Gerenciamento de Banda Passante
em Servidores de Vídeo

Roberto de Almeida Façanha

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Gerenciamento de Banda Passante em Servidores de Vídeo

Roberto de Almeida Façanha[1]

Agosto de 1998

Banca Examinadora:

- Prof. Dr. Nelson Luís Saldanha da Fonseca
  Instituto de Computação, Unicamp (Orientador)
- Prof. Dr. Luiz Fernando Gomes Soares
  Departamento de Informática, PUC-Rio
- Prof. Dr. Edmundo Roberto Mauro Madeira
  Instituto de Computação, Unicamp
- Prof. Dr. Célio Cardoso Guimarães
  Instituto de Computação, Unicamp (Suplente)

[1] O autor é Bacharel em Ciências da Computação pela Universidade Estadual do Ceará.

Gerenciamento de Banda Passante em Servidores de Vídeo

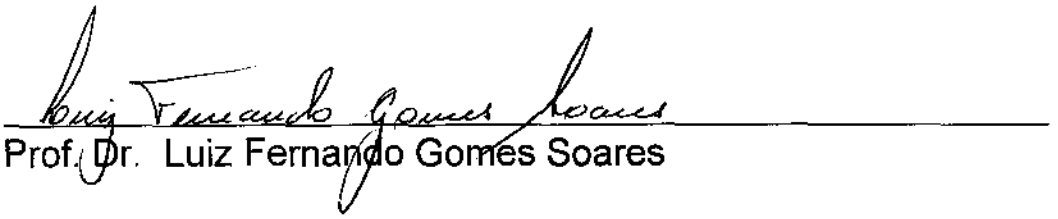
Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Roberto de Almeida Façanha e aprovada pela Banca Examinadora.

Campinas, 28 de agosto de 1998.

Prof. Dr. Nelson Luís Saldanha da Fonseca
Instituto de Computação, Unicamp
(Orientador)

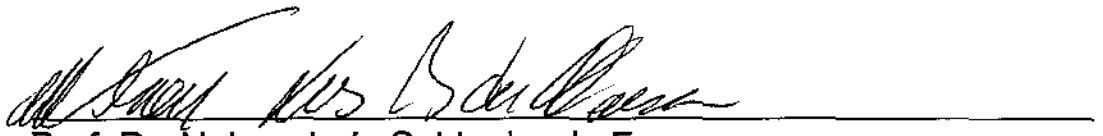
Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Tese de Mestrado defendida e aprovada em 24 de agosto de 1998 pela Banca Examinadora composta pelos Professores Doutores



Prof. Dr. Luiz Fernando Gomes Soares

Prof. Dr. Edmundo Roberto Mauro Madeira



Prof. Dr. Nelson Luís Saldanha da Fonseca

Aos meus pais e à minha noiva

"Se a nossa opção é progressista, se estamos a favor da vida e não da morte, da eqüidade e não da injustiça, do direito e não do arbítrio, da convivência com o diferente e não de sua negação, não temos outro caminho senão viver plenamente a nossa opção. Encarná-la, diminuindo assim a distância entre o que dizemos e o que fazemos."

Paulo Freire

Agradecimentos

Ao meu orientador, pelas palavras de incentivo nos momentos de indefinição da bolsa e por sua paciência nos meus momentos de imaturidade. Agradeço especialmente a oportunidade de sua orientação neste trabalho.

Ao Dr. M. T. Shing, por sua atenção e esclarecimentos prestados sobre os conceitos contidos em artigos de sua autoria que foram utilizados como referências neste trabalho.

Aos professores da Universidade Estadual do Ceará, especialmente Everardo Maia, Fernando de Carvalho, Marcelo Aragão, Marcos Negreiros e Roberta Pontes e ao professor Mauro Oliveira do CEFET-Ce, pelos conhecimentos transmitidos e pelo despertar para a investigação científica.

À minha família, especialmente meus pais (José Luiz Pinto Façanha e Lucília de Almeida Façanha), irmã (Suely de Almeida Façanha) e tio (Armando Alves de Almeida), por toda a confiança e apoio incondicionais, que em muito me ajudaram a concluir este trabalho.

À minha noiva (Karol Silva de Moura), por toda a compreensão, paciência e amor nestes quase três anos furtados de nosso convívio e também pelas palavras confortadoras nos momentos de maior dificuldade. Você foi fonte de estímulo e força durante todo este tempo.

Aos amigos, pelos momentos agradáveis de convivência dentro e fora da Unicamp, em especial, aos conterrâneos Cláudio N. de Menezes e Gutemberg B. G. Filho e ao Prof. Dr. Pedro J. de Rezende, pela ajuda e discussões sobre alguns problemas deste trabalho.

À amiga Jusmeri e sua família, por sua acolhida em seu lar, me proporcionando um convívio agradável no período final de minha jornada.

Ao CNPq e FAPESP, através do processo Nº 96/09739-1, pelo suporte financeiro deste trabalho, sem o qual sua realização não teria sido possível.

A Deus, que tornou possíveis todos os agradecimentos anteriores.

Resumo

Vídeo sob Demanda é a aplicação que permite ao usuário selecionar um vídeo dentre uma grande coleção, transmitindo-o até sua casa através de uma rede de telecomunicações. Várias técnicas foram propostas de modo a reduzir a grande demanda de banda passante requerida para a transmissão de vídeo digital. Dentre elas destacam-se as técnicas de *Batching* e *Piggybacking*, que visam reduzir a demanda de banda passante através do compartilhamento de fluxos de vídeo.

Esta dissertação endereça o problema do gerenciamento de banda passante em servidores de Vídeo sob Demanda. Inicialmente, apresenta-se um estudo enfocando a técnica de *Piggybacking*, no qual são introduzidas duas novas políticas. A primeira, política S^2, é uma generalização da política Algoritmo *Snapshot*, pois realiza a mesclagem de fluxos resultantes dos intervalos *Snapshot*. A segunda política, denominada Híbrida, realiza otimizações no interior dos intervalos *Snapshot*. Analisa-se também o algoritmo de programação dinâmica, utilizado para a construção de árvores de mesclagem pela política *Snapshot*, com o objetivo de se reduzir a sua complexidade. Em seguida, desenvolve-se um estudo sobre a técnica de *Batching*, no qual se propõe uma nova política que visa maximizar o número de usuários suportados por um servidor de vídeo. Finalmente, realiza-se um estudo que integra ambas as técnicas. Apresentam-se alguns comentários sobre os fatores que influenciam a abordagem integrada de *Batching* e *Piggybacking* e o esquema utilizado.

Abstract

Video on Demand (VoD) enables users to select and watch movies stored in a remote video server. Several techniques have been proposed in order to reduce the huge bandwidth demand required by VoD. Amongst them, we mention Batching and Piggybacking. Such techniques reduce the bandwidth demand by sharing video streams.

In this work we focuse on the problem of bandwidth management in Video on Demand storage servers. Firstly, we introduce two new piggybacking policies. The first policy, S^2, is a generalization of the Snapshot Algorithm policy. The second policy carries out optimizations in the Snapshot intervals. We also analyze algorithms to reduce the complexity of building video merging trees. Moreover, a novel batching policy is introduced. This policy aims at maximizing the number of users supported by a video server. Finally, we study the integration of Batching and Piggybacking.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Multimídia refere-se à combinação de várias formas de representação de informação (áudio, gráficos, texto e vídeo) e à apresentação síncrona, manipulação e interação destas mídias. Se por um lado a rápida expansão do paradigma multimídia deve-se à capacidade crescente dos computadores pessoais (*desktops*); por outro, o oferecimento de serviços multimídia remotos em larga escala dependerá do efetivo dimensionamento das futuras redes de comunicação.

Uma das áreas mais promissoras para o oferecimento de serviços multimídia em redes é a *Loja de Locação de Vídeo Eletrônico* (*Eletronic Video Rental Store*) que oferece o serviço de Vídeo sob Demanda (*Video on Demand* — VoD). Através deste serviço, um usuário pode escolher um vídeo dentre uma grande coleção e ter o vídeo enviado através de uma rede de telecomunicações até sua casa.

Um sistema de VoD pode ser implementado de modo a oferecer uma grande diversidade de serviços. Em um extremo há o serviço mais sofisticado, no qual o vídeo é iniciado instantaneamente e o usuário pode executar operações de VCR. No outro extremo, encontra-se o serviço econômico, no qual a exibição do vídeo não é instantânea, mas ao contrário, o usuário deve esperar o início programado da apresentação. Neste caso o usuário não dispõe de operações VCR [1, 2].

A transmissão de vídeo requer grande demanda de banda passante, como pode ser visto na Tabela 1.1 [3]. Estes altos requerimentos restringem o oferecimento de serviços de vídeo a uma grande população de usuários devido aos recursos finitos alocáveis para transmissão na subrede de comunicação, bem como a capacidade finita de entrada/saída dos servidores de vídeo (canais lógicos) e de processamento. Assim sendo, diversas técnicas vêm sendo propostas a fim de reduzir a demanda de banda passante. Dentre elas pode-se citar Replicação e *Caching*, *Bridging*, *Batching* e *Piggybacking*.

As técnicas de replicação e *caching* visam reduzir a demanda de banda passante na rede de comunicação através da multiplicidade de cópias de objetos de vídeo, isto é, através

da distribuição de várias cópias de um mesmo objeto em diversos servidores. *Bridging* propõe a utilização de *buffers* de memória como mecanismo de redução da demanda de banda passante em servidores. Finalmente, mencionam-se as técnicas de *Batching* e *Piggybacking*, as quais utilizam o compartilhamento de fluxos de vídeo como mecanismo para a redução da demanda de banda passante em servidores de vídeo. Em outras palavras, um servidor de VoD pode atender a um conjunto de usuários que deseja assistir a um determinado vídeo utilizando uma quantidade de recursos inferior à requerida em caso de alocação de um fluxo para cada usuário. Este compartilhamento ocorre de forma transparente sem que a exibição a um usuário em particular seja prejudicada.

