0.707\sqrt{N}$. Portanto, temos que o número de fluxos *multicast* cresce aproximadamente com a raiz quadrada da taxa de chegada.

A componente de *patch* está associada ao processo de chegada dos clientes. Fa-

zemos então sua estimativa como uma variável aleatória. Nosso objetivo agora é, pois, determinar a distribuição desta variável aleatória. Considere novamente a Figura 3.2. Para determinarmos quantos fluxos de *patch* concorrentes são transmitidos em cada unidade do k-ésimo intervalo, precisamos também considerar os fluxos de *patch* do $(k-1)$-ésimo intervalo. Os demais intervalos anteriores não precisam ser considerados, pois os respectivos fluxos de *patch* eventualmente iniciados terminam antes que o k-ésimo intervalo comece. Considere que o k-ésimo intervalo seja dividido em dois subintervalos: $[1,\ w - \frac{T}{N}]$ e $[w - \frac{T}{N} + 1,\ w + \frac{T}{N}]$. Note que apenas no primeiro subintervalo podemos ter fluxos iniciados no $(k-1)$-ésimo intervalo. A Figura 3.2 mostra um exemplo em que uma chegada ocorre na última unidade do $(k-1)$-ésimo intervalo. O fluxo de *patch* iniciado por esta chegada é o mais longo possível e tem duração w. Note que quanto maior é o valor $\frac{T}{N}$, em relação ao valor w, menos unidades do k-ésimo intervalo transmitem fluxos que são iniciados no $(k-1)$-ésimo intervalo. No limite, quando $\frac{T}{N} = w$ (veja Equação 3.4), todos os fluxos de *patch* se extinguem dentro do próprio intervalo em que são iniciados. Note também que não precisamos considerar cenários em que $T/N > w$, pois isto significa que $N = 1$ (veja Equação 3.4)) e, portanto, o uso de técnicas de compartilhamento de banda torna-se inócuo. Aqui focamos valores de N no intervalo de 10 a 100, como já mencionamos.

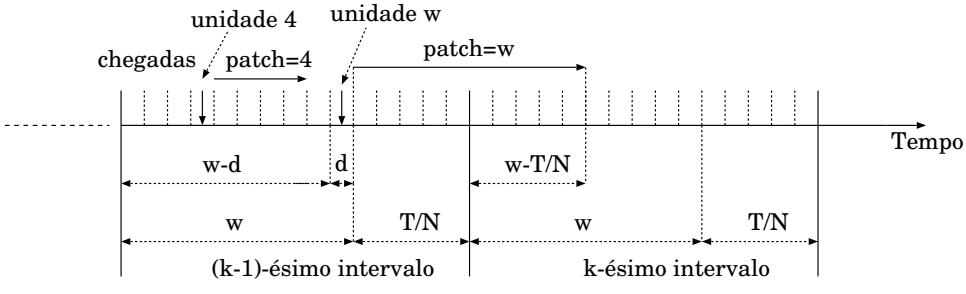


Figura 3.2: Análise dos fluxos de *patch* nos k-ésimo e $(k-1)$-ésimo intervalos

A distribuição do número de fluxos de *patch* concorrentes em cada unidade do k-ésimo intervalo é obtida em três passos. Primeiro, derivamos o número de fluxos concorrentes na i-ésima unidade dos k-ésimo e $(k-1)$-ésimo intervalos assumindo que há uma chegada (requisição) em cada unidade da janela w do $(k-1)$-ésimo intervalo e que não há chegadas no k-ésimo intervalo. Segundo, derivamos o número de fluxos concorrentes na i-ésima unidade do k-ésimo intervalo assumindo que há

uma chegada (requisição) em cada unidade das janelas w dos k-ésimo e $(k-1)$-ésimo intervalos, respectivamente. E terceiro, assumimos que o processo de chegada de requisições é representado por Poisson e verificamos a influência disto na abertura dos fluxos de *patch*.

Para o primeiro passo temos então a análise a seguir. Considere o $(k-1)$-ésimo intervalo e observe apenas os fluxos de *patch* lá iniciados. Por hipótese, como dito anteriormente, admitimos que ocorre uma chegada em cada uma das unidades da janela w deste intervalo. Note que uma chegada na j-ésima unidade do intervalo inicia, a partir da $(j+1)$-ésima unidade, um *patch* com duração igual a j unidades. A Figura 3.2 ilustra esta observação para o caso em que $j = 4$. Temos então que há um *patch* na i-ésima unidade do $(k-1)$-ésimo intervalo se, e somente se, $2j \geq i$ e uma chegada ocorre na j-ésima unidade deste intervalo, para $j = 1, 2, \ldots, i-1$. Além disso, o *patch* de mais longa duração, iniciado no $(k-1)$-ésimo intervalo, começa na unidade w e termina na unidade $2w$. Daí, as unidades a serem consideradas para análise vão na realidade desde $i = 1$ até $i = 2w$. Seja então $m_{i,w}$ a variável que denota o número de fluxos concorrentes de *patch* na i-ésima unidade do intervalo $[1, 2w]$, onde $i = 1, 2, ..., 2w$. O Lema a seguir determina uma expressão para $m_{i,w}$.

Lema 1 *Assumindo uma chegada (requisição) em cada unidade do intervalo [1, w], o número de fluxos concorrentes de patch na i-ésima unidade do intervalo [1, 2w] é dado pela variável $m_{i,w}$, calculada por:*

- *Caso 1:* $i \in [1, w]$

$$m_{i,w} = \sum_{j=1}^{i-1} \mathbf{1}\{2j \geq i\} = \lceil \tfrac{i-1}{2} \rceil \tag{3.5}$$

- *Caso 2:* $i \in [w+1, 2w]$

$$m_{i,w} = \sum_{j=1}^{w} \mathbf{1}\{2j \geq i\} = \lceil \tfrac{2w-(i-1)}{2} \rceil \tag{3.6}$$

onde $\mathbf{1}$ *denota a função indicadora.*

Prova

Mostramos por indução matemática que as Equações 3.5 e 3.6 são verdadeiras.

- Caso 1: $i = 1, 2, \ldots, w$.

  Base

  É direto verificar que para $i = 1$ e todo valor de w, temos que $m_{1,w} = 0$. Este resultado está de acordo com o resultado da Equação 3.5.

  Hipótese Indutiva

  Assuma que o lema é verdadeiro para $1 \leq i \leq k$, onde k é par, i.e, $k = 2c$ e $c \in \{1, 2, \ldots\}$. Podemos então escrever: $m_{2c,w} = \lceil \frac{2c-1}{2} \rceil = c$, e $m_{2c-1,w} = \lceil \frac{2c-1-1}{2} \rceil = c - 1$.

  Agora assuma que o lema é verdadeiro para $1 \leq i \leq k$, onde k é ímpar, i.e, $k = 2c - 1$ e $c \in \{1, 2, \ldots\}$. Podemos então escrever: $m_{2c-1,w} = \lceil \frac{2c-1-1}{2} \rceil = c - 1$, e $m_{2c-2,w} = \lceil \frac{2c-2-1}{2} \rceil = c - 1$.

  Daí, obtemos as seguintes relações:

  - Se i é par, então $m_{i,w} = m_{i-1,w} + 1$;
  - Se i é ímpar, então $m_{i,w} = m_{i-1,w}$.

  Avaliação da Equação 3.5 para $i + 1$

  Usando as suposições da hipótese indutiva mostraremos que o resultado da Equação 3.5 também vale para $i + 1$. Como antes, seja $c \in \{1, 2, \ldots\}$.

  - i é ímpar, então $i + 1$ é par.

    Temos então $i = 2c - 1$ e $i + 1 = 2c$. Usando a Equação 3.5, obtemos $m_{2c,w} = \lceil \frac{2c-1}{2} \rceil = c$. Agora usando o resultado da hipótese indutiva, obtemos: $m_{2c,w} = m_{2c-1,w} + 1 = c - 1 + 1 = c$.

  - i é par, então $i + 1$ é ímpar

    Temos então $i = 2c$ e $i + 1 = 2c + 1$. Usando a Equação 3.5, obtemos $m_{2c+1,w} = \lceil \frac{2c+1-1}{2} \rceil = c$. Agora usando o resultado da hipótese indutiva, obtemos: $m_{2c+1,w} = m_{2c,w} = c$.

Isto completa a prova da Equação 3.5.

- Caso 2: $i = w+1, w+2, \ldots, 2w$.

  Base

  É direto verificar que para $w = 1$ e $i = 2$, temos $m_{2,1} = 1$. Este resultado está em acordo com a Equação 3.6.

  Hipótese Indutiva

  Sejam c_1 e $c_2 \in \{1, 2, \ldots\}$.

  - Assuma que o lema é verdadeiro para $1 \leq i \leq k$, onde k é par, i.e., $k = 2c_2$.

    Para w par, i.e., $w = 2c_1$, podemos então escrever: $m_{2c_2,2c_1} = \lceil \frac{4c_1-2c_2+1}{2} \rceil = \lceil 2c_1 - c_2 + 1/2 \rceil = 2c_1 - c_2 + 1$, e $m_{2c_2-1,2c_1} = \lceil \frac{4c_1-2c_2+2}{2} \rceil = \lceil 2c_1 - c_2 + 1 \rceil = 2c_1 - c_2 + 1$.

    Para w ímpar, i.e., $w = 2c_1 - 1$, podemos então escrever: $m_{2c_2,2c_1-1} = \lceil \frac{4c_1-2-2c_2+1}{2} \rceil = \lceil 2c_1 - c_2 - 1/2 \rceil = 2c_1 - c_2$, e $m_{2c_2-1,2c_1-1} = \lceil \frac{4c_1-2-(2c_2-2)}{2} \rceil = \lceil 2c_1 - c_2 \rceil = 2c_1 - c_2$.

  - Assuma agora que o lema é verdadeiro para $1 \leq i \leq k$, onde k é ímpar, i.e, $k = 2c_2 - 1$.

    Para w par, i.e., $w = 2c_1$, podemos então escrever: $m_{2c_2-1,2c_1} = \lceil \frac{4c_1-2c_2+2}{2} \rceil = \lceil 2c_1 - c_2 + 1 \rceil = 2c_1 - c_2 + 1$, e $m_{2c_2-2,2c_1} = \lceil \frac{4c_1-2c_2+2+1}{2} \rceil = \lceil 2c_1 - c_2 + 1 + 1/2 \rceil = 2c_1 - c_2 + 2$.

    Para w ímpar, i.e., $w = 2c_1 - 1$, podemos então escrever: $m_{2c_2-1,2c_1-1} = \lceil \frac{4c_1-2-(2c_2-2)}{2} \rceil = \lceil 2c_1 - c_2 \rceil = 2c_1 - c_2$, e $m_{2c_2-2,2c_1-1} = \lceil \frac{4c_1-2-(2c_2-2)}{2} \rceil = \lceil 4c_1 - 2c_2 + 1/2 \rceil = 2c_1 - c_2 + 1$.

  Daí, obtemos as seguintes relações:

  - Se i é par, então $m_{i,w} = m_{i-1,w}$;
  - Se i é ímpar, então $m_{i,w} = m_{i-1,w} - 1$.

  Avaliação da Equação 3.6 para $i+1$

  Usando as suposições da hipótese indutiva mostraremos que o resultado da Equação 3.6 também vale para $i+1$. Como antes, sejam c_1 and $c_2 \in \{1, 2, \ldots\}$.

– i é ímpar, então $i+1$ é par. Temos então: $i = 2c_2 - 1$ e $i+1 = 2c_2$.

Para w par, ou seja, $w = 2c_1$, podemos escrever como se segue. Da Equação 3.6 obtemos: $m_{2c_2,2c_1} = \lceil \frac{4c_1-2c_2+1}{2} \rceil = 2c_1 - c_2 + 1$. E o resultado da hipótese indutiva fornece: $m_{2c_2,2c_1} = m_{2c_2-1,2c_1} = 2c_1 - c_2 + 1$.

Para w ímpar, ou seja, $w = 2c_1 - 1$, podemos escrever como se segue. Da Equação 3.6 obtemos: $m_{2c_2,2c_1-1} = \lceil \frac{4c_1-2-2c_2+1}{2} \rceil = 2c_1 - c_2$. E o resultado da hipótese indutiva fornece: $m_{2c_2,2c_1-1} = m_{2c_2-1,2c_1-1} = 2c_1 - c_2$.

– i é par, então $i+1$ é ímpar. Temos então: $i = 2c_2$ e $i+1 = 2c_2 + 1$.

Para w par,ou seja, $w = 2c_1$, podemos escrever como se segue. Da Equação 3.6 obtemos: $m_{2c_2+1,2c_1} = \lceil \frac{4c_1-2c_2-1+1}{2} \rceil = 2c_1 - c_2$. E o resultado da hipótese indutiva fornece: $m_{2c_2+1,2c_1} = m_{2c_2,2c_1} - 1 = 2c_1 - c_2 + 1 - 1 = 2c_1 - c_2$.

Para w ímpar, ou seja, $w = 2c_1 - 1$, podemos escrever como se segue. Da Equação 3.6 obtemos: $m_{2c_2+1,2c_1-1} = \lceil \frac{4c_1-2-2c_2-1+1}{2} \rceil = 2c_1 - c_2 - 1$. E o resultado da hipótese indutiva fornece: $m_{2c_2+1,2c_1-1} = m_{2c_2,2c_1-1} - 1 = 2c_1 - c_2 - 1$.

Isto completa a prova da Equação 3.6.

□

Passemos agora para o segundo passo da nossa análise, i.e., a derivação do número de fluxos concorrentes na i-ésima unidade do k-ésimo intervalo, assumindo que há uma chegada (requisição) em cada unidade das janelas w dos k-ésimo e $(k-1)$-ésimo intervalos, respectivamente. Seja então $l_{i,w}$ o número de fluxos de *patch* concorrentes na i-ésima unidade do k-ésimo intervalo $[1,\ w + T/N]$, considerando uma chegada em cada unidade da janela w deste e do $(k-1)$-ésimo intervalos. A variável $l_{i,w}$ pode ser diretamente obtida pela superposição adequada dos resultados apresentados no Lema 1.

A Figura 3.3 ilustra a superposição dos k-ésimo e $(k-1)$-ésimo intervalos. Vemos que o número de fluxos de *patch* concorrentes na primeira unidade do k-ésimo intervalo é igual a $m_{1,w} + m_{w+T/N+1,w}$, na segunda unidade é igual a $m_{2,w} + m_{w+T/N+2,w}$, e assim por diante. A formulação final para $l_{i,w}$ é apresentada no Lema 2.

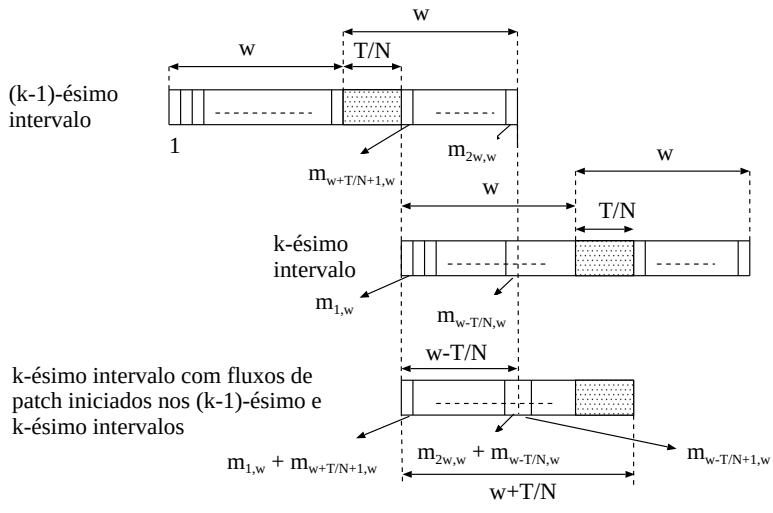


Figura 3.3: Superposição dos k-ésimo e $(k-1)$-ésimo intervalos

Lema 2 *Para $i = 1, 2, ..., w + \frac{T}{N}$ e assumindo uma chegada (requisição) em cada unidade da janela w do k-ésimo e $(k-1)$-ésimo intervalos, o número de fluxos de patch concorrentes na i-ésima unidade do k-ésimo intervalo é dado pela variável $l_{i,w}$, calculada por:*

- *Caso 1:* $i \in [1, w - \frac{T}{N}]$

$$l_{i,w} = \begin{cases} \frac{w-\frac{T}{N}}{2} + 1 & , \text{se } (w - \frac{T}{N}) \text{ e } i \text{ são pares} \\ \lceil \frac{w-\frac{T}{N}}{2} \rceil & , \text{caso contrário} \end{cases} \tag{3.7}$$

- *Caso 2:* $i \in [w - \frac{T}{N} + 1, w]$

$$l_{i,w} = \lceil \frac{i-1}{2} \rceil \tag{3.8}$$

- *Case 3:* $i \in [w + 1, w + \frac{T}{N}]$

$$l_{i,w} = \lceil \frac{2w - (i-1)}{2} \rceil \tag{3.9}$$

Prova

- Caso 1: $i \in [1, w - \frac{T}{N}]$

  Temos que $l_{i,w} = m_{i+w+T/N,w} + m_{i,w}$, onde $m_{i+w+T/N,w}$ é o número de fluxos de patch concorrentes relativo ao $(k-1)$-ésimo intervalo, e $m_{i,w}$ é o número

de fluxos de patch concorrentes relativo ao k-ésimo intervalo. Substituindo as Equações 3.5 e 3.6 do Lema 1 na expressão obtida para $l_{i,w}$, temos:

$$l_{i,w} = \lceil \frac{2w-(i+w+T/N-1)}{2} \rceil + \lceil \frac{i-1}{2} \rceil = \lceil \frac{w-T/N-i-1}{2} \rceil + \lceil \frac{i-1}{2} \rceil \tag{3.10}$$

Agora consideremos os valores possíveis (se par ou ímpar) para $w - \frac{T}{N}$ e i:

- $(w - \frac{T}{N})$ e i são pares.

$$l_{i,w} = \lceil \frac{w-T/N}{2} - \frac{i}{2} + \frac{1}{2} \rceil + \lceil \frac{i}{2} - \frac{1}{2} \rceil = \frac{w-T/N}{2} - \frac{i}{2} + 1 + \frac{i}{2} = \frac{w-T/N}{2} + 1 \tag{3.11}$$

- $(w - \frac{T}{N})$ é ímpar e i é par.

$$l_{i,w} = \lceil \frac{w-T/N}{2} - \frac{i}{2} + \frac{1}{2} \rceil + \lceil \frac{i}{2} - \frac{1}{2} \rceil = \frac{w-T/N}{2} + \frac{1}{2} - \frac{i}{2} + \frac{i}{2} = \lceil \frac{w-T/N}{2} \rceil \tag{3.12}$$

- $(w - \frac{T}{N})$ e i são ímpares.

$$l_{i,w} = \lceil \frac{w-T/N}{2} - \frac{i}{2} + \frac{1}{2} \rceil + \lceil \frac{i}{2} - \frac{1}{2} \rceil = \frac{w-T/N}{2} - \frac{i-1}{2} + \frac{i-1}{2} = \lceil \frac{w-T/N}{2} \rceil \tag{3.13}$$

- $(w - \frac{T}{N})$ é par e i é ímpar.

$$l_{i,w} = \lceil \frac{w-T/N}{2} - \frac{i}{2} + \frac{1}{2} \rceil + \lceil \frac{i}{2} - \frac{1}{2} \rceil = \lceil \frac{w-T/N}{2} \rceil - \frac{i-1}{2} + \frac{i-1}{2} = \lceil \frac{w-T/N}{2} \rceil \tag{3.14}$$

Isto completa a prova da Equação 3.7.

- Caso 2: $i \in [w - \frac{T}{N} + 1, w]$

Temos que $l_{i,w} = m_{i,w}$, pois neste subintervalo todos os fluxos de patch iniciados no $(k-1)$-ésimo intervalo já terão terminado. Da Equação 3.5 do Lema 1, temos então:

$$l_{i,w} = \lceil \frac{i-1}{2} \rceil \tag{3.15}$$

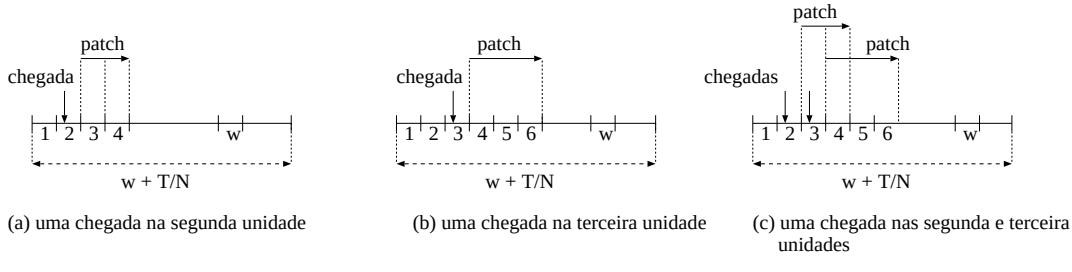
- Caso 3: $i \in [w + 1, w + \frac{T}{N}]$

Temos também que $l_{i,w} = m_{i,w}$, pois neste subintervalo todos os fluxos de patch iniciados no $(k-1)$-ésimo intervalo já terão terminado. Da Equação 3.6 do Lema 1, temos então:

$$l_{i,w} = \lceil \frac{2w - (i-1)}{2} \rceil \tag{3.16}$$

Isto completa a prova das Equações 3.8 e 3.9. □

Passemos agora para o último passo da nossa análise, i.e., consideremos que o processo de chegada (requisições) é representado por um processo de Poisson. Seja então X_i a variável aleatória que denota o número de fluxos de *patch* concorrentes na i-ésima unidade do intervalo $[1,\ w + \frac{T}{N}]$, onde $i = 1, ..., w + \frac{T}{N}$. Considere o cenário apresentado na Figura 3.4 e examinemos, por exemplo, os possíveis valores de X_4. (Para simplificar a figura, nem os fluxos *multicast* e nem os fluxos de *patch* do intervalo anterior são representados). Podemos ver que os valores possíveis para X_4 são 0, 1 ou 2, a depender das chegadas ocorridas nas unidades anteriores. $X_4 = 1$ quando há uma chegada ou na segunda ou na terceira unidades (Figuras 3.4(a) e 3.4(b)). Já a Figura 3.4(c) ilustra a situação em que $X_4 = 2$. Neste caso ocorre chegada tanto na segunda quanto na terceira unidade. Finalmente, $X_4 = 0$ quando não há chegadas nem na segunda e nem na terceira unidades, independentemente de ter havido ou não uma chegada na primeira unidade, pois o fluxo de *patch* relativo a essa primeira unidade não seria suficientemente longo para influenciar o valor de X_4. De forma geral, X_i depende, portanto, do número de unidades anteriores à i-ésima unidade que têm uma chegada, dado que os fluxos de *patch* então iniciados por estas chegadas são suficientemente longos para se estenderem sobre a i-ésima unidade. O Teorema a seguir determina a distribuição de X_i.



(a) uma chegada na segunda unidade

(b) uma chegada na terceira unidade

(c) uma chegada nas segunda e terceira unidades

Figura 3.4: Fluxos concorrentes na quarta unidade do intervalo $[1, w + T/N]$.

Teorema 1 *X_i tem distribuição binomial com parâmetros $p = 1 - e^{-\frac{N}{T}}$ e $n_{i,w}$, $i = 1, \ldots, w + \frac{T}{N}$. O parâmetro $n_{i,w}$ é dado por:*

1. *Caso 1:* $w = \frac{T}{N}$, *então:*

$$n_{i,w} = \begin{cases} \lceil \frac{i-1}{2} \rceil & , \text{ se } i \in [1, w] \\ \\ \lceil \frac{2w-(i-1)}{2} \rceil & , \text{ se } i \in [w+1, w+\frac{T}{N}] \end{cases}, \tag{3.17}$$

2. *Caso 2:* $w > \frac{T}{N}$, *então:*

   - $i \in [1, w - \frac{T}{N}]$, *então:*

$$n_{i,w} = \begin{cases} \frac{w-\frac{T}{N}}{2} + 1 & , \text{ se } (w - \frac{T}{N}) \text{ e } i \text{ são pares} \\ \\ \lceil \frac{w-\frac{T}{N}}{2} \rceil & , \text{ caso contrário} \end{cases} \tag{3.18}$$

   - $i \in [w - \frac{T}{N} + 1, w]$, *então:*

$$n_{i,w} = \lceil \frac{i-1}{2} \rceil \tag{3.19}$$

   - $i \in [w + 1, w + \frac{T}{N}]$, *então:*

$$n_{i,w} = \lceil \frac{2w-(i-1)}{2} \rceil \tag{3.20}$$

Prova

Como mencionado anteriormente, o intervalo $[1, w+\frac{T}{N}]$ está dividido em unidades de tamanho d. Seja então A a variável aleatória que denota o número de chegadas (requisições) na unidade d. Temos que $P[A > 0] = 1 - e^{-\frac{N}{T}}$, pois a chegada de requisições segue um processo de Poisson e λ é normalizado em relação ao valor de d. Então temos uma seqüência de unidades de tamanho d, onde cada uma destas unidades tem uma probabilidade $p = 1-e^{-N/T}$ de *sucesso*, onde este *sucesso* significa que pelo menos uma chegada ocorreu na unidade. O número total de unidades que podem iniciar um fluxo de *patch* que se estende sobre a i-ésima unidade do intervalo $[1, w + \frac{T}{N}]$ é dado pela variável $n_{i,w}$, estimada da forma a seguir:

- Caso 1: $w = \frac{T}{N}$

Todos os fluxos de *patch* iniciados em um dado intervalo $[1,\ w + \frac{T}{N}]$ são transmitidos inteiramente dentro deste mesmo intervalo. Daí, $n_{i,w} = m_{i,w}$, dado pelas Equações 3.5 e 3.6 obtidas no Lema 1.

- Caso 2: $w > \frac{T}{N}$

  Os fluxos de *patch* iniciados em um dado intervalo $[1,\ w + \frac{T}{N}]$ não são necessariamente transmitidos por inteiro neste mesmo intervalo. Daí, $n_{i,w} = l_{i,w}$, dado pelas Equações 3.7, 3.8 e 3.9 obtidas no Lema 2.

Podemos então concluir que X_i tem distribuição binomial com parâmetros $n_{i,w}$ e $p = 1 - e^{-\frac{N}{T}}$, onde $i = 1, \ldots, w + \frac{T}{N}$.

□

Do Teorema 1, temos, portanto, que o número de fluxos concorrentes de *patch* na i-ésima unidade do intervalo $[1,\ w + \frac{T}{N}]$ é uma variável aleatória binomial X_i, $i = 1, \ldots, w + \frac{T}{N}$. Note que as variáveis X_i possuem o mesmo parâmetro p. No entanto, o parâmetro $n_{i,w}$ pode variar dependendo do valor de i e w.

Examinemos os valores de $n_{i,w}$ para o Caso 2 do Teorema 1. Analisamos o Caso 2 porque w é maior que T/N nos cenários de interesse, i.e., $10 \leq N \leq 100$ (veja Equação 3.4 para referência). A partir das equações deste caso, notamos que a computação de $n_{i,w}$ varia de acordo com o subintervalo. Para o primeiro subintervalo, i.e., $[1,\ w - T/N]$, o parâmetro $n_{i,w}$ não depende de i e pode assumir no máximo dois diferentes valores. Para os dois outros subintervalos, i.e., $[w - T/N + 1,\ w]$ e $[w + 1,\ w + T/N]$, o parâmetro $n_{i,w}$ depende de i e pode assumir mais que dois valores distintos. Esta condição de diferente variabilidade de $n_{i,w}$, a depender do subintervalo considerado, é explicada devido à influência do intervalo anterior, a qual só existe para o primeiro subintervalo (Figura 3.2). Decorre então que, quanto menor for o valor de T/N, menor é a variabilidade associada a $n_{i,w}$ em todo o intervalo base de análise $[1,\ w + T/N]$.

Para ilustrar este aspecto, considere $T = 10^4$ (lembramos que $w = \frac{W}{d}$ e $T = \frac{D}{d}$). Calculamos então a proporção do intervalo $[1,\ w + T/N]$ correspondente ao primeiro e aos dois seguintes subintervalos. A Tabela 3.2 mostra os resultados obtidos

para $N = 10, 25, 50, 100, 200, 400, 700$. Como esperado, notamos que o percentual do primeiro subintervalo (onde $n_{i,w}$ assume no máximo dois valores distintos) é diretamente proporcional ao aumento de N.

N	$[1, w - T/N]$	$[w - T/N + 1, w + T/N]$
10	56%	44%
25	72%	28%
50	80%	20%
100	86%	14%
200	90%	10%
400	93%	7%
700	95%	5%

Tabela 3.2: Tamanho dos subintervalos para diferentes valores de N

Por exemplo, a Figura 3.5(a) mostra $P[X_i = k]$ para $N = 50$, $T = 10^3$ e $k = 1, 2$. O valor $P[X_i = k]$ é constante para o subintervalo $[1, w - T/N]$, pois $n_{i,w} = \lceil \frac{w-T/N}{2} \rceil$ para estas unidades. Na Figura 3.5(b), o valor de $P[X_i = k]$, para $N = 100$, $T = 10^3$ e $k = 1, 2$, oscila entre dois valores para o subintervalo $[1, w - T/N]$, pois $n_{i,w}$ é computado a partir da Equação 3.18. Em ambas as figuras, ao final do intervalo, vemos que $P[X_i = k]$ apresenta maior variabilidade. Como já explicado, este comportamento se dá devido a maior variabilidade do parâmetro $n_{i,w}$ no subintervalo $[w - T/N + 1, w + T/N]$.

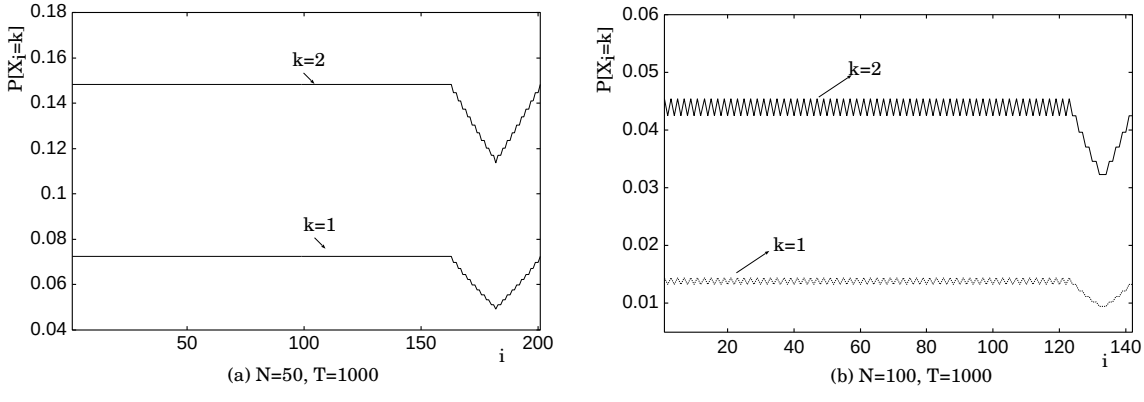


Figura 3.5: $P[X_i = k]$ para o intervalo $[1, w + T/N]$.

Baseando-se na análise acima, aproximamos então a distribuição do número de

fluxos de *patch* concorrentes no intervalo $[1,\ w + T/N]$ por uma variável aleatória binomial com parâmetros p e a esperança de $n_{i,w}$. Note que, quanto maior é o valor de N, mais precisa tende a ser esta aproximação, pois a variabilidade do parâmetro $n_{i,w}$ tende a reduzir-se. Mais formalmente, seja X a variável aleatória que representa o número de fluxos de *patch* concorrentes no intervalo $[1, w + T/N]$. A variável X tem então distribuição binomial de parâmetros $p = 1 - e^{-N/T}$ e $n_w = \lceil \frac{\sum_{i=1}^{w+T/N} n_{i,w}}{w+T/N} \rceil$. A expressão da distribuição de X é escrita na Equação 3.21.

$$F_X(k) = \sum_{j=0}^{k} \binom{n_w}{j} (1 - e^{-\frac{N}{T}})^j (e^{-\frac{N}{T}})^{n_w - j} \quad , 0 \leq k \leq n_w \tag{3.21}$$

Finalmente, o número total de fluxos concorrentes da técnica *Patching* é a soma dos fluxos concorrentes para transmissão dos fluxos de *patch* mais os fluxos concorrentes para transmissão do objeto completo (i.e, fluxos *multicast*). Esta segunda parcela, conforme já vimos, é aproximada por $M_C = \frac{N}{\sqrt{2N+1}}$. Definimos então Z como a variável aleatória que representa o total de fluxos concorrentes gerados pela técnica de *Patching*. A distribuição de Z é a distribuição da variável aleatória X deslocada para a direita do valor M_C. Isto acontece porque em nosso modelo consideramos o total de fluxos concorrentes para transmissão do objeto completo sendo dado pela constante M_C. Segue então diretamente que $P(Z \leq M_C - 1) = 0$. A distribuição de Z é dada na Equação 3.22. Destacamos que tanto X quanto Z dependem apenas dos parâmetros N e T do objeto O.

$$P(Z \leq k) = \begin{cases} 0 & , 0 \leq k \leq v - 1 \\ \sum_{j=v}^{k} \binom{n_w}{l} (1 - e^{-\frac{N}{T}})^l (e^{-\frac{N}{T}})^{n_w - l} & , v \leq k \leq n_w + v \end{cases} , \tag{3.22}$$

onde k denota o número de fluxos concorrentes, $v = M_C = \frac{N}{\sqrt{2N+1}}$, $n_w = \lceil \frac{\sum_{i=1}^{w+T/N} n_{i,w}}{w+T/N} \rceil$ e $l = j - v$.

Como já mencionado, em nosso modelo o intervalo $[1,\ w + \frac{T}{N}]$ é dividido em unidades de tempo de duração d. Note que quanto menor for o valor de d, mais próximo o modelo é da técnica *Patching*. No limite, quando $d \to 0$, o modelo é idêntico à técnica *Patching*. Definimos então dois critérios possíveis para estimar o valor de d. O primeiro é baseado na probabilidade de ocorrer mais de uma chegada

(requisição) na unidade de duração d, e o segundo é baseado no valor do erro relativo entre a esperança de Z e o valor da banda média B obtida a partir da Equação 3.3.

Em relação ao primeiro critério, apresentamos a Figura 3.6. Esta figura mostra a probabilidade de haver mais de uma chegada em uma unidade de duração d considerando vários cenários (diferentes valores de N e T). Lembramos que $T = \frac{D}{d}$. Notamos que para $T \geq 10^4$, a probabilidade de haver mais de uma chegada em uma unidade de duração d é menor que 10^{-4} para todos os valores de N examinados.

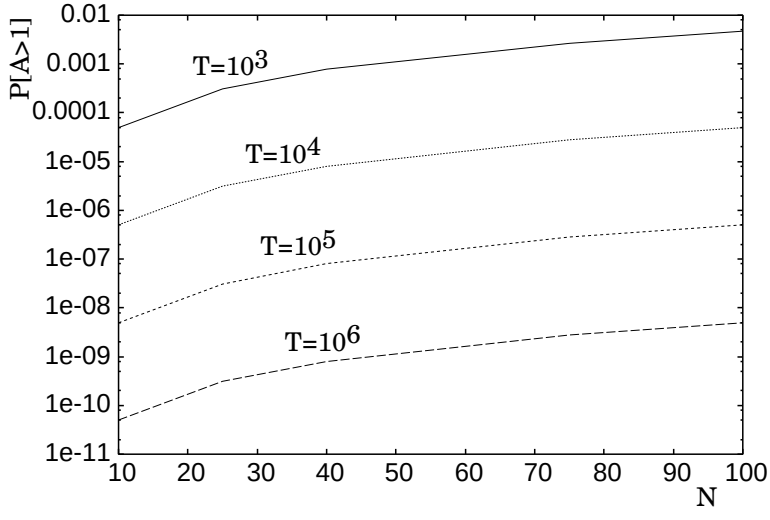


Figura 3.6: Probabilidade de mais de uma chegada na unidade de duração d.

Consideremos agora o segundo critério. A esperança da variável aleatória Z é dada por:

$$E[Z] = n_w(1 - e^{-\frac{N}{T}}) + \frac{N}{\sqrt{2N+1}} \ , \tag{3.23}$$

onde a primeira parcela da soma é a esperança da variável aleatória X, e a segunda parcela é o total de fluxos concorrentes para transmissão do objeto completo.

Definimos o erro relativo entre $E[Z]$ e B como $E_{RB} = \frac{|B-E[Z]|}{B}$. Na Figura 3.7(a), plotamos o erro relativo para as seguintes popularidades: $N = 10, 25, 40, 75$ e 100. Na Figura 3.7(b), mostramos a CCDF da variável aleatória Z, para $N = 100$ e vários valores de T. De forma geral, como já esperado, o erro relativo diminui a medida que T aumenta. No entanto, podemos observar que este erro não apresenta reduções significativas para $T > 10^4$. Isto ocorre porque, para $T > 10^4$, a Equação 3.23 passa a ser dominada essencialmente pelo valor da segunda parcela (i.e., $N/\sqrt{2N+1}$), independentemente do valor de N . Podemos também notar

que o erro calculado permanece praticamente o mesmo para $N \leq 40$. Neste caso, a Equação 3.23 também passa a ser dominada essencialmente pelo valor da segunda parcela (i.e., $N/\sqrt{2N+1}$), tendo em vista que a razão N/T já se torna desprezível para $T \geq 10^3$. Por fim, vemos as CCDFs são praticamente as mesmas para $T \geq 10^4$. Novamente lembramos que T representa o número de unidades de tamanho d em que o objeto é dividido, i.e., $T = D/d$.