Finalmente, deve-se mencionar que existe uma diferença entre Vídeo sob Demanda e Filme sob Demanda (*Movie on Demand* — MoD). Enquanto VoD possibilita a utilização de serviços como ensino à distância, jogos, *home shopping*, dentre outros; MoD constitui-se numa categoria de VoD em que os objetos de vídeo acessados são exclusivamente filmes [4].

Codificação	Faixa Aproximada da Taxa de Transmissão
CCIR[a] (agora ITU-R) 601/D-1[b]	140–270 Mbps
H.261[c]	64 kbps–2 Mbps
Motion JPEG[d]	10–20 Mbps
MPEG-1	1.2–2.0 Mbps
MPEG-2	3–10 Mbps (TV regular), 15–20 Mbps (HDTV[e])
Compressão de *software* (*Small Windows*)	45 Mbps

[a]CCIR - *Consultative Committee on International Radio.*
[b]Padrão para digitalização de vídeo sem compressão.
[c]Codificação para vídeo-conferência e vídeo-telefonia.
[d]Imagens médicas.
[e]*High Definition TV.*

Tabela 1.1: Faixas aproximadas de taxas de vários padrões de codificação de vídeo.

1.1 Objetivos da Dissertação

O objetivo desta dissertação é avaliar aspectos de desempenho em servidores de sistemas de VoD, nos quais técnicas de compartilhamento de fluxo são empregadas como mecanismo para a redução da demanda de banda passante. Mais especificamente, avaliam-se:

- Requisitos de complexidade do algoritmo utilizado pela política de *Piggybacking*, Algoritmo *Snapshot*, para a construção da árvore ótima de mesclagens;
- O impacto da utilização de novas políticas de *Piggybacking*;
- O impacto da utilização de uma nova política de *Batching*; e
- A implementação integrada de ambas as técnicas.

1.2 Organização da Dissertação

Esta dissertação está organizada como segue. Aspectos conceituais de sistemas de VoD e trabalhos relacionados são apresentados no capítulo 2. Nos capítulos 3 e 4, apresentam-se estudos direcionados às técnicas de *Piggybacking* e *Batching*, respectivamente. A integração destas técnicas é apresentada no capítulo 5. No capítulo 6 nossas conclusões e considerações finais são apresentadas. O apêndice A apresenta a demonstração de complexidade da heurística proposta para o problema da construção da árvore de mesclagem de fluxos. O apêndice B contém o código da heurística implementado em linguagem C e o apêndice C contém a lista de acrônimos. Finalmente, o apêndice D apresenta as tabelas com os símbolos utilizados nesta dissertação.

Capítulo 2

Conceitos de Sistemas de Vídeo sob Demanda

Televisão é o meio de comunicação mais amplamente difundido e seu serviço é bastante simples: escolha de um programa dentre um conjunto ofertado sem controle do usuário. Sistemas de Vídeo sob Demanda diferem de um sistema convencional de TV, pois permitem ao usuário selecionar um programa dentre uma grande coleção e ter controle sobre o mesmo através de operações de VCR. Estes sistemas tornam-se disponíveis comercialmente e competitivos com a TV à medida em que se obtêm avanços nas diversas tecnologias envolvidas (dispositivos de armazenamento, técnicas de compressão e estruturas de suporte à transmissão digital).

Este capítulo aborda resumidamente diversos aspectos de um sistema de VoD e apresenta, de forma integrada, trabalhos relacionados. Inicialmente, apresentam-se as definições de alguns termos utilizados ao longo deste capítulo (seção 2.1). Na seção 2.2 são apresentados alguns componentes da arquitetura de um sistema de VoD. Aspectos relacionados aos servidores de vídeo são abordados numa seção específica (seção 2.3), em virtude de sua importância para um sistema de VoD e das diversas questões que envolvem seu projeto. Em seguida, comentários são feitos sobre esquemas que viabilizam a interatividade entre usuário e sistema, através de operações de VCR (seção 2.4). Finalmente, são relacionadas as diversas técnicas de redução da demanda de banda passante (seção 2.5).

2.1 Definições Premilinares

Esta seção tem por objetivo apresentar diversos conceitos utilizados durante a introdução do conteúdo deste capítulo. Os conceitos relacionados foram extraídos das referências [3, 4, 5, 6].

2.1.1 ATM (*Asynchronous Transfer Mode*)

ATM ou *Asynchronous Transfer Mode* é o padrão para as Redes Digitais de Serviços Integrados de Faixa Larga (RDSI-FL). O ATM é uma tecnologia de multiplexação e comutação de grande largura de banda e baixo retardo que vem tornando-se disponível para redes públicas e privadas. Os princípios do ATM e as plataformas fundamentadas nesta tecnologia formam a base para o oferecimento de uma variedade de serviços de comunicação multimídia de alta velocidade.

As unidades básicas de transferência de dados são chamadas de *células* e possuem o tamanho de 53 bytes ou octetos, de modo que os 5 primeiros bytes formam o cabeçalho (que contém informações sobre nível de prioridade, comutação, etc), e os demais 48 bytes compõem a parte de dados ou *payload*. As células são transmitidas através de conexões com circuitos virtuais. O ATM busca maximizar a utilização da rede através da multiplexação estatística.

As redes ATM dispõem de mecanismos de controle de admissão de conexões (*Connection Admission Control* — CAC), responsáveis pela determinação da aceitação ou recusa de uma nova conexão mediante os parâmetros de "contrato" declarados pela fonte e dos recursos disponíveis, e de mecanismos para o controle do tráfego das conexões estabelecidas (mecanismos de policiamento).

2.1.2 Redes SONET

SONET ou *Synchronous Optical Networks* é o padrão que especifica taxas de relógio e níveis de banda passante, sinais óticos, características de fibra (ótica) e procedimentos operacionais, incluindo mecanismos de auto correção (desejáveis quando anéis de fibra são utilizados para a transmissão síncrona de dados), permitindo a interconexão de equipamentos heterogêneos. O bloco básico de transmissão é o *quadro SONET* que possui 810 bytes e é transmitido a cada 125 μs gerando um sinal de 51,840 Mbps conhecido como *Sinal de Transporte Síncrono de Nível 1* — STS-1.

2.1.3 Codificação MPEG

MPEG ou *Motion Picture Expert Group* é uma família de padrões internacionais definido para a compressão e transmissão de áudio e imagens em movimento. Um fluxo de vídeo consiste de uma seqüência de imagens estáticas, as quais possuem altos níveis de correlação. O padrão MPEG explora esta característica e, através das codificações *intraframe*, que elimina a redundância espacial, e *interframe*, a qual elimina ambas as redundâncias (espacial e temporal), alcança índices altos de compressão, gerando-se diferentes tipos de quadros, que são citados a seguir:

Intracoded Frames — *I*: Sofrem apenas codificação *intraframe*. Por serem auto-contidos funcionam como quadros de referência e apresentam os menores índices de compressão.

Predictive Frames — *P*: São reconstruídos a partir de quadros I ou P anteriores, e;

Bidirectional Predictive Frames — *B*: Constituem-se em interpolações dos quadros I ou P adjacentes, por isso, normalmente apresentam os maiores índices de compressão.

Os quadros I, P e B podem ser organizados de diversas formas, compondo o conceito de grupo de imagens (*Group of Pictures* — GOP), como ilustrado na Figura 2.1.

Existem diversas variantes do padrão. MPEG-1 foi projetado para se obter uma taxa de transmissão de até 1,5 Mbps, sendo adequado para uma boa qualidade visual (resolução média). MPEG-2 estabelece uma codificação com melhor qualidade de vídeo, cujas taxas alcançam até 20 Mbps, no caso de suporte à HDTV. O MPEG-4 foi projetado com o objetivo de permitir taxas de transmissão extremamente baixas, as quais variam de 4,8 Kbps a 64 Kbps, porém com uma qualidade visual razoável.

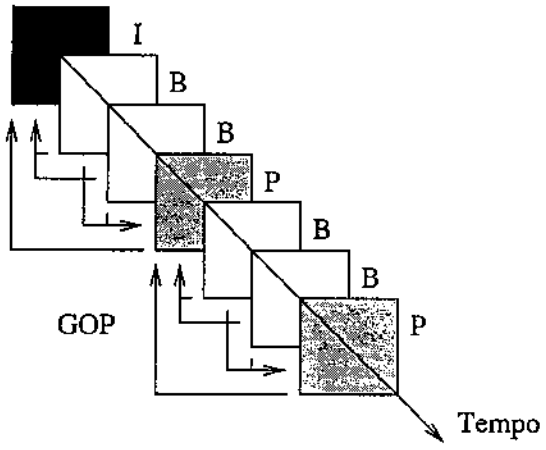


Figura 2.1: Ilustração da organização de quadros I, B e P em um GOP.

2.2 Arquitetura de Vídeo sob Demanda

Nesta seção descreve-se uma arquitetura de VoD [3, 7, 8] (Figura 2.2) que provê serviço interativo fim-a-fim utilizando tecnologia ATM ou SONET (definidas nas seções 2.1.1 e 2.1.2). O sistema inclui servidor(es) de vídeo, uma rede de distribuição com funções de gerenciamento e subredes locais de acesso ao *backbone*, além do equipamento utilizado pelo usuário (*set top box* e controle remoto).

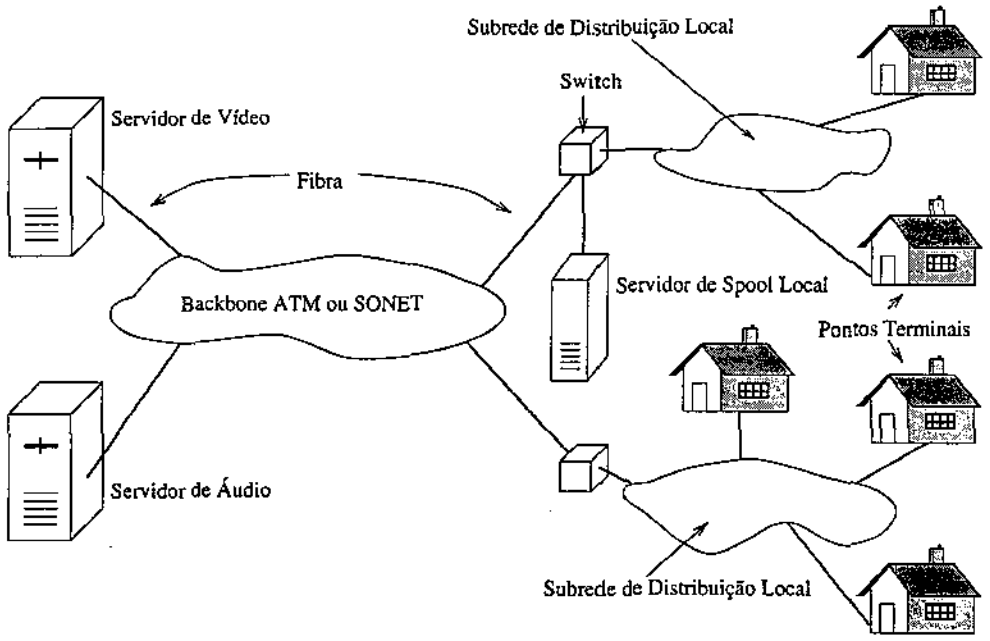


Figura 2.2: Elementos da arquitetura de um sistema de VoD.

2.2.1 Rede de Distribuição

A rede de distribuição é o conjunto de *switches* e cabeamento existentes entre o servidor de vídeo e o equipamento do usuário, consistindo de um *backbone* ATM ou SONET (ou ATM sobre SONET) interligado a uma rede de distribuição local. É necessário que o *backbone* possua grande largura de banda e suporte à transmissão de fluxos de dados isócronos. As redes SONET, em virtude da transmissão síncrona, garantem a inexistência da oscilação dos tempos de transmissão de células (*jitter*), porém nas redes ATM garantias de Qualidade de Serviço (*Quality of Service* — QoS) são obtidas através de um eficiente controle de tráfego.

Quanto às subredes locais de acesso, compreendem várias estruturas a partir das quais interliga-se o *backbone* ao equipamento do usuário. As soluções existentes mais comuns são:

ADSL (*Asymmetric Digital Subscriber Line*): Propõe-se a utilização de par trançado para a transmissão de vídeo, de tal forma que se possa utilizar o cabeamento de telefonia já instalado. No entanto, esta estrutura não suporta a transmissão de fluxos de vídeo MPEG devido ao seu comprimento típico de 10Km. A solução ADSL é eliminar eco e outros ruídos através da tecnologia de processamento eletrônico de sinais. ADSL-1, padronizado pelo ANSI (*American National Standards Institute*), oferece um canal para

transmissão de dados a 1,536 Mbps, um canal para sinais de controle com taxa de 16kbps, além de um canal analógico para telefonia que opera com freqüência de 4kHz. ADSL-2 e ADSL-3 encontram-se em processo de padronização.

FTTC (*Fiber To The Curb*): Utiliza-se uma estrutura híbrida composta de fibra ótica e par trançado. O segmento de fibra ótica interliga o *backbone* até um dispositivo nas proximidades (*Optical Network Unit* — ONU) das instalações dos usuários, deste ponto em diante, utilizam-se conexões ponto a ponto com o dispositivo do usuário através de par trançado.

FTTH (*Fiber To The Home*): Neste esquema utiliza-se fibra ótica desde o servidor de vídeo até os pontos terminais. Esta alternativa aparece como a solução definitiva pois oferece largura de banda suficiente para a transmissão de dados e sinais de controle, além de ser um meio com taxa de erros extremamente reduzida. Porém, é bastante caro e sua implantação não deve ocorrer a curto prazo.

HFC (*Hybrid Fiber/Coax*): Diferentemente das alternativas anteriores, que empregam abordagem ponto a ponto, esta solução propõe a utilização de um cabo coaxial compartilhado que interliga diversos usuários a um segmento de fibra ótica ligado ao *backbone*. Transmissões podem ocorrer na forma analógica ou digital, ou seja, possui suporte para a transmissão de TV a cabo e VoD. Entretanto, em virtude do acesso compartilhado ao meio de transmissão, esquemas de acesso múltiplo como TDMA (*Time Division Multiple Access*) ou FDMA (*Frequency Division Multiple Access*) precisam ser utilizados para os sinais de controle.

2.2.2 Equipamento do Usuário

O equipamento utilizado por um usuário final inclui a *set top box*, conectada a um aparelho de TV ou vídeo cassete, e um controle remoto. A *set top box* permite aos usuários selecionar o serviço ou vídeo de sua preferência, possui uma unidade de descompressão para decodificar a seleção realizada, geralmente codificada em MPEG-1 ou MPEG-2, e enviá-la ao dispositivo de exibição, geralmente a TV.

2.3 Aspectos de Servidores de Vídeo

Num sistema de VoD, os vídeos (que compõem uma biblioteca previamente digitalizada) devem ser selecionados em tempo real e controlados como num aparelho de vídeo

através de operações de VCR. Portanto, é necessário que os servidores de vídeo possuam alta capacidade de armazenamento e taxa de transferência de dados.

O subsistema de armazenamento do servidor precisa suportar um grande número de usuários simultaneamente, assim como o controle de *hardware* e *software* para recepção de requisições, localização de vídeos, transferência de dados entre dispositivos na hierarquia de armazenamento, contabilização de custos, controle de admissão, gerenciamento de operações de VCR e criptografia (necessária para evitar o acesso a informações sigilosas, como senha no processo de identificação do usuário, e garantir que apenas usuários pagantes assistam vídeos quando utiliza-se meio de transmissão com acesso compartilhado).

O servidor de vídeo precisa manipular múltiplos fluxos de saída sem variação[1] no tempo necessário para a transferência de bits, quadros ou células de informação. Para manter a vazão requerida faz-se necessária a utilização de (*i*) um processador poderoso, (*ii*) um barramento de alta velocidade e (*iii*) canais e dispositivos de entrada/saída de alta capacidade. Contudo, ainda podem ocorrer flutuações nos fluxos de informação. Para minimizar tais flutuações, os servidores normalmente utilizam grandes *buffers* de saída. Com isso, os dados são acessados rapidamente em grandes blocos dos discos rígidos, armazenados temporariamente no *buffer* e enviados numa taxa pré-definida.

Nas próximas seções, diversos componentes de um servidor de vídeo são apresentados, com especial ênfase na hierarquia de armazenamento.

2.3.1 Componentes de um Servidor

Um servidor de vídeo compreende:

Hierarquia de Armazenamento: Os recursos de armazenamento são vistos como uma hierarquia (composta por armazenamento terciário, secundário e primário, além de memória RAM e *cache*), geralmente representada graficamente por um triângulo. Neste modelo, dispositivos com baixo tempo de acesso são posicionados no topo, enquanto que dispositivos de desempenho e custos inferiores e maior capacidade de armazenamento localizam-se nos níveis mais próximos à base. A Figura 2.3 ilustra um esquema de gerenciamento de memória compatível com os dispositivos atualmente existentes, em que o nível de *cache* de arquivos ocorre no equipamento do usuário; o *cache* de servidor é realizado através da distribuição de diversas cópias de objetos de vídeo em servidores de pequeno porte na rede; enquanto que os demais níveis são implementados num servidor de grande porte. Na ilustração, discos rígidos são substituídos por RAIDs (*Redundant Arrays of Inexpensive Disks*) e *jukeboxes* óticos compõem o nível intermediário entre a

[1]Na realidade, uma pequena variação é tolerável e sua amplitude está limitada pelo *buffer* existente na *set top box*.

memória terciária e primária.

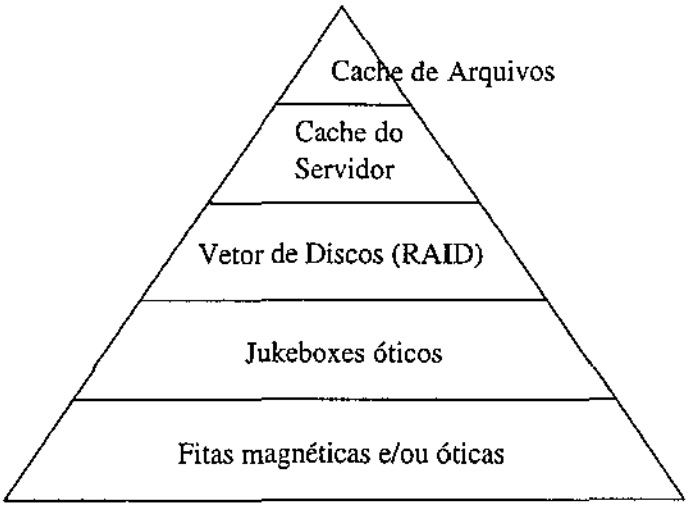


Figura 2.3: Esquema de gerenciamento de memória de uma arquitetura de vídeo sob demanda.

Controladores de Fluxos: Até o advento do ATM, mais precisamente do protocolo AAL5, adaptado à transmissão de vídeo, o servidor de vídeo precisava produzir um fluxo de bits contínuo isócrono para o usuário. Nos sistemas antecessores ao ATM, o controlador de fluxos atua como o gerente de tráfego para o servidor. Este dispositivo precisa suportar altas taxas de transações[2].

Barramento: A maioria dos processadores utiliza arquiteturas baseadas em barramento. Dado que a informação relacionada a muitos vídeos migra do subsistema de discos para os módulos de comunicação, e que o vídeo produz um fluxo isócrono com taxa média previsível, a capacidade de transferência do barramento é um fator crítico de desempenho. Portanto, uma vez que o tráfego demandado para os usuários precisa passar pelo barramento, a inexistência de um projeto detalhado pode transformá-lo no ponto crítico de desempenho do sistema.

Processador: O processador principal controla o fluxo de dados entre os vários componentes do servidor, as funções de gerenciamento e de contabilização de custos. Novas arquiteturas de computadores podem ser requeridas para suportar o número alto de usuários

[2]Operações de busca e transferência de dados (entre os diversos níveis de armazenamento) e transmissão dos mesmos.

simultâneos em uma área metropolitana ampla. Alguns sistemas utilizam *mainframes* ou estações RISC (*Reduced Instruction Set Computer*), porém computadores massivamente paralelos são apontados como a plataforma ideal.