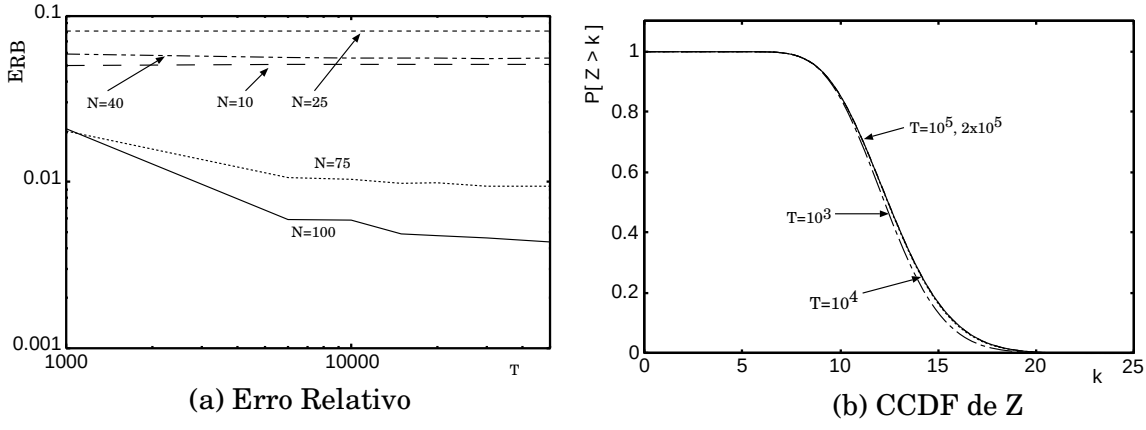


Figura 3.7: Estudo do valor de T.

3.3.2 Distribuição para Múltiplos Objetos

Um servidor usualmente possui mais de um objeto. Isto motiva a derivação da distribuição do número de fluxos concorrentes para o caso em que existem chegadas de requisições para vários objetos. Esta distribuição pode ser bastante útil, por exemplo, para a configuração global do sistema.

Na seção anterior vimos que as distribuições de X e Z dependem de dois parâmetros: N e T. Entretanto, podemos considerar T como uma constante de valor definido de acordo com os critérios apresentados ao final daquela seção. Assim, X e Z passam a depender exclusivamente de N. Aqui assumimos que todos os objetos no servidor possuem a mesma taxa de exibição. Esta é uma suposição bastante comum de trabalhos anteriores (e.g., [47, 25, 15, 14]) e é justificada por pesquisas recentes (p. ex., [73, 77]) voltadas para a análise de cargas de servidores reais.

Seja Y_N a variável aleatória que denota o número de fluxos de *patch* concorrentes devido a todos os objetos de popularidade N, onde $N \in \{1, \ldots, N_{max}\}$ e N_{max} é

o maior valor de popularidade de objeto no servidor. Desde que o valor T seja considerado constante e o mesmo para todos os objetos no servidor, temos então que Y_N é uma soma de variáveis aleatórias binomiais idênticas. Mais precisamente, Y_N é uma variável aleatória binomial com parâmetros $p = 1 - e^{-\frac{N}{T}}$ e $n_{N,w} = m_N \times \lceil \frac{\sum_{i=1}^{w+T/N} n_{i,w}}{w+\frac{T}{N}} \rceil$, onde m_N denota o número total de objetos de popularidade N no servidor [78].

É importante entender que considerar T como um valor constante e igual para todos os objetos no servidor não implica que todos os objetos tenham o mesmo tamanho D. Lembramos que $T = D/d$ (veja Tabela 3.1 para referência). Como temos um modelo analítico e fazemos uma análise em estado estacionário, o valor de D não afeta a distribuição da banda dos objetos. Para ilustrar, na Figura 3.8, temos as curvas de CCDFs (obtidas via simulação) para um objeto de popularidade $N = 5$, considerando D =100 min, 200 min, 500 min. Podemos notar que as curvas são praticamente as mesmas. Comportamento semelhante é observado para diferentes valores de D e um mesmo valor de N no intervalo $[10, 100]$.

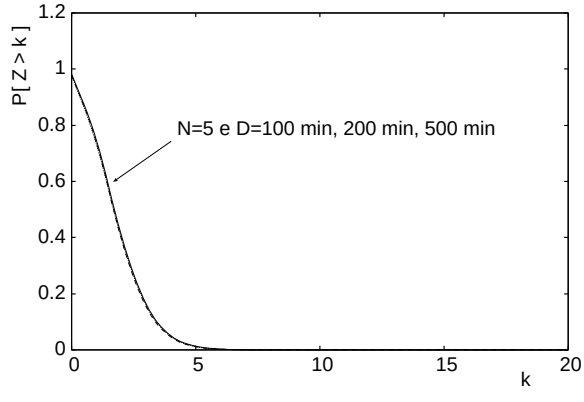


Figura 3.8: CCDFs para diferentes valores de D e um mesmo N.

Agora seja Y a variável aleatória que denota o número total de fluxos concorrentes de *patch* devido à transmissão de todos os objetos armazenados no servidor. Y é portanto a soma de potencialmente N_{max} variáveis binomiais de diferentes parâmetros. Como não existe uma solução fechada para a distribuição desta soma, a distribuição de Y pode então ser calculada a partir da convolução das PMFs de Y_N, para $N = 1, \ldots, N_{max}$. Esta convolução pode ser eficientemente implementada usando a Transformada Rápida de Fourier [49].

O número total de fluxos concorrentes da técnica *Patching* é a soma dos fluxos concorrentes para transmissão dos fluxos de *patch* mais os fluxos para transmissão dos objetos completos (i.e., fluxos *multicast*). Analogamente ao caso de um único objeto, esta última parcela é aproximada por $f = \sum_{N=1}^{N_{max}} \frac{m_N N}{\sqrt{2N+1}}$, que corresponde à soma de todas as componentes *multicast* individuais relativas aos objetos no servidor. Definimos então S como a variável aleatória que denota o número total de fluxos concorrentes necessários para a transmissão de todos os objetos armazenados no servidor. Decorre então que a distribuição de S é a distribuição de Y deslocada para a direita do valor constante f. Semelhantemente à análise do caso de um único objeto, o deslocamento ocorre porque assumimos que o número de fluxos *multicast* é uma constante dada por f. Segue então diretamente que $P(S \leq f - 1) = 0$.

3.4 Resultados

Nesta seção validamos o modelo analítico proposto usando simulação e mostramos como podemos, de forma prática, empregar a distribuição obtida para, por exemplo, (i) alocar banda para garantir um determinado nível de QoS, (ii) estimar o impacto na QoS quando alguns parâmetros do sistema se modificam dinamicamente, e (iii) configurar globalmente o sistema. Mencionamos que os resultados de simulações são obtidos usando o ambiente de modelagem Tangram-II [79, 80], e têm intervalos de confiança de 95% que estão dentro de 5% dos valores reportados. As chegadas dos clientes são geradas de acordo com um processo de Poisson, os objetos têm 100 minutos de duração, e fazemos $T = 10^4$. Por último, como *Patching* é uma técnica orientada a requisições, focamos mais especialmente valores de N no intervalo de 10 a 100.

Tangram-II é um ambiente de modelagem e experimentação de sistemas computacionais/comunicações, desenvolvido na Universidade Federal do Rio de Janeiro (UFRJ), com participação de UCLA/USA, com propósitos de pesquisa e educação. Este ambiente combina uma interface de usuário sofisticada em um paradigma de orientação a objeto e novas técnicas de solução para análise de performance e disponibilidade. O usuário especifica um modelo em termos de objetos que intera-

gem por meio de um mecanismo de troca de mensagens. Uma vez que o modelo seja compilado, pode ser resolvido analiticamente, se for Markoviano ou pertencer a uma classe de modelos não-Markovianos, ou resolvido via simulação. Há alguns *solvers* disponíveis para o usuário, tanto para estado estacionário como para análise transiente.

A organização desta seção é detalhada a seguir. A Subseção 3.4.1 faz a validação do modelo analítico proposto e exemplifica como empregar a distribuição para garantir um determinado nível de QoS para um grupo de clientes recuperando um objeto do servidor. Na Subseção 3.4.2 estudamos o comportamento da distribuição de uso da banda considerando diferentes valores de janelas w para uma dada popularidade de objeto. Isto permite-nos avaliar os requisitos de banda quando o servidor opera com uma janela diferente da ótima e, assim, quantificar o impacto na QoS do sistema. Esta análise é motivada pelo fato de que o cálculo da janela ótima da técnica *Patching* depende da correta estimativa da taxa de chegada de requisições λ. Entretanto, esta estimativa pode não ocorrer de forma instantânea e o servidor então opera temporariamente fora da condição de janela ótima. Por fim, a Seção 3.4.3 avalia o uso da distribuição da banda para múltiplos objetos para atingir um determinado nível de QoS, considerando diferentes cenários. Neste caso, o estímulo é a configuração global e eficiente do sistema.

3.4.1 Validação do Modelo e Reserva de Banda

Para avaliar a acurácia do nosso modelo usamos dois importantes critérios: Erro Relativo e MSE (*Mean Squared Error*). Em relação ao primeiro critério, comparamos a banda média B, computada a partir da Equação 3.3 (teórica), com aquela computada a partir do nosso modelo $E[Z]$ (analítica - Equação 3.23). Calculamos o erro relativo usando $Error = \frac{|B-E[Z]|}{B}$. A Figura 3.9 apresenta B e $E[Z]$, e a Figura 3.10 apresenta os erros relativos obtidos para $10 \leq N \leq 100$. Podemos observar que eles se situam entre 10^{-2} e 10^{-3} e são, portanto, satisfatórios. Na Tabela 3.3 ilustramos os valores dos parâmetros p e n_w para alguns valores de N no intervalo de 10 a 100.

Tabela 3.3: Valores dos parâmetros p e n_w

N	p	n_w
10	9.99e-04	1401
25	2.50e-03	1057
40	3.99e-03	890
50	4.98e-03	816
75	7.47e-03	692
100	9.95e-03	613

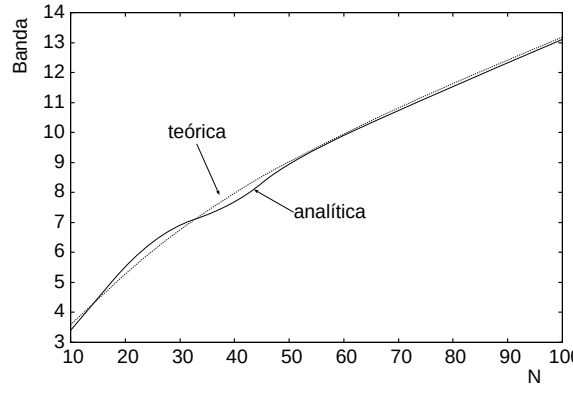


Figura 3.9: Banda média.

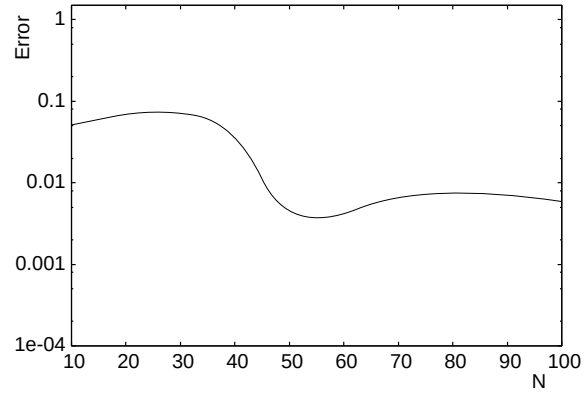


Figura 3.10: Erros relativos para banda média.

Em relação ao segundo critério, calculamos os valores de MSE para os casos de único objeto e de múltiplos objetos a partir de $MSE_{single} = \frac{\sum_{i=1}^{M}|(1-F_Z(z))-(1-F_{Z_s}(z))|^2}{M}$ e $MSE_{multiple} = \frac{\sum_{i=1}^{M}|(1-F_S(s))-(1-F_{S_s}(s))|^2}{M}$, respectivamente, onde M é o número de amostras, e o índice subscrito s das variáveis aleatórias Z e S é usado para se referir aos resultados provenientes de simulação. Lembramos que Z e S representam o número de fluxos concorrentes para os casos de único objeto e múltiplos objetos, respectivamente. A Figura 3.11 mostra a CCDF do número de fluxos concorrentes, considerando $N = 10, 25, 40, 50, 75$ e 100. Nesta figura temos os resultados analíticos e de simulação. E, na Figura 3.12, temos os resultados computados para os valores de MSE no intervalo $10 \leq N \leq 100$.

Note que os valores de MSE são em sua maioria menores que 10^{-3}. Isto indica que os resultados analíticos são satisfatoriamente próximos daqueles obtidos via simulação. Ainda, a partir das Figuras 3.12 e 3.11, podemos observar que a acurácia do modelo tende a aumentar com o crescimento do valor de N. Em outras palavras, o modelo proposto tende a ser mais preciso para objetos mais populares, i.e., para os objetos que significativamente impactam na banda do servidor. Enfatizamos que isto é uma tendência confirmada pelos experimentos realizados. Não podemos, portanto, sempre afirmar que, para qualquer valor de N, quanto maior ele for, mais preciso o modelo será.

Lembramos que a esperança do parâmetro $n_{i,w}$ é utilizada para parametrizar a distribuição binomial. Este parâmetro possui maior variabilidade no subintervalo $[w - T/N + 1, w + T/N]$, conforme mostrado na Seção 3.3.1. Decorre então que a acurácia do modelo é principalmente afetada pelo tamanho do subintervalo $[w - T/N+1, w+T/N]$ e pela variabilidade de $n_{i,w}$ neste mesmo subintervalo. O tamanho do subintervalo $[w - T/N + 1, w + T/N]$ decresce com o valor de N. Entretanto, a variabilidade de $n_{i,w}$ nem sempre decresce com o valor de N. Por exemplo, o coeficiente de variação de $n_{i,w}$ para $N = 10$ é menor que aquele para $N = 25$. Neste caso o comportamento de MSE termina sendo uma conseqüência principalmente da variabilidade de $n_{i,w}$.

A Figura 3.11 também demonstra que, em geral, os resultados analíticos subestimam os resultados de simulação. A explicação para isto é que utilizamos valores

médios para determinar a componente *multicast* e, também, para calcular o parâmetro n_w da distribuição binomial. Considere duas variáveis aleatórias binomiais Y_1 and Y_2 com mesmo parâmetro p (probabilidade de sucesso) e distintos parâmetros m_1 e m_2 (número de lançamentos), onde $m_1 < m_2$. Temos então que Y_1 sempre subestima Y_2 [78]. Em nosso modelo aproximamos o número de fluxos concorrentes de *patch* por uma variável aleatória X com parâmetros p e a esperança de $n_{i,w}$. Lembramos que o número de *patches* na i-ésima unidade do intervalo base de análise $[1, w + T/N]$ é representado por uma variável aleatória X_i com parâmetros p e $n_{i,w}$. Temos então a condição mencionada acima, i.e., a variável aleatória X subestima as variáveis aleatórias X_i com $n_{i,w} > n_w$. Semelhantemente, o valor médio provido por M_C subestima valores instantâneos da componente *multicast*.

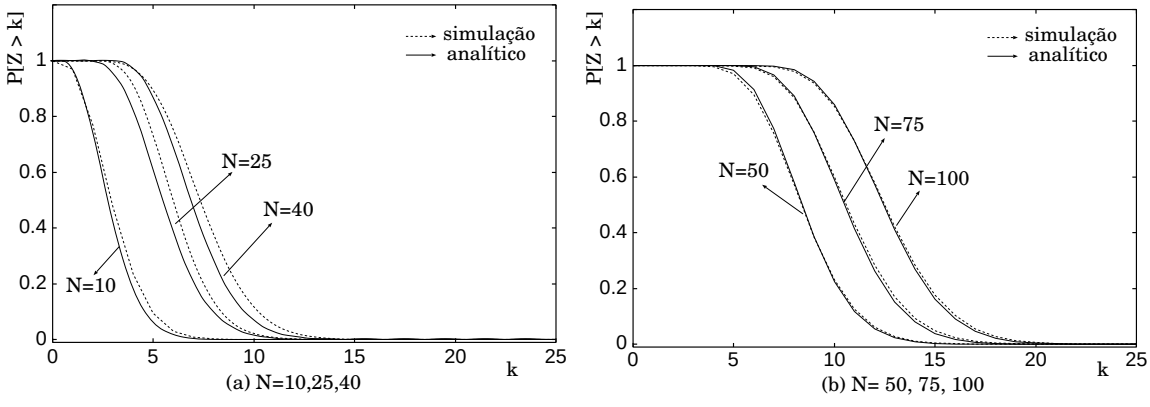


Figura 3.11: CCDF do número de fluxos concorrentes para um único objeto.

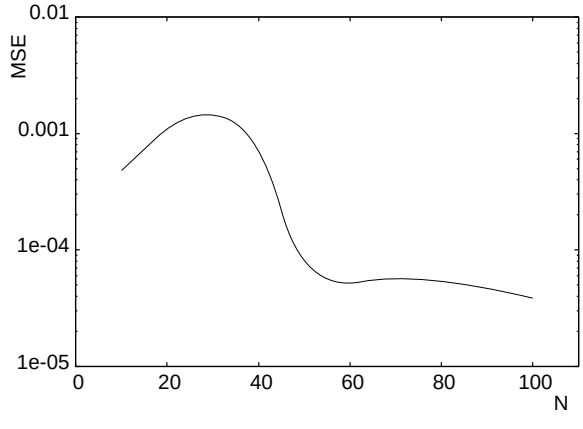


Figura 3.12: Mean Squared Error (MSE).

Na Figura 3.13 temos a CCDF para o caso de múltiplos objetos. Neste exemplo consideramos conjuntamente os mesmos seis objetos analisados na Figura 3.11. As

curvas analítica e de simulação estão bem próximas entre si e o MSE é igual a 2.32e-04, que é bastante satisfatório. Podemos notar também que este valor é menor que aqueles obtidos individualmente para $N \leq 40$. Isto ocorre porque, apesar da banda consumida por um objeto poder variar significativamente ao longo do tempo, a banda total (i.e., a soma das bandas individuais) para transmitir um grande número de objetos simultaneamente possui um menor coeficiente de variação ao longo do tempo, considerando objetos requisitados independentemente e taxa de chegada constante [5, 78]. A seguir focamos aplicações relativas ao caso do único objeto e deixamos a discussão do caso de múltiplos objetos para mais adiante.

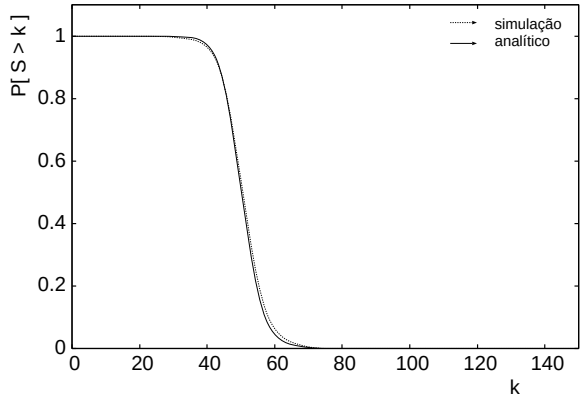


Figura 3.13: CCDF do número de fluxos concorrentes para múltiplos objetos.

Considerando nosso modelo analítico, temos a análise que se segue. A Figura 3.14 ilustra a PMF do número de fluxos concorrentes para transmissão de um objeto, considerando dois valores de popularidade N. A partir desta figura, podemos estimar individualmente o número de fluxos concorrentes (canais) utilizados e sua respectiva probabilidade de ocorrência. Este tipo de informação pode ser útil, por exemplo, para avaliarmos e compararmos pontualmente valores específicos da distribuição.

Já por meio da CCDF do número de fluxos concorrentes podemos reservar banda (canais) para um grupo de clientes tal que a probabilidade deles não serem atendidos imediatamente seja baixa, garantindo assim um determinado nível de QoS do sistema. Assuma, por exemplo, que desejamos reservar k canais no servidor para um grupo de clientes recuperando um dado objeto O de tal sorte que $P[Z > k] = 10^{-3}$, i.e., a probabilidade do número de fluxos concorrentes ser maior que k é igual a 10^{-3}. Usando a distribuição podemos então construir a Tabela 3.4 para, por exemplo, $N = 25, 50, 100$, e diretamente obter o valor de k desejado. Esta tabela mostra

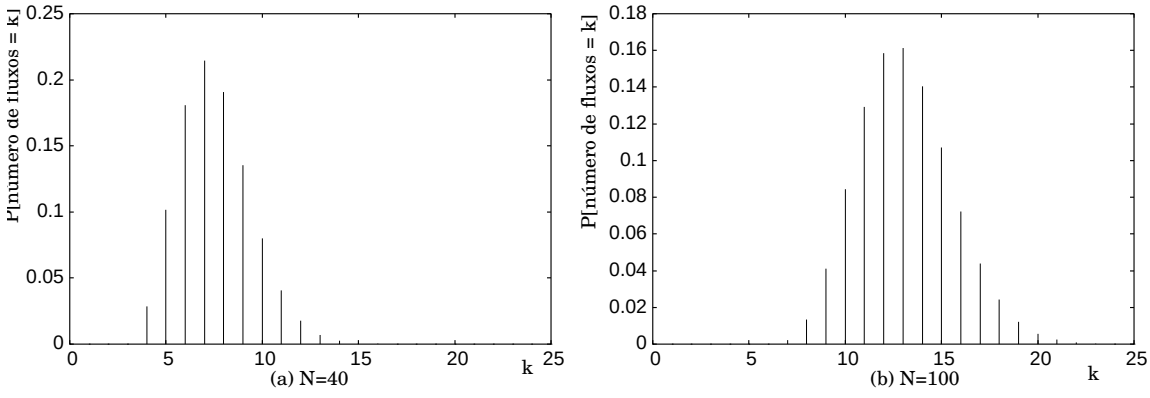


Figura 3.14: PMF do número de fluxos concorrentes para um objeto.

a banda estimada (**EB**), a probabilidade do número de fluxos concorrentes exceder a banda estimada ($P[Z > EB]$), e o incremento necessário (**NI**) na banda média (calculada através da Equação 3.3) para que se atinja o valor de probabilidade desejado.

Por exemplo, para $N = 50$, é necessário um incremento de 88.88% na banda média para se atingir $P[Z > EB] =$ 3.1e-04. Vemos que se a reserva de banda é feita com base no valor da banda média, a probabilidade do número de fluxos exceder a banda reservada é maior que 0.38 para todos os valores de N considerados. Isto implica que os clientes têm uma probabilidade significativa de esperar até que um canal possa ser alocado para serví-los. Por exemplo, notamos que, para N=25, aproximadamente 50% dos clientes não são atendidos imediatamente. Este experimento ressalta a importância da utilização da distribuição e, também, a inadequação do valor médio da banda para este fim.

3.4.2 Análise da Distribuição para Diferentes Janelas

Considere um servidor que dinamicamente estima o valor da taxa de chegada de requisições λ e, daí, determina a janela (limiar de tempo) a ser utilizada de acordo com o valor da janela ótima da Equação 3.2. Suponha que o valor inicialmente obtido para N seja igual a 75. Da Figura 3.15, a banda (número de canais) necessária para se ter $P[Z > k] \approx 10^{-2}$ é 17. Agora assuma que o valor da taxa λ se modifica e que o servidor não perceba imediatamente. O servidor continua então a operar

N=25			N=50			N=100		
EB	$P[Z > EB]$	**NI**	**EB**	$P[Z > EB]$	**NI**	**EB**	$P[Z > EB]$	**NI**
6	4.9e-01	-	9	3.8e-01	-	13	4.1e-01	-
7	2.7e-01	16.67%	10	2.3e-01	11.11%	14	2.7e-01	7.69%
8	1.3e-01	33.33%	11	1.2e-01	22.22%	15	1.6e-01	15.38%
9	5.2e-02	50.00%	12	5.5e-02	33.33%	16	8.9e-02	23.08%
10	1.8e-02	66.67%	13	2.3e-02	44.44%	17	4.6e-02	30.76%
11	5.7e-03	83.33%	14	8.9e-03	55.56%	19	9.6e-03	46.15%
12	1.6e-03	100.00%	15	3.1e-03	66.67%	21	1.5e-03	61.54%
13	4.1e-04	116.67%	17	3.1e-04	88.89%	22	5.6e-04	69.23%

Tabela 3.4: Avaliação da banda para $N = 25, 50, 100$.

com a janela ótima determinada para $N = 75$ e mantém a mesma reserva de banda original. Considerando este cenário, objetivamos responder as seguintes questões: (i) Qual é a qualidade de serviço provida pela reserva de banda original para os clientes que agora acessam o sistema? (ii) Qual é o impacto de operar com um limiar que não corresponde à janela ótima? e quais são os reais requisitos de banda deste limiar?

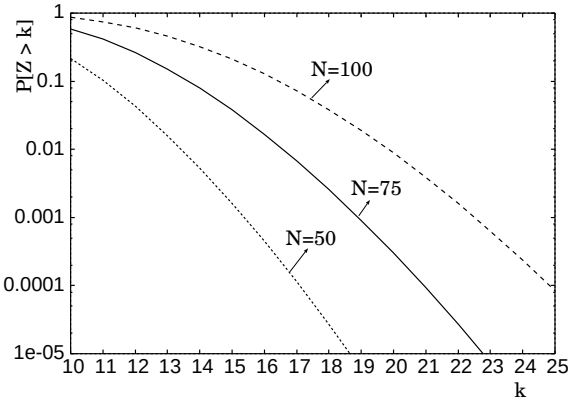
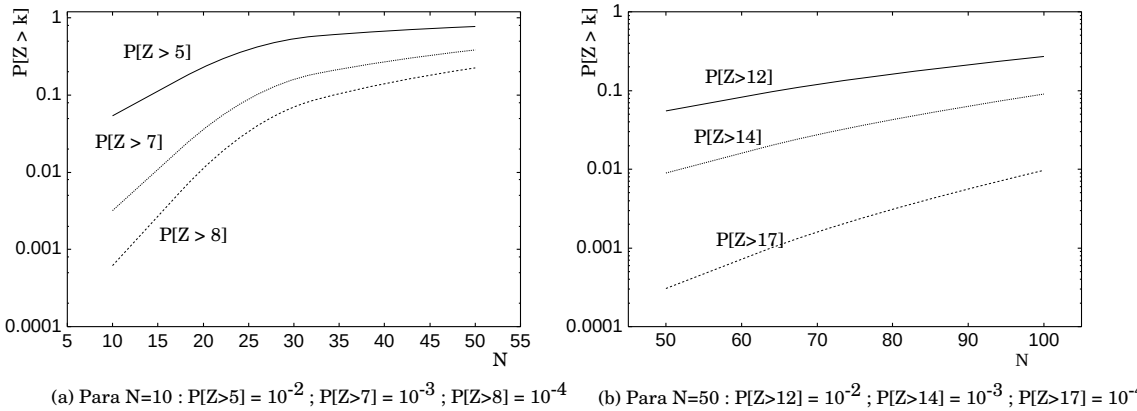


Figura 3.15: CCDF para diferentes valores de N e janela ótima w definida para $N = 75$.

As respostas para estas questões podem ser diretamente obtidas a partir da distribuição da banda do servidor. Analisemos a primeira questão. Em nosso exemplo, assumimos que $N = 75$, a reserva de banda é igual a k canais, e o limiar de tempo

é igual a janela ótima calculada para $N = 75$. Se a banda e o limiar de tempo são calculados considerando a popularidade igual a 75 e a taxa de chegada λ aumenta (diminui), observamos a partir Figura 3.15 que a probabilidade do número de fluxos concorrentes (ou seja, de canais) ser maior que o valor originalmente reservado aumenta (diminui). Isto é um resultado naturalmente já esperado, todavia, utilizando o modelo proposto, podemos precisamente quantificar a QoS oferecida quando a taxa varia e o servidor não detecta imediatamente. Suponha, por exemplo, que o número de canais reservados seja 17. Temos que $P[Z > 17] = 6.8\text{e-}03$ para $N = 75$. Se λ decresce de tal sorte que $N = 50$ e o servidor não percebe, decorre que $P[Z > 17] = 1.1\text{e-}04$. Portanto, a probabilidade que sejam necessários mais canais do que os reservados decresce em uma ordem de magnitude. Por outro lado, se λ aumenta de tal sorte que $N = 100$, decorre que $P[Z > 17] = 7.2\text{e-}02$, i.e., a probabilidade aumenta em uma ordem de magnitude. Mencionamos que as distribuições para $N = 50$ e $N = 100$ da Figura 3.15 são calculadas usando a janela ótima definida para $N = 75$.

A Figura 3.16 apresenta um outro exemplo que ilustra a qualidade de serviço oferecida aos clientes quando a taxa de chegadas λ se modifica. Neste exemplo atribuímos um valor inicial para λ e avaliamos a QoS quando λ varia. Na Figura 3.16(a) o servidor é inicialmente configurado para $N = 10$ e, na Figura 3.16(b), para $N = 50$. Notamos que, para $N = 10$ e $N = 50$, um aumento de aproximadamente 20% em λ não afeta de forma significativa a qualidade oferecida aos clientes. O mesmo comportamento é observado para outros valores de N no intervalo $[10, 100]$.



(a) Para N=10 : P[Z>5] = 10^{-2} ; P[Z>7] = 10^{-3} ; P[Z>8] = 10^{-4} (b) Para N=50 : P[Z>12] = 10^{-2} ; P[Z>14] = 10^{-3} ; P[Z>17] = 10^{-4}

Figura 3.16: QoS em função do crescimento de N.

Agora apresentamos a análise da segunda questão. A Figura 3.17 mostra a dis-

tribuição do número de fluxos concorrentes Z quando a janela utilizada varia em torno de 20% do valor da janela ótima para $N = 50$ e $N = 100$. Notamos que estas distribuições são bem semelhantes. Daí, se o servidor não detectar mudanças no valor da taxa λ, e conseqüentemente não atualizar o valor da janela utilizada, os requisitos de banda para este valor *não-ótimo* ainda são próximos daqueles relacionados ao valor ótimo. Resultados qualitativamente semelhantes são obtidos para outros valores de N no intervalo de 10 a 100. Temos então que a janela definida para a técnica *Patching* pode variar dentro de certos limites do valor ótimo sem afetar significativamente os requisitos de banda. Note que estes resultados corroboram aqueles do parágrafo anterior.

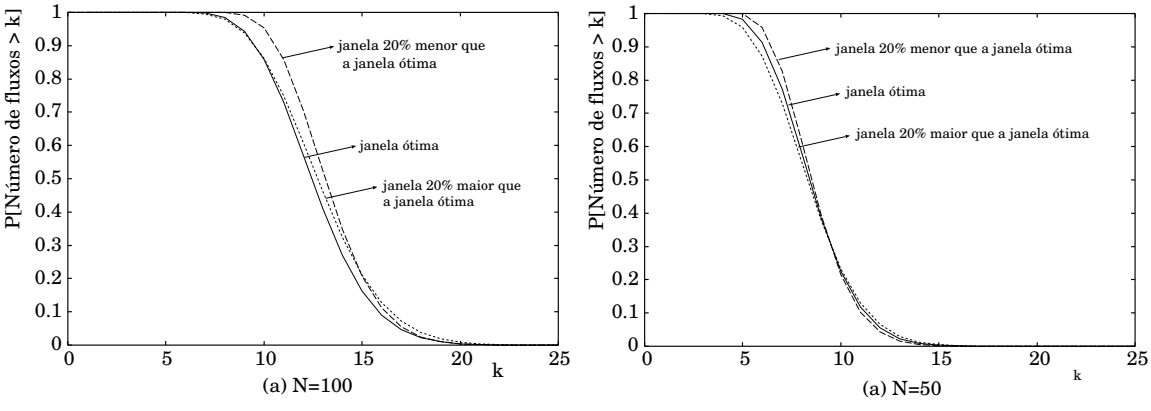


Figura 3.17: Distribuição do número de fluxos concorrentes para $N = 50$ e $N = 100$ quando a janela varia em um intervalo de 20% em torno da janela ótima.

Resumindo, o cálculo da distribuição do número de fluxos concorrentes permite que seja feita uma reserva de banda para um dado objeto ou para um grupo de clientes com o objetivo de garantir uma certa qualidade de serviço do sistema. É possível também prever a qualidade que será oferecida para os clientes caso a taxa de chegada de requisições se altere e o servidor não detecte rapidamente esta mudança. Além disso, aumentos de aproximadamente até 20% na taxa de chegada não impactam consideravelmente na qualidade de serviço do sistema. Um outro resultado importante está relacionado ao valor da janela. Através dos resultados obtidos, constatamos que alterações em torno de até 20% do valor da janela ótima não alteram de forma significativa a distribuição.

3.4.3 Configuração do Sistema

Como mencionado, a distribuição da banda do servidor no caso de múltiplos objetos pode ser empregada, por exemplo, para a configuração global do sistema. Para exemplificar este aspecto, consideramos quatro cenários específicos baseados em estudos de casos reais apresentados em [73]. Lá foi mostrado que a distribuição da freqüência de acesso para os objetos dos servidores de educação a distância BIBS e e-Teach pode ser aproximada pela concatenação de duas distribuições Zipf [81] (i.e., a probabilidade de selecionar o objeto de categoria i é igual a $\frac{1}{i^{1-z}\sum_{j=1}^{K}\frac{1}{j^{1-z}}}$, onde K é o número total de objetos no servidor e z é denominado de $skew factor$ da distribuição. A primeira Zipf é usada para modelar a freqüência de acesso dos objetos populares e a segunda para os objetos pouco populares. Consideramos em nosso experimento apenas a distribuição relativa aos objetos mais populares, pois são aqueles de maior interesse. Os objetos pouco populares possuem freqüência de acesso muito baixa e, portanto, são de pequena relevância para fins de análise de compartilhamento. Os cenários examinados são sumarizados na Tabela 3.5. Eles se referem aos objetos mais populares de tamanho 50-55 min do servidor BIBS. Esta tabela apresenta o número de objetos de cada popularidade e também o *skew factor* z da distribuição.

Tabela 3.5: Número de objetos em cada cenário

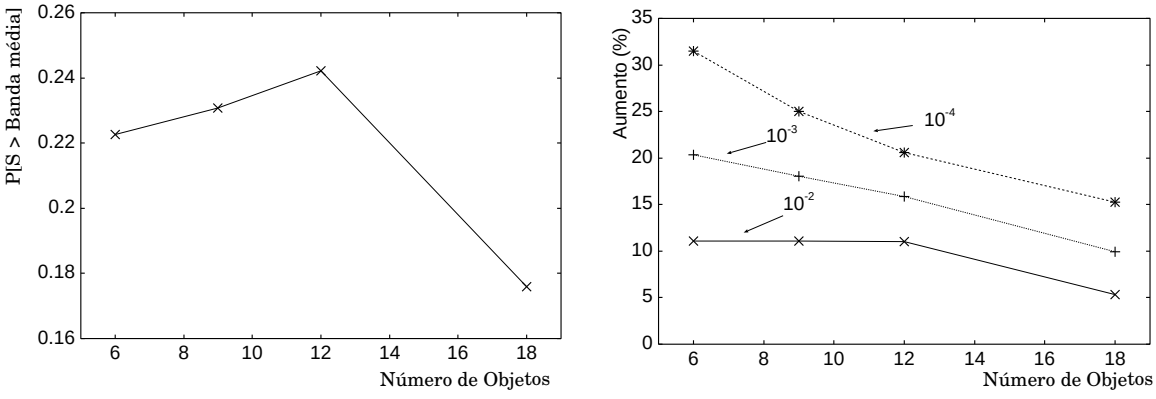
N	Cenário 1	Cenário 2	Cenário 3	Cenário 4
	z=0.60	z=0.55	z=0.80	z=0.65
100	1	2	2	4
75	1	2	2	3
50	1	2	2	3
40	1	1	2	3
25	1	1	2	3
10	1	1	2	2

A Figura 3.18(a) mostra a probabilidade do número total de fluxos concorrentes exceder a banda média total, calculada pela soma das bandas médias individuais

obtidas para cada valor de popularidade (para referência, veja Equação 3.3). A Figura 3.18(b) apresenta o incremento necessário em relação à banda média total para atingir uma dada probabilidade do cliente não ser imediatamente atendido por indisponibilidade de canais. Observamos que a probabilidade de exceder a banda média total é maior que 0.18 em todos os cenários. E o aumento necessário na banda média total para se ter uma probabilidade baixa do cliente não ser imediatamente atendido (i.e., $P[S > k = 10^{-4}]$) vai de 32% (Cenário 1) a 15% (Cenário 4). Os experimentos também mostram que quanto maior é o número de objetos a serem servidos, menor é o incremento necessário para atingir uma determinada QoS. Isto advém do fato de que, como já comentado, o coeficiente de variação da distribuição do número de fluxos concorrentes para múltiplos objetos diminui conforme aumentamos o número de objetos sendo transmitidos. Estes resultados ficam em sintonia com aqueles apresentados em [5, 25].

Outro exemplo de configuração global consiste em pensarmos no percentual de atendimento que queremos oferecer aos clientes do sistema. Isto decorre naturalmente da condição de que o servidor de um sistema de VoD tem uma banda de acesso limitada e, portanto, pode estar sujeito à indisponibilidade de canais [52, 53], semelhantemente ao que ocorre em um sistema telefônico convencional em que chamadas são perdidas por indisponibilidade de linhas tronco de acesso. Daí, pode ser desejável realizar um projeto de dimensionamento do sistema em função da estimativa de atendimento de clientes que queiramos oferecer. Neste caso, o conhecimento do valor médio, mesmo que seja próximo do real, não é útil para tais fins. Esta abordagem só pode ser dada, portanto, se conhecermos a distribuição da banda do servidor. Para exemplificar este aspecto utilizamos o cenário fictício descrito a seguir.

Consideramos um total de 20 objetos sendo transmitidos. A distribuição Zipf, com *skew factor* 0.271 [53, 82, 73, 83], é utilizada para determinar o número de objetos de cada popularidade (Tabela 3.6). Na Figura 3.19(a), plotamos a PMF do número de fluxos concorrentes. Podemos observar que o número de fluxos está aproximadamente no intervalo de 170 a 230. E na Figura 3.19(b) temos a CCDF do número de fluxos concorrentes. Neste exemplo, para atendermos os clientes



(a) P [S > Banda média]

(b) % aumento na banda do servidor para ter P[S>k]=10^{-2}, 10^{-3} or 10^{-4}.

Figura 3.18: Análise do caso de múltiplos objetos.

N	Num. de objetos
10	2
25	2
40	2
50	3
75	4
100	7

Tabela 3.6: Cenário do servidor de múltiplos objetos

com probabilidade aproximada de 0.50, precisamos de uma banda total (**BT**) de pelo menos 199 canais, e para atendermos com uma probabilidade aproximada de 0.95, precisamos de uma banda total de pelo menos 215 canais, conforme valores apresentados na Tabela 3.7.