Condicionador de Fluxos de Saída: Adapta o servidor para o sistema de transmissão, utilizando *buffers* para a determinação do *clock* da taxa de transmissão constante adequada. Com a utilização do protocolo AAL5, o condicionador precisará adotar um comportamento mais estatístico.

2.3.2 Hierarquia de Armazenamento

Os diversos níveis que compõem a hierarquia de armazenamento em um servidor de vídeo são citados a seguir.

Armazenamento Terciário: O armazemamento terciário geralmente consiste de aparelhos de vídeo cassete, fitas magnéticas e discos óticos organizados na forma de *jukeboxes*. Também chamado de armazenamento *off-line*. Estes dispositivos possuem como características principais:

- baixo custo;
- grande capacidade de armazemamento;
- baixo desempenho (tempo de acesso elevado); e
- baixa taxa de transferência.

Armazenamento Secundário: O armazenamento secundário tipicamente consiste de tecnologia de armazenamento ótico de alta velocidade ou sistemas RAID. Sistemas mais sofisticados dispensam o nível de armazenamento terciário e armazenam todas as informações neste nível. Outros sistemas preferem utilizar os três elementos da hierarquia de armazenamento. Também chamado de armazenamento *near-line*.

Armazenamento Primário: Constitui-se de discos rígidos convencionais organizados ou não na forma de RAID. Também chamado de armazenamento *on-line*.

Memória *RAM* e *cache*: Além da utilização convencional, num servidor de vídeo a memória RAM é alocada para fluxos individuais ou difusão ampla (*broadcast*), também chamada *movieRAM*. Esta memória é utilizada pelo controlador de fluxos para gerenciar mais eficientemente a informação associada aos segmentos de vídeos mais requisitados.

Os próximos items abordam mecanismos específicos[3] que podem ser utilizados numa hierarquia de memória.

***Jukeboxes* Óticos:** Muitos CD-ROMs podem ser agrupados em *jukeboxes*. Estes dispositivos são construídos posicionando-se os discos em estantes acessadas por dispositivos robóticos. Quando um arquivo precisa ser acessado, o sistema de gerenciamento de arquivos identifica onde encontrá-lo e o braço mecânico substitui o disco atual pelo que contém o arquivo a ser acessado.

Memórias Holográficas: Dispositivos magnéticos ou óticos são inerentemente limitados a vetores unidimensionais de bits (organizados na forma de espirais num disco). O método de armazenamento holográfico constitui-se num esquema muito eficiente de armazenamento em três dimensões utilizando a "profundidade" do dispositivo (mídia). Esta técnica também provê robustez e insensibilidade a erros, uma vez que setores defeituosos apenas reduzem a razão sinal/ruído no processo de leitura, ao invés de danificá-lo. No esquema de armazenamento tridimensional, grande largura de banda pode ser obtida através da manipulação dos dados como vetores de bits.

O meio de armazenamento pode ser um cristal fotorrefratário (para múltiplas operações de escrita) ou fotopolímeros (para escrita única). A informação é armazenada como um holograma, o qual é reconstruído para leitura. Para seu armazenamento, realiza-se a transformada de Fourier dos vetores bidimensionais de pontos luminosos. Utiliza-se este mecanismo com o objetivo de reduzir o montante de informação gravada e prover imunidade a defeitos físicos.

A memória holográfica é um dispositivo orientado a páginas, as quais são vetores bidimensionais de pontos luminosos. Múltiplas páginas são multiplexadas holograficamente de modo a criar uma pilha de páginas, todas armazenadas no espaço normalmente utilizado para armazenar uma única imagem bidimensional. Com este objetivo, altera-se o ângulo do feixe de luz de referência pela fração de um grau ou utilizam-se soluções proprietárias.

Outros aspectos do projeto de servidores que ainda podem ser mencionados relacionam-se com (*i*) o projeto de sistemas de arquivos multimídia e organização de dados em disco [9, 10, 11], (*ii*) balanceamento de carga em discos através da replicação de

[3] *Jukeboxes* e memórias holográficas receberam destaque por serem, respectivamente, um dispositivo menos comum que os demais e um tipo de memória ainda em estágio de pesquisa.

Operação	Descrição
Play/Resume	Exibe o vídeo a partir do início ou de um ponto intermediário.
Stop	Interrompe a apresentação. Sem exibição de imagem e som.
Pause	Interrupção temporária. Com exibição de imagem.
Jump Forward	Salta para um ponto posterior. Sem exibição de imagem e som.
Jump Backward	Salta para um ponto anterior. Sem exibição de imagem e som.
Speed Up	Exibição acelerada com imagem e som (*fast-forward*).
Slow Down	Apresentação lenta com imagem e som.
Reverse	Exibição invertida com imagem e som.
Fast Reverse	Apresentação invertida acelerada com imagem e som.
Slow Reverse	Exibição invertida lenta com imagem e som.

Tabela 2.1: Operações de VCR.

segmentos de arquivos [12], (*iii*) gerenciamento de banda passante disponível em discos com o objetivo de aumentar a vazão (*throughput*) [13], (*iv*) políticas de escalonamento de discos, gerenciamento de *buffer* e questões de barramento de entrada/saída [14, 15], (*v*) além de outras questões sobre arquitetura [16, 17].

2.4 Interatividade em Sistemas de Vídeo sob Demanda

A interação entre usuário e sistema abordada nesta seção diz respeito às operações disponíveis a um usuário para o controle de uma sessão de VoD iniciada, ou seja, as operações de VCR. O advento do vídeo digital tornou possível a existência de novos paradigmas, isto é, novos tipos de operações tornam-se disponíveis. A Tabela 2.1 contém uma relação de operações de VCR [18]. Almeroth [19] divide as operações de VCR em duas categorias:

- Operações *Contínuas*: Um usuário possui controle completo sobre a duração de uma operação de VCR, e;
- Operações *Descontínuas*: A duração das operações é discretizada em múltiplos inteiros de um determinado intervalo de tempo.

São diversos os esquemas para o suporte completo ou parcial a operações de VCR propostos na literatura. Philip Yu *et al.* [20, 21] apresentam um mecanismo integrado com *Batching* em que usuários que realizam operações de *pause/resume* possuem garantias determinísticas, através do *caching* de fluxos, da retomada da apresentação; uma

flexibilização para oferecer garantias probabilísticas também é considerada. Towsley *et al.* [22] apresentam uma estratégia de *Batching* com oferecimento de operações de VCR com reserva de fluxos para ambos os conjuntos de vídeos populares e não populares além de um contingente extra, utilizado para as retomadas de exibição após operações de VCR. Neste mesmo trabalho, apresenta-se uma solução analítica para a determinação da janela de *Batching* para os vídeos populares. Em [23], Towsley *et al.* apresentam um esquema em que parte da banda passante do sistema é reservada para operações de VCR, sendo compartilhada pelos usuários que realizam tais operações. Duas abordagens são consideradas. Na primeira, *Delay Scheme*, há limitação do número de usuários que compartilham a porção de banda passante utilizada para operações de VCR; enquanto que na segunda abordagem, *Loss Scheme*, não havendo banda passante disponível para uma nova requisição, a banda passante de cada fluxo é reduzida para acomodar o novo usuário. Li *et al.* [24] introduzem um esquema em que usuários que compartilham um fluxo de vídeo podem "destacar-se" deste fluxo em decorrência de uma operação de VCR. Em seguida, busca-se a ressincronização de tais usuários através da utilização de *buffers* circulares, chamados de *buffers de sincronização.*

Nussbaummer *et al.* [25, 26] apresentam uma classificação contendo quatro modalidades para o oferecimento de serviços de VoD em que cada uma delas possui diferentes compromissos entre custo e o nível de interatividade oferecida ao usuário, citadas a seguir:

TVOD (*True Video on Demand*): Nesta modalidade é permitido ao usuário executar quaisquer operações de VCR. No esquema TVOD, para cada usuário que requisita acesso a um vídeo é alocado um canal lógico. Para permitir interação total, nenhum outro usuário pode acessar este canal simultaneamente. A ocorrência de uma requisição sem a disponibilidade de canais lógicos implica em descarte sumário do pedido. Este fato é chamado de *blocking.*

Uma variação deste esquema é a criação de uma fila com as requisições pendentes. Para isso, o usuário deve ser notificado do tempo de espera estimado, sendo que sua inclusão na fila só ocorre se o tempo máximo de espera for menor que o comunicado.

NVOD (*Near Video on Demand*): O esquema NVOD foi projetado de modo a oferecer um serviço de menor custo que TVOD. Seu princípio básico consiste em iniciar um fluxo de cada vídeo a cada Δt minutos em canais distintos (cada fluxo está Δt minutos atrasado/adiantado em relação ao anterior/posterior), de modo que um usuário que faz um pedido precisa que esperar no máximo Δt minutos para ser atendido. Suas vantagens são simplicidade e redução de custos; porém, não dispõe de interatividade, com exceção da possibilidade de "saltar" para fluxos anteriores/posteriores.

PVOD (*Partitioned Video on Demand*): Este esquema constitui-se da utilização conjunta dos anteriores combinando suas vantagens. Parte dos canais é utilizada com o esquema NVOD para exibição dos vídeos populares, os demais canais são utilizados para exibição dos vídeos não populares através do esquema TVOD. Os usuários que requisitam vídeos exibidos pela partição NVOD são atendidos por ela, enquanto que os demais, pela partição TVOD.

DAVOD (*Dynamically Allocated Video on Demand*): Possui comportamento similar ao PVOD. Contudo, esta modalidade permite o deslocamento de usuários da partição NVOD para TVOD por ocasião da solicitação de operações de VCR.

2.5 Técnicas de Redução da Demanda de Banda Passante

O oferecimento de serviços de VoD em larga escala depende em grande parte do controle de admissão e gerenciamento dos recursos alocáveis; porém, em virtude da grande demanda de banda passante requerida, a utilização de outras estratégias torna-se necessária. Nos Estados Unidos (continental) estima-se que o número de aparelhos de TV ligados simultaneamente em horários de pico totaliza 77.000.000. Num sistema de VoD com alocação de fluxos sob demanda, o montante de banda passante requerido totalizaria 462 Tbps para a codificação MPEG-2 NTSC (6Mbps/fluxo), 770 Tbps utilizando-se MJPEG NTSC (10Mbps/fluxo) e 1,54 Pbps para MPEG-2 no formato de HDTV (20Mbps/fluxo) [27]. Estes altos requerimentos evidenciam a necessidade da utilização de técnicas de redução da demanda de banda passante isoladamente ou de forma conjunta. As próximas seções discorrem sobre estas técnicas, com ênfase nas técnicas que propõem o compartilhamento de fluxo.

2.5.1 Replicação e *Caching*

A técnica de replicação [27, 28, 29] propõe a distribuição de diversas cópias de um mesmo objeto de vídeo em vários servidores, objetivando-se reduzir o custo de transmissão (volume de dados) em segmentos da rede. Por um lado, o custo de transmissão está diretamente relacionado à topologia de rede empregada; por outro, o grau de replicação é influenciado pelo custo de armazenamento das diversas cópias. Deste modo, a técnica de replicação objetiva a minimização do custo de transmissão na rede sem, contudo, aumentar demasiadamente o custo de armazenamento.

A técnica de *caching* [30, 31, 32] consiste na implementação da replicação, porém de forma dinâmica. Para isso, faz-se necessário o monitoramento constante dos padrões

de requisições dos usuários para determinar quais vídeos serão replicados e em quais servidores da rede. Neste contexto, pode-se explorar o escalonamento prévio de filmes para distribuir suas cópias em períodos de tráfego reduzido.

2.5.2 *Bridging*

A técnica de *Bridging* [20, 21, 33, 34] propõe o armazenado dos últimos k minutos de um vídeo em um *buffer* de memória, de tal forma que se possa atender requisições (por novos fluxos ou de retomada de exibição após operações de VCR) que cheguem ao sistema num intervalo com duração máxima de k minutos após o quadro de filme atualmente exibido. O atendimento de tais requisições pelos dados armazenados no *buffer* reduz, desta forma, a demanda de banda passante nos dispositivos de armazenamento dos servidores de vídeo.

2.5.3 Compartilhamento de Fluxo

Uma abordagem diferente das anteriores é proposta pelas técnicas que propõem o compartilhamento de fluxo como mecanismo para a redução da demanda de banda passante. Com vistas a este objetivo (compartilhamento de fluxo), tais técnicas levam em consideração a probabilidade de um conjunto de requisições pelos vídeos populares chegar ao sistema dentro de um intervalo de tempo relativamente curto, de forma a atender às requisições com um único fluxo. Neste grupo encontram-se as técnicas de *Batching* e *Piggybacking*, detalhadas a seguir.

Batching

A técnica de *Batching* [20, 21, 22, 35, 36, 37, 38, 39, 40] consiste em reter requisições por um determinado intervalo de tempo com o objetivo de agrupar o máximo número de requisições e servir ao grupo com um único fluxo de vídeo. A este intervalo dá-se o nome de **Janela** ou **Intervalo de** *Batching*[4] e seu dimensionamento representa um compromisso entre a minimização do tempo esperado pelo usuário e a maximização do **Efeito de** *Batching*, ou seja, o número médio de usuários servidos por fluxo.

A seguir estão relacionadas as políticas de *Batching* propostas na literatura.

Espera Forçada: Exige que pelo menos uma das requisições seja retida durante a janela de *Batching*. Ao final da mesma, aloca-se um fluxo para o vídeo solicitado. Este

[4]Não há uma nomenclatura padronizada para se referir ao tempo de espera a que um usuário é submetido enquanto o sistema aguarda a chegada de novas requisições do mesmo vídeo. Nesta dissertação, ambos os termos são utilizados indistintamente.

funcionamento pode causar, nos horários de pico, o fenômeno da formação de ciclos, que consistem em momentos de intensa alocação de canais seguidos por longos intervalos de espera, isto é, índices altos de rejeição de pedidos. Os ciclos decorrem da alocação independente de canais para vídeos distintos, uma vez que o intervalo de espera é aplicado em cada fila independentemente das demais.

Controle de Taxa Puro: Política que (*i*) impõe um limite superior na taxa a qual canais podem ser alocados, e (*ii*) limita o número total de canais que podem ser alocados durante um intervalo de tempo fixo (*intervalo de medida*) com o objetivo de evitar a formação de ciclos (como descrito anteriormente).

Controle de Taxa Desviado: Difere da política anterior, pois permite que a taxa de alocação, durante os horários de pico, esteja momentaneamente acima da taxa de alocação máxima com o objetivo de evitar o abandono de usuários (sem serviço) caso um canal não seja alocado em um curto intervalo de tempo.

Maior Tamanho de Fila (*Maximum Queue Length*) — MQL: Busca-se maximizar o efeito de *Batching* alocando-se o fluxo para a fila cujo número de requisições pendentes seja maior.

Fila de Maior Índice (*Maximum Factored Queue Length*) — MFQL: Define-se um índice que determina qual das filas de requisições receberá o fluxo a ser alocado, este índice representa um compromisso entre a maximização do efeito de *Batching* e o tempo de espera na fila. Busca-se, desta forma, maximizar a utilização dos recursos do sistema sem desprezar os vídeos menos populares.

Primeiro a Chegar, Primeiro a ser Atendido (*First Come First Served*) — FCFS: Nesta política, considera-se principalmente o aspecto de justiça entre os vídeos (populares e não populares) quando da alocação de fluxos. Assim sendo, todas as requisições são inseridas em uma única fila e a alocação é realizada conforme a ordem de chegada.

Paradigma de Tolerância de Espera: Busca-se maximizar o efeito de *Batching*. Com este objetivo, define-se um *intervalo de espera* (*wait threshold*), ou seja, um intervalo de tempo no qual requisições são retidas, explorando-se o tempo estimado de espera dos usuários. Definem-se duas classes de políticas:

Max_Batch: Um vídeo é escalonado para alocação se pelo menos uma de suas requisições foi retida durante o intervalo de espera. Caso mais de um vídeo satisfaça esta condição, aloca-se o fluxo utilizando-se o critério de maior tamanho de fila (*Max Batch MQL* — MBQ) ou de maior número estimado de abandonos (*Max Batch with Minimal Loss* — BML). Caso contrário, um fluxo permanece disponível até que uma requisição satisfaça a este critério.

Min_Idle: Dividem-se os vídeos em dois conjuntos: populares, $\mathcal{H}$ (*hot videos*), e não populares, $\mathcal{C}$ (*cold videos*). Aplica-se *Batching* apenas sobre o conjunto $\mathcal{H}$ sem a restrição de tempo mínimo de espera no conjunto $\mathcal{C}$. Um vídeo é inserido em $\mathcal{H}$ se *i*) é popular, *ii*) sua fila possui mais de uma requisição, e *iii*) a única requisição de sua fila excedeu o intervalo de espera. Se o conjunto $\mathcal{H}$ não estiver vazio, aloca-se um vídeo utilizando-se o critério de maior tamanho de fila (*Min Idle MQL* — IMQ) ou maior número esperado de perdas (*Min Idle with Minimal Loss* — IML). Caso contrário, aloca-se um fluxo para um vídeo do conjunto $\mathcal{C}$ utilizando-se a política FCFS.

Piggybacking

A técnica de *Piggybacking* [33, 34, 41, 42] baseia-se no fato de que alterações da ordem de 5% na taxa de exibição não são perceptíveis aos usuários. Desta forma, as requisições são atendidas imediatamente e o fluxo compartilhado é obtido através da superposição de fluxos dessincronizados, ou seja, altera-se a taxa de exibição dos fluxos de vídeo de tal forma que estes venham a exibir um mesmo quadro de filme em um determinado instante, descartando-se um dos fluxos após a sincronização.

Se por um lado o *Piggybacking* não proporciona retardo inicial como no *Batching*, por outro, não é tão eficiente em termos de redução da demanda de banda passante. Isto porque, na chegada de uma requisição, um canal lógico é alocado imediatamente (se houver algum disponível) e este somente será liberado por ocasião de uma mesclagem ou finalização da exibição do vídeo.

Na técnica de *Piggybacking* denomina-se **Janela de Mesclagem** (*Catchup Window*), $W_{Pol}(f_i)$, a distância, dada em quadros e calculada relativamente à posição f_i, que indica a possibilidade de sincronização de dois fluxos para a política *Pol*. Quanto às taxas de exibição, definem-se as taxas mínima, normal e máxima, em quadros por segundo, denotadas respectivamente por, S_{min}, S_n e S_{max}.

Um aspecto importante a ser considerado constitui-se em como obter as variações nas taxas de exibição dos quadros dos vídeos. Uma solução simples seria alterar o parâmetro de freqüência de amostragem dos quadros durante a codificação e armazenar versões alteradas juntamente com as originais, aumentando-se o custo de armazenamento. Uma segunda alternativa para se obter a alteração das taxas de apresentação dos quadros

consiste em utilizar dispositivos que possam inserir ou retirar quadros, obtendo assim expansão ou compactação através de interpolação. As desvantagens desta última abordagem são que *i*) o *layout* dos dados no disco geralmente está adequado a uma determinada taxa de transmissão. Assim sendo, o suporte a diferentes taxas pode prejudicar o escalonamento e/ou requisição de espaço adicional de *buffer* de armazenamento; *ii*) o suporte à modificação em tempo real implica em custo adicional com *hardware* especializado.

A seguir estão relacionadas as políticas propostas na literatura para implementação de *Piggybacking*.

Par-Ímpar: Define-se $W_{pi}(0)$ como a janela de mesclagem para a política par-ímpar relativa ao início de um vídeo. Para cada fluxo iniciado pelo sistema em W_{pi}, atribuem-se as "velocidades" S_{min} e S_{max} alternadamente, de modo que ao primeiro fluxo atribui-se S_{min} (Figura 2.4). Pela formulação desta política, o ganho máximo obtido é de 50%.