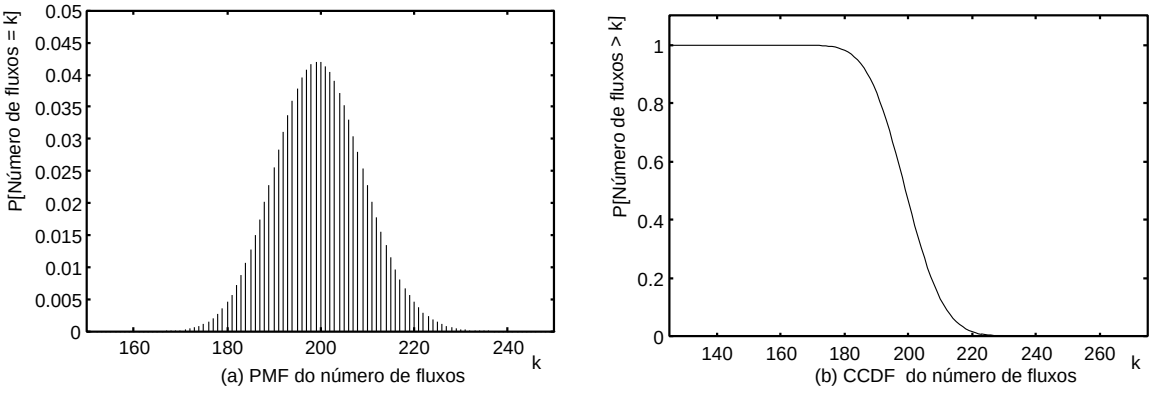


Figura 3.19: PMF e CCDF do número de fluxos concorrentes

$P[S \leq BT]$	**BT**
49.54e-02	199
49.54e-02	199
69.53e-02	204
90.80e-02	212
94.83e-02	215

Tabela 3.7: Probabilidade de atendimento

3.5 Conclusões

Neste capítulo desenvolvemos um modelo analítico preciso e simples para calcular a distribuição do número de fluxos concorrentes no servidor devido ao emprego da técnica *Patching*, ou equivalentemente, a distribuição de uso da banda do servidor. No caso do servidor de único objeto, mostramos que esta distribuição pode ser modelada por uma distribuição binomial parametrizada pela taxa de chegada de requisições e duração do objeto. Também abordamos o caso do servidor de múltiplos objetos e mostramos que, neste caso, a distribuição é dada por uma soma de

variáveis aleatórias binomiais independentes que pode ser implementada através da Transformada Rápida de Fourier. Por meio de experimentos, validamos o modelo analítico proposto, comprovamos a importância da utilização da distribuição em projetos reais e, ainda, exemplificamos seu uso, principalmente, para a obtenção de qualidade de serviço. Os principais resultados são destacados a seguir.

- De forma simples, podemos utilizar a distribuição de uso da banda para dinamicamente determinarmos o número de canais necessários para garantir um determinado nível de QoS do sistema para um grupo de clientes recuperando um objeto.

- A partir da distribuição de uso da banda no caso de um único objeto, verificamos que a probabilidade de se exceder o valor médio calculado é bem significativa (≈ 0.4). Isto quer dizer que a qualidade de serviço é severamente afetada se alguma reserva de canais é feita tendo por base valores médios.

- Através da análise da distribuição, é possível predizer a qualidade de serviço se a taxa de chegada de requisições se alterar e o servidor não for capaz de imediatamente perceber esta alteração. Constatamos que variações inferiores a 20% não impactam severamente na qualidade de serviço do sistema. Em sintonia com esta constatação, a janela de tempo da técnica *Patching* pode variar também em torno de 20% do valor ótimo sem significativamente afetar a distribuição de uso da banda.

- Nos cenários de múltiplos objetos considerados, observamos que a banda total do servidor necessária para garantir uma baixa probabilidade do cliente não ser atendido ($\approx 10^{-4}$) é de 15% a 32% maior que o valor da banda média total, o que evidencia a importância da utilização da distribuição em projetos reais.

Capítulo 4

Interatividade

4.1 Introdução

ESTE capítulo tem por principal objetivo a apresentação das novas técnicas para interatividade *Patching* Interativo Eficiente - PIE, *Patching* Interativo Completo - PIC e *Merge* Interativo - MI. Sua organização é descrita a seguir.

A Seção 4.2 introduz conceitos básicos relativos a cenários com interatividade e revisa os algoritmos de operação de duas técnicas já propostas na literatura. As novas técnicas PIE, PIC e MI são apresentadas formalmente na Seção 4.3. A Seção 4.4 volta-se para a discussão dos limiares de tempo usados nas novas técnicas propostas. Na Seção 4.5 estão os mais importantes resultados de simulação, obtidos nos diferentes cenários de interatividade considerados. Por fim, as conclusões constituem a Seção 4.6.

4.2 Conceitos Básicos

Nesta seção explicamos a forma mais usual de atendimento de requisições em cenários com interatividade e, também, revisamos os algoritmos de operação de duas recentes técnicas de compartilhamento de banda que podem ser empregadas para o serviço imediato em cenários com interatividade: *Patching* Interativo - PI e

Closest Target - CT . Conforme já comentamos, a técnica PI é baseada no paradigma de *Patching* e a técnica CT no paradigma de HSM. Estas técnicas são utilizadas nas análises competitivas que realizamos mais adiante com as novas técnicas propostas.

A escolha de PI e CT deu-se por serem técnicas relativamente recentes e, assim, conseguirem agregar simultaneamente dois aspectos essenciais: eficiência e simplicidade. A técnica PI possui um mecanismo razoavelmente elaborado, em termos de decisões que visam o compartilhamento de fluxos e da utilização do *buffer* local do cliente, sem perder com isso a simplicidade da estrutura de união de fluxos em dois níveis, conforme a técnica original *Patching*. Já a técnica CT apresenta resultados de performance comparáveis a escalonamentos ótimos *off-line*, considerando o acesso seqüencial, e pode ser facilmente empregada em cenários com interatividade.

4.2.1 Cenário com Interatividade

Considere um servidor de vídeo com interatividade e um objeto multimídia nele armazenado. Para efeito de análise, admita que o objeto esteja dividido em unidades (blocos) de dados de mesmo tamanho. Admita ainda que cada cliente possui um *buffer* local capaz de armazenar pelo menos metade do objeto requisitado e que sua banda corresponde a duas vezes a taxa de exibição deste objeto. Todos os fluxos iniciados (*unicast* ou *multicast*) transmitem dados na taxa de exibição do objeto. Ademais, o cliente é livre para praticar acesso seqüencial e não-seqüencial, ou seja, o cliente pode iniciar ou terminar a exibição do objeto em qualquer uma de suas unidades, além de praticar ações de interatividade como, por exemplo, *Pause*, *Jump Forwards* e *Jump Backwards*. Este é o modelo utilizado deste ponto em diante de nosso trabalho.

A Figura 4.1 ilustra o cenário básico para entendermos a forma atual mais comum de atendimento das requisições para um mesmo objeto (p. ex., [12, 28, 29]). Uma requisição ocorre em $t = t_0$ e existem n fluxos *multicast* em andamento. Seja u_d a unidade do objeto solicitada por esta requisição e, sem perda de generalidade, admita que todos os fluxos S_i, $i = 1, \ldots, n$, estão transmitindo unidades de dados posteriores a u_d. Por fim, seja $dist_i$ a distância entre u_d e a unidade sendo atualmente

transmitida pelo fluxo *multicast* S_i.

De forma geral, o atendimento da solicitação de u_d pode ser visto como realizado de duas formas básicas: serviço *descontínuo* ou serviço *contínuo* [53, 30]. No caso de serviço descontínuo, a solução consiste em simplesmente identificar o fluxo que tenha associado o menor valor de $dist_i$, $i = 1, \ldots, n$. Seja então S_{min} este fluxo e $dist_{min}$ o valor de distância associado. Note que se $dist_{min} > 0$ então o cliente que fez a solicitação por u_d receberá na realidade uma unidade posterior à mesma e distante $dist_{min}$, caracterizando o serviço descontínuo por não estar recebendo exatamente aquilo que foi solicitado. No caso do serviço contínuo, o valor $dist_{min}$ é determinado e então é verificado se está ou não dentro de um limiar de tempo permitido, aqui denominado de δ_{after}. Caso esteja, o cliente é atendido pelo fluxo correspondente S_{min} e a diferença $dist_{min}$ (i.e., *patch*) é enviada através de um fluxo *unicast*. Caso contrário, um novo fluxo *multicast* $S_{n+1} = S_{new}$ é então aberto para atender a requisição ocorrida.

A determinação formal do valor de δ_{after} não foi ainda formalmente discutida na literatura em termos analíticos. As pesquisas existentes restringem-se à proposição de técnicas que utilizam um δ_{after} possuindo valores em conformidade com o tamanho do *buffer* disponível para o cliente (p. ex., [28, 29]). Netto e Gorza [70, 71] chegaram a fazer estudos experimentais na tentativa de obter estimativas considerando valores em função da janela ótima de *Patching* (Equação 3.2), o que julgamos não ser sempre o mais adequado devido ao fato do tamanho de δ_{after} está mais diretamente vinculado ao tamanho médio solicitado na requisição que, por sua vez, pode ser muito maior ou menor que o tamanho da janela calculada. Neste trabalho mostramos, mais adiante, como obter uma estimativa analítica para este importante parâmetro.

Uma alternativa possível para evitar a criação de um fluxo S_{new} consiste em admitir que também seja verificado no sistema se há algum fluxo em andamento que esteja transmitindo uma unidade de dados anterior, mas suficientemente próxima à unidade u_d. Notadamente, a estimativa do que seja suficientemente próxima é bastante subjetiva e está atrelada ao grau de tolerância do cliente. Seja δ_{before} o limiar de tempo utilizado para mensurar esta proximidade. A utilização deste

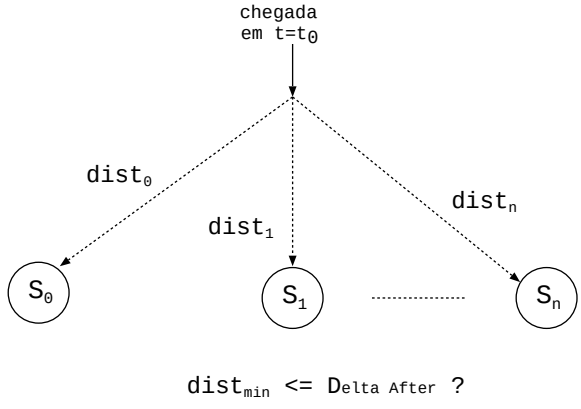


Figura 4.1: Cenário básico.

conceito traz naturalmente a descaracterização daquilo que chamamos de vídeo sob demanda verdadeiro (*True Video On Demand*) [53], pois obriga o cliente a assistir a uma parte do objeto que ele não solicitou, o que redunda em descontinuidade. Dependendo do tipo de aplicação e do tamanho desta descontinuidade, a utilização deste conceito pode não ser tão grave pelo fato do cliente não perder efetivamente qualquer informação e de poder prover algum nível de otimização da banda do servidor. Os trabalhos de Netto e Gorza [70, 71], por exemplo, consideram $\delta_{before} = 60\,\mathrm{s}$ para objetos de 6000–4200 s. Isto representa apenas 1.0%–1.4% do objeto e parece ser tolerável, por exemplo, para aplicações do tipo filme sob demanda. Estimulando a utilização de δ_{before}, Almeroth e Ammar [12] inclusive sugerem que clientes de sistemas interativos de fato podem não se importar com pequenas descontinuidades em relação ao serviço oferecido. Que tamanho exato deve ter δ_{before} e em que tipo de aplicação pode ser utilizado ainda é uma questão aberta na literatura.

4.2.2 Técnica Patching Interativo – PI

Nesta técnica a chegada de uma requisição é o único evento que pode provocar uma união de fluxos no sistema, conforme explicamos a seguir. Seja u_W a unidade de dados do objeto correspondente ao tamanho da janela ótima W da técnica *Patching* original [22, 23] (p. ex., se $W = 12\,\mathrm{s}$ e o objeto é dividido em unidades de dados de 1 s cada, então $u_W = 12$), e seja S_W um fluxo *multicast* que transmite, no instante da chegada da requisição, uma unidade de dados anterior ou igual à u_W. Para efeito de análise, assuma que uma requisição ocorre para uma dada unidade u_r do objeto. A

tomada de decisões neste algoritmo se dá em função de dois casos: $u_r = 1$ e $u_r \neq 1$, onde 1 representa a primeira (inicial) unidade do objeto.

- Caso 1: $u_r = 1$ – Se S_W existe, e está transmitindo uma unidade de dados posterior à u_r, então a requisição é atendida pelo mesmo e as unidades eventualmente perdidas (*patch*) são enviadas através de um fluxo *unicast*. Se S_W não existe então um novo fluxo *multicast* é aberto para atender a requisição.

- Caso 2: $u_r \neq 1$ – Neste caso inicialmente é verificado se existe um fluxo *multicast* S_{before}, i.e., um fluxo transmitindo uma unidade de dados anterior à u_r dentro de um limiar de tempo δ_{before}. Se este fluxo existe então a requisição é atendida pelo mesmo. Caso contrário, é verificado se existe um fluxo *multicast* S_{after}, i.e., um fluxo transmitindo uma unidade de dados posterior à u_r dentro de um limiar de tempo δ_{after}. Se este fluxo existe então o servidor informa ao cliente para escutar S_{after} e abre um novo fluxo *unicast* para transmitir as unidades inicialmente perdidas (*patch*). Por outro lado, se S_{after} não existe então um novo fluxo *multicast* S_{new} é aberto para servir a requisição por u_r.

4.2.3 Técnica Closest Target – CT

Diferentemente de PI, e conforme já mencionado, a técnica CT é baseada no paradigma de HSM. Seu mecanismo de tentativa de compartilhamento de dados é mais exaustivo que o de PI, pois está sempre buscando um fluxo para ser simultaneamente escutado com aquele que está efetivamente servindo a requisição então ocorrida. Além disso, todos os fluxos abertos são *multicast* e há ainda o fato desta técnica não utilizar limiar de tempo restritivo para a busca por fluxos no sistema. A técnica CT possui três eventos para união de fluxos, descritos a seguir.

- *evento 1*: admita que ocorra uma requisição para uma dada unidade de dados u_r do objeto. O servidor imediatamente abre um novo fluxo multicast S_{new} para atender esta requisição. Simultaneamente o servidor também tenta localizar um outro fluxo em andamento no sistema que esteja transmitindo uma unidade de dados posterior à u_r, sem considerar qualquer limiar de tempo.

Seja S' este fluxo. Se S' existe então o cliente de S_{new} também vai escutá-lo de tal sorte que S_{new} e S' possam, eventualmente, ser unidos mais à frente.

- *evento 2*: admita o término precoce de um fluxo S_j que era o fluxo alvo de um outro fluxo S_i. Em outras palavras, S_j termina antes de ser alcançado por S_i. Os clientes de S_i então tornam-se *órfãos* e os conteúdos respectivamente armazenados em seus *buffers*, devido à escuta de S_j, são descartados. O servidor imediatamente busca por um outro fluxo em andamento no sistema que transmite uma unidade de dados posterior àquela atual do fluxo S_i. Seja S_{subs} este fluxo. Se este fluxo existe então ele torna-se o novo alvo de S_i. Caso S_{subs} também termine precocemente, i.e., termine antes de ser alcançado por S_i, o processo de busca por um outro fluxo S_{subs} é repetido.

- *evento 3*: admita que ocorre a união de dois fluxos no sistema: S_i e seu alvo S_j. Seja S_m o fluxo resultante desta união. Os eventuais conteúdos dos *buffers* dos clientes de S_m, que originalmente eram clientes de S_j, são descartados e o servidor imediatamente busca no sistema por um fluxo que esteja transmitindo uma unidade de dados posterior àquela do fluxo S_m, sem considerar qualquer limiar de tempo. Seja S' este fluxo. Se S' existe então todos os clientes de S_m vão escutá-lo também de tal sorte que S_m e S' possam ser unidos mais à frente.

Note que os *eventos 1* e *3* conferem à árvore de união de fluxos a característica de ter profundidade ilimitada. Por exemplo, um fluxo S_i, recém aberto para atender a requisição por u_r, pode ter como alvo um outro fluxo S_j que, por sua vez, já pode ter como alvo um outro fluxo S_k, constituindo assim uma estrutura de profundidade três.

4.3 Novas Técnicas para Interatividade

4.3.1 Técnica Patching Interativo Eficiente – PIE

Semelhantemente à técnica PI, aqui a chegada de uma requisição é o único evento que provoca uma união de fluxos no sistema. Também é baseada no clássico esquema

Patching e tem, como principal premissa, a manutenção da estrutura de união de fluxos em no máximo dois níveis. Como antes, para efeito de análise, admita que ocorre uma requisição para uma dada unidade de dados u_r do objeto.

O algoritmo inicialmente busca por um fluxo *multicast* S_{before}, i.e., um fluxo transmitindo uma unidade de dados anterior à u_r e dentro de um limiar de tempo δ_{before}. Se este fluxo existe então a requisição é atendida pelo mesmo. Porém, se S_{before} não existe, é então verificado se existe um fluxo *multicast* S_{after}, i.e., um fluxo transmitindo uma unidade de dados posterior à u_r e dentro de um limiar de tempo δ_{after}. Se este fluxo existe, então o servidor informa ao cliente para escutar S_{after} e abre um novo fluxo *unicast* para transmitir as unidades inicialmente perdidas (*patch*). Entretanto, se S_{after} não existe então um novo fluxo *multicast* S_{new} é aberto para servir a requisição por u_r. Nesta situação é ainda verificado se existe um fluxo multicast S_{merge}, i.e., um fluxo transmitindo uma unidade de dados anterior à u_r e dentro de um limiar de tempo δ_{merge}. O fluxo S_{merge} não pode possuir *patches* associados a ele, pois um dos objetivos é manter a estrutura de união de fluxos em no máximo dois níveis. Se S_{merge} existe então o fluxo S_{new} é o seu alvo e os clientes de S_{merge} devem escutar também o fluxo S_{new} para que, eventualmente, S_{merge} se una ao fluxo S_{new}.

Note que o algoritmo de PIE é mais elaborado que o de PI por introduzir o conceito do fluxo S_{merge}, o que cria a expectativa de uma maior otimização de banda dado que é uma possibilidade a mais para a união de fluxos. Note, também, que mesmo a técnica CT, apesar de seu esquema exaustivo e irrestrito, não utiliza a conceituação do fluxo S_{merge}. Por fim, observe que em PIE não há tratamento diferenciado devido ao recebimento de requisição para a unidade inicial do objeto (i.e., $u_r = 1$), conforme ocorre em PI.

4.3.2 Técnica Patching Interativo Completo – PIC

Esta técnica também é baseada no clássico esquema *Patching* e, apesar de bem mais elaborada que as técnicas PI e PIE, também tem como premissa básica a manutenção da estrutura de união de fluxos em no máximo dois níveis. Três eventos

são considerados para união de fluxos no sistema, conforme explicamos a seguir.

- *evento 1*: admita que ocorra uma requisição para uma dada unidade de dados u_r do objeto. Nesta situação o algoritmo procede de forma semelhante ao de PIE com a diferença básica de que, em vez de buscar por um único fluxo S_{merge}, aqui busca-se por um conjunto de fluxos deste tipo. A expectativa natural é que mais otimização de banda possa ser conseguida desde que, potencialmente, mais fluxos são unidos.

- *evento 2*: admita o término precoce de um fluxo *multicast* S_j que era o fluxo alvo de um outro fluxo *multicast* S_i. Em outras palavras, S_j termina antes de ser alcançado por S_i. Os clientes de S_i então tornam-se *órfãos* e os conteúdos respectivamente armazenados em seus *buffers*, devido à escuta de S_j, são descartados. O servidor imediatamente busca por um outro fluxo em andamento no sistema que transmite uma unidade de dados posterior àquela atual do fluxo S_i e dentro do limiar de tempo δ_{merge}. Seja S_{subs} este fluxo. Se este fluxo existe então ele torna-se o novo alvo de S_i. Caso S_{subs} também termine precocemente, i.e., termine antes de ser alcançado por S_i, o processo de busca por um outro fluxo S_{subs} é repetido. Note a semelhança entre o procedimento realizado neste evento e aquele descrito para a técnica CT. A diferença reside basicamente na utilização do limiar de tempo δ_{merge}.

- *evento 3*: admita o término de todos os eventuais fluxos de *patch* associados a um fluxo denotado por S_{source}. Nesta situação imediatamente o servidor tenta localizar um fluxo que transmite uma unidade de dados posterior àquela do fluxo S_{source} e dentro de um determinado limiar de tempo. Seja S_{sink} este fluxo e seja δ_{merge} o limiar de tempo utilizado. Se S_{sink} existe então ele torna-se alvo de S_{source}. Ou seja, os clientes de S_{source} passam a escutar também o fluxo S_{sink} de tal sorte que S_{source} e S_{sink} possam ser unidos mais à frente.

4.3.3 Técnica Merge Interativo – MI

Aqui apresentamos a técnica *Merge* Interativo - MI. Assim como CT, esta técnica é baseada no paradigma de HSM e, portanto, tem sua estrutura de união de fluxos de

profundidade ilimitada (i.e., número de níveis ilimitado). Os seguintes três eventos são considerados para efeito de união de fluxos no sistema.

- *evento 1*: admita que ocorra uma requisição para uma dada unidade de dados u_r do objeto. O servidor imediatamente abre um novo fluxo S_{new} para atender esta requisição. Simultaneamente, o servidor busca por um fluxo em andamento no sistema que esteja transmitindo uma unidade de dados posterior à u_r e dentro de um limiar de tempo δ_{MI}. Seja S' este fluxo. Se S' existe então o cliente de S_{new} também vai escutá-lo de tal sorte que S_{new} e S' possam ser unidos mais à frente.

- *evento 2*: admita o término precoce de um fluxo S_j que era o fluxo alvo de um outro fluxo S_i. Neste caso o tratamento é análogo ao descrito no *evento 2* de PIC, sendo que aqui utiliza-se o limiar de tempo δ_{MI} para busca do substituto de S_j.

- *evento 3*: admita que ocorra a união de dois fluxos no sistema: S_i e seu alvo S_j. Seja S_m o fluxo resultante desta união. O servidor imediatamente busca no sistema por um fluxo que esteja transmitindo uma unidade de dados posterior àquela do fluxo S_m dentro de um limiar de tempo δ_{MI}. Seja S' este fluxo. Se S' existe então os clientes de S_m, que originalmente pertenciam ao fluxo S_i, vão escutá-lo também para que possam alcançar S' e então se unir a ele. Já os clientes de S_m, que originalmente pertenciam ao fluxo S_j, não são afetados.

Note que existem duas principais diferenças em relação à técnica CT explicada anteriormente: a consideração de um limiar de tempo δ_{MI} e a não fusão de grupos de clientes pertencentes a fluxos distintos na ocorrência do *evento 3*. Estas duas diferenças são discutidas em maiores detalhes a seguir. Em relação à primeira diferença, i.e., a utilização de um limiar de tempo denotado por δ_{MI}, argumentamos o seguinte. Embora a técnica CT não faça uso de qualquer limiar de tempo, uma série de outras técnicas baseadas em HSM o fazem na intenção de otimizar as decisões relacionadas a abertura e união de fluxos no sistema (p. ex., [9, 59]). Visualizamos com esta diferença a possibilidade de uma otimização da técnica MI.

Já em relação à segunda diferença, i.e., a não fusão de grupos de clientes pertencentes a fluxos distintos na ocorrência do *evento 3*, temos a seguinte análise. Suponha que um fluxo S_i se una com seu alvo S_j. Seja S_m o fluxo resultante desta união. Considere que S_j já tivesse um fluxo alvo S_k antes de ser alcançado por S_i. Na técnica CT, conforme já explicado, um fluxo alvo para S_m é selecionado e toda informação (i.e., unidades de dados do objeto) recebida pelos clientes originalmente pertencentes à S_j (enquanto estavam escutando S_k antes da ocorrência da união) é simplesmente descartada. Para um acesso seqüencial, esta decisão de descarte não aumenta a banda do servidor (muito embora impacte na banda do cliente), pois esta mesma informação precisa ser retransmitida de qualquer forma para atender os clientes originalmente pertencentes ao fluxo S_i. Por outro lado, para um acesso não-seqüencial, esta decisão pode aumentar a banda do servidor, pois a informação descartada não é mais necessariamente retransmitida, conforme explicamos a seguir.

Assuma que, logo após a união dos fluxos S_i e S_j, que resulta no fluxo S_m, todos os clientes originalmente pertencentes à S_i decidam, por exemplo, realizar um salto para trás (ou para frente) (i.e., *Jump Backwards* ou *Jump Forwards*) do atual ponto em exibição do objeto. Neste caso, se os clientes originalmente pertencentes à S_j descartarem a informação em *buffer* (recebida de S_k), como ocorre na técnica CT, esta mesma informação precisará ser retransmitida novamente, não por causa dos clientes originalmente pertencentes à S_i, mas por causa dos clientes originalmente pertencentes à S_j. A decisão de descarte nesta situação é, portanto, indevida, pois os clientes de S_j já tinham esta informação em *buffer* antes da união. Para resolver isso, na técnica MI, conforme já explicamos, um fluxo alvo S_t é então selecionado e designado exclusivamente para os clientes de S_m que originalmente pertenciam à S_i, e os clientes que originalmente pertenciam à S_j não são afetados, i.e., não fazem descarte de dados e continuam tendo S_k como fluxo alvo. Note que S_t pode ser um fluxo diferente de S_k, pois os clientes têm um acesso não-seqüencial e assim podem, portanto, abrir fluxos de dados em qualquer instante e em qualquer unidade de dados do objeto. Assim, temos em MI a possibilidade de um mesmo fluxo com diferentes grupos de clientes associados, cada grupo com um fluxo alvo distinto.

No entanto, é preciso dizer que existem duas situações específicas, advindas desta segunda diferença acima explicada, que podem favorecer a técnica CT em relação à técnica MI. A primeira situação relaciona-se ao fato de que o eventual novo fluxo alvo S_t, designado para os clientes de S_i depois que S_i se une à S_j, pode estar relativamente mais próximo ao fluxo S_j do que aquele originalmente designado para o próprio S_j. Por exemplo, admita que, antes da união de S_i com S_j, o fluxo alvo de S_j, que é o fluxo S_k, esteja à distância de 20 unidades de dados. Admita que imediatamente antes da união de S_i com S_j, um novo fluxo S_t é aberto no sistema e este está à distância de apenas 3 unidades de S_j. Considere que neste instante ocorre a união de S_i com S_j, resultando no fluxo S_m. Considerando este cenário, analisemos a seguir o que ocorre nas técnicas MI e CT.

Na técnica CT, todos os clientes de S_m (originalmente pertencentes ao fluxo S_i ou S_j) são beneficiados pelo fato da identificação do novo alvo indicar o fluxo mais próximo S_t. Na técnica MI, os clientes de S_m, originalmente pertencentes à S_i, são beneficiados pela identificação do fluxo mais próximo S_t; entretanto, os clientes de S_m, originalmente pertencentes à S_j, permanecem tendo S_k como fluxo alvo. Nesta situação, CT pode apresentar uma melhor otimização de banda que MI, pois o fluxo alvo mais próximo atende indistintamente a todos os clientes de S_m. Note que uma maior proximidade do fluxo alvo leva a um menor número de fluxos no sistema, pois o fluxo que tenta alcançar o alvo pode ser extinto mais cedo, redundando conseqüentemente em otimização de banda.

A segunda situação é descrita a seguir. Na técnica MI, se os clientes de S_j não possuem um fluxo alvo, estes clientes ainda permanecerão sem um fluxo alvo mesmo após a união de S_i com S_j. Na técnica CT, isto não ocorre, pois o fluxo alvo é escolhido para todos os clientes de S_m, independentemente de serem originalmente pertencentes à S_i ou S_j. Contudo, este fato, não traz considerável vantagem para CT. A explicação é que, devido à operação intrínseca do paradigma de HSM (abertura e busca por fluxos de forma exaustiva) e ainda à condição de termos interatividade, a probabilidade de um cliente não ter um fluxo alvo associado é praticamente desprezível. Para analisar melhor este aspecto, implementamos neste trabalho uma variante de MI, que denominamos MI-var. A única diferença para MI

é que, na técnica MI-var, após ocorrer a união de S_i com S_j, resultando no fluxo S_m, os clientes originalmente pertencentes à S_j, que não possuem um fluxo alvo, são também considerados para a designação de um eventual novo fluxo alvo S_t.

4.4 Limiares de Tempo Delta

Nesta seção discorremos sobre os limiares de tempo inerentes às novas técnicas propostas, a saber: δ_{after}, δ_{before}, δ_{merge} e δ_{MI}. Para o primeiro, fazemos a derivação analítica de um valor aproximado e, para os demais, discutimos sobre a importância do parâmetro em si e sobre patamares de valores aceitáveis.

4.4.1 Delta After – δ_{after}

Determinar um valor ótimo para o parâmetro δ_{after} não é uma tarefa trivial. Isto porque não existe um modelo analítico simples para representar o comportamento interativo do cliente. Alternativamente, fazemos então a derivação de um valor aproximado para este parâmetro. A questão de quão distante está a aproximação obtida do valor ideal é discutida na seção referente aos resultados de simulação (Seção 4.5). Lá apresentamos indícios de que esta aproximação é satisfatória a depender do nível de interatividade do cenário.

As seguintes suposições são consideradas em nossa derivação analítica: (i) o processo de chegadas de requisição é representado por uma processo de Poisson; (ii) seja X uma variável aleatória que representa a primeira unidade (unidade inicial) de um segmento (intervalo de unidades consecutivas de dados) do objeto solicitado por uma requisição. Assumimos que X tem distribuição uniforme. Em outras palavras, a probabilidade de acesso a cada uma das unidades do objeto é exatamente a mesma; (iii) o padrão de acesso é tal que cada requisição solicita sempre um segmento de tamanho médio L do objeto; (iv) apenas requisições, cujos segmentos associados possuem o mesmo valor de X, podem compartilhar fluxos entre si.

Em relação a primeira suposição, temos a argumentação a seguir. Almeida et al. [73] mostram que o intervalo entre chegadas para o servidor de ensino a distância

eTeach [84] segue uma distribuição Pareto. Outrossim, estudos que realizamos a partir de *logs* reais do sistema MANIC [85, 86], da Universidade de Massachusetts, mostraram-nos que a distribuição do intervalo entre chegadas é aproximadamente Lognormal (veja exemplos da Figura 4.2). Nestes estudos utilizamos o Método dos Momentos e o conceito de *Mean Squared Error* [87, 88, 89]. Diante das informações que dispomos, o intervalo entre chegadas em sistemas reais com interatividade tende, portanto, a possuir uma significativa variância, sendo assim adequadamente modelado por uma distribuição cuja função de distribuição complementar tem um decaimento mais lento que o exponencial. Dessa forma, a distribuição exponencial de mesma média leva a estimativas pessimistas em relação a utilização da banda do servidor. Isto porque chegadas em rajada tendem a utilizar menos banda, pois os *patches* necessários tendem a ser menores (Figura 4.2). Por último, registramos ainda que esta primeira suposição também foi considerada em trabalhos anteriores, como [72, 26], que analisam as influências de um acesso não-seqüencial do cliente sobre a banda do servidor.

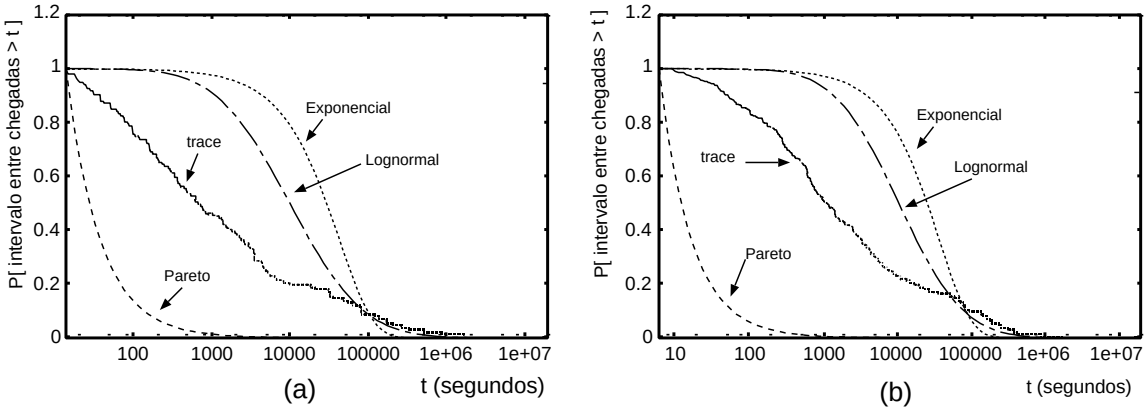


Figura 4.2: Distribuições do intervalo entre chegadas.

Nossa suposição de mesma probabilidade de acesso para as unidades do objeto é baseada nos resultados dos trabalhos de Almeida et al. [73, 77], considerando o servidor de ensino a distância eTeach [84]. É verificado que esta tendência torna-se mais acentuada com o aumento da popularidade do objeto. Já a suposição de que as requisições sejam para segmentos de tamanho médio L do objeto encontra suporte nos trabalhos anteriores de Almeida et al. [73], Vernon et al. [72], Jin e Bestavros [26], e Gupta et al. [90]. Estes trabalhos analisam e/ou constatam o padrão de acesso aqui escolhido, além de outros possíveis de ocorrência em um

ambiente de interatividade. E, por último, a suposição de que apenas as requisições, cujos intervalos associados têm a mesma unidade inicial (i.e., que possuem o mesmo valor da variável aleatória X), são permitidas de compartilhar o fluxo inicialmente disparado por uma delas, é vista como uma consideração pessimista em termos de otimização da banda. Isto porque chegadas, cujos segmentos possuem diferentes unidades iniciais, naturalmente poderiam também compartilhar fluxos entre si. A principal vantagem é de fato a simplicidade matemática que é então advinda desta suposição.

Com as suposições argumentadas acima, fazemos a derivação do valor aproximado do parâmetro δ_{after} a seguir. Considere um servidor de objeto único. Seja a taxa de chegadas de requisições no servidor igual a β e ainda admita o objeto dividido em T unidades (indivisíveis), i.e., o objeto é constituído das unidades $u_1, \ldots, u_T$. Como as chegadas obedecem a um processo de Poisson e o acesso a essas unidades é uniforme, temos então que o processo de chegadas para cada unidade u_i, $i \in \{1, \ldots, T\}$, é um processo de Poisson de taxa $\frac{\beta}{T}$. Este resultado é baseado na *propriedade de decomposição* de um processo de Poisson [78].

Considere agora um cenário em que existem apenas chegadas de requisições para a unidade u_i, $i \in \{1, \ldots, T\}$, do objeto. Admita que a primeira chegada ocorre no instante $t = t_0$ e que esta solicita um tamanho médio L do objeto. Este tamanho é então enviado pelo servidor através de um fluxo *multicast*. Seja então D_i o limiar de tempo permitido para que chegadas posteriores possam também escutar o fluxo aberto inicialmente pela primeira chegada e então recebam a parte perdida (*patch*) diretamente do servidor através de um fluxo *unicast*. A próxima chegada, após o limiar de tempo D_i, reinicia o processo descrito. A derivação do valor ótimo de D_i corresponde ao valor aproximado de δ_{after} que almejamos obter. Como esta derivação independe da unidade específica u_i, doravante omitimos o índice i e, por simplicidade de notação, fazemos $\gamma = \frac{\beta}{T}$. O cenário de transmissão do tamanho médio L do objeto e dos *patches* associados é simplificadamente ilustrado na Figura 4.3.

Note que em média existem $1 + \gamma D$ chegadas compartilhando o fluxo *multicast* inicialmente aberto. Dentre estas, γD chegadas são atendidas também por fluxos *unicast*. Portanto, a taxa de chegadas para fluxos *unicast* é $\gamma(\frac{\gamma D}{1+\gamma D})$. Como cada

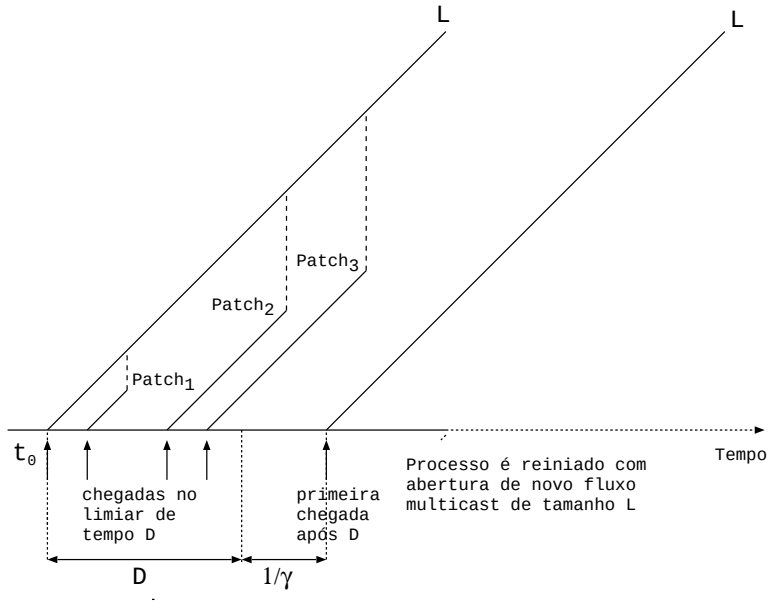


Figura 4.3: Cenário de transmissão

fluxo *unicast* existente tem uma duração média de $\frac{D}{2}$ [91], através da formulação de Little [76], obtemos então o número médio de fluxos *unicast* N_u dado na Equação 4.1.

$$N_u = \gamma(\frac{\gamma D}{1+\gamma D})\frac{D}{2} \tag{4.1}$$

Além disso, como os fluxos *multicast* de tamanho L são iniciados em média a cada intervalo $D + \frac{1}{\gamma}$, temos, também através da formulação de Little, o número médio de fluxos *multicast* N_m dado na Equação 4.2.

$$N_m = \frac{L}{D+\frac{1}{\gamma}} \tag{4.2}$$

Somando-se as Equações 4.1 e 4.2 temos o resultado apresentado na Equação 4.3. E, finalmente, minimizando esta soma e substituindo $\gamma = \frac{\beta}{T}$, temos o valor ótimo de D dado na Equação 4.4. Este valor corresponde ao valor aproximado do parâmetro δ_{after}.

$$N_{u+m} = \frac{L+\frac{\gamma D^2}{2}}{D+\frac{1}{\gamma}} \tag{4.3}$$

$$D_{otimo} = \frac{\sqrt{2\frac{\beta}{T}L+1}-1}{\frac{\beta}{T}} \tag{4.4}$$

Lembramos que β é a taxa do processo de Poisson relacionado ao processo de chegadas de requisições (considerando a interatividade inclusive) que chegam no sistema, L é o tamanho médio solicitado pelas requisições, e T é o número de unidades (indivisíveis) do objeto armazenado no servidor.