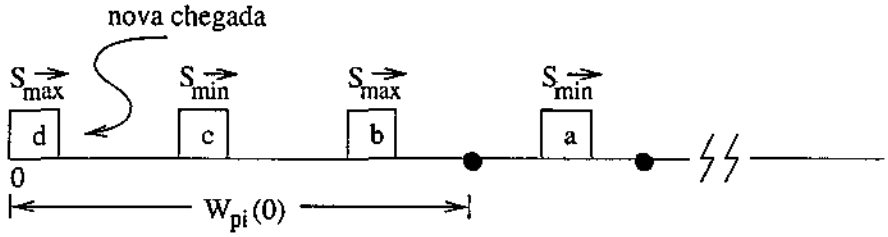


Figura 2.4: Ilustração do funcionamento da política par-ímpar.

Mesclagem Simples: Definem-se duas janelas de mesclagem, $W_{ms}(0)$ e $W^m_{ms}(0)$ (janela máxima de mesclagem da política mesclagem simples), calculadas com relação ao início do vídeo. $W^m_{ms}(0)$ indica a última posição em que dois fluxos podem sincronizar-se, ou seja, se o fluxo i chega ao sistema e encontra o fluxo j $W_{ms}(0)$ quadros à sua frente, i e j são sincronizados no limite final de $W^m_{ms}(0)$.

Esta abordagem associa fluxos a "grupos de mesclagem", em que os fluxos são superpostos formando um único fluxo resultante antes de deixarem $W^m_{ms}(0)$. Se o grupo é composto de n fluxos, o primeiro terá sua taxa ajustada em S_{min} e os demais em S_{max} (Figura 2.5), até o instante da última mesclagem, quando o fluxo resultante passa a ter velocidade S_n.

Gulosa: Esta política realiza mesclagens sempre que possível durante a exibição do vídeo. Com este objetivo, definem-se janelas de mesclagem adicionais além da inicial,

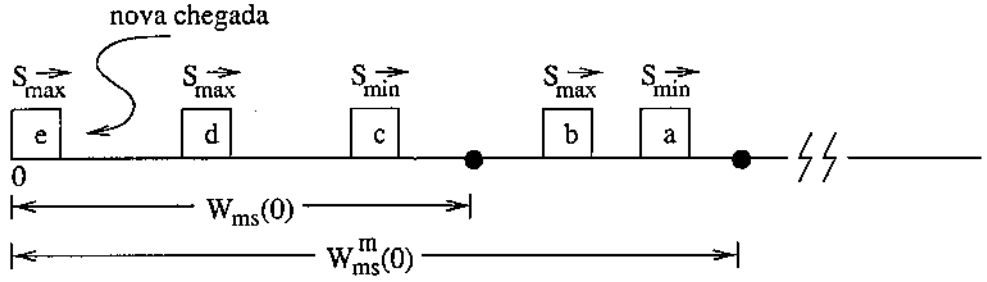


Figura 2.5: Ilustração do funcionamento da política mesclagem simples.

$W_g(0)$, calculadas relativas à uma posição f_i, $W_g(f_i)$. Esta janela de mesclagem é utilizada como um indicativo para as mesclagens adicionais.

A política gulosa funciona como segue: se ao cruzar a janela os fluxos ainda não estiverem sincronizados, calcula-se $W_g(W_g(0))$ para verificar a possibilidade de mesclagens com fluxos anteriores. Quando uma mesclagem ocorre na posição f_i, uma nova janela de mesclagem é calculada, $W_g(f_i)$. Se não houver fluxos na janela, a taxa de exibição é ajustada para S_n, caso contrário, a velocidade do fluxo anterior é ajustada em S_{min} e a do fluxo atual em S_{max}.

A Figura 2.6 ilustra a política gulosa. Neste caso, os fluxos b e d já foram mesclados com a e c, respectivamente.

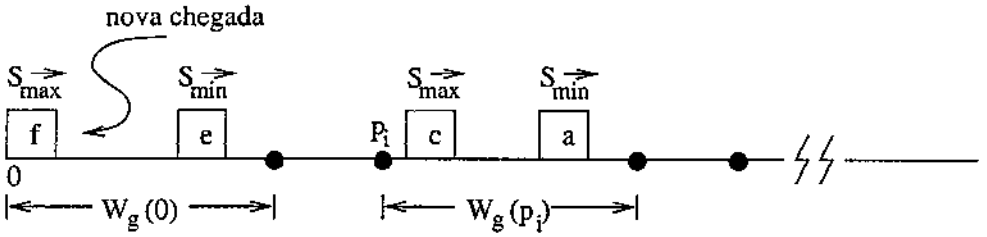


Figura 2.6: Ilustração do funcionamento da política gulosa.

Mesclagem Simples Generalizada: Define-se a janela de mesclagem máxima, W_m, como a última posição em quadros que permite a ocorrência de uma mesclagem antes do fim do vídeo. Dada a diferença das taxas, calcula-se W_m como:

$$W_m = \frac{S_{max} - S_{min}}{S_{max}} \cdot L,$$

em que L é o número de quadros do vídeo.

O dimensionamento da janela de mesclagem é um fator importante porque:

- Quando o tamanho da janela é grande, um número maior de fluxos pode ser mesclado em um único fluxo. Porém, as mesclagens ocorrem próximas do final do vídeo;
- Quando o tamanho da janela é muito pequeno, as mesclagens tendem a ocorrer próximas do início do vídeo, porém, em quantidade reduzida.

Pode-se perceber que ambas as situações não favorecem à redução do número de quadros exibidos por um conjunto de fluxos. Aggarwal *et al.* [41, 42] apresentam um método analítico para otimizar o tamanho da janela de mesclagem, o qual leva em consideração o tamanho do vídeo e a taxa de chegada prevista, assumindo que as chegadas são modeladas por um processo de Poisson com parâmetro λ. Com isso, a janela de mesclagem derivada otimiza o número e a posição, relativamente ao tamanho do vídeo, em que mesclagens ocorrem; e, deste modo, propicia uma maior redução na demanda de banda passante.

Assim sendo, define-se a política mesclagem simples generalizada em função do parâmetro W ($0 < W \leq W_m$) chamado **Janela Ótima de Mesclagem**, sendo esta a única diferença entre esta política e sua versão original, definida em função de W_m. A Figura 2.7 ilustra o dimensionamento da janela ótima de mesclagens em função da janela máxima. Observa-se que à medida em que o intervalo médio entre chegadas assume valores maiores, o tamanho de W aproxima-se do tamanho de W_m.

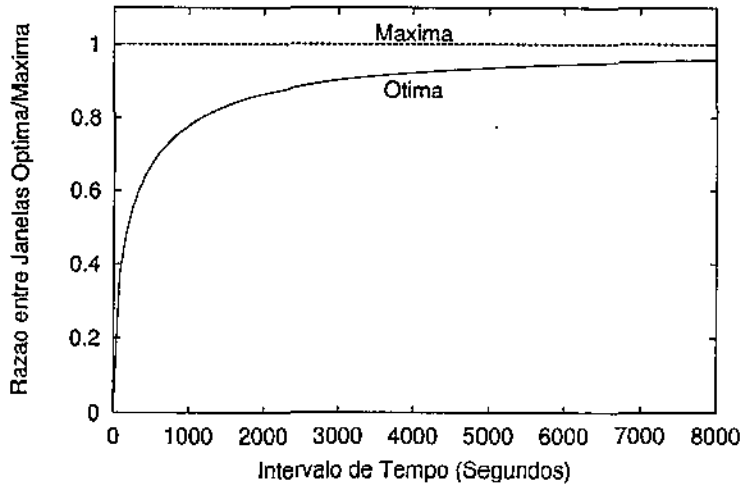


Figura 2.7: Dimensionamento da janela ótima relativo à janela máxima em função do intervalo médio entre requisições.

Quanto às taxas de exibição, S_{max} é associada a um fluxo se já existir um outro em W com taxa S_{min}, caso contrário, associa-se S_{min}. Quando um fluxo "rápido" sincroniza-se com um "lento", descarta-se o rápido.

Algoritmo *Snapshot*: Dentre as políticas de *Piggybacking*, a política *Snapshot* [41, 42] é a que busca minimizar o número de quadros exibidos por um conjunto de fluxos de vídeo de forma mais eficiente.

A computação realizada pelas políticas de *Piggybacking* pode ser vista como uma árvore binária em que as folhas correspondem aos fluxos, os nós internos são as mesclagens intermediárias e a raiz é a mesclagem final formando o fluxo resultante do conjunto (Figura 2.8). Estas representações gráficas das estratégias de mesclagem de um conjunto de fluxos são chamadas de **Árvores de Mesclagens**. Portanto, o número de possíveis árvores formadas a partir de um conjunto de fluxos constitui-se no número de políticas de *Piggybacking* potencialmente ótimas e é dado pelo $(n-1)$-ésimo número de **Catalan** [41, 42, 43]:

$$Catalan(n-1) = \frac{(2n-2)!}{(n-1)!n!} \tag{2.1}$$

o que implica que o número de árvores cresce rapidamente inviabilizando uma busca exaustiva da estratégia ótima e sua árvore binária correspondente.