Note que a formulação obtida para D_{otimo} é semelhante àquela obtida para a janela ótima de *Patching* (Equação 3.2). Ela é tão mais precisa quanto maior for a proximidade entre o modelo analítico que utilizamos para sua derivação e o modelo da carga real. Esta proximidade pode ser avaliada a partir de estatísticas da carga real. Por exemplo, se o desvio padrão de L é significativo ou a distribuição de X não é uniforme, o valor obtido para D_{otimo} pode ser muito diferente do valor ideal do parâmetro δ_{after}. Nestes casos, alternativamente, redefinimos T como o número de valores diferentes que a variável aleatória X pode assumir.

Para ilustrar, suponha que ocorre um número significativo de requisições para um objeto de tamanho $T = 1000$ armazenado no servidor. Entretanto, X assume apenas três valores distintos dentre os 1000 possíveis. Daí, fazemos T igual a 3 em vez de 1000. Esta redefinição empírica de T reflete a tentativa de adequação do modelo analítico, que admite uma distribuição de acesso uniforme, à verdadeira distribuição de acesso presente na carga real. Ainda podemos usar os valores D_{piso} e D_{teto}, definidos na Equação 4.5, com fins de estabelecer um intervalo de confiança para o valor obtido para D_{otimo} nesta situação. Note que quanto menor for o desvio padrão associado, maior aplicabilidade prática possui o intervalo de confiança estimado.

$$D_{valor} \triangleq \frac{\sqrt{2\frac{\beta}{T}L_{valor}+1}-1}{\frac{\beta}{T}} \ , \tag{4.5}$$

onde $valor = piso$ ou $teto$, $L_{piso} \triangleq (L - \sigma)$, $L_{teto} \triangleq (L + \sigma)$, e σ é o desvio padrão de L.

Notadamente, os resultados advindos das Equações 4.4 e 4.5 pressupõem o pleno conhecimento prévio dos parâmetros associados à carga real, o que na prática não é sempre verdade. A solução para isso é a estimativa dinâmica destes parâmetros à

medida que ocorre o atendimento das requisições ao longo do tempo. Por exemplo, podemos adaptar o procedimento de determinação dinâmica de [41] e utilizá-lo neste nosso caso. A idéia base seria estimar um valor inicial (p. ex., o valor de L), e a partir deste, fazer incrementos ou decrementos ao mesmo tempo que observa-se a distribuição da banda, para verificar se alguma otimização pode ou não ser conseguida.

4.4.2 Delta Before – δ_{before}

A determinação do limiar de tempo δ_{before} é, como já dito anteriormente, influenciada pela tolerância que o cliente tem em relação à descontinuidade do serviço que ele sofre por receber uma parte do filme que não é solicitada. Enfatizamos que esta descontinuidade se refere ao fato do cliente receber um ponto anterior ao solicitado, ou seja, o cliente não deixa de receber o que solicitou, mas eventualmente recebe algo à mais do que de fato foi pedido.

Muito embora ainda seja uma lacuna na literatura, conforme comentamos anteriormente, é possível conjeturamos que esta tolerância se estabelece em conformidade com o tipo de aplicação. Por exemplo, se pensarmos em filmes sob demanda é de se esperar que descontinuidades em torno de, por exemplo, até 3-5% do tamanho total do objeto não cause um nível de insatisfação tão alto no cliente. Já para uma aplicação como ensino a distância, 3-5% pode ser considerado bastante irritante, pois o aluno pode estar interessado justamente em um ponto muito específico da aula.

Nos nossos experimentos (Seção 4.5) fixamos o valor de δ_{before} em 10 s. Fomos mais rigorosos que Netto e Gorza [70, 71] que consideraram 60 s para este limiar. A explicação para este maior rigor deu-se por objetivarmos estar mais próximos do chamado serviço contínuo e, ao mesmo tempo, termos uma análise mais *pura* da eficiência real das técnicas propostas. Por análise mais *pura* entenda-se aquela capaz de mensurar a otimização conseguida pela capacidade de compartilhamento da técnica, e não pela descontinuidade de serviço que o cliente venha eventualmente a tolerar. Reconhecendo, contudo, este rigor, ao final de todos os experimentos

também fizemos análises específicas para avaliar o quanto poderíamos ganhar em otimização da banda ao aumentarmos este valor de 10 s para valores maiores e dentro de um intervalo que conjeturamos como razoável para aplicações diversas. Conforme veremos mais à frente, os resultados obtidos são bastante atrativos.

4.4.3 Delta Merge – δ_{merge} e Delta MI – δ_{MI}

Para determinação do valor ideal destes limiares, fazemos experimentações conforme explicamos a seguir.

- Para δ_{merge}: inicialmente fazemos este limiar igual a δ_{after}. Note que δ_{after} é aplicado para ação de união de um cliente com um fluxo *multicast* e, portanto, pode ser considerado como um valor mínimo para δ_{merge} pelo fato deste relacionar-se a ações de união entre fluxos *multicast* e, daí, potencialmente de um conjunto de clientes. Nos experimentos seguintes, fazemos então δ_{merge} assumir valores em função do valor de δ_{after} e analisamos a eficiência conseguida a partir das distribuições de banda. Daí, observando os resultados, identificamos os valores ideais.

- Para δ_{MI}: atribuímos diferentes valores, inclusive admitindo a sua não restrição, que corresponde a não empregar este limiar. A partir das distribuições de banda, observamos então o valor que provê o maior nível de otimização.

Esta conduta experimental se justifica pela impossibilidade da derivação analítica destes limiares de forma suficientemente trivial devido a complexidade inerente do modelo de interatividade do cliente, conforme também comentamos quando da determinação de δ_{after}.

4.5 Avaliação de Performance

4.5.1 Organização

Esta seção apresenta os principais experimentos realizados durante a elaboração deste trabalho e seus resultados. Consideramos, para efeito de análise competitiva com as novas técnicas propostas PIE, PIC e MI (MI-var), as seguintes outras técnicas: clássico esquema de *Patching*, PI e CT.

A decisão de incluir o clássico esquema de *Patching* é respaldada pelo propósito principal de mensurar a inadequação de uma técnica predominantemente de acesso seqüencial em cenários com interatividade. Em relação à técnica PI, a justificativa principal, conforme já comentado na Subseção 4.2.2, deve-se ao fato de ser uma técnica bem recente e poder ser vista como uma evolução natural de propostas anteriores (p. ex., SAM *protocol* [28] e a proposta de Abram-Profeta e Shin [30]) no que se refere ao uso planejado do *buffer* local do cliente nas decisões de união e abertura de fluxos do sistema; além de também ser significativamente mais simples em sua concepção que, por exemplo, a técnica SRMDRU [67], que necessita de fluxos com taxas de transmissões diferenciadas, e que a técnica BEP [29], a qual utiliza-se de uma estrutura de união de fluxos baseada em árvores de até três níveis. Por fim, a técnica CT é também incluída por se tratar possivelmente de um dos mais eficientes esquemas baseados em HSM, e ainda apresentar uma simplicidade extremamente atrativa no que se refere ao mecanismo de decisões para compartilhamento de fluxos, conforme comentado anteriormente na Subseção 4.2.3.

A seguir, apresentamos a organização desta seção. A Subseção 4.5.2 discute as principais métricas utilizadas para a análise competitiva entre as técnicas. Na Subseção 4.5.3 encontram-se os principais objetivos dos experimentos realizados. As cargas sintéticas, obtidas a partir de servidores multimídia reais são descritas na Seção 4.5.4. Por fim, os experimentos realizados estão na Subseção 4.5.5.

4.5.2 Métricas

As análises competitivas realizadas neste trabalho utilizam as seguintes principais métricas: consumo médio de banda do servidor, valor de pico de banda do servidor, distribuição de banda do servidor, razões competitivas entre as distribuições de banda do servidor, banda agregada dos clientes e trabalho médio do servidor.

Consumo médio e valor de pico de banda são duas métricas comumente utilizadas em experimentos comparativos. Já a distribuição de banda torna-se importante pelo fato de considerarmos em nossos experimentos a interatividade dos clientes. Isto faz com que a banda possa variar significativamente ao longo do tempo. Conseqüentemente, o simples cômputo do valor médio e/ou do valor de pico não permite(m) uma análise mais detalhada das técnicas. As razões competitivas [65] aqui utilizadas são assim definidas [50]: $RMax = max(\frac{P[Banda>k]_{tec_1}}{P[Banda>k]_{tec_2}})$ e $RMin = min(\frac{P[Banda>k]_{tec_1}}{P[Banda>k]_{tec_2}})$, para todo k inteiro desde que $P[Banda > k] \geq 10^{-4}$ (por precisão da simulação), onde tec_1 refere-se à técnica em relação a qual queremos comparar a técnica tec_2. Note que estas razões representam um meio eficiente de quantificar a diferença entre as respectivas distribuições de banda de duas técnicas tec_1 e tec_2 que desejamos comparar. Definimos a banda agregada dos clientes como o número de clientes *ativos*, ou seja, o número de clientes realizando *streaming* no sistema. Esta métrica nos dá um sentimento do uso real de canais por parte dos clientes. Por fim, a métrica trabalho médio é utilizada para mensurar a complexidade do sistema e definimos da forma explicada a seguir.

O trabalho médio é função do número médio de mensagens recebidas pelo servidor e das operações executadas para o tratamento das mesmas. A operação de maior custo é a busca por um fluxo *multicast*. Faz-se então a análise de pior caso e admite-se que a operação de busca seja executada em tempo $O(n)$, onde n é o número de fluxos *multicast* em andamento no sistema. Aspectos relacionados a construção de árvores multicast e procedimentos de busca por meio de estruturas de dados otimizadas não fazem do escopo de discussão deste texto. Nosso objetivo é uma avaliação de cunho comparativo e não absoluto. O servidor pode receber quatro tipos de mensagens: solicitação de unidade de dados do objeto, solicitação de término de processo de união de fluxos, solicitação de término de *patch* e, por

último, solicitação de fim de transmissão (i.e., o cliente pára de receber dados do servidor, podendo ficar em estado de *pause*, ler do *buffer* local, ou ainda terminar sua sessão). TB, TM, TP e TF, definidos individualmente na Tabela 4.1, denotam os números médios de mensagens no sistema e são obtidos, assim como também n, a partir do modelo de simulação. A Tabela 4.2(a) sintetiza as complexidades de tempo relativas às operações devido a cada tipo de mensagem, onde $O(C)$ denota uma complexidade de tempo constante. A Tabela 4.2(b) apresenta as equações para calcular o trabalho médio. Estes valores, por sua vez, são obtidos/derivados a partir da análise de pior caso, apresentada em seguida. Por último, convém destacar que, apesar de não aparecerem explicitamente nas equações, os limiares de tempo δ têm influência direta no trabalho médio, pois podem modificar o número de mensagens no sistema.

Símbolo	Definição
TB	núm. médio de msg. solicitando por unidade de dados
TM	núm. médio de msg. solicitando término de processo de união de fluxos
TP	núm. médio de msg. solicitando término de *patch*
TF	núm. médio de msg. solicitando fim de transmissão

Tabela 4.1: Número médio de mensagens

Técnica	TB	TM	TP	TF
Patching	$O(n)$	—	$O(C)$	$O(C)$
PI	$O(n)$	—	$O(C)$	$O(C)$
PIE	$O(n)$	$O(C)$	$O(C)$	$O(C)$
PIC	$O(2n)$	$O(n)$	$O(n)$	$O(n)$
MI, MI-var	$O(2n)$	$O(n)$	—	$O(n)$
CT	$O(2n)$	$O(n)$	—	$O(n)$

(a) Complexidade de tempo

Técnica	Fórmulas
Patching	$TB * n + TP + TF$
PI	$TB * n + TP + TF$
PIE	$TB * n + TM + TP + TF$
PIC	$TB * 2n + (TM + TP + TF) * n$
MI, MI-var	$TB * 2n + (TM + TF) * n$
CT	$TB * 2n + (TM + TF) * n$

(b) Trabalho médio

Tabela 4.2: Complexidade de tempo e trabalho médio do servidor

Análise de Pior Caso

Aqui fazemos a análise de complexidade de tempo de pior caso para as técnicas *Patching*, PI, PIC, PIE, MI (MI-var) e CT. Para tanto, focamos simplificadamente

apenas as operações que de fato impactam na complexidade final. Lembramos que os detalhes relativos aos algoritmos de operação destas técnicas estão apresentados anteriormente nas Subseções 4.2.2 e 4.2.3, e Seção 4.3.3.

- a) Mensagem relacionada à solicitação para unidade de dados (TB).

  - Para PI (*Patching*): (i) se o cliente estiver em algum grupo, desalocá-lo do grupo; (ii) realizar uma busca por um fluxo alvo; (iii) caso exista um fluxo alvo, abrir um *patch* para receber a parte inicialmente perdida e fazer o cliente escutar o fluxo alvo simultaneamente. Caso contrário, abrir um novo fluxo. A complexidade final é então $O(n)$.

  - Para PIE: (i) se o cliente estiver em algum grupo, desalocá-lo do grupo; (ii) realizar uma busca por um fluxo alvo; (iii) caso exista um fluxo alvo, abrir um *patch* para receber a parte inicialmente perdida e fazer o cliente escutar o fluxo alvo simultaneamente. Caso contrário, abrir um novo fluxo e verificar se existe um fluxo suficientemente próximo para unir-se a este novo fluxo. A complexidade final é então $O(n)$.

  - Para PIC: (i) se o cliente estiver em algum grupo, desalocá-lo do grupo; (ii) caso existam órfãos, realizar uma busca por um fluxo substituto para órfãos; (iii) realizar uma busca por um fluxo alvo; (iv) caso exista um fluxo alvo, abrir um *patch* para receber a parte inicialmente perdida e fazer o cliente escutar o fluxo alvo simultaneamente. Caso contrário, abrir um novo fluxo e verificar se existem fluxos suficientemente próximos para unirem-se a este novo fluxo. A complexidade final é então $O(2n)$.

  - Para MI (MI-var e CT): (i) se o cliente estiver em algum grupo, desalocá-lo do grupo; (ii) caso existam órfãos, realizar uma busca por um fluxo substituto para órfãos; (iii) realizar uma busca por um fluxo alvo; (iv) caso exista um fluxo alvo, abrir um novo fluxo *multicast* para receber a parte inicialmente perdida e fazer o cliente escutar o fluxo alvo simultaneamente. Caso contrário, abrir um novo fluxo *multicast*. A complexidade final é então $O(2n)$.

- b) Mensagem relacionada à solicitação para fim de *patch* (TP).

- Para PI (*Patching*): extinguir um fluxo de *patch* associado a um grupo. A complexidade final é então constante e igual a $O(C)$.

- Para PIE: extinguir um fluxo de *patch* associado a um grupo. A complexidade final é então constante e igual a $O(C)$.

- Para PIC: (i) extinguir um fluxo de *patch* associado a um grupo; (iii) caso o grupo não tenha mais *patches* associados, realizar a busca por um fluxo destino. A complexidade final é então $O(n)$.

- c) Mensagem relacionada à solicitação para fim de união de fluxos (TM).

  - Para PIE: passar o cliente para o grupo alvo e retirá-lo do grupo anterior. A complexidade é constante e igual a $O(C)$.

  - Para PIC: (i) passar o cliente para o grupo alvo e retirá-lo do grupo anterior; (ii) caso existam órfãos no grupo anterior, realizar uma busca por um fluxo substituto. A complexidade final é $O(n)$.

  - Para MI (MI-var e CT): situação idêntica àquela discutida para PIC e a complexidade é também $O(n)$.

- d) Mensagem relacionada à solicitação de fim de transmissão (TF).

  - Para PI (*Patching*): desalocar cliente do grupo. A complexidade final é então constante e igual a $O(C)$.

  - Para PIE: desalocar cliente do grupo. A complexidade final é então constante e igual a $O(C)$.

  - Para PIC: (i) desalocar cliente do grupo; (ii) caso existam órfãos, realizar uma busca por um fluxo substituto para órfãos. A complexidade final é $O(n)$.

  - Para MI (MI-var e CT): (i) desalocar cliente do grupo; (ii) caso existam órfãos, realizar uma busca por fluxo substituto para órfãos. A complexidade final é então $O(n)$.

A partir desta análise de pior caso, observamos a importância da realização de busca por um fluxo no sistema, pois termina por definir as complexidades de tempo que não são constantes.

4.5.3 Objetivos

Para os diferentes cenários de interatividade e considerando o emprego das novas técnicas, nossos experimentos têm os objetivos macros comentados a seguir.

- 1) Utilizando as métricas discutidas há pouco, realizar análises competitivas entre as novas técnicas PIE, PIC, MI (MI-var) e as seguintes técnicas já propostas na literatura: clássico *Patching*, PI e CT. O propósito principal é identificar a técnica mais eficiente do paradigma de *Patching* e do paradigma de HSM e, também, a diferença de performance entre estas.

- 2) Quantificar a importância da utilização de *buffer* local, de tamanho igual ao do objeto, com fins de armazenar permanentemente todas as unidades (do objeto) enviadas pelo servidor. Com isto, uma unidade anteriormente armazenada pode ser recuperada em qualquer instante de tempo desejado. A idéia é então que, antes de se fazer uma solicitação ao servidor, o cliente primeiro faça uma pesquisa em seu *buffer* local para saber sobre a disponibilidade ou não da unidade desejada.

- 3) Determinar valores aproximadamente ideais para os parâmetros δ_{after}, δ_{merge} e δ_{MI} e, também, verificar a acurácia da derivação analítica feita neste trabalho para o valor aproximado de δ_{after}.

- 4) Avaliar o nível de otimização conseguido na banda do servidor devido à tolerância a pequenas *descontinuidades* no serviço de VoD oferecido. Estas descontinuidades são implementadas a partir da atribuição de diferentes valores para o parâmetro δ_{before}.

Em relação ao *objetivo 2* cabe ainda acrescentar o seguinte. Convém relembrar que as técnicas PIE, PIC e MI, assim como também PI, CT e *Patching*, fazem uso em sua concepção base de um modelo em que o *buffer* local do cliente é de tamanho equivalente a pelo menos metade do tamanho do objeto, e é usado estritamente para fins de sincronismo e, conseqüentemente, união com o fluxo alvo escolhido. Ressaltamos então que, com a utilização da idéia de *buffer* local do cliente de tamanho

do objeto e armazenamento de todas as unidades recebidas, a função de sincronismo continua existindo. O que ocorre é a adição da possibilidade de armazenar todas as unidades do objeto com fins de evitar, como já comentado, que unidades já armazenadas tenham de ser novamente requisitadas ao servidor.

Para maior simplicidade de referência e clareza de discussão, utilizamos o sufixo SB (*Simple Buffer*) para indicar que a técnica faz uso apenas do *buffer* simples convencional, de tamanho igual a pelo menos metade do objeto e com fins exclusivos de sincronismo, conforme originalmente proposto nos modelos. E usamos o sufixo CB (*Complete Buffer*), para indicar que a técnica relacionada faz uso de *buffer* completo, de tamanho do objeto e utilizado para prover sincronismo e, também, para armazenar todas as unidades do objeto enviadas pelo servidor, evitando assim solicitações por unidades anteriormente já armazenadas.

A banda do servidor é aqui sempre medida em unidades da taxa de exibição do objeto, ou equivalentemente, em número de canais em uso simultâneo no servidor, e os valores TB, TM, TP e TF em mensagens por minuto. Destacamos que nos experimentos com *buffer* simples (SB), é possível observar que os valores TB e TF tendem a ser independentes da técnica utilizada, pois passam a refletir mais puramente o comportamento interativo dos clientes, sem quaisquer influências do sistema. Por fim, devido ao significativo número de experimentos e resultados obtidos, resolvemos, para maior facilidade de exposição e entendimento global, selecionar apenas os principais resultados para apresentar neste capítulo, e deixamos os demais contidos nos apêndices deste texto.

4.5.4 Cargas Sintéticas

Nossas simulações usam quatro cargas sintéticas obtidas a partir de servidores reais nos trabalhos de [77, 74]. Dois destes servidores são de ensino a distância (eTeach e MANIC). O outro é o *Universo Online* (UOL), um dos maiores provedores de conteúdo da américa latina e volta-se essencialmente para a área de entretenimento. Os objetos são divididos em unidades de dados de 1 s e o cliente pode executar as seguintes ações VCR: *Play, Stop, Pause/Resume, Jump Forwards* e *Jump Backwards*.

Cada uma das cargas utilizadas pode ser percebida como um cenário distinto de análise. As principais características das cargas são apresentadas na Tabela 4.3. Note que elas são estatisticamente diferentes, conforme comentamos a seguir.

Tabela 4.3: Cargas sintéticas

Estatística	eTeach	MANIC-1	MANIC-2	UOL
tamanho do objeto T (s)	2199	4175	4126	226
popularidade do objeto N	98	99	100	97
número total de requisições	5146	677	2366	1114
número médio de req/sessão	10.29	1.35	4.73	2.23
tamanho médio do segmento L (s)	118	1190	496	134
desvio padrão de L (s)	143	1184	473	91
coef. de variação de L	1.21	1.00	0.95	0.68
número de diferentes valores assumidos pela v.a. X	≈ 2199	≈ 24	≈ 98	≈ 24

Para a Carga eTeach, temos as Figuras 4.10 e 4.11. Podemos, por exemplo, perceber que a utilização de *buffer* local completo (CB) deve prover alguma otimização da banda, pois existe localidade de acesso (i.e., acesso para as mesmas unidades) não desprezível nos *retornos* (Figura 4.10(a)). Também é possível deduzir-se que praticamente todas as unidades do objeto são acessadas de forma inicial (veja Figura 4.11). O acesso de forma inicial a uma dada unidade i significa que esta unidade aparece como a primeira unidade do segmento (intervalo) de tamanho médio L associado à requisição ocorrida, ou seja, a variável aleatória X assume o valor i (veja detalhes na Seção 4.4.1). Por fim, também ainda pode-se deduzir que há uma certa uniformidade na distribuição do número de acessos às unidades do objeto (veja Figura 4.10(b)).

No caso da primeira carga do servidor MANIC, doravante chamada de MANIC-1, temos as Figuras 4.4 e 4.5. É possível notar, por exemplo, que existe localidade de acesso referente aos *retornos* observados. Entretanto, devido ao baixo nível de interatividade existente (número de requisições por sessão < 1.4), a utilização de *buffer* local completo no cliente pode terminar por não prover a otimização de banda eventualmente imaginada em um primeiro instante de análise. Também a partir

desta caracterização, é possível deduzir-se que não existe, comparativamente com o último cenário, uniformidade na distribuição do número de acessos às unidades do objeto (veja Figura 4.4(b)), e que apenas aproximadamente 24 unidades do objeto são acessadas de forma inicial (veja Figura 4.5).

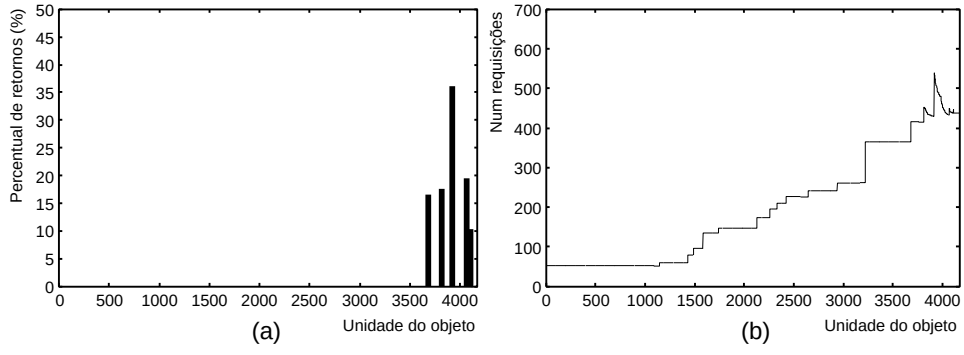


Figura 4.4: (a) Percentual de retornos e (b) Número de requisições.

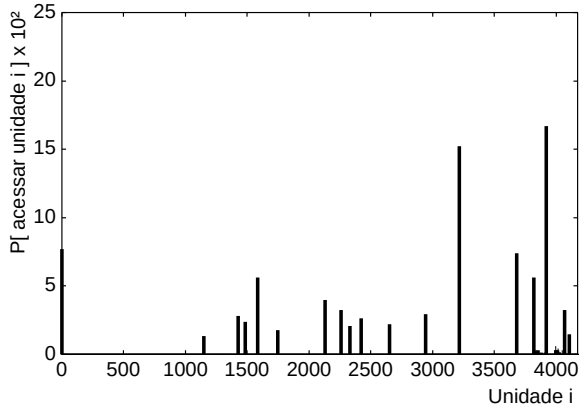


Figura 4.5: Percentual de acesso de forma inicial a unidade i do objeto.

Para a segunda carga do servidor MANIC, doravante chamada de MANIC-2, apresentamos as Figuras 4.6 e 4.7. Note que esta caracterização nos diz que poucas (≈ 98) unidades distintas do objeto são efetivamente acessadas de forma inicial pelas requisições existentes (veja Figura 4.7), e que existe pouca localidade de acesso referente aos *retornos* (Figura 4.6(a)), o que leva a intuir que a utilização de *buffer* local completo no cliente não deve prover significativa otimização na banda do servidor. Também é possível deduzir-se que existe, comparativamente com o último cenário, alguma uniformidade na distribuição do número de acessos às unidades do objeto (veja Figura 4.6(b)).

Por fim, para a carga do servidor UOL temos as Figuras 4.8 e 4.9. É possível

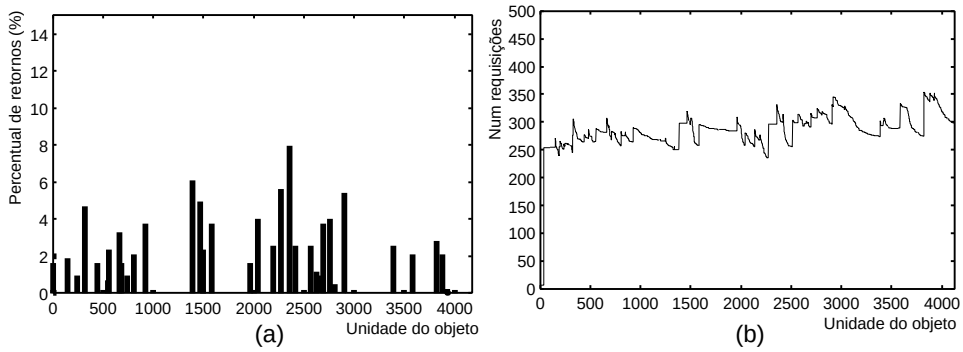


Figura 4.6: (a) Percentual de retornos; (b) Número de requisições.

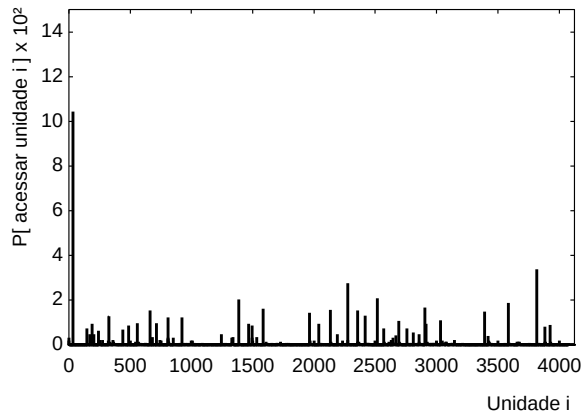


Figura 4.7: Percentual de acesso de forma inicial a unidade i do objeto.

notar que existe significativa localidade de acesso referente aos *retornos* observados (Figura 4.8(a)), implicando que o uso de *buffer* local completo no cliente se mostre bastante conveniente. Também, a partir desta caracterização, é possível destacar a significativa localidade para início de exibição (Figura 4.9), implicando no comprometimento da otimização de banda advinda do uso do parâmetro δ_{before}.

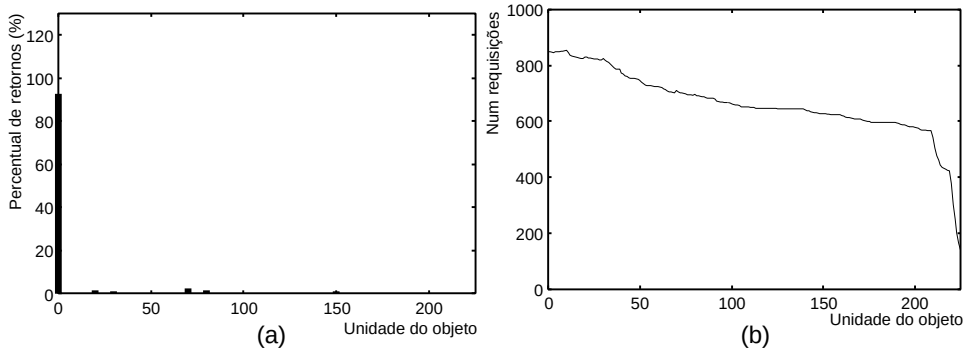


Figura 4.8: (a) Percentual de retornos e (b) Número de requisições.

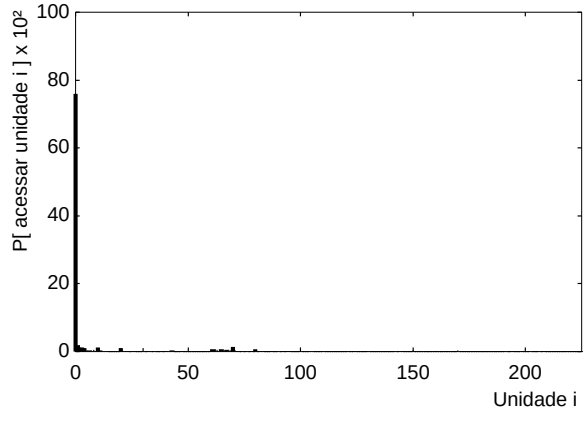


Figura 4.9: Percentual de acesso de forma inicial à unidade i do objeto.

4.5.5 Experimentos

Os resultados de simulação são obtidos usando a ferramenta Tangram-II [80], sobre a qual inclusive já comentamos na Subseção 3.4. Para facilidade de exposição, optamos por apresentar os resultados de acordo com os objetivos elencados na Subseção 4.5.3 e, salvo informado diferentemente, admitimos:

- A chegada de 500 clientes no total seguindo um processo de Poisson de taxa

$\alpha = \frac{N}{T}$. Lembramos que N é a popularidade do objeto e T a duração do mesmo em unidades de dados.

- Um servidor de um único objeto constituído de unidades (indivisíveis) de 1 s cada.
- δ_{merge} igual a δ_{after}, e $\delta_{before} = 10\,\text{s}$.
- δ_{after} fixo e igual a 50% do valor da janela ótima de *Patching* W. Observamos que estimativas diferentes de 50% de W para δ_{after} redundam em resultados comparativos qualitativamente semelhantes.
- A análise do uso do *buffer* local (SB ou CB) considera apenas as técnicas PIE e PIC, e deixa-se, por facilidade de exposição, a análise envolvendo MI para mais à frente na seqüencia dos experimentos.
- Para efeito de justiça, nas análises competitivas específicas entre as técnicas PIE e MI, utilizamos os valores ideais obtidos experimentalmente para os limiares δ_{after} e δ_{merge}.

Análise do Paradigma de Patching

Em todas as cargas consideradas, PI, PIE e PIC apresentam melhor performance que a clássica técnica *Patching*. Este resultado já era esperado, pois *Patching* é um esquema para acesso seqüencial e, portanto, não é capaz de lidar eficientemente com a interatividade dos clientes.

Esta superioridade em performance é quantificada por meio das razões competitivas computadas: $RMin = 1.0$ e $RMax \in [120.2, 3634.9]$ (veja Tabela 4.4). Por exemplo, para o caso de uso de *buffer* completo, $P[Banda > k]$ chega a ser até algumas ordens de grandeza maior para *Patching* do que para PIE. Ilustramos ainda esta diferença especificamente para PIE, por meio das Figuras 4.12(a) e 4.12(b). A primeira mostra a distribuição (CCDF) da banda para PIE-SB e *Patching*-SB, e a segunda mostra para PIE-CB e *Patching*-CB, ambas para a Carga eTeach.

As técnicas PIE e PIC têm performance semelhante em todas as cargas e são ambas, de forma geral, mais eficientes que PI. Esta superioridade em performance

Tabela 4.4: RMin e RMax de PIE, PIC e PI para *Patching*

Técnica	eTeach		MANIC-1		MANIC-2		UOL	
	RMin	RMax	RMin	RMax	RMin	RMax	RMin	RMax
PIE-SB	1.0	450.0	1.0	126.8	1.0	1514.7	1.0	163.3
PIE-CB	1.0	120.2	1.0	261.0	1.0	154.2	1.0	377.7
PIC-SB	1.0	786.8	1.0	117.5	1.0	3634.9	1.0	163.3
PIC-CB	1.0	382.3	1.0	142.0	1.0	390.4	1.0	377.7
PI-SB	1.0	393.87	1.0	7.76	1.0	638.8	1.0	188.3
PI-CB	1.0	210.3	1.0	10.12	1.0	60.6	1.0	346.51

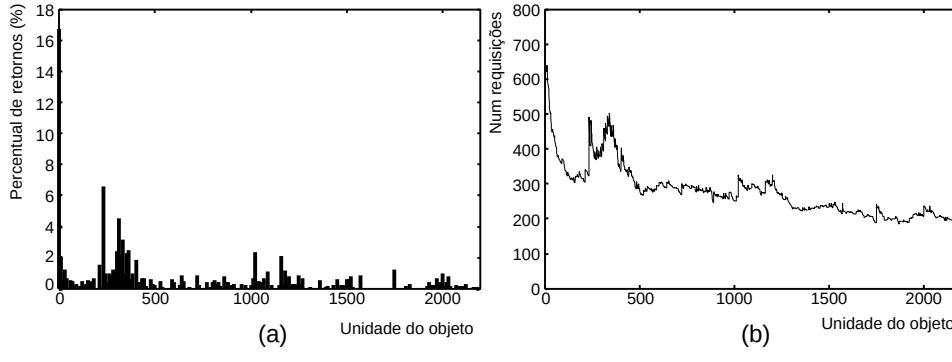


Figura 4.10: (a) Percentual de retornos e (b) Número de requisições.

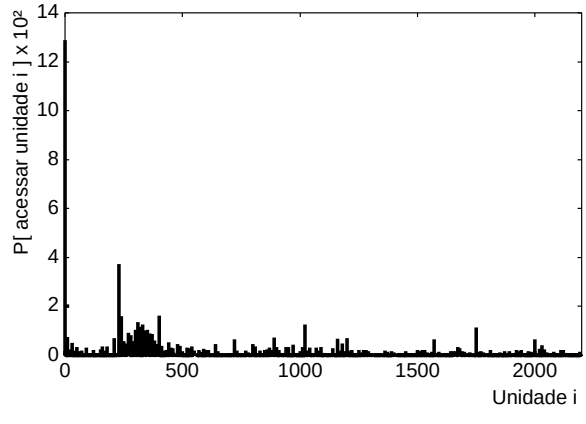


Figura 4.11: Percentual de acesso de forma inicial a unidade i do objeto.

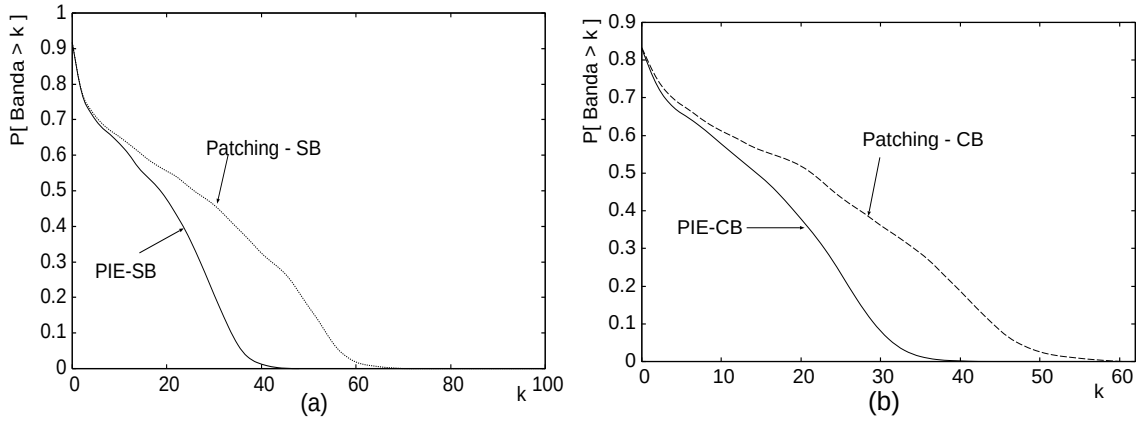


Figura 4.12: (a) CCDF para PIE-SB e *Patching*-SB em eTeach; (b) CCDF para PIE-CB e *Patching*-CB em eTeach.

é quantificada por meio das razões competitivas computadas: $RMin \in [0.5, 1.0]$ e $RMax \in [1.6, 25.8]$ (veja Tabela 4.5). Também para ilustrar este fato e considerando a Carga MANIC-1, apresentamos a distribuição (CCDF) das técnicas PIE e PIC nas Figuras 4.13(b) e 4.14(b). As respectivas curvas de distribuição de banda praticamente se confundem. Devido a sua maior simplicidade de implementação, este fato termina favorecendo PIE.