Deste modo, a política *Snapshot* constrói a árvore ótima de mesclagem de fluxos de vídeo da seguinte forma: considere um conjunto de n fluxos de um mesmo vídeo (o qual é composto de L quadros) e suas posições dadas em quadros e denotadas por $f_1, f_2, \ldots, f_n$, em que $f_1 \geq f_2 \geq \cdots \geq f_n$ em um determinado instante de tempo. Sejam i e j dois fluxos entre 1 e n, com $i \leq j$. Denota-se $P(i,j)$ como a posição de mesclagem (em quadros), na qual os fluxos i e j apresentam o mesmo quadro de filme. Uma vez que o fluxo i possui velocidade S_{min} e o fluxo j, S_{max}, a posição de mesclagem é dada por:

$$P(i,j) = f_i + \frac{S_{min} \cdot (f_i - f_j)}{S_{max} - S_{min}} \tag{2.2}$$

Seja $C(i,j)$ o custo de uma política e $\mathcal{A}(i,j)$ sua árvore binária correspondente. O custo para cada fluxo é dado por:

$$C(i,i) = L - f_i \tag{2.3}$$

Para se obter o custo $C(i,j)$ mínimo é necessário que o princípio de otimalidade seja satisfeito, ou seja: *Existe um fluxo k, com $i \leq k < j$, tal que as subárvores esquerda $(i,\ldots,k)$ e direita $(k+1,\ldots,j)$ também são ótimas.* Estas subárvores são denotadas por $\mathcal{A}(i,k)$ e $\mathcal{A}(k+1,j)$. O custo da árvore que corresponde ao conjunto de fluxos $i,\ldots,j$ é dado por:

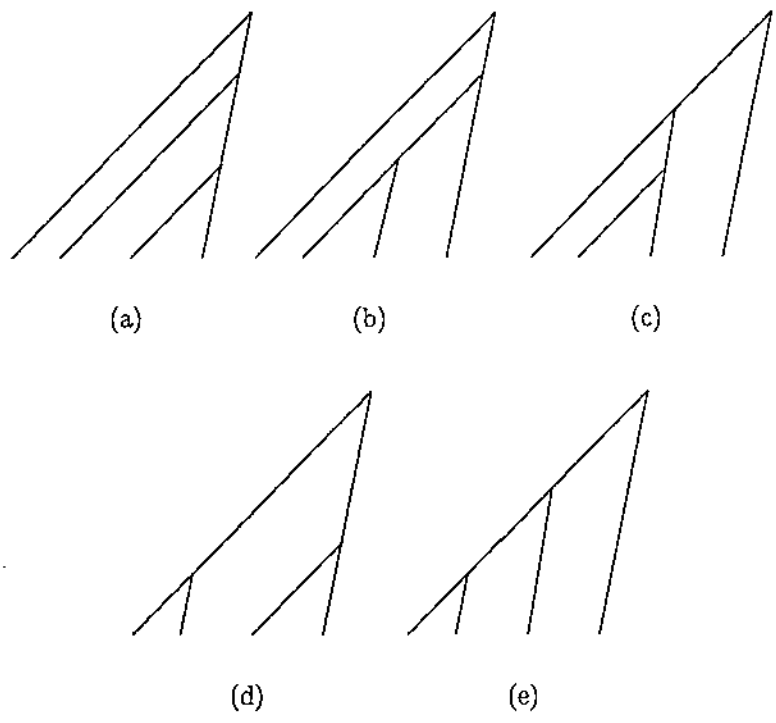


Figura 2.8: Possíveis árvores de mesclagens para quatros fluxos.

$$C(i,k) + C(k+1,j) - \max(L - P(i,j), 0) \tag{2.4}$$

Portanto, a política ótima para os fluxos $i, \ldots, j$ possui subárvores $i, \ldots, k^*$ e $k^* + 1, \ldots, j$, em que:

$$k^* = \text{argmin}_{i \leq k < j}\{C(i,k) + C(k+1,j) - \max(L - P(i,j), 0)\} \tag{2.5}$$

Deste modo, o custo para o conjunto dos n fluxos, $C(1,n)$, pode ser calculado de forma *bottom-up* através de um algoritmo de programação dinâmica.

Assim como as demais políticas de *Piggybacking*, a política *Snapshot* atribui velocidades aos fluxos que são disparados pelo sistema dentro de um intervalo, neste caso o intervalo *Snapshot*, denotado por I. O intervalo *Snapshot* é dado por $I = W/S_{max}$, em que W é o valor ótimo da janela de mesclagem derivado para a política mesclagem simples generalizada [42, 41]. Dentro do intervalo I, a política *Snapshot* comporta-se como a política mesclagem simples. Ao final do intervalo os procedimentos de otimização descritos anteriormente são aplicados.

2.6 Síntese do Capítulo

Sistemas de Vídeo sob Demanda constituem-se numa das principais aplicações das futuras RDSI-FL. Num sistema de VoD, um usuário pode escolher um vídeo dentre uma grande coleção e controlá-lo através de operações de VCR. Os diversos componentes de um sistema de VoD são o servidor de vídeo, a rede de comunicação e subredes de acesso aos pontos de recepção e o equipamento do usuário, composto pela *set top box* e controle remoto. O oferecimento de operações de VCR implica em grande complexidade no projeto dos servidores de vídeo, fato que pode ter como conseqüência um aumento substancial do custo dos serviços oferecidos.

Diversas técnicas foram propostas com o objetivo de reduzir a grande demanda de banda passante das aplicações de vídeo. Estas técnicas baseiam-se em princípios diferentes e são aplicáveis em pontos distintos de uma arquitetura de VoD. Particularmente, destacam-se as técnicas de *Batching* e *Piggybacking* que propõem a utilização compartilhada de fluxos vídeo. Ambas podem ser implementadas de diversas formas, as quais são definidas por suas políticas.

Capítulo 3

Piggybacking

A técnica de *Piggybacking* tem como principal característica o atendimento imediato das requisições por vídeos, aspecto amplamente desejável em um sistema de VoD. Porém, para uma efetiva redução da demanda de banda passante, faz-se necessária a utilização de políticas sofisticadas. Políticas eficientes devem considerar fatores como o dimensionamento da janela de mesclagem, a topologia (que minimize o custo) da árvore de mesclagem construída, otimizações na janela de mesclagem e a possibilidade da implantação de múltiplos níveis de mesclagem. Dentre as várias políticas de *Piggybacking* destaca-se a política Algoritmo *Snapshot* [41, 42]. Esta política otimiza a demanda de banda passante considerando apenas os fatores de dimensionamento da janela e topologia da árvore de mesclagem.

Neste capítulo apresentam-se duas extensões à política *Snapshot* que consideram os fatores mencionados anteriormente. A primeira política, denominada S^2 (seção 3.1), utiliza dois níveis de mesclagem (otimização), a segunda extensão constitui-se na política Híbrida (seção 3.2), que busca otimizar as mesclagens no interior da janela de mesclagem (intervalo *Snapshot*). Adicionalmente, consideram-se aspectos de complexidade do algoritmo de construção da árvore ótima de mesclagens da política *Snapshot* (seção 3.3). Analisam-se duas abordagens de redução desta complexidade que são: (*i*) emprego de algoritmos para o problema da parentização ótima do produto de matrizes (solução através de triangulação de polígonos — seção 3.3.3), e (*ii*) elaboração de uma heurística com custo computacional inferior para a construção das árvores de mesclagem (seção 3.3.4).

3.1 A Política S^2

A política *Snapshot* visa minimizar o número de quadros exibidos pelos fluxos iniciados durante o último intervalo I, de tal modo que é necessário aguardar o final deste intervalo. É interessante mencionar que quando o tamanho de W é pequeno com relação

ao valor da janela máxima de mesclagem, W_m, a redução no número de quadros exibidos por todos os fluxos no sistema não é ótima, mesmo que as reduções locais (em cada intervalo) sejam.

É fácil verificar que, quanto menor o intervalo médio entre requisições (taxas de chegadas maiores) as janelas ótimas de mesclagem serão cada vez menores, como pode ser visto na Figura 2.7, podendo haver uma ou várias janelas ótimas contidas numa mesma janela máxima de mesclagem. A possibilidade da superposição de dois fluxos separados por no máximo W_m quadros permite que ganhos superiores aos da política *Snapshot* possam ser obtidos. Assim sendo, propõe-se a política S² que busca otimizar o número de quadros exibidos por um conjunto de fluxos disparados pelo sistema numa **Janela Máxima Alterada** de mesclagem, W'_m, em que $W \leq W'_m \leq W_m$, e não somente no intervalo I. A janela máxima alterada é constituída por um número inteiro de janelas ótimas, isto é, a janela máxima alterada possui $\lfloor W_m/W \rfloor$ janelas ótimas. Em outras palavras, a política S² introduz um segundo nível de mesclagens, ou seja, a mesclagem das resultantes dos intervalos I da política *Snapshot*.

O funcionamento da política S² é como se segue: aplica-se primeiramente o algoritmo *Snapshot* sobre os fluxos iniciados pelo sistema nos intervalos *Snapshot* (como proposto originalmente). Em seguida, aplica-se também o algoritmo de programação de dinâmica sobre os fluxos resultantes destes intervalos. É interessante enfatizar que, de acordo com a definição da própria política *Snapshot*, atribui-se a velocidade S_{max} a estes fluxos resultantes com exceção daquele gerado pela primeira janela ótima contida em W'_m. Outro aspecto importante é que, ao contrário do que ocorre com o primeiro nível de otimização, o ponto de aplicação do procedimento de otimização não ocorre ao final de intervalos de duração fixa. Estes intervalos são denotados por I_{S^2} e sua duração é determinada pelo padrão das requisições por vídeos em cada janela W'_m (Figura 3.1).

Uma generalização natural da política S² seria considerar n níveis de otimização. No entanto, os ganhos obtidos com a implementação destes níveis seriam praticamente nulos dado que os fluxos nestes níveis estariam separados por valores bem próximos a W_m quadros (ou ainda maiores) o que implica em mesclagens próximas do (ou após o) final do vídeo. Portanto, a introdução de níveis extras de mesclagens não proporciona reduções efetivas na demanda de banda passante.