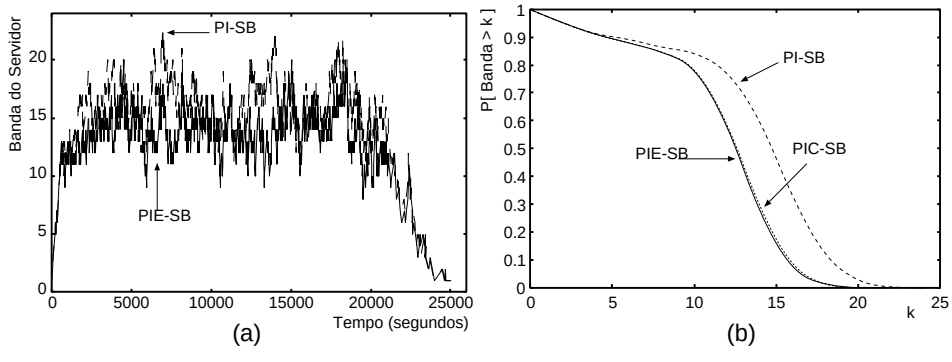


Figura 4.13: (a) Banda para PIE-SB e PI-SB em MANIC-1; (b) CCDF da banda para PIE-SB, PIC-SB e PI-SB em MANIC-1.

A justificativa para que PIC não consiga uma melhor performance que PIE, mesmo sendo uma técnica mais elaborada, se dá pelo fato de TM ser geralmente pequeno, comparativamente com TP. Isto que dizer que os compartilhamentos de fluxos se dão em sua grande maioria por meio de *patches*, tornando PIE e PIC funcionalmente bem parecidas. Entendemos que a tentativa de aumentar a eficiência de PIC passa necessariamente pela mudança da estrutura de união de fluxos de

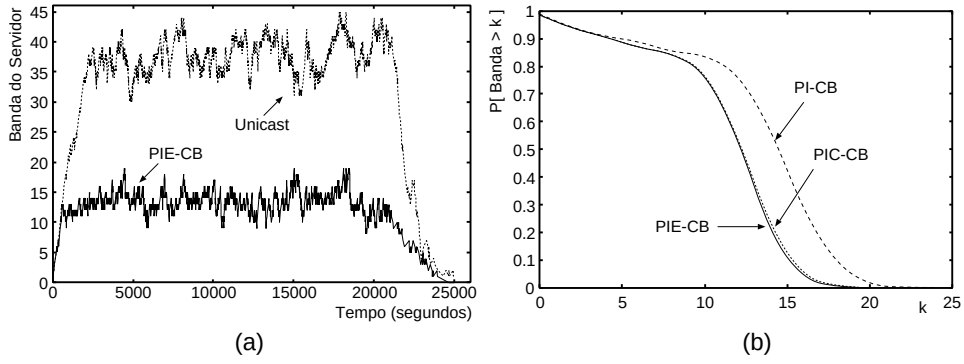


Figura 4.14: (a) Banda do servidor para PIE-CB em MANIC-1; (b) CCDF da banda para PIE-CB, PIC-CB e PI-CB em MANIC-1.

tal sorte que sejam permitidas estruturas de mais de dois níveis. Contudo, essa permissividade pode levar a necessidade de definição de outros limiares de tempo, dificultando a sua implementação prática.

Temos, portanto, como conclusão geral, que PIE é a de melhor performance, seguida proximamente por PIC e com uma satisfatória vantagem sobre PI.

Tabela 4.5: RMin e RMax de PIE e PIC para PI

Técnica	eTeach		MANIC-1		MANIC-2		UOL	
	RMin	RMax	RMin	RMax	RMin	RMax	RMin	RMax
PIE-SB	1.0	1.8	1.0	16.3	0.5	1.6	0.7	2.0
PIE-CB	0.6	1.6	1.0	25.8	0.9	4.0	0.9	14.5
PIC-SB	1.0	2.0	1.0	15.1	0.7	1.7	0.7	2.0
PIC-CB	1.0	2.5	1.0	14.1	1.0	8.9	0.9	14.5

Análise da Importância de *Buffer* Completo (CB)

Para fins de otimização de banda do servidor, o emprego de CB é naturalmente importante nas cargas em que há um número significativo de retornos (*Jump Backwards*) para unidades já transmitidas e armazenadas localmente. Em nossos experimentos, este fato é principalmente notado nas cargas com maior nível de interatividade, i.e., maior número médio de requisições por sessão. Por exemplo, considerando a Carga UOL, observamos que o benefício trazido pelo emprego de *buffer* local completo é bem explícito, conforme ilustrado na Figura 4.15. Por outro

lado, para a Carga MANIC-1, o benefício alcançado, avaliado através das distribuições de banda mostradas na Figura 4.16, é bem discreto por não haver significativo número de retornos, conforme havíamos observado (Figura 4.4(a)).

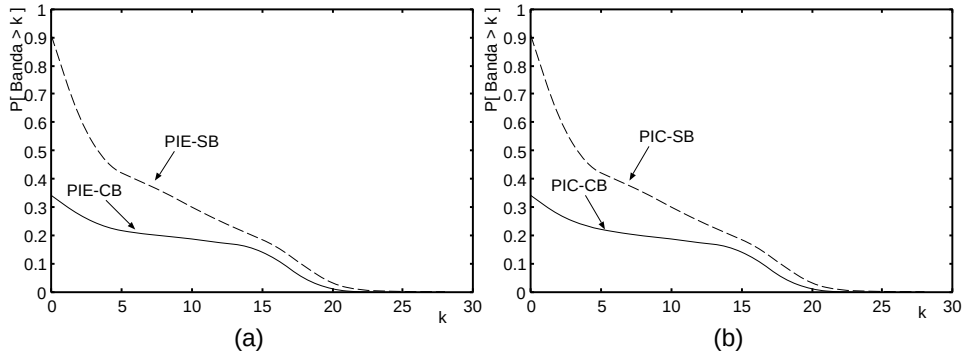


Figura 4.15: (a) CCDF da banda para PIE em UOL; (b) CCDF da banda para PIC em UOL.

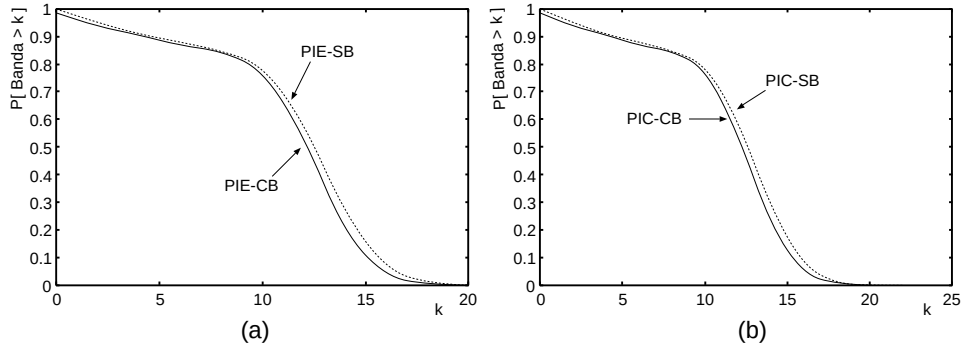


Figura 4.16: (a) CCDF da banda PIE em MANIC-1; (b) CCDF da banda para PIC em MANIC-1.

Em relação a influência de CB na banda agregada dos clientes, observamos reduções significativas sob a mesma condição geral de haver retornos para unidades anteriormente transmitidas, comentada no parágrafo anterior. A Figura 4.17 exemplifica este fato para a técnica PIE e carga UOL. Podemos observar que existe uma otimização significativa do lado do cliente. O valor de pico da banda agregada dos clientes é reduzido em 18.85% (de 122 para 99 canais) devido ao emprego de CB. Já para a Carga MANIC-1, o benefício é bem discreto, conforme pode ser observado na Figura 4.18, onde também é considerada a técnica PIE. Neste cenário as curvas praticamente se confundem. A otimização conseguida é de apenas 2.22% no pico, ficando em sintonia com a conclusão do parágrafo anterior de que a otimização da banda do servidor se dá também de forma pouco aparente.

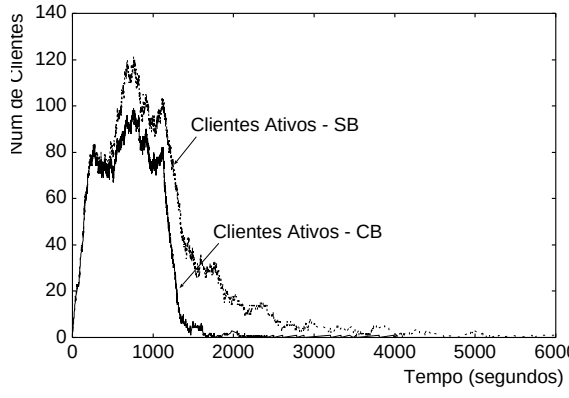


Figura 4.17: Banda agregada (clientes ativos) para PIE em UOL.

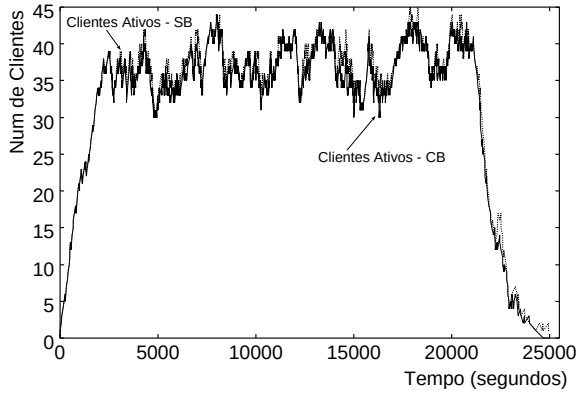


Figura 4.18: Banda agregada (clientes ativos) para PIE em MANIC-1.

Por fim, de forma bem interessante, o emprego de CB em certos cenários provoca o aumento da complexidade de mensagens, enquanto que em outros provoca a redução de complexidade. Por exemplo, para PIE e Carga eTeach, o valor TB aumenta em 31% e para PIC este aumento é de 26%. Já para Carga UOL, percebemos uma redução de complexidade de mensagens devido a CB. Por exemplo, TB e TF têm reduções de 61.76% e 40.92%, respectivamente. Estes resultados estão apresentados nas Tabelas 4.6 e e 4.7. Isto acontece porque existe dois efeitos contrários que se manifestam quando da utilização do CB. Por um lado, o cliente tende a fazer menos requisições por já ter a informação em *buffer* local e, portanto, diminui o número de mensagens. Por outro lado, se houver fragmentação da informação armazenada, o cliente tenderá a alternar a leitura em *buffer* com a leitura a partir de um canal do servidor.

Técnica	B_{max}	**TB**	**TM**	**TP**	**TF**
PIC-SB	48	25.40	0.99	20.64	13.54
PIE-SB	48	25.40	0.72	20.69	13.54
PIC-CB	43	31.90	1.39	24.45	28.91
PIE-CB	45	33.26	0.93	25.96	30.26

Tabela 4.6: Estatísticas para Carga eTeach

Técnica	B_{max}	**TB**	**TM**	**TP**	**TF**
PIC-SB	27	35.30	0.10	23.16	21.92
PIE-SB	27	35.30	0.06	23.17	21.92
PIC-CB	24	13.15	0.04	8.01	12.95
PIE-CB	24	13.13	0.01	7.99	12.93

Tabela 4.7: Estatísticas para carga UOL

Devido à simplicidade de implementação do CB, temos como conclusão geral que o seu emprego é recomendável, pois as chances de otimização que podem ser conseguidas são bem claras e, mesmo que não sejam, o seu emprego não acarreta malefícios severos para o sistema.

Análise do Limiar Delta After - δ_{after}

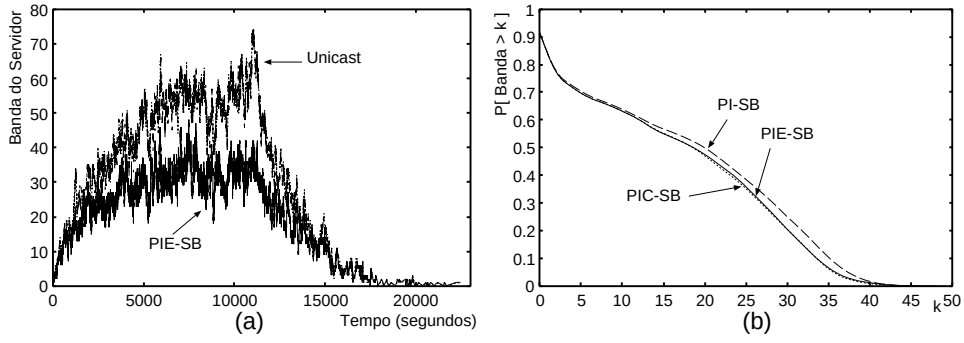


Figura 4.19: (a) Banda de PIE-SB em eTeach; (b) CCDF da banda para PIE, PIC e PI em eTeach.

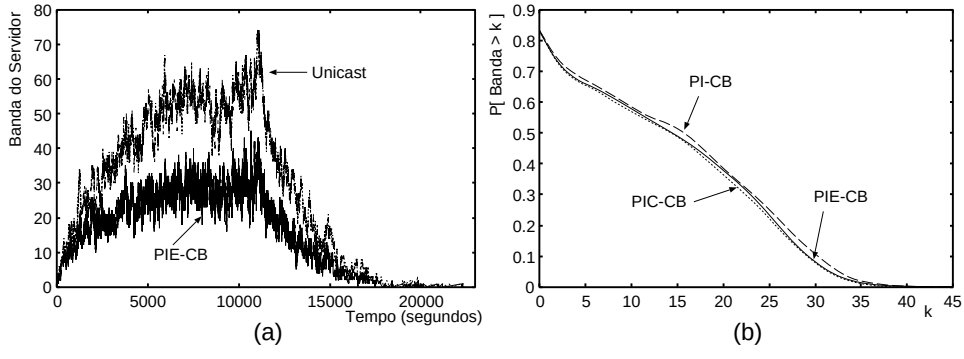


Figura 4.20: (a) Banda de PIE-CB em eTeach; (b) CCDF da banda para PIE, PIC e PI em eTeach.

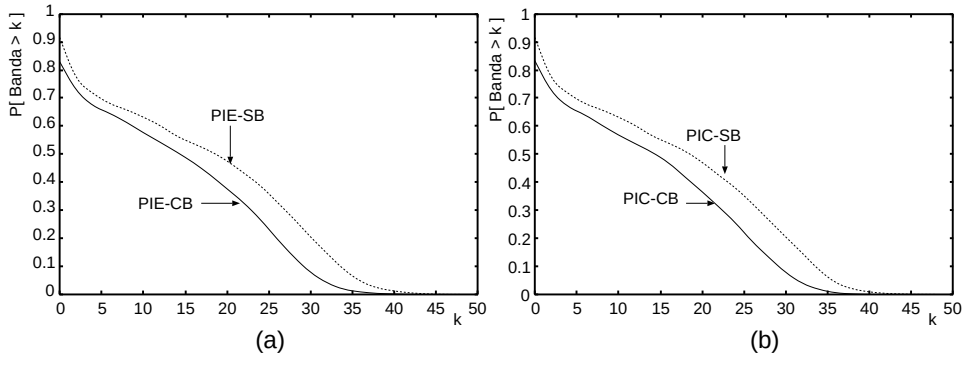


Figura 4.21: CCDF da banda para (a) PIE e (b) PIC em eTeach.

Para determinação do valor ideal de δ_{after}, fizemos experimentos apenas com as técnicas PIE e/ou PIC. Lembramos que a técnica MI não possui este parâmetro. Consideramos o intervalo de 0–100% da janela ótima W de *Patching* para o valor de δ_{after}. Valores fora deste intervalo excedem em demasia o tamanho médio do segmento solicitado por requisição em todos os cenários e se mostram ineficientes para fins de otimização de banda.

Por exemplo, para a Carga eTeach, apresentamos as distribuições com os valores mais relevantes para este estudo nas Figuras 4.23 (para PIC) e 4.24 (para PIE). As Tabelas 4.9 (para PIC) e 4.10 (para PIE) apresentam os respectivos valores médios de mensagens registrados. Desta análise temos então indicações de que o valor ideal para ambas as técnicas está aproximadamente em 30–40% da janela ótima, pois estabelece a melhor relação de benefício entre as distribuições e os valores médios aferidos, muito embora o tamanho correspondente a 20% tenha provido um valor mínimo de pico da banda dentre aqueles ilustrados. Estes resultados também corroboram a condição de significativa semelhança de performance das técnicas PIE e PIC.

Ainda para a Carga eTeach, por meio da Equação 4.4, calculamos o valor aproximado de δ_{after}. O desenvolvimento é apresentado em (4.7). Sabendo-se que a janela ótima da técnica *Patching* é igual a 292.50 s, concluímos que o valor D_{otimo} corresponde a 39.89% da janela ótima. Este valor pertence ao intervalo do valor ideal obtido nos experimentos (30–40% da janela ótima). Para este cenário, temos então que o valor calculado é uma estimativa bem próxima para o valor ideal de δ_{after}. Este fato deve-se à condição do cenário analisado ser relativamente próximo do modelo analítico utilizado para derivação do valor calculado, principalmente porque, conforme comentado no início desta seção, existe uma certa uniformidade no acesso às unidades do objeto e, praticamente, todas as unidades do mesmo são acessadas (veja Figuras 4.10(b) e 4.11). De forma interessante, note também que o valor médio L termina redundando em uma boa estimativa para o valor de δ_{after}.

Porém, ao realizarmos este mesmo experimento com a Carga MANIC-1, temos o seguinte. As distribuições com os valores mais relevantes para este estudo são mostradas na Figura 4.22. Conjuntamente com os valores apresentados na Tabela 4.8,

temos indicações de que o valor ideal está aproximadamente em 70–80% da janela ótima, pois estabelece a melhor relação de benefício entre as distribuições e os valores médios de mensagens aferidos, muito embora o tamanho correspondente a 90% tenha provido um valor mínimo de pico da banda dentre aqueles ilustrados. Para o cálculo do valor aproximado de δ_{after}, utilizamos a Equação 4.4 com $T = 24$ (número de unidades distintas acessadas de forma inicial), pois neste cenário temos localidade de acesso (veja Figura 4.4(b)), conforme explicado no início desta seção. Este desenvolvimento está apresentado em (4.6). Sabendo-se que a janela ótima da técnica *Patching* é igual a 552.73 s (Equação 3.2), concluímos que o valor D_{otimo} está fora do intervalo do valor ideal obtido experimentalmente (70%–80% da janela ótima). A estimativa de um intervalo de confiança (Equação 4.5) não se faz necessária devido ao significativo desvio padrão associado a L.

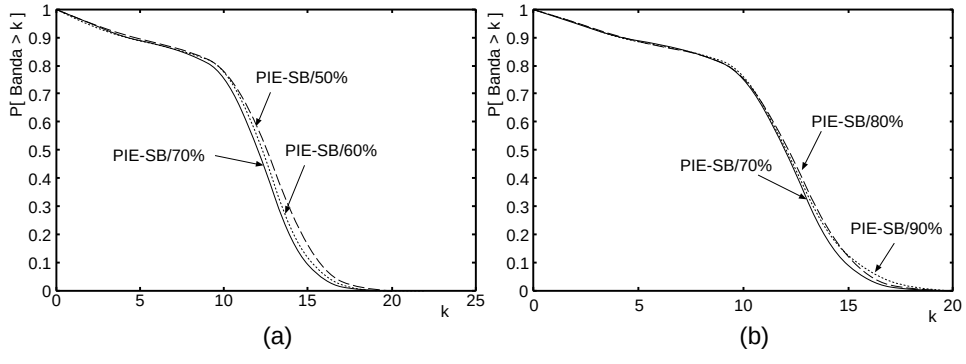


Figura 4.22: Estudo para determinação do valor ideal para δ_{after}

δ_{after} (%)	B_{max}	**TB**	**TM**	**TP**	**TF**
50	20	1.70	0.69	1.27	1.57
60	21	1.70	0.52	1.34	1.57
70	20	1.70	0.42	1.40	1.57
80	20	1.70	0.32	1.45	1.57
90	19	1.70	0.29	1.45	1.57

Tabela 4.8: Principais estatísticas para PIE-SB em MANIC-1

$$D_{otimo(s)} = \frac{\sqrt{2\frac{\beta}{T}L+1}-1}{\frac{\beta}{T}} = \frac{\sqrt{2\frac{1.70/60}{24}1190+1}-1}{\frac{1.70/60}{24}} = 806.27 \tag{4.6}$$

Como conclusão geral, os experimentos revelam que a aproximação analítica que derivamos $D_{optimal}$ é satisfatória para as Cargas eTeach e MANIC-2 e inadequada para as duas outras cargas (Tabela 4.11). Isto ocorre devido as premissas do modelo utilizado para a derivação. Por exemplo, assumimos que a distribuição de X é uniforme e isto não se aplica as Cargas UOL e MANIC-1. Note que estas premissas foram observadas como verdadeiras nos cenários de interatividade moderada-alta, i.e., num. requisições por sessão > 4.7.

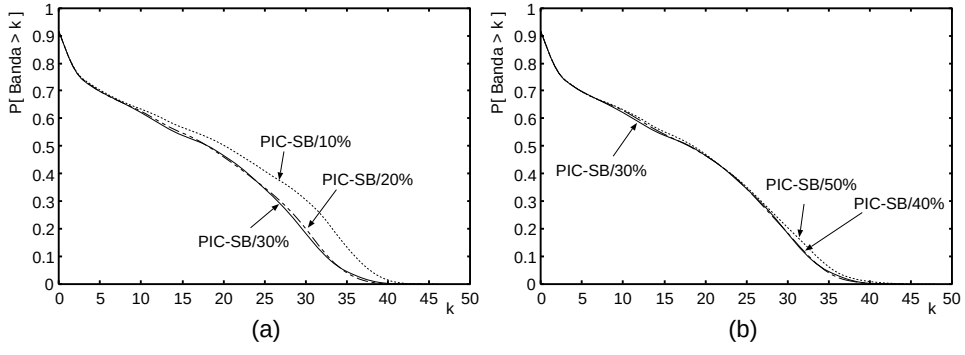


Figura 4.23: Estudo do tamanho de δ_{after} para PIC em eTeach.

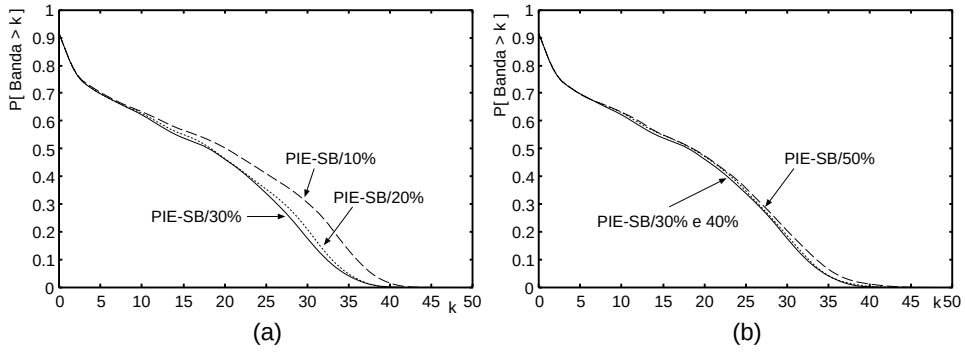


Figura 4.24: Estudo do tamanho de δ_{after} para PIE em eTeach.

$$D_{otimo(s)} = \frac{\sqrt{2\frac{\beta}{T}L+1}-1}{\frac{\beta}{T}} = \frac{\sqrt{2\frac{25.40/60}{2199}118+1}-1}{\frac{25.40}{2199}} = 116.69 \tag{4.7}$$

Análise do Limiar Delta Merge - δ_{merge}

Para determinação do valor ideal para δ_{merge}, realizamos experimentos apenas com a técnica PIE e/ou PIC. Lembramos A técnica MI não possui este parâmetro. Experimentamos valores percentuais do valor ideal de δ_{after} obtido na análise anterior e observamos as respectivas funções de distribuição da banda para cada uma das

δ_{after} (%)	B_{max}	**TB**	**TM**	**TP**	**TF**
10	46	25.40	2.04	10.55	13.54
20	43	25.40	2.33	15.20	13.54
30	44	25.40	1.71	17.99	13.54
40	44	25.40	1.34	19.52	13.54
50	48	25.40	0.99	20.64	13.54

Tabela 4.9: Estatísticas de PIC-SB em eTeach

δ_{after} (%)	B_{max}	**TB**	**TM**	**TP**	**TF**
10	46	25.40	1.95	10.55	13.54
20	43	25.40	1.94	15.27	13.54
30	44	25.40	1.32	17.98	13.54
40	46	25.40	1.08	19.46	13.54
50	48	25.40	0.72	20.69	13.54

Tabela 4.10: Estatísticas de PIE-SB em eTeach

Valor	**eTeach**	**MANIC-1**	**MANIC-2**	**UOL**
Experimental (% W)	30–40	70–80	50–60	100-110
Analítico (% W)	40.0	146.0	74.0	244.0

Tabela 4.11: Limiar Delta After - δ_{after}

cargas. Valores fora deste intervalo se mostram ineficientes para fins de otimização da banda. Os resultados gerais obtidos mostram que δ_{merge} pode ser simplesmente aproximado pelo valor de δ_{after}. Isto simplifica tremendamente a implementação e a operação *online* de PIE e PIC, pois na prática passa a existir apenas a necessidade da determinação de um único parâmetro.

Como exemplificação, as distribuições com os valores mais relevantes neste experimento, considerando a Carga eTeach e MANIC-2, são apresentadas nas Figuras 4.25 e 4.26. Vemos que o valor ideal de δ_{merge} está de fato em aproximadamente 100% do valor de δ_{after}, pois já estabelece um adequado nível de otimização da banda.

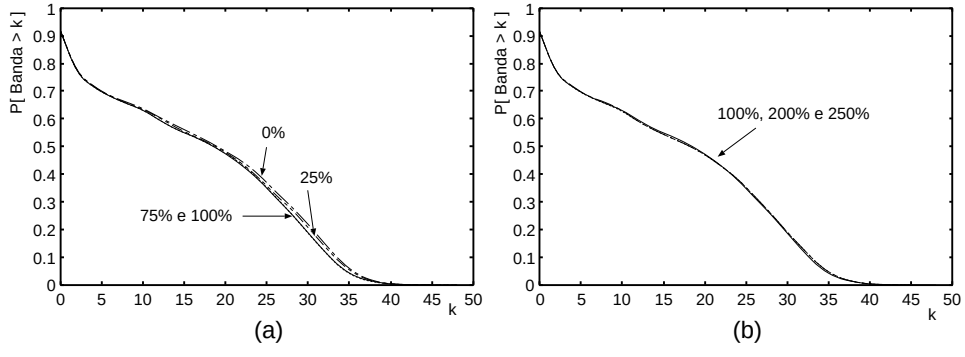


Figura 4.25: Estudo para determinação de δ_{merge} para PIE em eTeach.

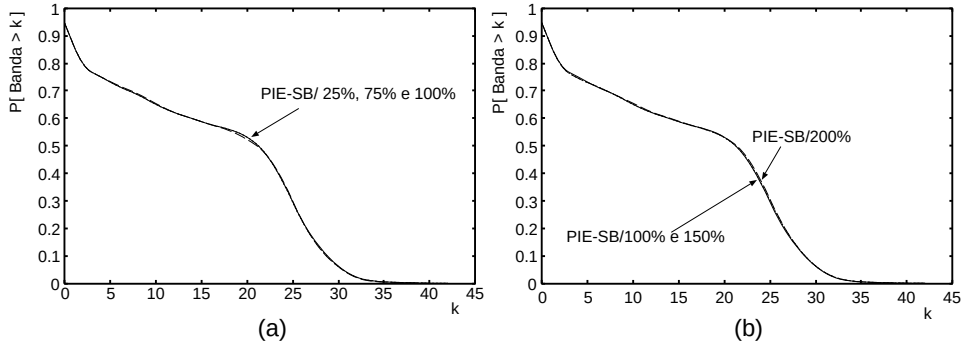


Figura 4.26: Estudo para determinação de δ_{merge} para PIE em MANIC-2.

Análise do Limiar δ_{MI}

Para determinação do valor ideal de δ_{MI}, realizamos a experimentação descrita seguir. Variamos seu valor no intervalo de 25–150% do valor de L, e também consideramos o não uso de δ_{MI} que corresponde a não utilizarmos restrição para seu valor.

As Figuras 4.28(a) e 4.27(a) e as Tabelas 4.12 e 4.13 Tabelas 4.14 e 4.15 trazem os resultados deste experimento considerando a Carga MANIC-2 e eTeach, respectivamente. Como observamos também nos dois outros cenários, estes experimentos mostram que, ao simplesmente não limitarmos o valor de δ_{MI}, já obtemos a performance otimizada ou bastante próxima da mesma. Também a diminuição do *trabalho* médio realizado não é significativa ao utilizarmos valores específicos para δ_{MI}.

A constatação final é portanto que, diferentemente do observado no paradigma de *Patching*, a técnica MI dispensa a determinação de limiares de tempo para decisões de união e abertura de fluxos no sistema quando usado em cenários com interatividade. Isto naturalmente favorece a sua implementação, pois exclui a necessidade de formulações matemáticas para cálculo de limiares de tempo. Assim, deixamos implícito nas análise seguintes o não uso deste limiar.

δ_{MI} (%)	B_{max}	**TB**	**TM**	**TF**
100	35	6.41	11.90	3.95
150	34	6.41	12.38	3.95
200	33	6.41	12.55	3.95
não uso	33	6.41	12.61	3.95

Tabela 4.12: Principais estatísticas para MI em MANIC-2.

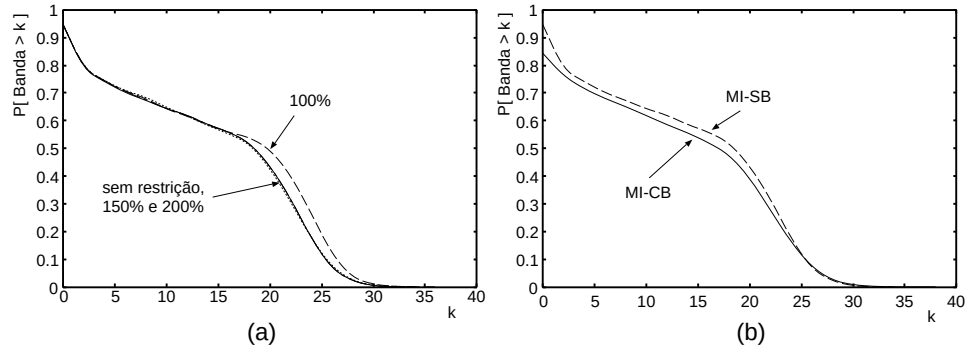


Figura 4.27: (a) Estudo do valor de δ_{MI} em MANIC-2 e (b) CCDF da banda de MI em MANIC-2.

Análise do Paradigma HSM

Ao compararmos MI e CT, notamos que, excetuando-se a Carga MANIC-1, ao empregarmos o *Complete Buffer* (CB), a técnica MI é mais eficiente que CT e, por

Parâmetro	não uso	200%	150%	100%
max{num fluxos multicast}	33	33	34	35
trabalho realizado	969.5	967.6	991.1	1003.5

Tabela 4.13: Estudo para determinação do parâmetro δ_{MI} em MANIC-2

δ_{MI} (%)	B_{max}	TB	TM	TF
25	53	25.40	12.62	13.54
50	48	25.40	20.18	13.54
75	45	25.40	25.21	13.54
100	45	25.40	28.51	13.54
150	41	25.40	32.89	13.54
não uso	42	25.40	37.50	13.54

Tabela 4.14: Principais estatísticas para MI em eTeach

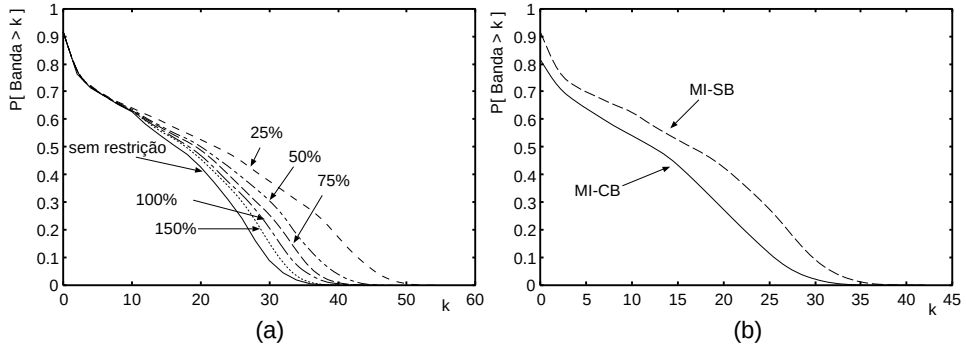


Figura 4.28: (a) Estudo de δ_{MI} em eTeach; (b) CCDF para MI em eTeach.

Parâmetro	max{num fluxos multicast}	trabalho realizado
não uso	42	4277.3
150%	41	3986.4
100%	45	4178.2
75%	48	4057.0
50%	53	4078.9

Tabela 4.15: Complexidade de MI em função de δ_{MI} em eTeach

outro lado, quando usamos o *Simple Buffer* (SB), a técnica CT é mais eficiente que MI. Este resultado geral está embasado na segunda diferença de MI com relação à CT, conforme detalhado na Subseção 4.3.3. Em linhas gerais, quando usamos *buffer* completo, existe uma tendência de termos um menor número de fluxos ativos no sistema e daí uma menor probabilidade de selecionar novos fluxos alvos, em substituição a fluxos alvos originais que de fato venham a prover economia de banda.

Para ilustrar o ponto acima apresentamos os experimentos realizados nas Cargas eTeach e UOL. Nas Figuras 4.29(a) e 4.29(b) temos eTeach. Para o caso de emprego de *buffer* simples (SB), a técnica CT é mais eficiente que MI. Para o caso que usamos *buffer* completo (CB), vemos, de forma contrária, uma vantagem de MI sobre CT. Nas Figuras 4.30(a) e 4.30(b) temos a Carga UOL. As observações anteriores permanecem verdadeiras, sendo que a vantagem de MI sobre CT é ainda mais significativa, corroborando o enunciado no parágrafo anterior de que que quanto mais eficiente for o emprego de CB, mais eficiente será MI em relação a CT.

A exceção observada para a Carga MANIC-1 advém do fato de que o emprego de *buffer completo* não é eficiente neste cenário. Em outras palavras, para efeito de compartilhamento de dados, as situações com e sem o emprego de *buffer* completo são praticamente idênticas.

Agora, de forma global, desde que a implementação de *buffer* completo é relativamente simples e vantajosa, devido à eventual economia de banda que pode ser atingida, decorre como conclusão natural que MI deve ser considerada uma opção mais adequada. E, mesmo na situação em que CB não for eficiente, como observado em MANIC-1, a performance de CT e MI terminam sendo bem próximas, conforme ilustrado na Figura 4.31.

Comparando MI com MI-var, notamos performances semelhantes em todos os cenários, ou seja, a otimização, quando existente, de MI-var em relação à MI é insignificante. Isto mostra que a probabilidade de que um cliente não tenha um fluxo alvo é praticamente nula, conforme conjecturado na Subseção 4.3.3.

Para ilustrar este fato, consideramos as Cargas eTeach e MANIC-2. As Figuras 4.42(a) e 4.42(b) referem-se a primeira carga e as Figuras 4.42(a) e 4.42(b) a

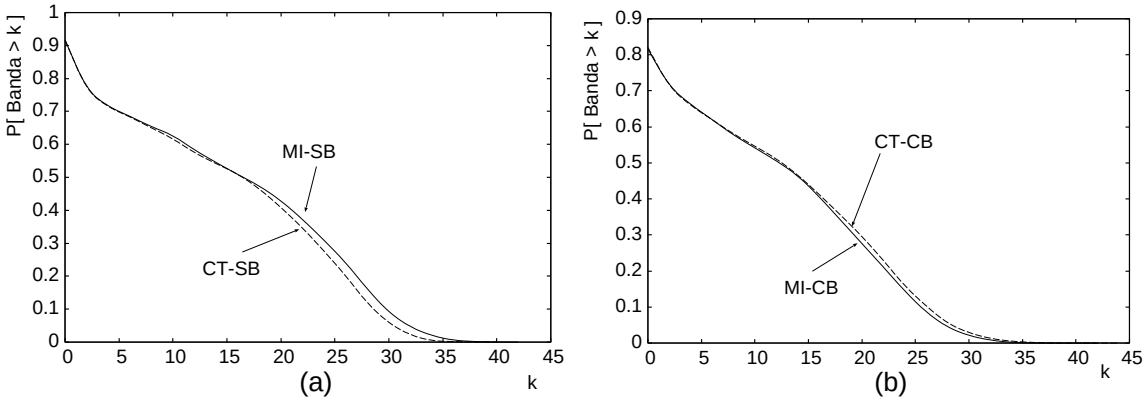


Figura 4.29: (a) CCDF para MI-SB e CT-SB em eTeach; (b) CCDF para MI-CB e CT-CB em eTeach.

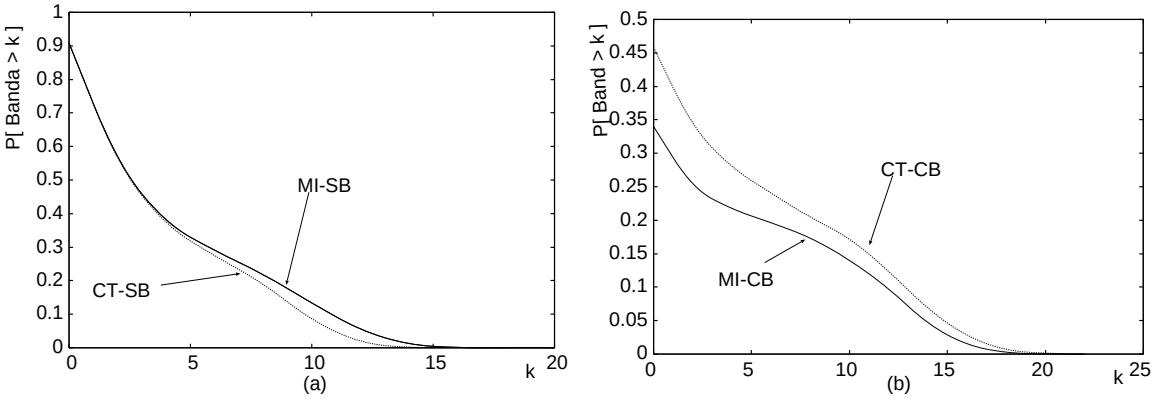


Figura 4.30: (a) CCDF para MI-SB e CT-SB em UOL; (b) CCDF para MI-CB e CT-CB em UOL.

segunda. Em ambos os casos, as curvas praticamente se superpõem. Isto naturalmente favorece MI por ter uma implementação mais simples que MI-var, pois retira a necessidade de verificação da existência de clientes sem fluxo alvo quando da ocorrência de união de fluxos.

Paradigma de Patching versus Paradigma de HSM

Aqui comparamos a técnica mais eficiente no paradigma de *Patching* com a mais eficiente no paradigma de HSM: técnicas PIE e MI, respectivamente. Em todos os cenários, observamos que MI tem melhor performance que PIE. Esta vantagem é quantificada através das razões competitivas apresentadas na Tabela 4.16. Também ilustramos este fato por meio das distribuições de banda nos quatro cenários analisados: Carga eTeach (Figura 4.34), Carga MANIC-1 (Figura 4.35), Carga MANIC-2 (Figura 4.36), e Carga UOL (Figura). Vemos que, tanto no caso de CB quanto de

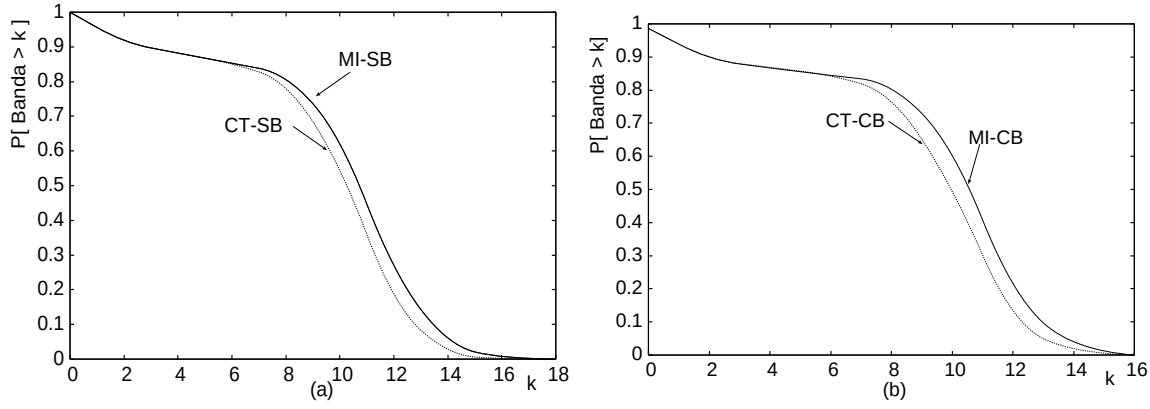


Figura 4.31: (a) CCDF para MI-SB e CT-SB em MANIC-1; (b) CCDF para MI-CB e CT-CB em MANIC-1.

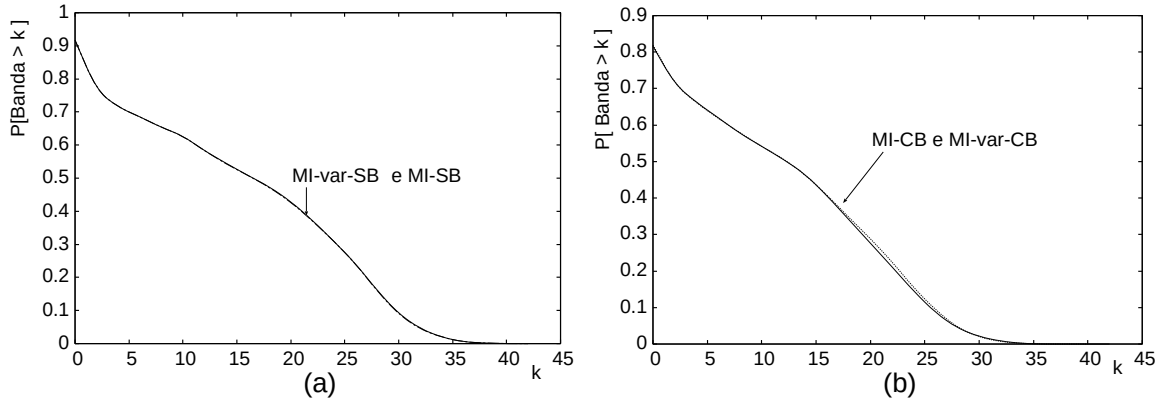


Figura 4.32: (a) CCDF para MI-SB e MI-var-SB em eTeach; (b) CCDF para MI-CB e MI-var-CB em eTeach.

SB, a técnica MI é sempre mais eficiente que PIE.

Ainda para efeito comparativo, temos a Tabela 4.17, onde podemos observar os valores de consumo médio de banda e valores de pico de banda nos quatro cenários. É interessante notar que o emprego de CB faz com que ocorra um aumento no pico da banda para a técnica MI. A explicação para este fato está possivelmente na fragmentação da informação em *buffer*, pois cria uma maior dificuldade para compartilhamento de fluxos, tendo em vista que os clientes terminam entrando em ciclos sucessivos de leitura local e leitura a partir de um canal do servidor. Perceba que este entendimento encontra-se em sintonia com aquele relativo ao aumento do número de mensagens devido a CB que comentamos anteriormente.

Já na Tabela 4.18 temos as reduções (%) em relação ao consumo médio de banda e aos valores de pico de banda registrados por PIE e MI em relação à técnica

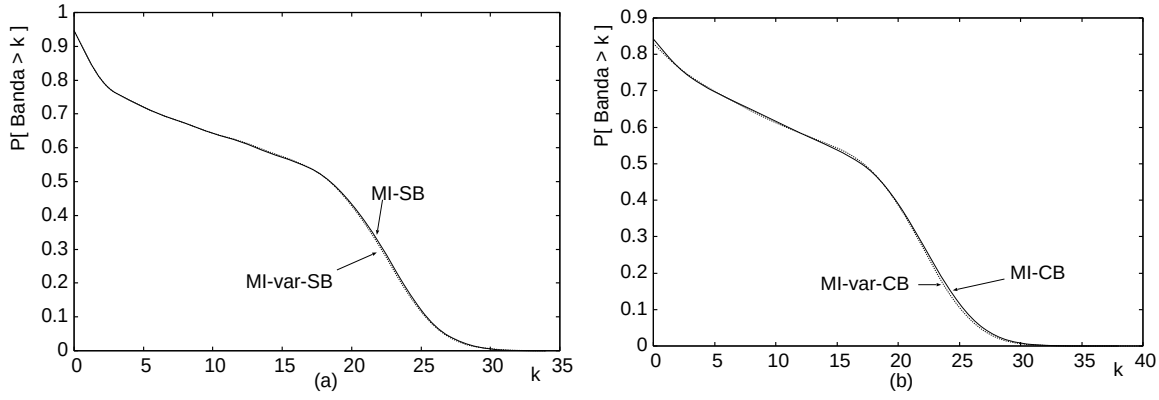


Figura 4.33: (a) CCDF para MI-SB e MI-var-SB em MANIC-2; (b) CCDF para MI-CB e MI-var-CB em MANIC-2.

Tabela 4.16: RMin e RMax de PIE para MI

Técnica	eTeach		MANIC-1		MANIC-2		UOL	
	RMin	RMax	RMin	RMax	RMin	RMax	RMin	RMax
PIE-SB	0.2	1.0	0.1	1.0	0.1	1.0	0.3	0.6
PIE-CB	0.2	1.0	0.2	1.0	0.1	1.0	0.1	1.0

Patching-SB. As reduções são de fato, como já esperado, bem expressivas tanto para o consumo médio de banda ($\approx$ 38%–42%), como para o valor de pico ($\approx$ 34%–52%). Estas reduções também são ilustradas por meio dos histogramas apresentados na Figura 4.38.

Entretanto, por meio do cômputo do trabalho médio apresentado na Tabela 4.19, também graficamente ilustrado na Figura **??**, podemos constatar uma maior sim-

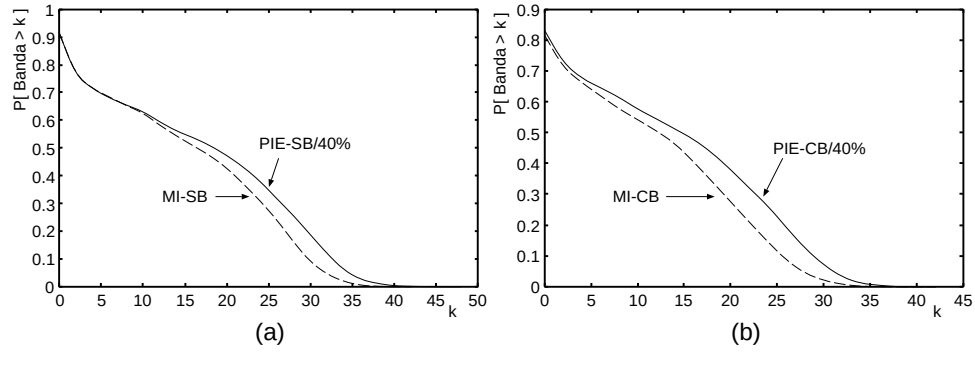


Figura 4.34: (a) CCDF da banda para PIE-SB e MI-SB em eTeach; (b) CCDF da banda para PIE-CB e e MI-CB em eTeach.

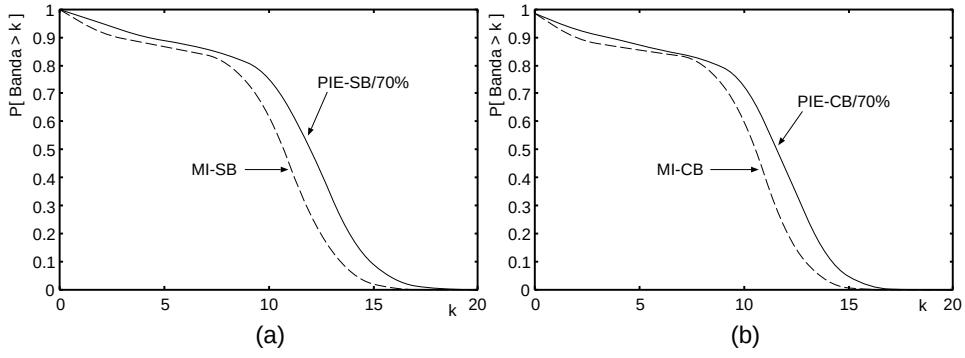


Figura 4.35: CCDF da banda para PIE-SB e MI-SB em MANIC-1; CCDF da banda para PIE-CB e MI-CB em MANIC-1.

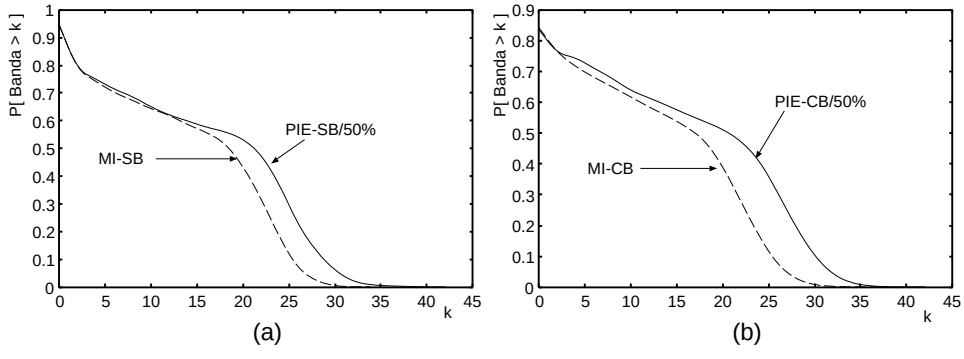


Figura 4.36: (a) CCDF da banda para PIE-SB e MI-SB em MANIC-2; (b) CCDF da banda para PIE-CB e MI-CB em MANIC-2.

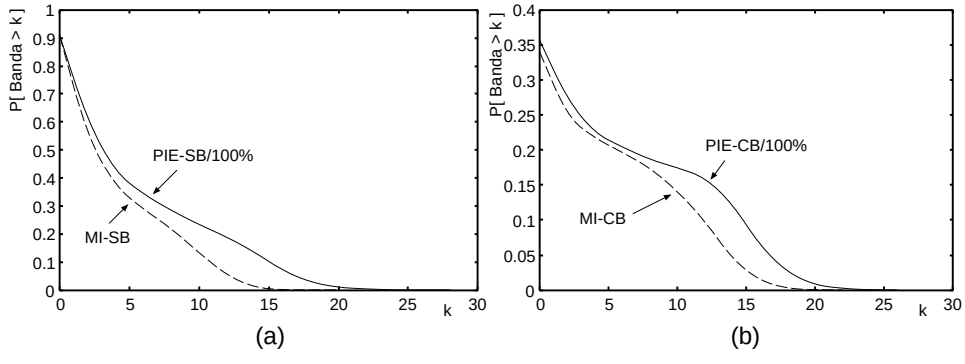


Figura 4.37: (a) CCDF da banda para PIE-SB e MI-SB em UOL; (b) CCDF da banda para PIE-CB e MI-CB em UOL.

Tabela 4.17: Consumo médio de banda (E_B) e valor de pico (V_P)

Técnica	eTeach		MANIC-1		MANIC-2		UOL	
	E_B	V_P	E_B	V_P	E_B	V_P	E_B	V_P
Patching-SB	26.20	71	28.32	43	32.83	68	8.51	33
Patching-CB	21.18	62	27.86	41	30.51	64	5.53	34
PI-SB	18.36	49	14.09	23	17.50	40	6.94	26
PI-CB	15.00	45	13.75	23	17.50	42	3.59	23
PIE-SB	17.18	44	11.58	20	17.00	42	5.93	27
PIE-CB	12.86	41	11.06	19	17.00	41	3.38	25
MI-SB	15.79	42	10.33	17	15.05	33	4.55	19
MI-CB	12.44	42	9.97	16	14.24	37	2.75	22

Tabela 4.18: Redução (%) em E_B e V_P com relação à *Patching*-SB

Técnica	eTeach		MANIC-1		MANIC-2		UOL	
	E_B	V_P	E_B	V_P	E_B	V_P	E_B	V_P
PIE-SB	34.43	38.03	59.11	53.49	48.22	38.24	30.32	18.18
PIE-CB	50.92	42.25	60.94	55.81	48.22	38.71	60.28	24.24
MI-SB	39.73	40.84	63.52	60.47	54.16	51.47	46.53	42.42
MI-CB	52.52	40.84	64.80	62.79	56.63	45.59	67.69	33.33

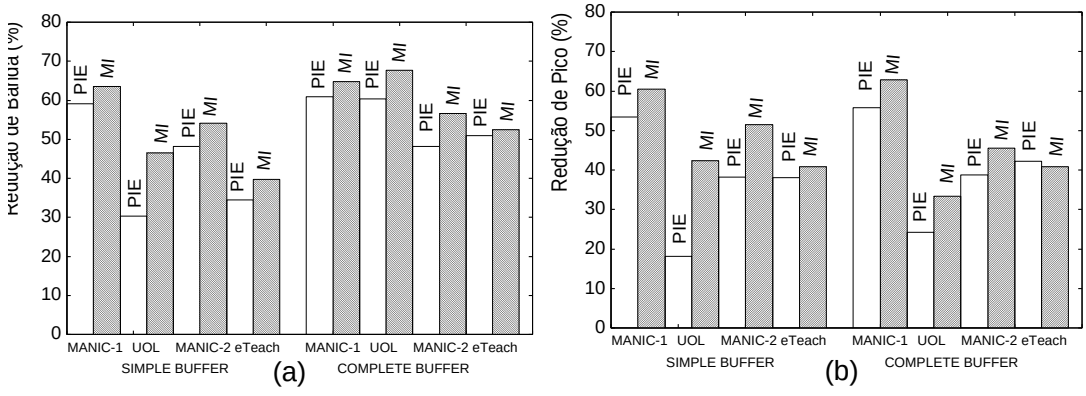


Figura 4.38: (a) Reduções de consumo médio de banda de PIE e MI em relação à Patching-SB; (b) Reduções de pico de banda de PIE e MI em relação à Patching-SB.

plicidade da técnica PIE. Isto porque apesar de ambas as técnicas apresentarem complexidades polinomiais, as constantes de multiplicação associadas são distintas e favorecem a técnica PIE, fazendo com que o *trabalho* seja até três ordens de grandeza maior para MI. Reforçando a constatação da maior simplicidade de PIE, também observamos em todos os cenários que o número de fluxos *multicast* ativos da técnica MI é bem maior que aquele da técnica PIE. Isto tem implicação direta na implementação real do serviço *multicast*, principalmente se considerarmos o serviço implementado na camada de aplicação. As Figuras 4.39 e 4.40 ilustram este aspecto ao evidenciar as distribuições do número de fluxos *multicast* associados à técnica MI considerando as Cargas eTeach e UOL, respectivamente.

Tabela 4.19: Trabalho Médio (T_m) e Número Máx Fluxos Multicast (F_m)

Parâmetro	eTeach		MANIC-1		MANIC-2		UOL	
	PIE	MI	PIE	MI	PIE	MI	PIE	MI
$T_m - SB$	618.3	4277.3	22.1	159.0	150.7	969.5	367.2	2962.1
$F_m - SB$	23	42	11	17	22	33	9	19
$T_m - CB$	804.4	9967.0	20.9	129.9	292.9	2508.6	191.9	2155.6
$F_m - CB$	23	42	11	16	20	37	9	22

É interessante ainda mencionar que a técnica PIE consegue se beneficiar, excetuando-se o cenário da Carga MANIC-2, mais de CB que a técnica MI, conforme graficamente apresentado na Figura 4.41. Imaginamos que esta constatação possa ter o seguinte entendimento. A técnica MI já encontra-se com nível de otimi-

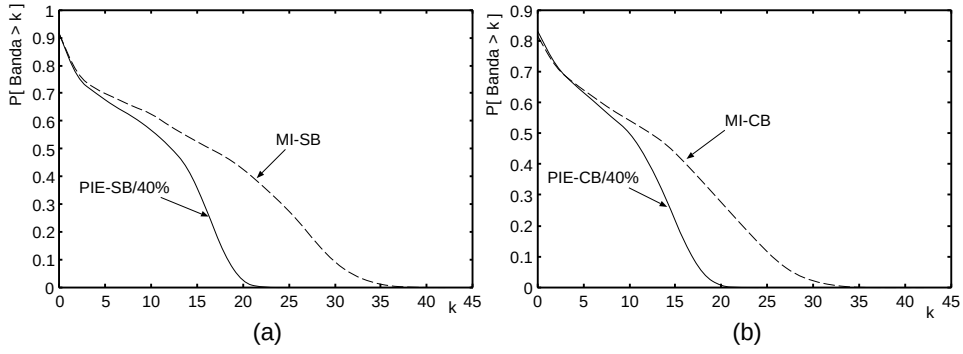


Figura 4.39: Distribuição do número de fluxos *multicast* para PIE e MI em eTeach.

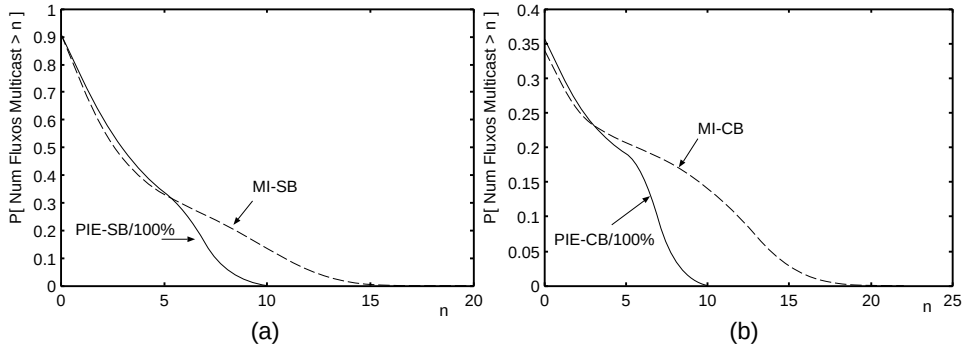


Figura 4.40: Distribuição do número de fluxos *multicast* para PIE e MI em UOL

zação acentuado e, portanto, apresenta-se com pouco espaço para otimizações. Já a técnica PIE ainda poderia ser otimizada através, por exemplo, da flexibilização em relação ao número de níveis permitidos em sua estrutura de união. Contudo, essa permissividade acarretaria a necessidade de definição de novos limiares de tempo, o que poderia inviabilizar a sua aplicação de forma prática e eficiente, conforme já comentamos anteriormente para PIC.

Como conclusão geral, acreditamos que deve existir um compromisso para a decisão da escolha da técnica mais apropriada. A técnica PIE tem como vantagens a sua simplicidade (menor complexidade) e a sua eficiência competitiva devido aos resultados de otimização de banda apresentados, enquanto que a técnica MI possui uma melhor performance absoluta à custa de uma gerência de sistema computacionalmente mais onerosa. Se a complexidade de mensagens não representar um gargalo no sistema, então MI é a técnica mais eficiente. Caso contrário, PIE pode ser considerada como uma opção.

Descontinuidades devido ao Limiar Delta Before - δ_{before}

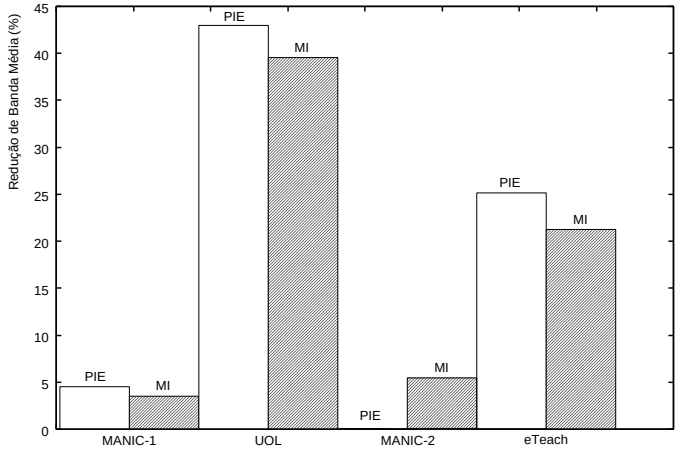


Figura 4.41: Redução do consumo médio de banda devido ao uso de CB.

Por ser a técnica mais eficiente do paradigma de *Patching*, aqui realizamos experimentos apenas com a técnica PIE. O procedimento consiste em variar o valor de δ_{before} e observar a distribuição da banda nos casos de *buffer* simples (SB) e *buffer* completo (CB) nos diferentes cenários.

Como observação geral, podemos notar que se utilizarmos $\delta_{before} \geq$ 1–3% (3–5%) de T, a técnica PIE-SB (PIE-CB) torna-se bastante competitiva com relação a MI-SB (MI-CB) em todas as cargas, exceto para UOL. Esta observação é graficamente ilustrada nas Figuras 4.42 e 4.43. Notamos ainda que, para tornar PIE mais eficiente que MI, o valor experimental obtido para δ_{before} é sempre maior no caso de *buffer* completo (CB) do que no caso de *buffer* simples (SB). Isto é explicado pelo fato de que, quando usamos o *buffer* completo, existe uma tendência de termos um menor número de fluxos no sistema e, portanto, uma maior dificuldade para efetuar o compartilhamento de dados. Esta condição favorece à técnica MI por ter um mecanismo de busca mais exaustivo que PIE e, conseqüentemente, cria a necessidade de um maior valor de $\delta_{anterior}$ para PIE com objetivo de compensar sua maior desvantagem em relação à MI no cenário de *buffer* completo.

O fato de δ_{before} não melhorar a performance de PIE na carga UOL é explicada a seguir. Avaliamos a probabilidade de $X = 1$, i.e., a probabilidade de que a primeira unidade (do segmento de tamanho médio L do objeto) solicitada pelo cliente seja igual à primeira unidade do objeto. O valor desta probabilidade é ≈ 0.78. Isto significa que a maioria dos clientes requisita a primeira unidade do objeto e, assim

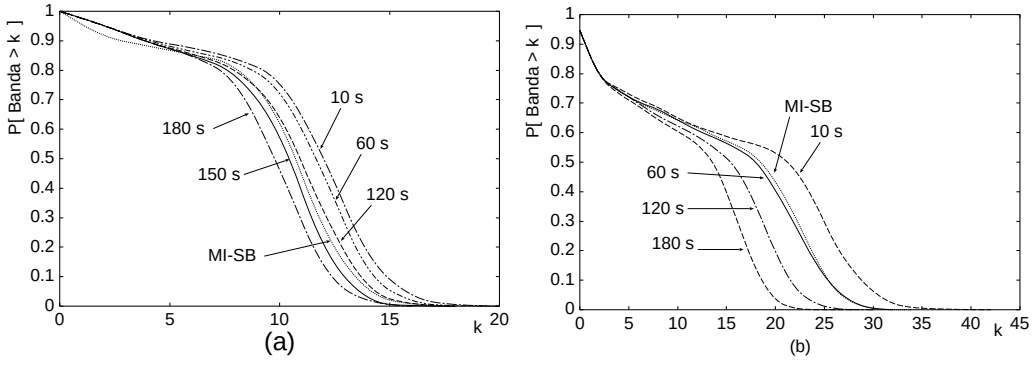


Figura 4.42: Análise de δ_{before}: (a) Carga MANIC-1 (SB); (b) Carga MANIC-2 (SB).

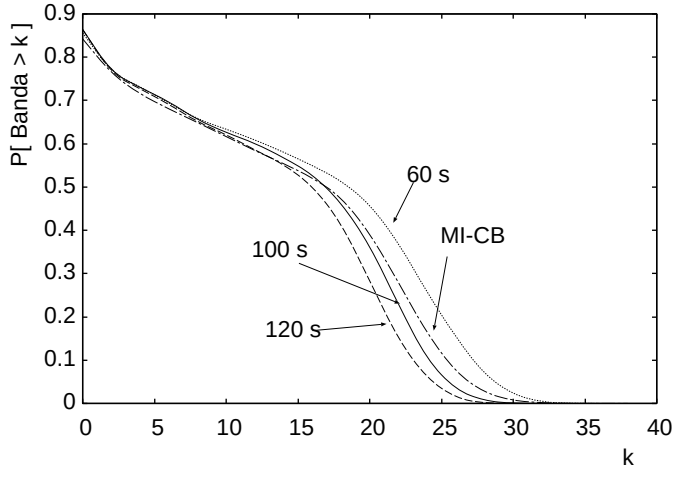


Figura 4.43: Análise de δ_{before}: Carga MANIC-2 (CB).

sendo, não há fluxos possíveis no limiar de δ_{before} para atender a requisição ocorrida. Portanto, a variação deste parâmetro termina não afetando a distribuição da banda.

Por fim, a Figura 4.44 faz uma síntese gráfica comparativa entre MI-SB e PIE-SB considerando dois resultados: (i) o trabalho médio em função do nível de interatividade (número médio de requisições por sessão) e (ii) o valor de δ_{before} para o qual temos praticamente a mesma distribuição de banda em ambas as técnicas. Note que, conforme o nível de interatividade aumenta, o trabalho médio de MI também aumenta enquanto que a descontinuidade introduzida por δ_{before} diminui. Comportamento semelhante é obtido também para o caso com *buffer* completo (CB). Este resultado geral mostra portanto que, apesar de MI ser de fato a técnica de melhor performance, PIE pode ser vista como uma técnica competitiva conforme o nível de interatividade aumenta.

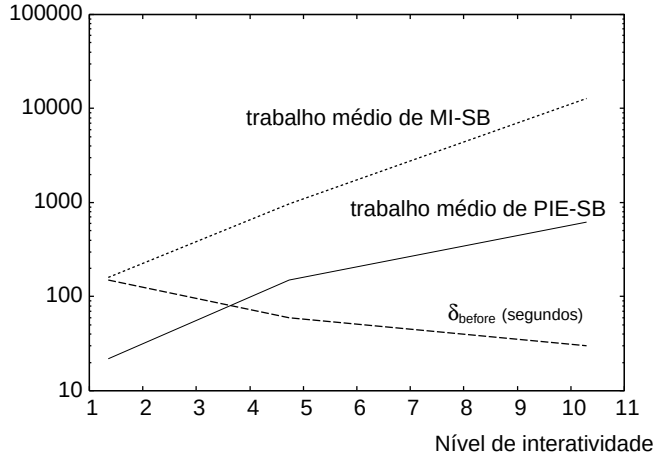


Figura 4.44: Comparação entre MI-SB e PIE-SB.

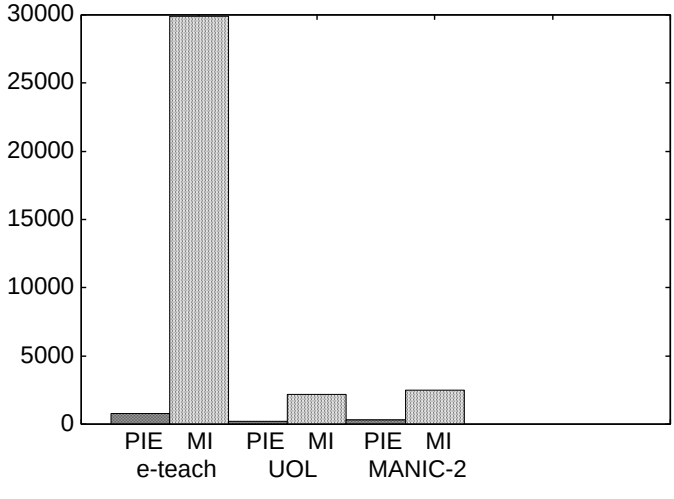


Figura 4.45: Trabalho médio para PIE-CB e MI-CB.

4.6 Conclusões

Este capítulo apresentou três novas técnicas de compartilhamento de banda para servidores de vídeo com interatividade: *Patching* Interativo Eficiente - PIE, *Patching* Interativo Completo - PIC e *Merge* Interativo - MI. As duas primeiras foram baseadas na idéia de *Patching* e a última nos esquemas de HSM. Obtivemos ainda uma aproximação analítica para um limiar de tempo usado em decisões de união e abertura de fluxos para as técnicas PIE e PIC (a saber: limiar δ_{after}), e também discutimos sobre outros limiares de tempo correlacionados (a saber: limiares δ_{merge}, δ_{before}). Além disso, também analisamos a necessidade do emprego de limiar de tempo para a proposta MI (a saber: limiar δ_{MI}) e, por fim, investigamos a importância do uso de *buffer* local no lado do cliente. As técnicas foram avaliadas e comparadas principalmente através de razões competitivas, distribuição da banda do servidor, valores médios de consumo de banda, banda agregada dos clientes (i.e., clientes ativos), valores de pico de banda e complexidade do sistema. Por meio de simulações em diferentes cenários de interatividade, pudemos fazer as principais constatações mencionadas a seguir.

- Paradigma *Patching*

  As técnicas PIE e PIC apresentaram níveis de otimização superiores àqueles da recente proposta *Patching* Interativo (PI) e do clássico esquema de *Patching* na maioria dos cenários. Também observamos que PIE e PIC apresentam desempenhos bem semelhantes. Entretanto, a concepção de implementação de PIC a torna mais complexa.

- Paradigma HSM

  A técnica MI apresentou níveis de otimização de banda satisfatórios. Na maioria dos cenários investigados, estes níveis foram superiores àqueles da proposta da literatura *Closest Target* (CT), sob a condição de utilizar a estratégia de armazenamento de toda a informação recebida do servidor (Complete *Buffer* - CB). Por se tratar de uma estratégia de simples implementação, visualizamos a técnica MI como mais eficiente deste paradigma.

- Serviço de VoD sem Descontinuidades

A técnica MI teve melhor performance que PIE e PIC à custa de uma maior complexidade do sistema, inferida a partir da avaliação do número médio de mensagens trocadas entre clientes e servidor.

- Serviço de VoD com Descontinuidades

  Admitindo descontinuidade de até $\approx$ 3–5% do tamanho total do objeto, PIE teve uma maior eficiência que MI na maioria dos cenários. Temos, portanto, que, apesar de MI ser de fato a técnica de melhor performance, PIE pode alternativamente ser vista como uma técnica competitiva a depender do grau de tolerância do cliente a descontinuidades.

- *Buffer* Completo (CB) versus *Buffer* Simples (SB)

  Os benefícios do emprego da estratégia CB dependem do cenário em questão e se refletiram nos três pontos comentados a seguir.

  – Influência no pico da banda do servidor, podendo aumentá-lo ou diminuí-lo. Isto é explicado pelo fato do uso do *buffer* provocar a redução do número de fluxos ativos no sistema e, com isso, reduzir circunstancialmente a probabilidade de compartilhamento e união de fluxos.

  – Otimização do uso banda do servidor ao longo do tempo.

  – Modificação da complexidade de mensagens, podendo aumentá-la ou diminuí-la, a depender do grau de fragmentação introduzido quando do armazenamento das informações.

  Considerando, contudo, ser de relativa simplicidade de implementação, a estratégia CB se mostrou mais oportuna, pois, nos casos em que não houve otimizações contundentes, as performances das técnicas com CB e SB se mostraram equivalentes.

- Limiar de Tempo δ_{after}

  A aproximação analítica para o limiar de tempo δ_{after}, relacionado às técnicas PIE e PIC, mostrou-se próxima da estimativa do intervalo do valor ideal obtido experimentalmente quando as premissas adotadas no modelo analítico estavam próximas do modelo real. Isto foi observado em cenários de maior nível de interatividade, i.e, num. requisições por sessão > 4.7.

Quando a acurácia da aproximação analítica para o limiar de tempo δ_{after} não é satisfatória, a idéia é utilizar o tamanho médio do segmento solicitado pela requisição como estimativa inicial, pois este é o tamanho máximo natural a ser considerado e, em seguida, realizar dinamicamente a busca pelo valor ideal. No caso de não se conhecer esse valor médio, pode-se ainda realizar uma estimativa inicial usando o valor da janela ótima da técnica *Patching* original, pois os valores obtidos nas simulações se mostraram confinados a percentuais deste tamanho.

- Para efeito de emprego prático das técnicas PIE e PIC, podemos admitir o seguinte em relação aos valores dos limiares de tempo definidos.

  – Para cenários de interatividade considerável, a aproximação analítica obtida para δ_{after} é satisfatória. Para cenários de baixa interatividade, deve-se realizar uma estimativa dinâmica de δ_{after}, baseada inicialmente ou no tamanho médio do segmento solicitado por requisição, ou no tamanho da janela ótima de *Patching*.

  – O limiar δ_{merge} deve ter seu valor igual ao valor de δ_{after}. Em relação à δ_{before}, estabelecemos 10 s por ser um valor aparentemente tolerável na maioria das aplicações de serviço de vídeo sob demanda, mas sua quantificação ideal é pertinente à aplicação em si e ainda é uma lacuna na literatura.

- Por último, independentemente do nível de interatividade do cenário, nenhum tipo de limiar de tempo deve ser empregado pela técnica MI, para efeito de maior simplicidade de operação *online*. Isto porque a otimização eventualmente conseguida devido ao limiar proposto δ_{MI} não é significativa.

Capítulo 5

Conclusões Gerais

5.1 Síntese Final

NESTE trabalho realizamos estudos visando analisar e otimizar a escalabilidade e o desempenho de servidores de vídeo sob demanda. As principais contribuições foram: o cálculo da distribuição de uso da banda do servidor e a proposta de três técnicas de compartilhamento de banda considerando a interatividade dos clientes.

A distribuição de uso da banda do servidor se faz importante em cenários onde os requisitos têm significativa variância. Nestes casos, o valor médio por si só não é um parâmetro ideal para o projeto de sistemas reais. Um outro aspecto importante a se considerar são as necessidades de estimativas percentuais para atendimento dos clientes como, por exemplo, acontece nos sistemas telefônicos. Nesta situação, mesmo que a variância associada seja baixa, o valor médio não é de grande utilidade e, mais uma vez, há então a necessidade do conhecimento da distribuição.

Para obtenção desta distribuição, consideramos que o servidor utiliza a popular técnica de compartilhamento de banda *Patching*. A escolha desta técnica deu-se pela sua simplicidade de implementação e eficiência, em termos de economia de banda, para objetos de baixa e média popularidade, quando comparada com técnicas mais sofisticadas da literatura. No caso de um objeto, mostramos que a distribuição pode ser aproximada por uma binomial com parâmetros que dependem da popularidade

do objeto. Usando simulações, observamos que o modelo analítico proposto é bastante satisfatório e que, quanto maior é a popularidade do objeto, mais preciso o modelo tende a ser. A distribuição de uso da banda no caso de múltiplos objetos foi mostrada como sendo uma soma de distribuições binomiais que pode ser calculada eficientemente através da Transformada Rápida de Fourier.

Em relação à interatividade nos servidores de vídeo sob demanda (VoD), as três técnicas propostas foram: *Patching* Interativo Eficiente - PIE, *Patching* Interativo Completo - PIC e *Merge* Interativo - MI. Para as duas primeiras, baseamos-nos no paradigma da técnica *Patching* original e, para a última, nos esquemas de HSM (*Hierarchical Stream Merging*). O paradigma de *Patching* é atrativo, como dissemos há pouco, pela sua simplicidade e performance competitiva, enquanto que o paradigma de HSM possui uma significativa eficiência em potencial. Esta eficiência de HSM se dá pelas características possíveis de pesquisa exaustiva e irrestrita por fluxos no intuito de compartilhamento de dados e, principalmente, pelo fato de todos os fluxos abertos no sistema serem *multicast* e assim, potencialmente, poderem ser fluxos alvos.

Os resultados obtidos em simulações confirmaram a satisfatória eficiência das técnicas PIE e PIC, inclusive superior à da recente proposta *Patching* Interativo (PI) na maioria dos cenários investigados. Os resultados também confirmaram a satisfatória eficiência de MI, inclusive superior, em determinados cenários, à da técnica *Closest Target* (CT). Como resultado geral, tivemos que MI é a de melhor performance absoluta, seguida por PIE e PIC. Estas duas últimas têm performances semelhantes, mas PIE possui maior simplicidade de implementação e gerência, decorrente da menor complexidade de mensagens. Obtivemos ainda uma aproximação analítica para um limiar de tempo usado em decisões de união e abertura de fluxos para as técnicas PIE e PIC (a saber: limiar δ_{after}), e também discutimos sobre outros limiares de tempo correlacionados (a saber: limiares δ_{merge} e δ_{before}). Além disso, analisamos também a necessidade do emprego de um limiar de tempo para a proposta MI (a saber: limiar δ_{MI}) e, ainda, investigamos a importância do uso de *buffer* local no lado do cliente para evitar que dados já armazenados tenham de ser novamente solicitados.

Em síntese, por meio das propostas das técnicas PIE, PIC e MI e, ainda, das análises e estudos realizados acreditamos ter estabelecido, por decorrência, um comparativo detalhado em relação à eficiência que pode ser alcançada pelo emprego das concepções de *Patching* e de HSM na implementação de novas técnicas para aplicações de VoD com interatividade.

5.2 Direcionamentos Futuros

Visualizamos as possibilidades para trabalhos futuros mencionadas a seguir.

- **1)** Análise qualitativa e quantitativa da influência do limiar de tempo δ_{before} das técnicas PIE e PIC sobre a percepção do cliente.

Neste trabalho realizamos um estudo experimental sobre a otimização conseguida na banda do servidor devido ao valor utilizado para δ_{before}, o qual diretamente limita o tamanho das descontinuidades a serem eventualmente introduzidas no serviço de VoD. Quanto maiores forem estas descontinuidades, maior é naturalmente o efeito negativo sobre a percepção do cliente. Assim sendo, torna-se importante estabelecer formalmente intervalos específicos para aceitabilidade destas descontinuidades por parte do cliente em função do tipo de aplicação em si. Isto com fins de aumentarmos os níveis de otimização de banda do servidor sem comprometermos a qualidade de serviço percebida pelo cliente.

- **2)** Estudo analítico da banda média do servidor para as técnicas PIE, PIC e MI.

Considerando o acesso não-seqüencial do cliente, a banda do servidor já foi alvo de discussão analítica em trabalhos anteriores como [26, 72, 92, 69]. Bestavros e Jin [26] mostraram que, para o padrão de acesso em que as requisições solicitam tamanhos fixos do objeto e começando de unidades aleatórias, a banda média do servidor (limite piso) cresce com a raiz quadrada da taxa de requisição. Vernon et al. [72] estenderam este último trabalho considerando outros padrões de acesso e

determinando o pior caso, o qual redunda também em um crescimento da banda com a raiz quadrada da taxa de requisição. Considerando a técnica SAM [28] e a proposta de Shin e Abram-Profeta [30], foram desenvolvidas expressões analíticas para quantificar separadamente os requisitos de canais para ações VCR e para união de fluxos no sistema [92]. Por último, um estudo analítico para derivação da banda média considerando um esquema de interatividade em especial é apresentado por Chang e Chan [69].

Daí, apesar de obtenção significativamente complexa, expressões analíticas para a banda média do servidor, considerando as técnicas PIE, PIC e MI, torna então possível a realização de análises competitivas analíticas com as outras propostas da literatura como, por exemplo, as mencionadas acima, e/ou o estabelecimento de um comparativo com os limites (piso) obtidos para os diferentes padrões de acesso examinados na literatura.

- **3)** Desenvolvimento de estratégias para emprego inteligente do *buffer* local do cliente.

Pesquisas anteriores já investigaram estratégias inteligentes para utilização do *buffer* local do cliente com fins de otimização da banda do sistema de VoD (p. ex., [45, 28, 30, 67, 50, 74, 75]). Em linhas gerais, estas estratégias baseiam-se, principalmente, em: (i) armazenamento de todas as unidades de dados transmitidas pelo servidor [50, 74], (ii) recebimento de unidades de dados quando o cliente está no estado de *Pause* ou lendo a partir de seu próprio *buffer* [75, 74], (iii) prolongamento de fluxos de dados [45, 28, 30, 67, 75, 74].

É importante destacar que essas otimizações podem, em sua maioria, ser implementadas em qualquer técnica de compartilhamento de banda. Acreditamos então ainda existir um caminho bem fértil na direção do desenvolvimento de outras estratégias e de sua aplicação com técnicas de compartilhamento de banda já propostas, criando novas perspectivas de níveis de otimização e, possivelmente, da realização de novas análises competitivas.

A título apenas de ilustração, imaginamos ser promissora uma concepção baseada em um conjunto de *buffers* compartilhados para toda uma rede local. Cada *buffer*

compartilhado se destinaria ao armazenamento exclusivo das unidades de dados de um dado objeto multimídia requisitado por um grupo de clientes, independentes entre si e conectados na mesma rede local. As unidades de dados, transmitidas pelo servidor de VoD, localizado remotamente, para atendimento de um dos clientes do grupo, seriam armazenadas no *buffer* compartilhado. Desta forma, qualquer um dos clientes pertencentes ao mesmo grupo poderia se beneficiar da solicitação anterior de um outro cliente. A quantidade de *buffers* seria estabelecida em função, por exemplo, do número de objetos mais populares disponíveis no provedor. Notadamente esta função deveria determinar de forma dinâmica a popularidade dos objetos e, conseqüentemente, sua decisão de alocação no conjunto de *buffers* da rede. A complexidade maior que imaginamos existir é a gerência individual (p. ex., política de acesso) de cada um dos *buffers* para permitir um compartilhamento de unidades de dados eficiente e eqüitativo, no sentido de garantir uma mesma QoS, satisfatória para todos os clientes da rede.

Apêndice A

Tabelas Complementares

Apresentamos a seguir tabelas que reúnem alguns resultados numéricos de experimentos realizados no Capítulo 5. Estes resultados são apresentados separadamente do referido capítulo para fins de melhor organização e, portanto, facilidade de leitura e entendimento do mesmo. Agrupamos as tabelas em acordo com o cenário a que se referem. O objetivo é apenas o de ilustração. As conclusões e discussões encontram-se no próprio Capítulo 5.

A.1 Tabelas para Carga eTeach

A Tabela A.1 apresenta o número médio de mensagens para as técnicas PIE e MI. São avaliados os caso CB e SB. É possível notar que, apesar da diminuição do pico da banda para PIE, existe um aumento na complexidade de mensagens. Isso possivelmente devido à fragmentação da informação armazenada localmente.

Técnica	B_{max}	**TB**	**TM**	**TP**	**TF**
PIE-SB	44	25.40	1.08	19.46	13.54
MI-SB	42	25.40	37.50	—	13.54
PIE-CB	41	32.60	1.15	23.98	29.46
MI-CB	42	59.48	61.78	—	56.57

Tabela A.1: Comparativo entre MI e PIE em eTeach

A.2 Tabelas para Carga MANIC-1

A Tabela A.2 apresenta o número médio de mensagens para as técnicas PIE e PIC. São avaliados os casos CB e SB. A Tabela A.3 apresenta o número médio de mensagens para as técnicas MI, considerando a variação do limiar δ_{MI}. Vemos que a mudança perceptível se dá quando não admitimos restrição para seu valor.

Técnica	B_{max}	**TB**	**TM**	**TP**	**TF**
PIC-SB	21	1.70	0.91	1.27	1.58
PIE-SB	20	1.70	0.69	1.27	1.58
PIC-CB	20	1.59	0.78	1.16	1.59
PIE-CB	19	1.58	0.66	1.15	1.58

Tabela A.2: Principais estatísticas da carga MANIC-1

δ_{MI} (%)	B_{max}	**TB**	**TM**	**TF**
100	18	1.70	4.30	1.57
150	17	1.70	4.35	1.57
200	17	1.70	4.38	1.57
não uso	17	1.70	4.38	1.57

Tabela A.3: Principais estatísticas para MI em MANIC-1

A Tabela A.3 apresenta o número médio de mensagens para as técnicas MI e PIE. São considerados os casos SB e CB. Podemos notar que a complexidade de mensagens se altera bem pouco, assim como também não há significativa redução do valor de pico da banda.

A.3 Tabelas para Carga MANIC-2

A Tabela A.5 apresenta o número médio de mensagens para as técnicas PIE e PIC. São considerados os casos SB e CB. Podemos notar que a complexidade de mensagens se altera de forma significativa. A Tabela A.6 traz a complexidade de

Técnica	B_{max}	TB	TM	TP	TF
PIE-SB	20	1.70	0.42	1.40	1.57
MI-SB	17	1.70	4.38	—	1.57
PIE-CB	19	1.60	0.42	1.26	1.60
MI-CB	16	1.38	3.98	—	1.38

Tabela A.4: Taxas de MI e PIE em MANIC-1

mensagens como função do limiar δ_{after} para as técnica PIC. A Tabela A.7 traz esta mesma informação só que para a técnica PIE. Por fim, a Tabela A.8 faz um comparativo entre as técnicas PIE e MI, avaliando as complexidades de mensagem.

Técnica	B_{max}	TB	TM	TP	TF
PIC-SB	41	6.41	0.65	5.31	3.95
PIE-SB	42	6.41	0.40	5.31	3.95
PIC-CB	38	12.86	0.76	10.89	12.21
PIE-CB	41	13.41	0.44	11.53	12.74

Tabela A.5: Estatísticas para PIE e PIC em MANIC-2

δ_{after} (%)	B_{max}	TB	TM	TP	TF
30	37	6.41	1.12	4.58	3.95
40	37	6.41	0.85	5.13	3.95
50	41	6.41	0.65	5.31	3.95
60	39	6.41	0.56	5.47	3.95
70	39	6.41	0.54	5.69	3.95

Tabela A.6: Principais estatísticas para PIC-SB em MANIC-2.

A.4 Tabelas para Carga UOL

A Tabela A.9 apresenta o número médio de mensagens para a técnicas PIE em função do limiar δ_{after}. A Tabela A.10 traz uma análise semelhante considerando

δ_{after} (%)	B_{max}	**TB**	**TM**	**TP**	**TF**
30	37	6.41	0.88	4.59	3.95
40	38	6.41	0.59	5.12	3.95
50	42	6.41	0.40	5.31	3.95
60	39	6.41	0.27	5.61	3.95
70	38	6.41	0.21	5.71	3.95

Tabela A.7: Principais estatísticas para PIE-SB em MANIC-2.

Técnica	B_{max}	**TB**	**TM**	**TP**	**TF**
PIE-SB	42	6.41	0.40	5.31	3.95
MI-SB	33	6.41	12.61	—	3.95
PIE-CB	41	13.41	0.44	11.53	12.74
MI-CB	37	16.35	19.27	—	15.83

Tabela A.8: Comparativo entre MI e PIE em MANIC-2

o limiar δ_{MI} e a técnica MI. Por fim, a Tabela A.11 faz um comparativo entre as técnicas PIE e MI, avaliando as complexidades de mensagem.

δ_{after} (%)	B_{max}	**TB**	**TM**	**TP**	**TF**
80	26	35.30	0.04	26.25	21.92
90	25	35.30	0.05	27.28	21.92
100	27	35.30	0.07	27.50	21.92
110	27	35.30	0.06	27.89	21.92
120	28	35.30	0.07	28.51	21.92
150	31	35.30	0.05	28.92	21.92

Tabela A.9: Principais estatísticas para PIE-SB em UOL.

δ_{MI} (%)	B_{max}	**TB**	**TM**	**TF**
50	19	35.30	60.51	21.92
100	19	35.30	62.90	21.92
150	19	35.30	63.19	21.92
não uso	19	35.30	63.38	21.92

Tabela A.10: Principais estatísticas para MI em UOL

Técnica	B_{max}	**TB**	**TM**	**TP**	**TF**
PIE-SB	27	35.30	0.07	27.50	21.92
MI-SB	19	35.30	63.38	—	21.92
PIE-CB	25	17.84	0.02	13.66	17.67
MI-CB	22	19.71	36.67	—	21.89

Tabela A.11: Comparativo entre MI e PIE em UOL

Apêndice B

Figuras Complementares

Apresentamos a seguir figuras que reúnem alguns resultados numéricos de experimentos realizados no Capítulo 5. Semelhantemente ao apêndice anterior, estes resultados são aqui apresentados a parte para fins de melhor organização e facilidade, portanto, de leitura e entendimento do próprio Capítulo 5. Agrupamos as figuras em acordo com o cenário a que se referem. O objetivo é apenas o de ilustração. As conclusões e discussões encontram-se no próprio Capítulo 5.

B.1 Figuras para Carga eTeach

Na Figura B.1 podemos observar o uso da banda ao longo do tempo devido ao emprego da técnica PIE, e, considerando PIE, PIC e PI, podemos observar as distribuições. Na Figura B.2 também temos as mesmas observações, com a diferença de que nesta última há o uso de CB.

Na Figura B.3 fazemos um comparativo das técnicas PIE e PIC. Na Figura B.4 observamos a banda agregada para PIC. Na Figura B.5 temos o uso da banda ao longo do tempo para MI. Por fim, a Figura B.6 ilustra as variações na distribuição de PIE devido ao emprego do limiar δ_{before}.

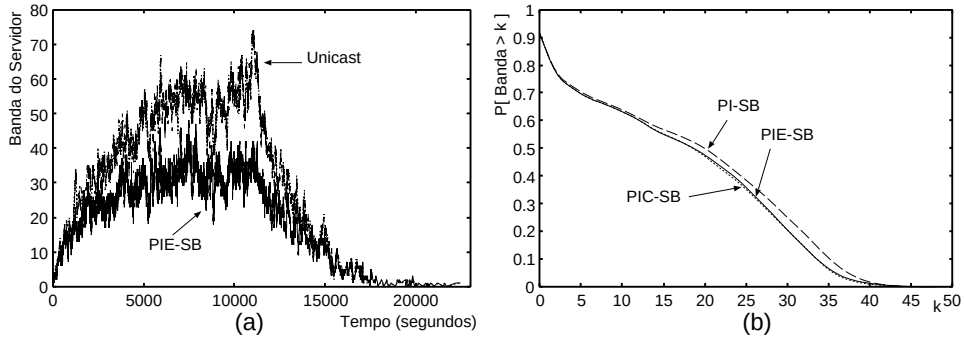


Figura B.1: (a) Banda de PIE-SB; (b) CCDF da banda para PIE, PIC e PI.

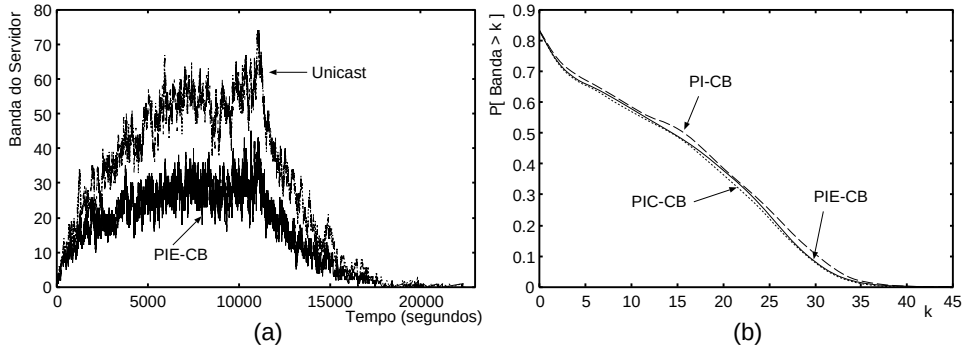


Figura B.2: (a) Banda de PIE-CB; (b) CCDF da banda para PIE, PIC e PI.

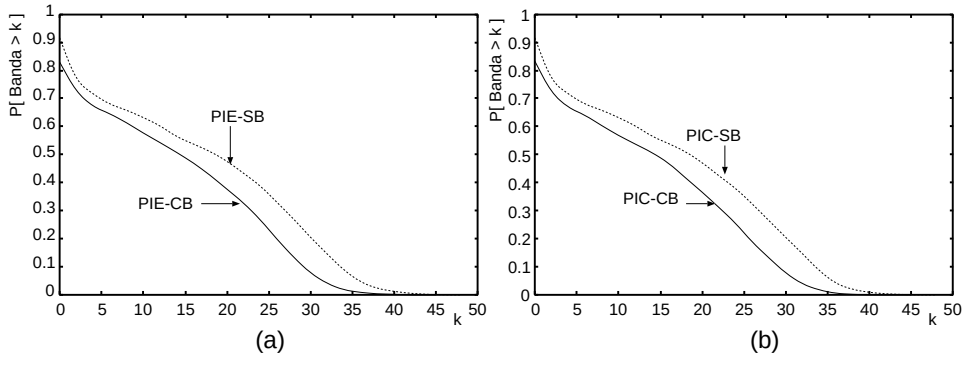


Figura B.3: CCDF da banda para (a) PIE e (b) PIC.

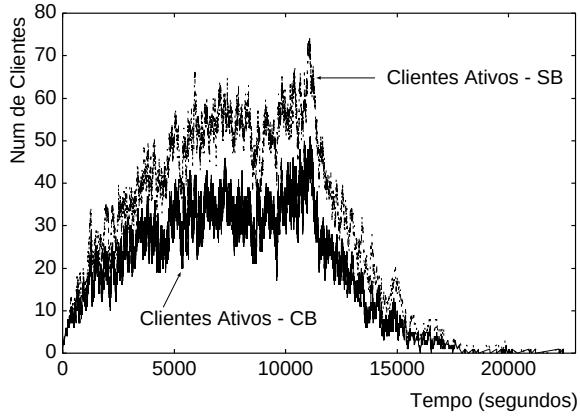


Figura B.4: Banda agregada (clientes ativos) para PIC em eTeach.

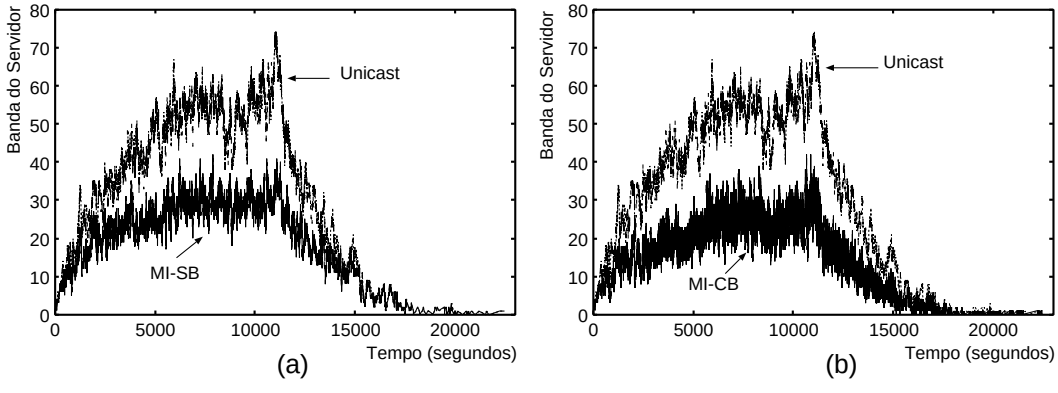


Figura B.5: Banda do servidor para MI em eTeach.

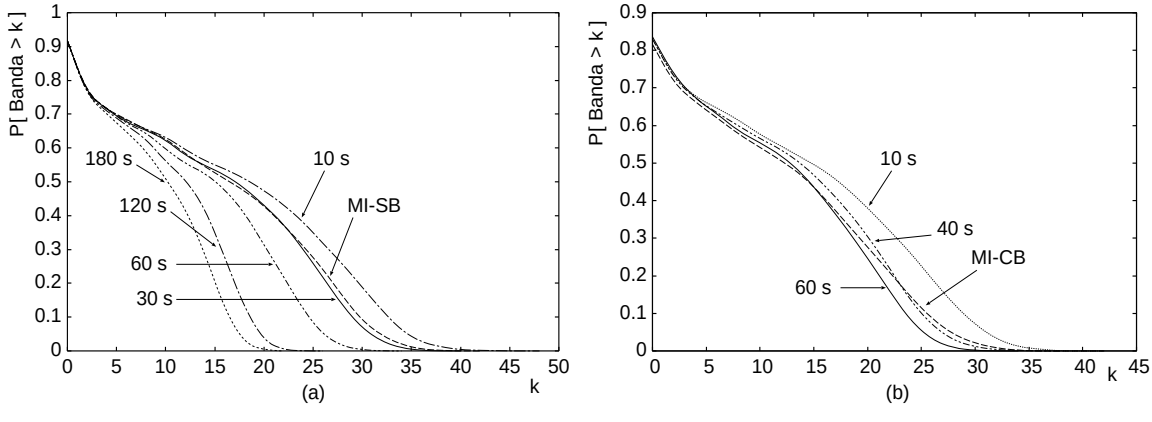


Figura B.6: (a) Estudo do valor de δ_{before} para PIE-SB em eTeach; (b) Estudo do valor de δ_{before} para PIE-CB em eTeach.

B.2 Figuras para Carga MANIC-1

Na Figura B.7 fazemos um comparativo das técnicas PIE e *Patching*. Na Figura B.8 observamos o uso da banda ao longo do tempo para PIE e PIC. Na Figura B.9 podemos observar as variações da distribuição devido ao limiar δ_{merge}. Por fim, a Figura B.10 traz as variações da distribuição de MI devido ao emprego do limiar δ_{MI}.

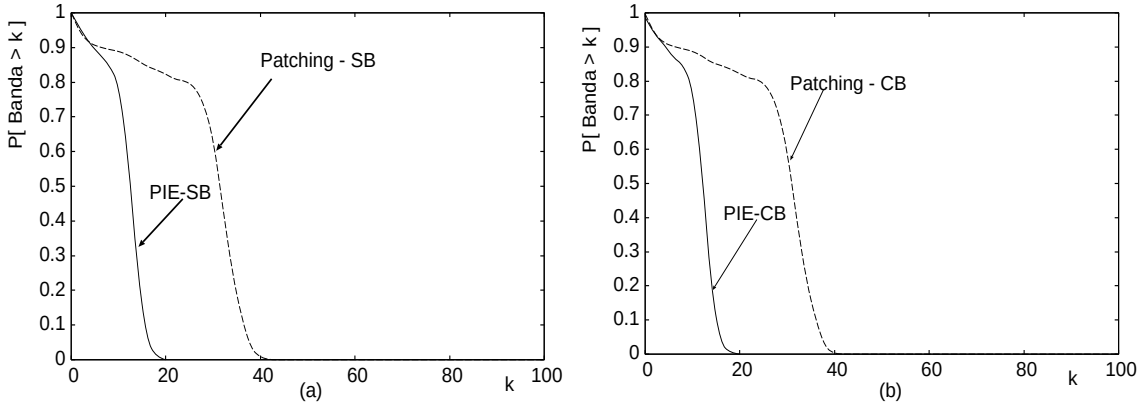


Figura B.7: (a) CCDF para PIE-SB e *Patching*-SB em MANIC-1; (b) CCDF para PIE-CB e *Patching*-CB em MANIC-1.

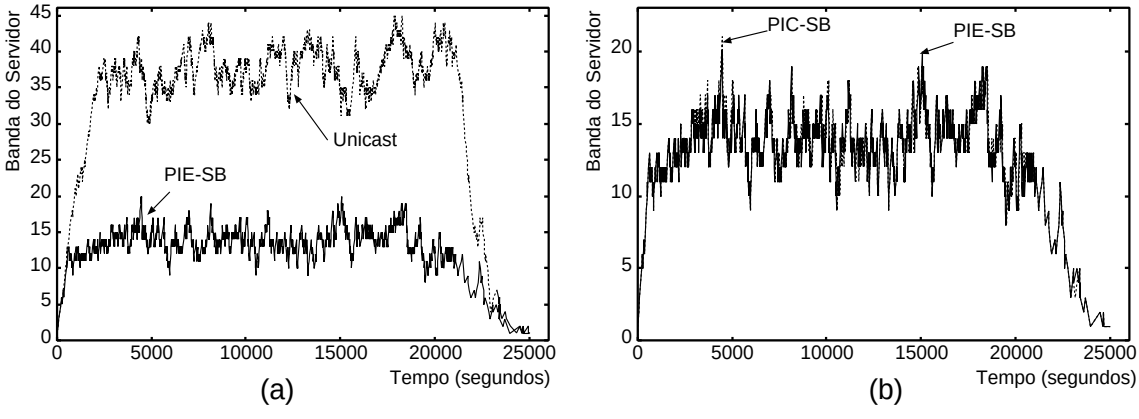


Figura B.8: (a) Banda do servidor para PIE-SB; (b) Banda para PIE-SB e PIC-SB.

Na Figura B.11 temos uma comparação entre as técnicas MI e MI-var. Na Figura B.12 observamos o uso da banda ao longo do tempo para MI. Na Figura B.13 temos a distribuição do número de fluxos *multicast* para PIE e MI. Na Figura B.13 temos a distribuição do número de fluxos *multicast* para PIE e MI. Por fim, a Figura B.14 traz as distribuições de banda considerando a técnica PIE para diferentes valores do limiar δ_{before}.

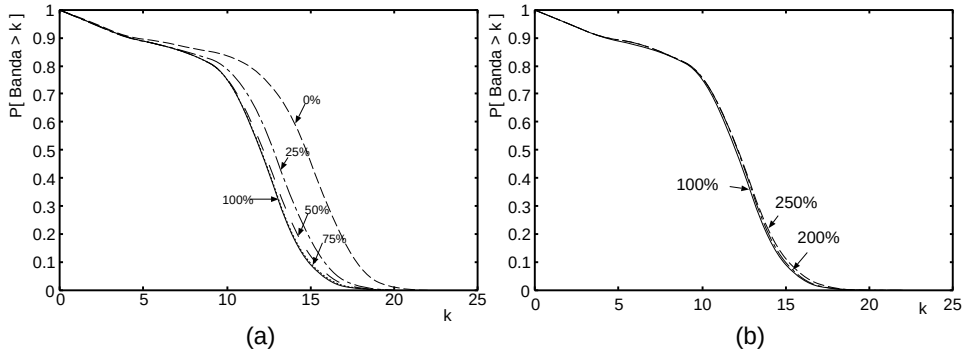


Figura B.9: Estudo do valor de δ_{merge} para PIE

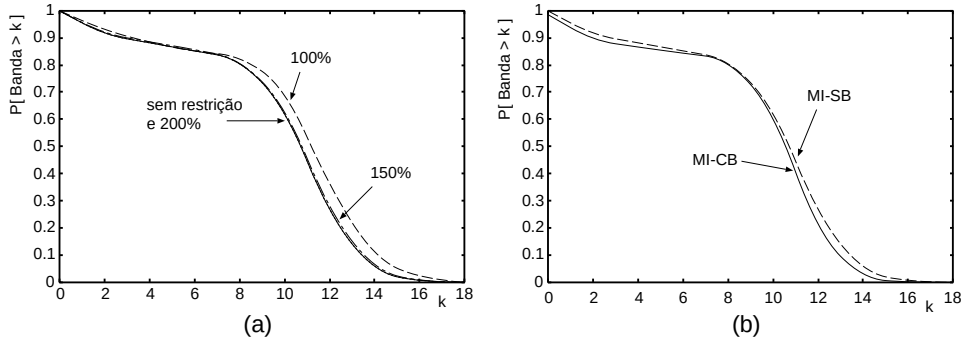


Figura B.10: (a) Estudo do valor de δ_{MI} em MANIC-1; (b) CCDF da banda para MI em MANIC-1.

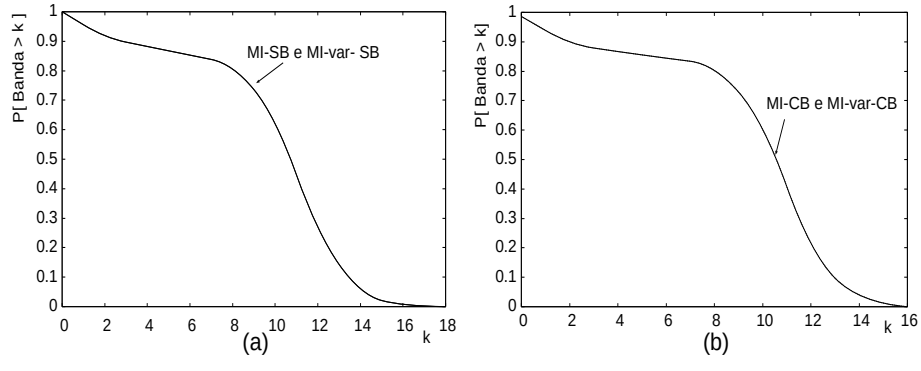


Figura B.11: (a) CCDF para MI-SB e MI-var-SB em MANIC-1; (b) CCDF para MI-CB e MI-var-CB em MANIC-1.

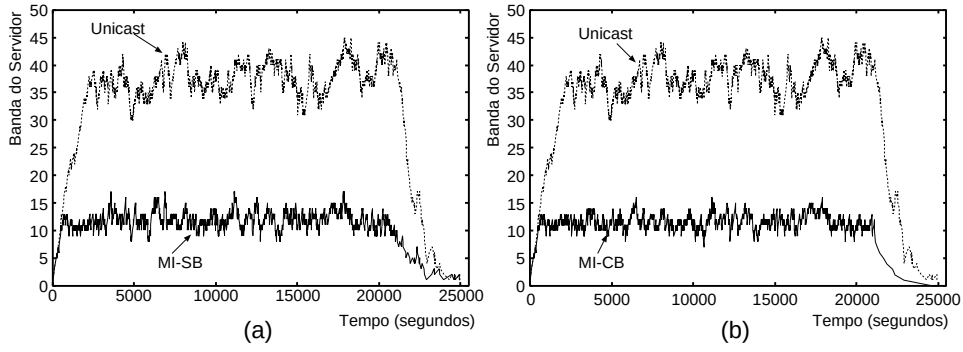


Figura B.12: Banda para a técnica MI em MANIC-1.

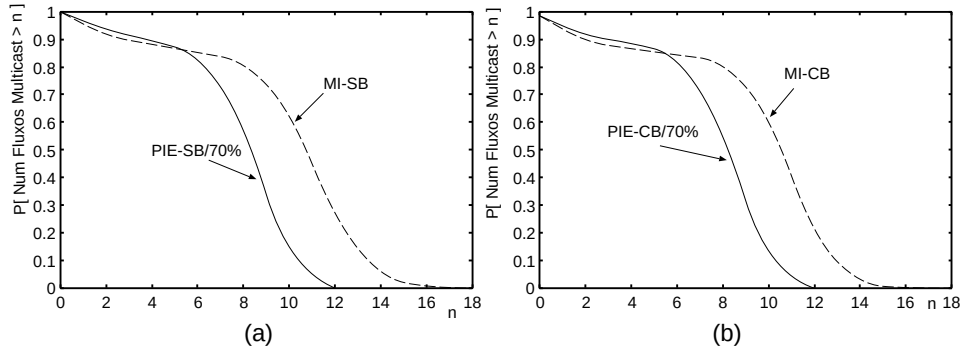


Figura B.13: Distribuições do número de fluxos *multicast* para PIE e MI na Carga MANIC-1.

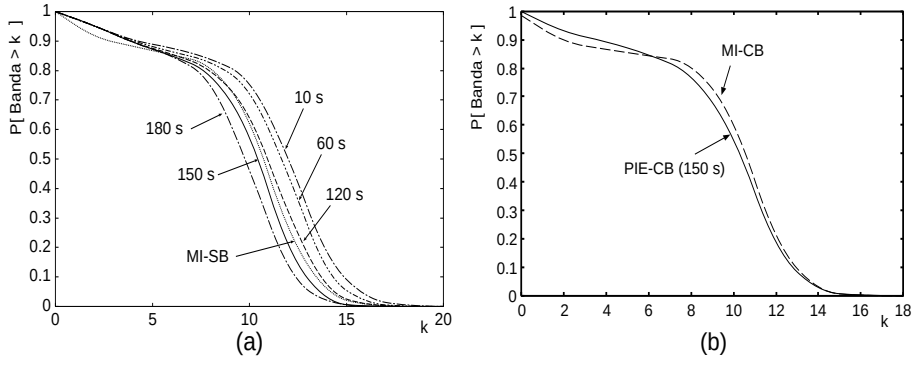


Figura B.14: (a) Estudo do valor de δ_{before} para PIE-SB em MANIC-1; (b) Estudo do valor de δ_{before} para PIE-CB em MANIC-1.

B.3 Figuras para Carga MANIC-2

Na Figura B.15 fazemos um comparativo das técnicas PIE e *Patching*. Na Figura B.16 observamos o uso da banda ao longo do tempo para PIC. Na Figura B.17 podemos observar as distribuições da banda para as técnicas PIE, PIC e PI. Na Figura B.18 temos as distribuições para PIE e PIC.

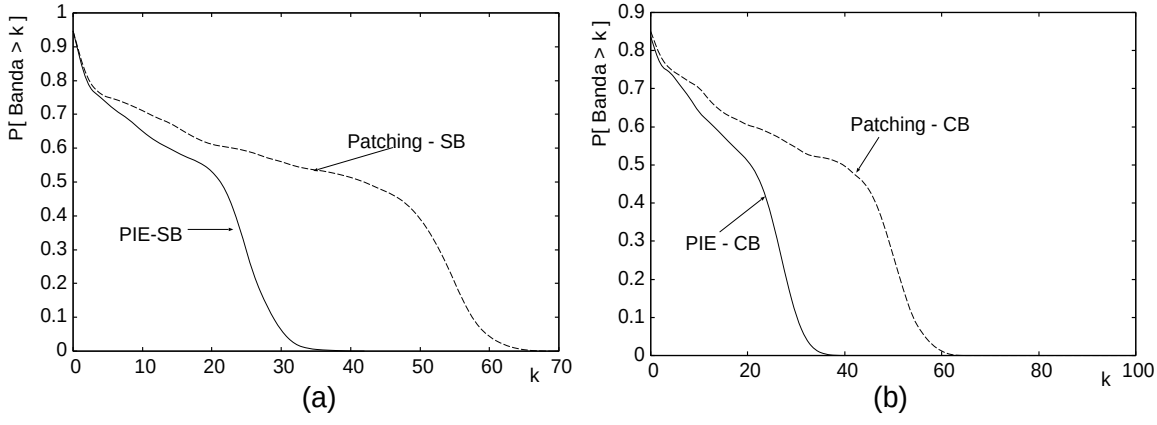


Figura B.15: (a) CCDF para PIE-SB e *Patching*-SB em MANIC-2; (b) CCDF para PIE-CB e *Patching*-CB em MANIC-2.

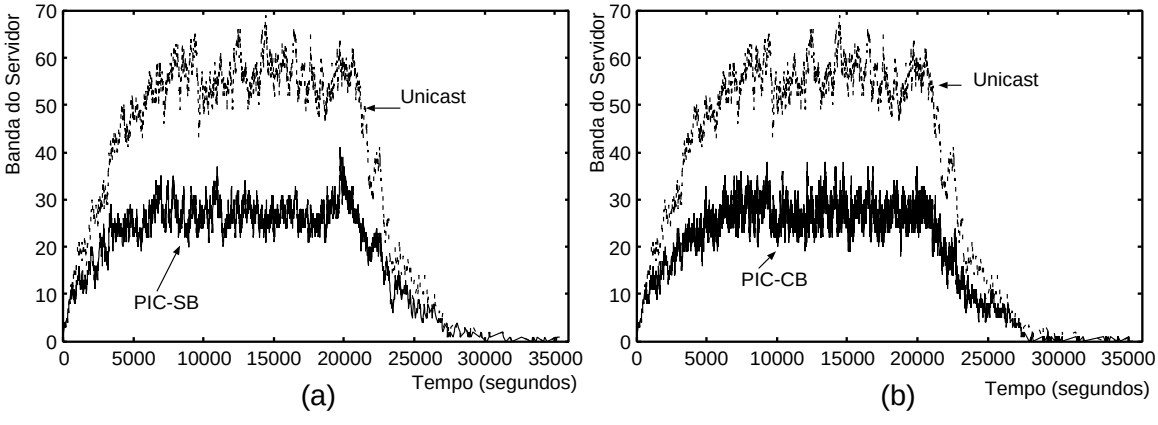


Figura B.16: Banda do servidor para PIC em MANIC-2.

Na Figura B.19 observamos a banda agregada de clientes para PIC. Na Figura B.20 temos um estudo para determinação do limiar δ_{after} para PIC. Na Figura B.21 temos o mesmo estudo só que para PIE. Na Figura B.22 temos as distribuições para MI e CT. Na Figura B.23 temos o uso da banda ao longo do tempo devido ao emprego da técnica MI. A Figura B.24 traz a distribuição do número de fluxos *multicast* para PIE e MI. Por fim, a Figura B.25 traz um estudo para

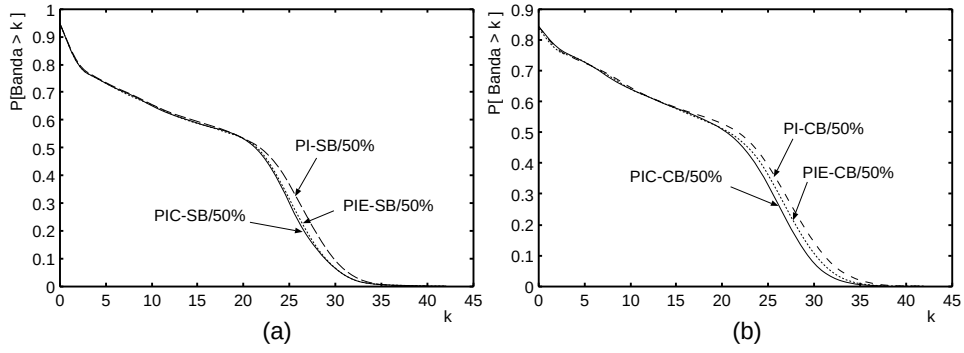


Figura B.17: Distribuição da banda para PIC, PIE e PI para MANIC-2.

determinação do limiar δ_{before} para a técnica PIE.

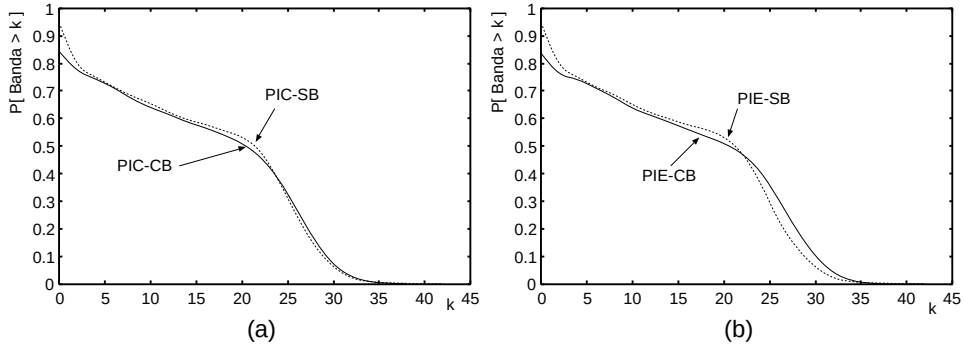


Figura B.18: (a) CCDF da banda para PIC em MANIC-2; (b) CCDF da banda para PIE em MANIC-2.

B.4 Figuras para Carga UOL

Na Figura B.26 temos as distribuições para PIE e *Patching*. Na Figura B.27 temos o uso da banda ao longo do tempo devido ao emprego da técnica PIC. A Figura B.28 traz a distribuição para PIC, PIE e PI. A Figura B.29 apresenta a análise para obtenção do valor ideal do limiar δ_{after} para a técnica PIE. Na Figura B.30 temos as distribuições para PIE objetivando a determinação do valor ideal de δ_{merge} para PIE. Na Figura B.31 temos um estudo semelhante para o limiar δ_{MI} para a técnica MI. A Figura B.32 traz um estudo comparativo a partir das distribuições de MI e MI-var. A Figura B.33 apresenta o uso da banda ao longo do tempo para MI. Por fim, a Figura B.34 apresenta um estudo a partir das distribuições para o limiar δ_{before} considerando a técnica PIE.

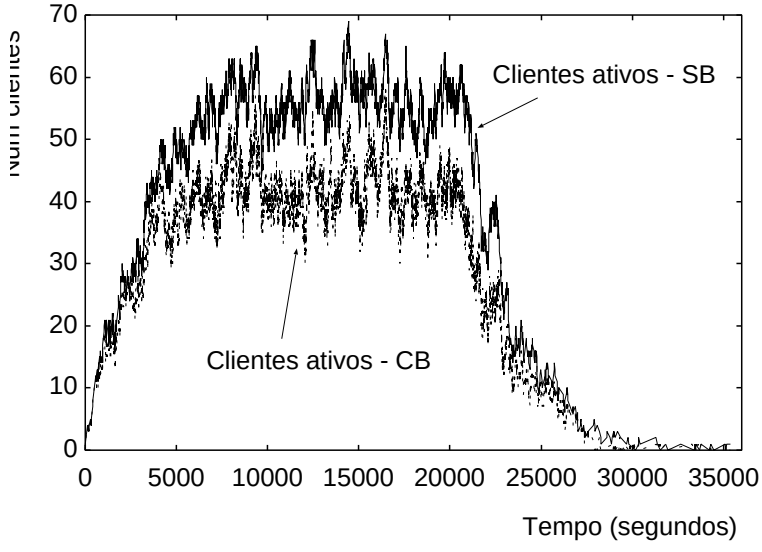


Figura B.19: Clientes ativos para PIC em MANIC-2.

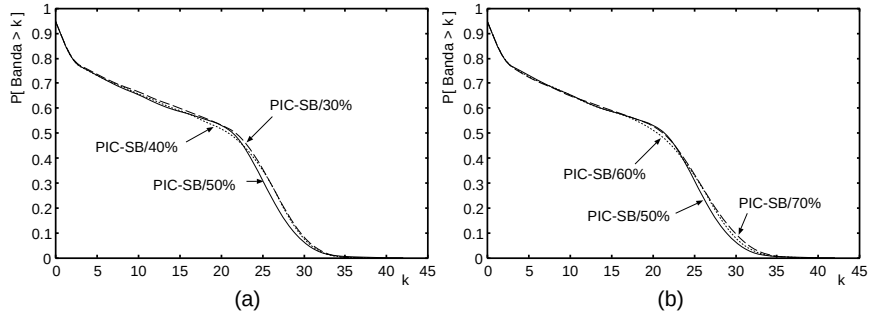


Figura B.20: Estudo para determinação do valor de δ_{after} para PIC em MANIC-2.

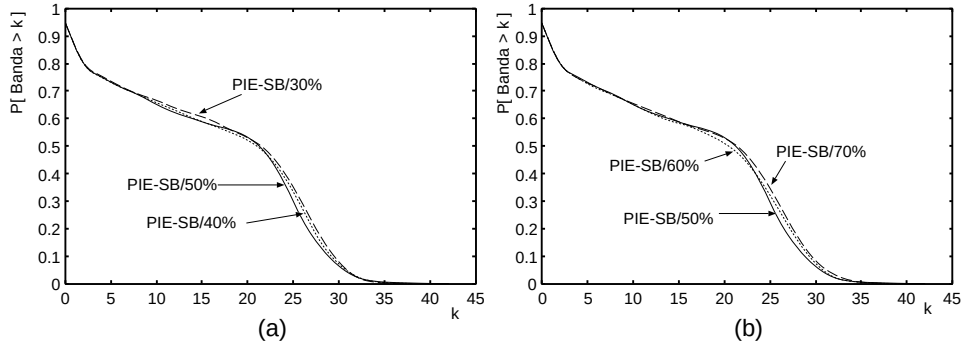


Figura B.21: Estudo para determinação do valor de δ_{after} para PIE em MANIC-2.

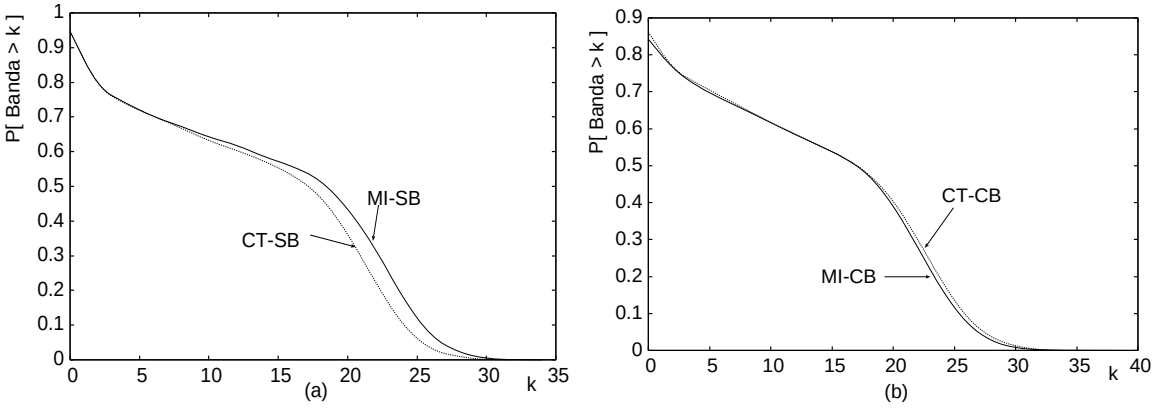


Figura B.22: (a) CCDF para MI-SB e CT-SB em MANIC-2; (b) CCDF para MI-CB e CT-CB em MANIC-2.

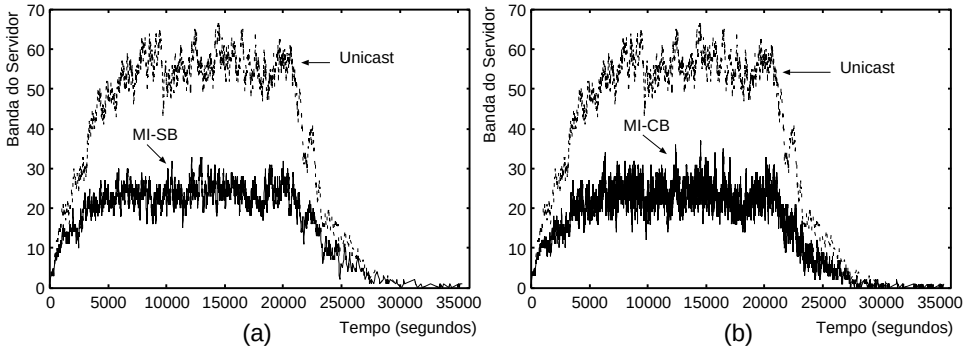


Figura B.23: Uso de banda ao longo do tempo para a técnica MI em MANIC-2.

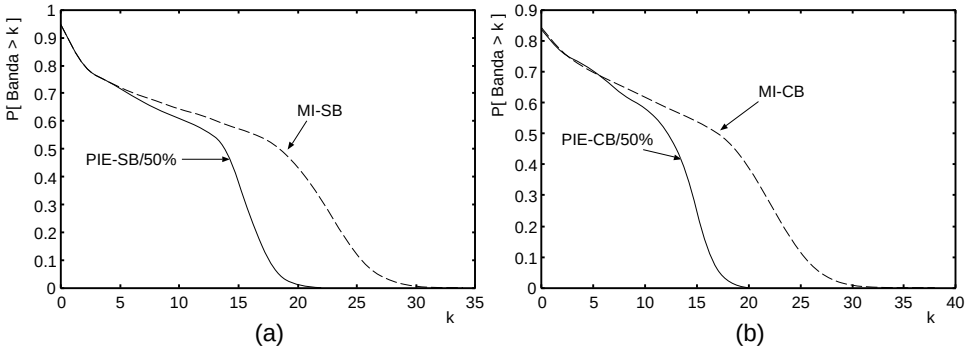


Figura B.24: CCDF do número de fluxos *multicast* para PIE e MI em MANIC-2.

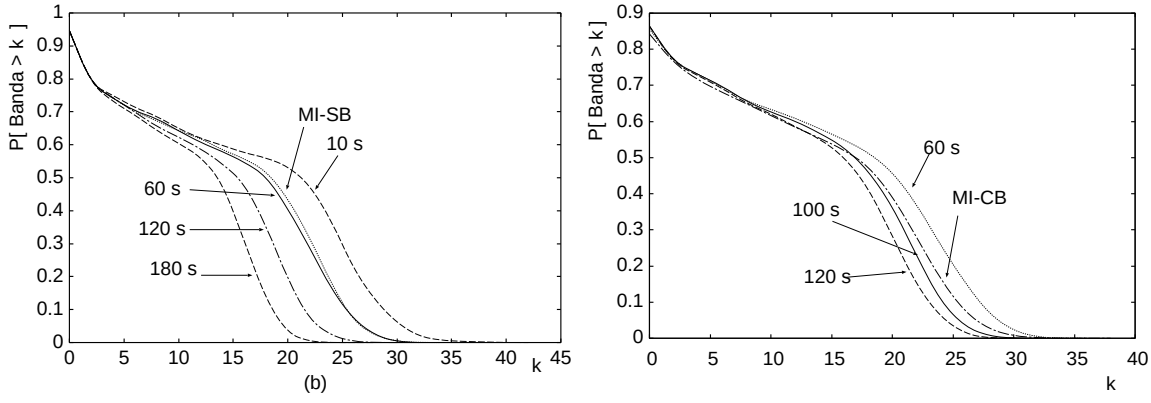


Figura B.25: (a) Estudo do valor de δ_{before} para PIE-SB em MANIC-2; (b) Estudo do valor de δ_{before} para PIE-CB em MANIC-2.

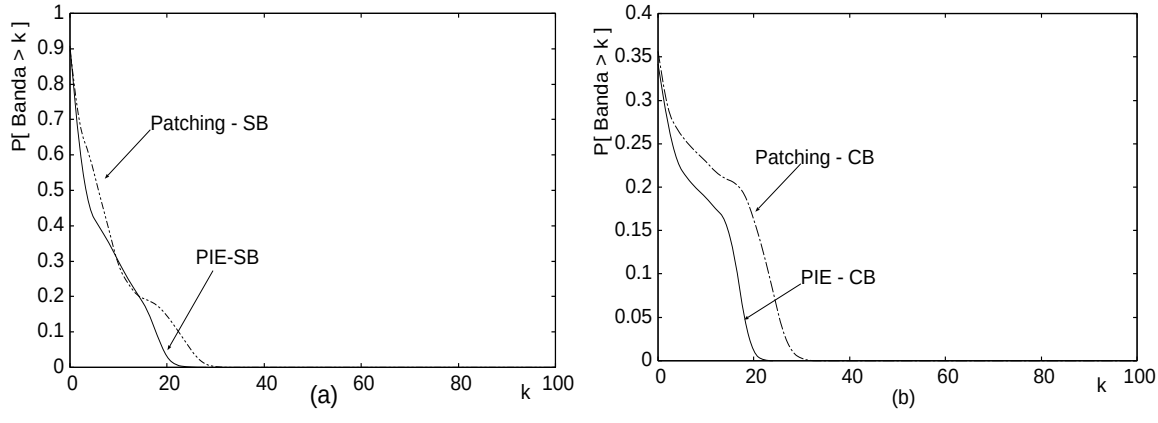


Figura B.26: (a) CCDF para PIE-SB e Patching-SB em UOL; (b) CCDF para PIE-CB e Patching-CB em UOL.

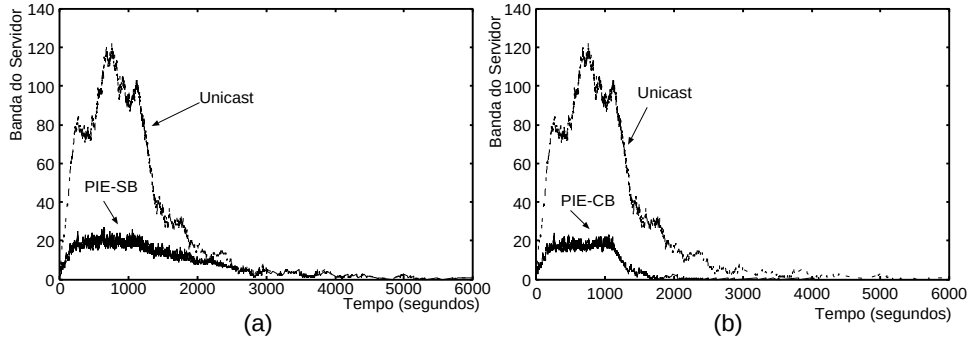


Figura B.27: Banda do servidor para PIC em UOL.

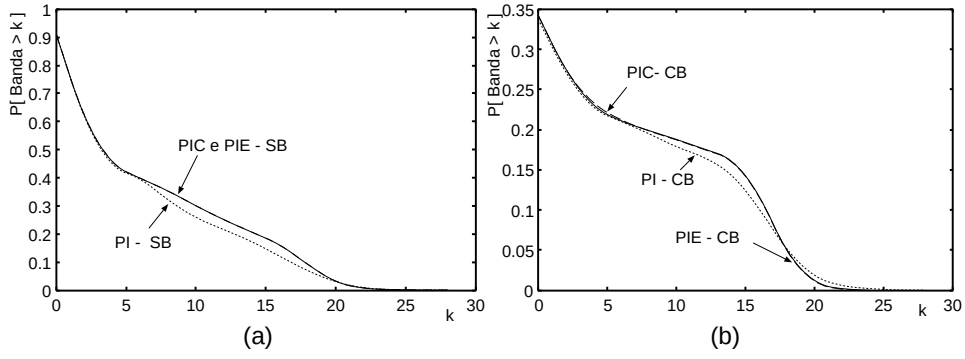


Figura B.28: CCDF da banda para PIC, PIE e PI em UOL.

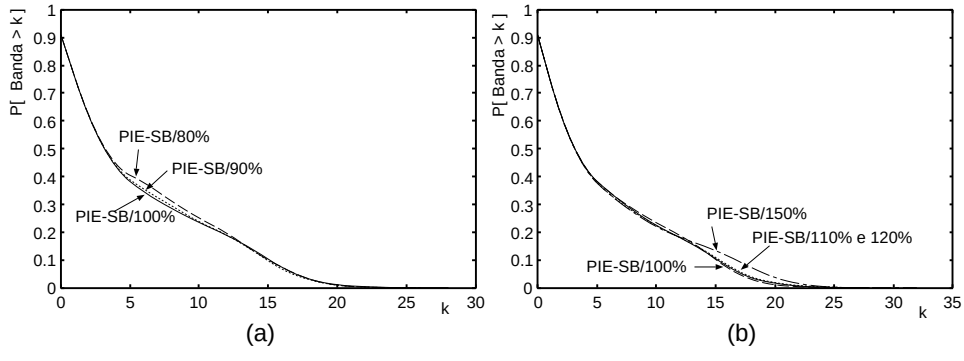


Figura B.29: Estudo para determinação de δ_{after} para PIE em UOL.

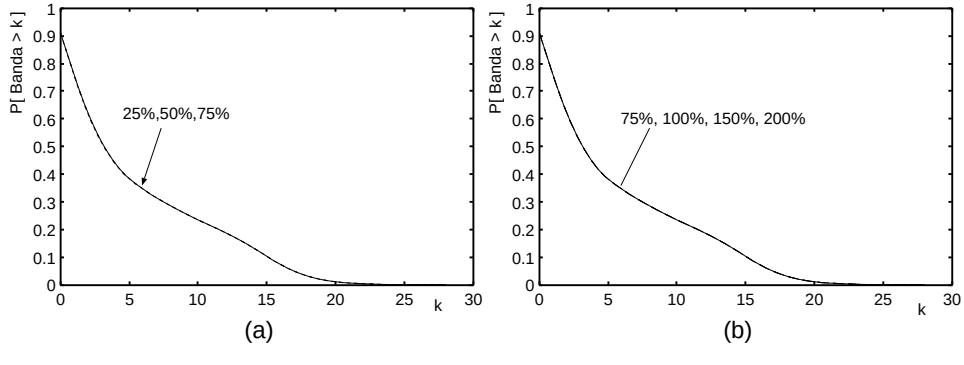


Figura B.30: Estudo para determinação de δ_{near} para PIE em UOL.

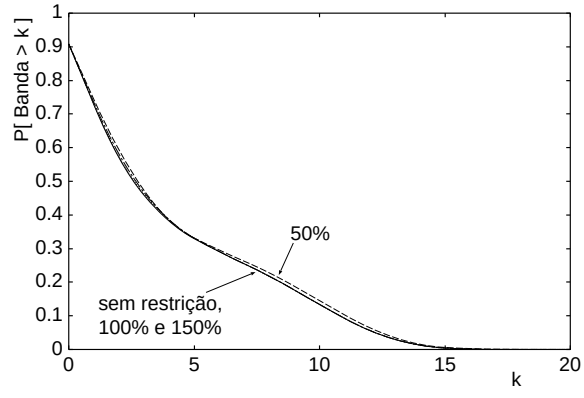


Figura B.31: Estudo para determinação do parâmetro δ_{MI} em UOL

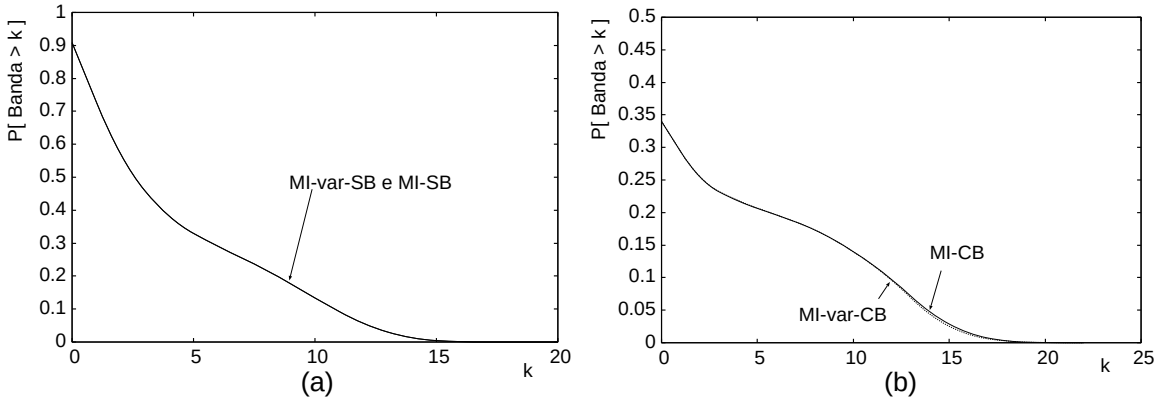


Figura B.32: (a) CCDF para MI-SB e MI-var-SB em UOL; (b) CCDF para MI-CB e MI-var-CB em UOL.

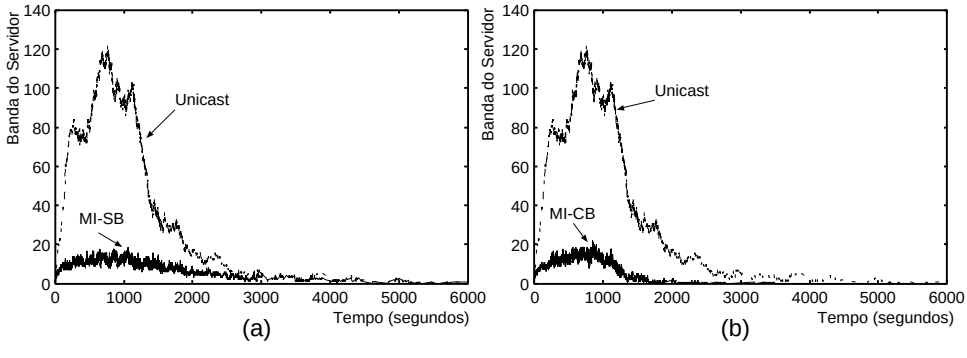


Figura B.33: Uso de banda ao longo do tempo para a técnica MI em UOL.

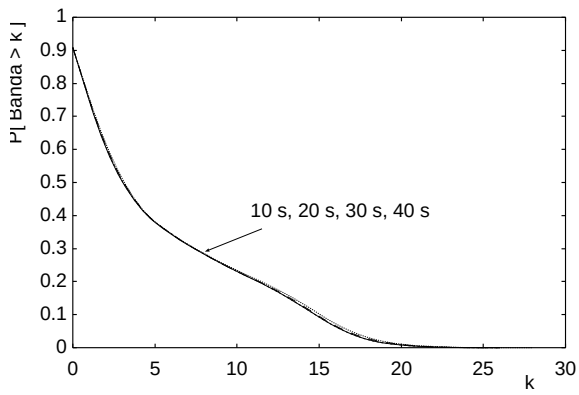


Figura B.34: Estudo do valor de δ_{before} para PIE-SB em UOL.

Referências Bibliográficas

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. On optimal piggyback merging policies for video-on-demand systems. In *Proc. 1996 ACM SIGMETRICS Conf. On Measurement and Modeling of Computer Systems*, pages 200–209, 1996.

[2] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true video-on-demand services. In *Proc. 6th ACM Int'l Multimedia Conf. (ACM MULTIMEDIA'98)*, pages 191–200, 1998.

[3] Y. Cai and K. A. Hua. An efficient bandwidth-sharing technique for true video on demand systems. In *Proc. of the 7th ACM Int'l Multimedia Conference (ACM Multimedia'99)*, pages 211–214, November 1999.

[4] S. Sen, L. Gao, J. Rexford, and D. Towsley. Optimal patching schemes for efficient multimedia streaming. In *Proc. 9th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99)*, June 1999.

[5] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. In *Proc. 5th Int'l Workshop on Multimedia Information Systems (MIS'99)*, pages 80–87, October 1999.

[6] D. Eager, M. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for video-on-demand servers. In *Proc. 7th ACM Int'l Multimedia Conference (ACM Multimedia'99)*, pages 199–202, November 1999.

[7] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth skimming: A technique for cost-effective video-on-demand. In *Proc. IS&T/SPIE MMCN'00*, pages 206–215, January 2000.

[8] A. Bar-Noy and R. E. Ladner. Competitive on-line stream merging algorithms for media-on-demand. In *Proc. of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2001.

[9] E. G. Coffman Jr., P. Jelenkovic, and P. Momcilovic. Provably efficient stream merging. In *Proc. 6th Int'l Workshop on Web caching and Content Distribution*, Boston, MA, June 2001.

[10] Wun-Tat Chan, Tak-Wah Lam, Hing-Fung Ting, and Wai-Ha Wong. A unified analysis of hot video schedulers. In *Proc. of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 179–188, 2002.

[11] Wun-Tat Chan, Tak-Wah Lam, Hing-Fung Ting, and Wai-Ha Wong. On-line stream merging, max span, and min coverage. In *Proc. of the 5th Conference on Algorithms and Complexity (CIAC)*, pages 337–346, 2003.

[12] K. C. Almeroth and M. H. Ammar. The use of multicast delivery to provide a scalable and interactive video-on-demand service. *IEEE Journal on Selected Areas in Communications*, 14(5):1110–1122, August 1996.

[13] Ailan Hu. Video-on-demand broadcasting protocols: A comprehensive study. In *Proc. IEEE Infocom*, pages 508–517, 2001.

[14] S. Viswanathan and T. Imielinski. Pyramid broadcasting for video-on-demand service. In *Proc. of the SPIE Multimedia Computing and Networking Conference*, pages 66–77, February 1995.

[15] K. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proc. ACM SIGCOMM*, September 1997.

[16] L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, 44(1):100–105, Mar 1998.

[17] L. Juhn and L. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3):268–271, Sept 1997.

[18] J.-F. Paris, S. W. Carter, and D. D. E. Long. A low bandwidth broadcasting protocol for video on demand. In *Proc. of IEEE Int'l Conference on Computer Communications and Networks (IC3N'98)*, 1998.

[19] J.-F. Paris, S. W. Carter, and D. D. E. Long. A hybrid broadcasting protocol for video on demand. In *Proc. 1999 Multimedia Computing Networking Conference (MMCN'99)*, January 1999.

[20] J.-F. Paris, S. W. Carter, and D. D. E. Long. A simple low-bandwidth broadcasting protocol for video on demand. In *Proc. 8th Int'l Conference on Computer Communications and Networks (IC3N'99)*, October 1999.

[21] R.M. Zafalão, N.L.S. Fonseca, and C.C. de Souza. The Polyharmonic Broadcasting Protocol subject to Bandwidth Constraints. In *Brazilian Computer Networks Conference*, pages 397–410, 2003. In Portuguese.

[22] L. Gao and D. Towsley. Supplying instantaneous video-on-demand services using controlled multicast. In *Proc. IEEE Multimedia Computing Systems'99*, pages 117–121, June 1999.

[23] Y. Chai, K. Hua, and K. Vu. Optimizing patching performance. In *Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, pages 204–215, January 1999.

[24] D. L. Eager, M. C. Ferris, and M. Vernon. Optimized regional caching for on-demand data delivery. In *Proc. ISET/SPIE Conf. Multimedia Computing and Networking (MMCN'99)*, pages 301–316, January 1999.

[25] H. Tan, D. Eager, M. Vernon, and H. Guo. Quality of service evaluations of multicast streaming protocols. In *Proc. ACM SIMETRICS 2002. Int'l Conference on Measurement and Modeling of Computer Systems*, June 2002.

[26] S. Jin and A. Bestavros. Scalability of multicast delivery for non-sequential streaming access. In *Proc. of ACM SIGMETRICS 2002*, Marina Del Rey, CA, June 2002.

[27] L. Gao, J. Hurose, and D. Towsley. Efficient schemes for broadcasting popular videos. In *Proc. IEEE NOSSDAV'98*, Cambridge, UK, July 1998.

[28] W. Liao and V. O. K. Li. The split and merge protocol for interactive video-on-demand. *IEEE Multimedia*, 4(4):51–62, Oct 1997.

[29] Huadong Ma and K. G. Shin. A new scheduling scheme for multicast true VoD service. *Lecture Notes in Computer Science*, 2195:708–715, Oct 2001.

[30] E. L. Abram-Profeta and K. G. Shin. Providing unrestricted VCR functions in multicast video-on-demand servers. In *Proc. of IEEE ICMCS*, pages 66–75, Austin, Texas, June 1998.

[31] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. Second ACM Int'l Multimedia (ACM Multimedia'94)*, pages 15–23, Oct 1994.

[32] A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic batching policies for an on-demand video server. *Multimedia Systems*, 4(3):112–121, June 1996.

[33] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. On optimal batching policies for video-on-demand storage server. In *Proc. of IEEE Conf. on Multimedia Systems*, 1996.

[34] R. O. Banker et al. Method of providing video-on-demand with VCR-like functions. U. S. Patent 5357276, 1994.

[35] C. K. da S. Rodrigues. Mecanismos para compartilhamento de recursos para transmissão de mídia contínua na internet. Exame de Qualificação, UFRJ, COPPE/PESC, Julho 2004.

[36] C. K. da S. Rodrigues and R. M. M. Leão. Cálculo da distribuição da banda para servidores de vídeo. In *III Workshop de Desempenho em Sistemas Computacionais e de Comunicação (WPerformance), XXIV Congresso da Sociedade Brasileira de Computação (SBC2004)*, pages 1362–1380, Salvador, BA, Brasil, Agosto 2004.

[37] C. K. da S. Rodrigues and R. M. M. Leão. Bandwidth Usage Distribution of Multimedia Servers using Patching. *Computer Networks*, 2006. Submitted.

[38] B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal proxy cache allocation for efficient streaming media distribution. *IEEE Transactions on Multimedia*, pages 1726–1735, 2004.

[39] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2cast: peer-to-peer patching scheme for VoD service. In *Proc. of the 12th World Wide Web Conference (WWW-03)*, May 2003.

[40] M. K. Bradshaw, B. Wang, S. Sen, L. Gao, J. Kurose, P. Shenoy, and D. Towsley. Periodic broadcast and patching services - implementation, measurement and analysis in an internet streaming video testbed. *ACM Multimedia Journal SI on Multimedia Distribution*, 2003.

[41] D. Guan and S. Yu. A two-level patching scheme for video-on-demand delivery. *IEEE Transactions on Broadcasting*, 50(1):11–15, March 2004.

[42] C. Griwodz, M. Liepert, M. Zink, and R. Steinmetz. Tune to lambda patching. *ACM SIGMETRICS Performance Evaluation Review*, 27(4):20–26, 2000.

[43] Y. W. Wong and Jack Y. B. Lee. Recursive Patching - An efficient technique for multicast video streaming. In *Proc. 5th International Conference on Enterprise Information Systems (ICEIS) 2003*, pages 23–26, Angers, France, April 2003.

[44] Y. Cai and K. Hua. Sharing multicast videos using patching streams. *Journal of Multimedia Tools and Applications*, 21(2):125–146, 2003.

[45] Y. Cai, W. Tavanapong, and K. Hua. Enhancing patching performance through double patching. In *Proc. Int'l. Conf. on Distributed Multimedia Systems*, pages 72–77, Miami, FL, USA, 2003.

[46] E. de Souza e Silva, R. M. M. Leão, and M. C. Diniz. The Required Capacity Distribution for the Patching Bandwidth Sharing Technique. In *Proc. of IEEE ICC2004*, May 2004.

[47] A. Bar-Noy, G. Goshi, R. E. Ladner, and K. Tam. Comparison of stream merging algorithms for media-on-demand. In *Proc. of MMCN 2002*, pages 18–25, San Jose, CA, January 2002.

[48] A. Bar-Noy and R. E. Ladner. Efficient algorithms for optimal stream merging for media-on-demand. *SIAM Journal on Computing*, 33(5):1011–1034, 2004.

[49] E. O. Brigham. *The Fast Fourier Transform and its applications*. Prentice-Hall, Inc., New Jersey, 1988.

[50] C. K. da S. Rodrigues and R. M. M. Leão. Novas técnicas de compartilhamento de banda para servidores de vídeo sob demanda com interatividade. In *XXIII Simpósio Brasileiro de Redes de Computadores (SBRC2005)*, pages 451–464, Fortaleza, CE, Brasil, Maio 2005.

[51] C. K. da S. Rodrigues and R. M. M. Leão. Techniques for Scalable Interactive Streaming. In *49th Annual IEEE Globecom Conference*, San Francisco, California, USA, December 2006. Submitted.

[52] E. de Souza e Silva, R. M. M. Leão, B. Ribeiro-Neto, and S. Campos. Performance issues of multimedia applications. *Lecture Notes in Computer Science*, 2459:374–404, July 2002.

[53] Huadong Ma and K. G. Shin. Multicast video-on-demand services. *ACM SIGCOMM Computer Communication Review*, 32(1):31–43, 2002.

[54] S. W. Carter and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. In *Proc. Sixth Int'l Conf. Computer Comm. and Networks (ICCCN'97)*, pages 200–207, September 1997.

[55] Y. Cai, K. Hua, and K. Vu. Optimizing Patching Performance. In *Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, pages 204–215, 1999.

[56] C. K. da S. Rodrigues and R. M. M. Leão. Técnicas para sistemas de vídeo sob demanda escaláveis. In *XXIV Simpósio Brasileiro de Redes de Computadores (SBRC2006)*, Curitiba, PR, Brasil, Maio 2006. Aceito.

[57] Wun-Tat Chan, Tak-Wah Lam, Hing-Fung Ting, and Wai-Ha Wong. A 5-competitive on-line scheduler for merging video streams. In *Proc. of the 15th Int'l Parallel and Distributed Processing Symposium*, 2001.

[58] Wun-Tat Chan, Tak-Wah Lam, Hing-Fung Ting, and Wai-Ha Wong. Competitive analysis of on-line stream merging algorithms. In *Proc. of the 27th Int'l Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 188–200, 2002.

[59] Wun-Tat Chan, Tak-Wah Lam, Hing-Fung Ting, and Wai-Ha Wong. On-line stream merging in a general setting. *Theoretical Computer Science*, 296(1):27–46, 2003.

[60] L. Golubchick, J. C. S. Lui, and R. R. Muntz. Reducing i/o demand in video-on-demand storage servers. In *Proc. ACM Sigmetrics'95*, pages 25–36, May 1995.

[61] L. Golubchick, J. C. S. Lui, and R. R. Muntz. Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers. *ACM Multimedia Systems*, 4(3):140–155, 1996.

[62] S. W. Lau, J. C. S. Lui, and L. Golubchik. Merging video streams in a multimedia storage server: Complexity and heuristics. *ACM Multimedia Systems Journal*, 6(1):29–42, 1998.

[63] D. L. Eager and M. K. Vernon. Dynamic skyscraper broadcasts for video-on-demand. In *Proc. 4th Int'l Workshop Multimedia Information Systems (MIS'98)*, pages 18–32, Sept. 1998.

[64] E. G. Coffman Jr., P. Jelenkovic, and P. Momcilovic. The dyadic stream merging algorithm. *Journal of Algorithms*, 43(1):120–137, 2002.

[65] A. Borodin and R. El-Yaniv. *On-line computation and competitive analysis.* Cambridge University Press, The Pitt Building, Trumpington Street, Cambridge, United Kingdom, 1998.

[66] A. Dan, D. Sitaram, P. Shahabuddin, and D. Towsley. Channel allocation under batching and VCR control in movie-on-demand servers. *Journal of Parallel and Distributed Computing*, 30(2):168–179, November 1995.

[67] W. W. F. Poon and K. T. Lo. Design of multicast delivery for providing VCR functionality in interactive video-on-demand systems. *IEEE Transactions on Broadcasting*, 45(1):141–148, March 1999.

[68] Huadong Ma, K. G. Shin, and W. B. Wu. Best-effort patching for multicast true VoD service. *Multimedia Tools Applications*, 26(1):101–122, 2005.

[69] S.-H. Gary Chan and Edward Chang. Providing scalable on-demand interactive video service by means of multicast and client buffering. In *Proc. IEEE ICC*, pages 11–15, Helsinki, June 2001.

[70] B. C. M. Netto. Patching interativo: Um novo método de compartilhamneto de recursos para transmissão de vídeo com alta interatividade. Tese de Mestrado, UFRJ, COPPE/PESC, Fevereiro 2004.

[71] M. L. Gorza. Uma técnica de compartilhamento de recursos para transmissão de vídeo com alta interatividade e experimentos. Tese de Mestrado, UFF, Departamento de Ciência da Computação, Dezembro 2003.

[72] H. Tan, D. Eager, and M. Vernon. Delimiting the range of effectiveness of scalable on-demand streaming. *Performance Evaluation*, 49:387–410, 2002.

[73] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon. Analysis of educational media server workloads. In *Proc. 11th Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'01)*, pages 21–30, June 2001.

[74] M. Rocha, M. Maia, I. Cunha, J. Almeida, and S. Campos. Scalable Media Streaming to Interactive Users. In *MULTIMEDIA'05: Proceedings of the 13th annual ACM international conference on Multimedia*, Singapore, November 2005.

[75] M. Rocha, M. Maia, J. Almeida, and S. Campos. Scalabilidade de Protocolos com Compartilahmento de Banda para Cargas de Mídia Contínua Realistas. In *IV Workshop de Desempenho em Sistemas Computacionais e de Comunicação (WPerformance), XXV Congresso da Sociedade Brasileira de Computação (SBC2005)*, São Leopoldo, RS, Julho 2005.

[76] Leonard Kleinrock. *Queueing Systems*, volume 1: Theory. John Wiley & Sons, New York, 1975.

[77] C. Costa, I. Cunha, A. Borges, C. Ramos, M. Rocha, J. M. Almeida, and B. Ribeiro-Neto. Analyzing client interactivity in streaming media. In *Proc. 13th ACM Int'l World Wide Web Conference*, pages 534–543, May 2004.

[78] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley & Sons, Inc., second edition, New York, 2002.

[79] R.M.L.R. Carmo, L.R. de Carvalho, E. de Souza e Silva, M.C. Diniz, and R.R. Muntz. Performance/Availability Modeling with the TANGRAM-II Modeling Environment. *Performance Evaluation*, 33:45–65, 1998.

[80] E. de Souza e Silva and Rosa M. M. Leão. The Tangram-II Environment. In *Proceedings of 11th International Conference TOOLS2000*, pages 366–369, 2000. LNCS 1786.

[81] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.

[82] A. L. Chervenak. *Tertiary storage: An evaluation of new applications*. PhD thesis, CSD, University of California at Berkeley, 1994.

[83] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming-media workload. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, 2001.

[84] *eTeach - Learning on Demand*. http://eteach.engr.wisc.edu/newEteach/home.html.

[85] J. Padhye and J. Kurose. An empirical study of client interactions with continuous-media courseware server. In *Proc. IEEE NOSSDAV*, Cambridge, UK, July 1998.

[86] J. F. Kurose, H. I. Lee, J. Steinberg, and M. Stern. Manic: Multimedia asynchronous networked individualized courseware. Technical Report 96-72, Department of Computer Science, University of Massachusetts, Oct 1996.

[87] W. Mendenhall and T. Sincich. *Statistics for Engineering and the Sciences.* Macmillan, third edition, 1991.

[88] Harold J. Larson. *Introduction to Probability Theory and Statistical Inference.* John Wiley & Sons, Inc., third edition, 1982.

[89] Raj Jain. *The art of Computer Systems Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley & Sons, Inc., New York, 1991.

[90] L. He, J. Grudin, and A. Gupta. Designing presentations for on-demand viewing. In *Proc. ACM 2000 Conf. On Computer Supported Cooperative Work*, pages 127–134, Philadelphia, PA, Dec 2000.

[91] Sheldon M. Ross. *Stochastic Processes.* John Wiley & Sons, Inc., second edition, New York, 1996.

[92] Huadong Ma and K. G. Shin. Performance analysis of the interactivity for multicast TVoD service. In *Proc. IEEE International Conference on Computer Communications and Networks (IEEE ICCCN'01)*, Phoenix, Oct 2001.