Para avaliar o impacto da introdução de um segundo nível de otimização, foi realizado um estudo comparativo através de simulação entre as políticas S², *Snapshot* original e a política *Snapshot* global, uma variante da política *Snapshot* que considera todos os fluxos de um mesmo vídeo sem o particionamento dos mesmos em subgrupos. A função objetivo utilizada neste estudo difere da considerada em [41], a qual contabiliza apenas o número de quadros exibidos após o intervalo I. A nova função objetivo contabiliza o número total de quadros apresentados por um conjunto de fluxos, ou seja, reflete a otimização realizada

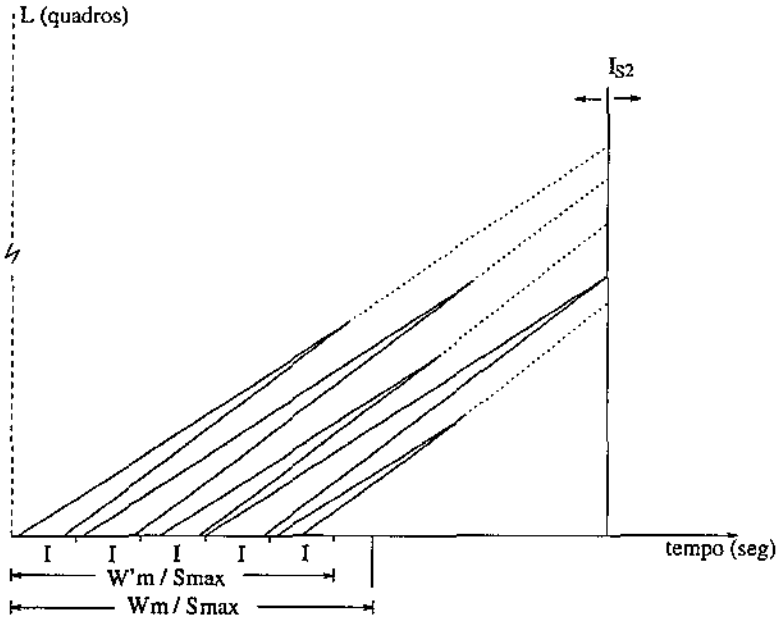


Figura 3.1: Possível situação da política S^2 na qual o último fluxo resultante gerado não corresponde ao fluxo resultante da última janela ótima contida na janela W'_m.

no intervalo I assim como a realizada pela política S^2 (posterior ao intervalo) e é dada por:

$$C(i,i) = L \tag{3.1}$$

para um fluxo i e para um conjunto de fluxos $i,\ldots,j$:

$$C(i,j) = Quadros_I + \sum_{k=\text{argmin}_{i\leq k<j}} (C(i,k) + C(k+1,j) - \max(L - P(i,j), 0)) \tag{3.2}$$

em que $Quadros_I$ é o somatório dos quadros apresentados pelos m fluxos descartados, $Q(\ell)$ com $\ell = 1,\ldots,m$, antes do final dos intervalos *Snapshot*:

$$Quadros_I = \sum_{\ell=1}^{m} Q(\ell) \tag{3.3}$$

3.1.1 Resultados Numéricos

Os resultados contidos nesta seção foram obtidos via simulação de eventos discretos, considerando-se um intervalo com nível de confiança de 95%, calculado através do método

de replicações independentes. Nos gráficos os intervalos de confiança foram omitidos para facilitar a visualização dos dados. Assume-se que as chegadas são modeladas por um processo de Poisson e utilizam-se $S_{min} = 28.5$ e $S_{max} = 31.5$ em quadros/segundos. Os gráficos mostram a redução percentual média no número de quadros apresentados em decorrência da aplicação das políticas de *Piggybacking*.

Foram realizados três experimentos:

- análise das políticas sob diversas taxas de chegadas;
- análise de sensibilidade da política S^2 ao tamanho da janela de mesclagem;
- análise do efeito da variação da duração de um filme em conjunto com a taxa de chegadas sobre a política S^2.

No gráfico da Figura 3.2, ilustra-se o comportamento das políticas sob diferentes taxas de chegada, isto é, varia-se o intervalo médio entre requisições de 15 a 500 segundos. Pode-se verificar que, à medida em que a duração destes intervalos aumenta, os ganhos com adoção de *Piggybacking* são reduzidos. Constata-se também que, enquanto existem pelo menos duas janelas ótimas contidas numa janela máxima, a política S^2 apresenta comportamento diferenciado da política *Snapshot*. Em situações como estas, em que são consideradas taxas altas de chegadas, o desempenho da política S^2 mostra-se superior ao da política *Snapshot*, de tal forma que a diferença percentual entre ambas pode alcançar valores de até 8%. Quando a introdução de um segundo nível de mesclagens não é mais possível, ou seja, existe somente uma janela ótima contida na janela máxima, as políticas apresentam comportamento semelhante. Quanto a política *Snapshot* Global, seu desempenho mostra-se inferior ao das demais em virtude da aplicação do algoritmo de geração da árvore de mesclagens somente ao final de intervalos de tempo bastante longos, de modo que ocorrem mesclagens próximas do final da exibição do vídeo.

A Figura 3.3 mostra o efeito da variação da janela de mesclagem sobre a política S^2. Neste experimento, considera-se o intervalo médio de chegadas de 30 segundos e a duração do vídeo de 2 horas. Pode-se verificar que a política S^2 é insensível à variação da janela de mesclagem. Isto se deve ao fato de que o segundo nível de otimização "recupera" eventuais perdas que ocorrem no primeiro nível em virtude de um dimensionamento inadequado da janela de mesclagem. Em outras palavras, quando a janela de mesclagem assume valores muito pequenos, ocorrem poucas mesclagens no primeiro nível de otimização, o que implica uma pequena redução no número de quadros exibidos. Porém, uma vez que existe um segundo nível, os fluxos resultantes do primeiro ainda podem ser superpostos, proporcionando assim uma redução ainda maior no número de quadros apresentados. À medida em que o tamanho da janela de mesclagem assume valores próximos dos da janela máxima, os ganhos são obtidos já no primeiro nível de otimização.

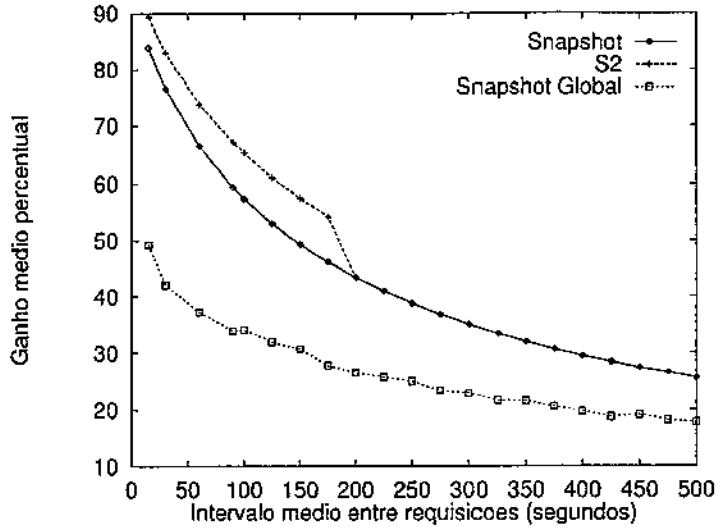


Figura 3.2: Comparação entre as políticas *Snapshot* original, S^2 e *Snapshot* global.

A insensibilidade à variação da janela de mesclagem constitui-se numa característica importante, pois a estimativa da taxa de chegadas é realizada à medida em que as requisições chegam ao sistema, podendo-se obter estimativas imprecisas em virtude de flutuações na taxa de chegadas. Nos casos em que a taxa é estimada de forma imprecisa, ocorre um dimensionamento inadequado da janela ótima, implicando numa possível degradação do desempenho das políticas baseadas neste conceito. Através desta análise, constata-se que a política S^2 é imune ao dimensionamento inadequado da janela ótima devido à existência de dois níveis de otimização.

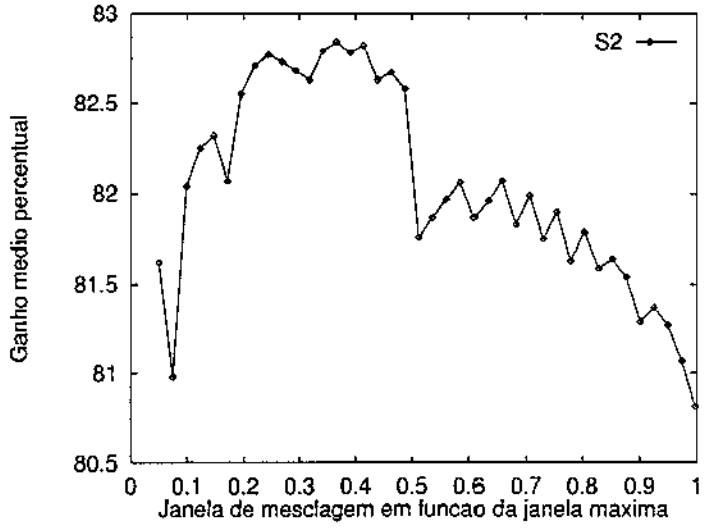


Figura 3.3: Influência do tamanho da janela sobre a política S^2.

A Figura 3.4 ilustra o efeito da variação da duração do vídeo em conjunto com a variação do intervalo médio entre requisições sobre a política S^2. Neste experimento, vídeos com duração de 30 minutos até 4 horas são submetidos a diversos intervalos médios entre chegadas, os quais assumem valores que variam de 15 a 500 segundos. Nota-se que são obtidas reduções maiores no número de quadros exibidos para filmes mais longos e submetidos a taxas de chegada maiores. A redução na demanda de banda passante varia desde 9% para vídeos com 30 minutos de duração e intervalo médio entre requisições de 500 segundos chegando até a 93% para filmes longos (4 horas de duração) com intervalo entre chegadas de 15 segundos.

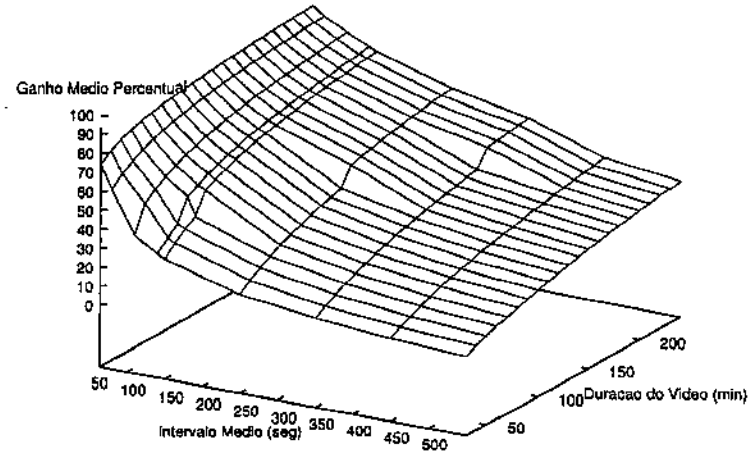


Figura 3.4: Influência da duração do vídeo em conjunto com a taxa de chegadas sobre a política S^2.

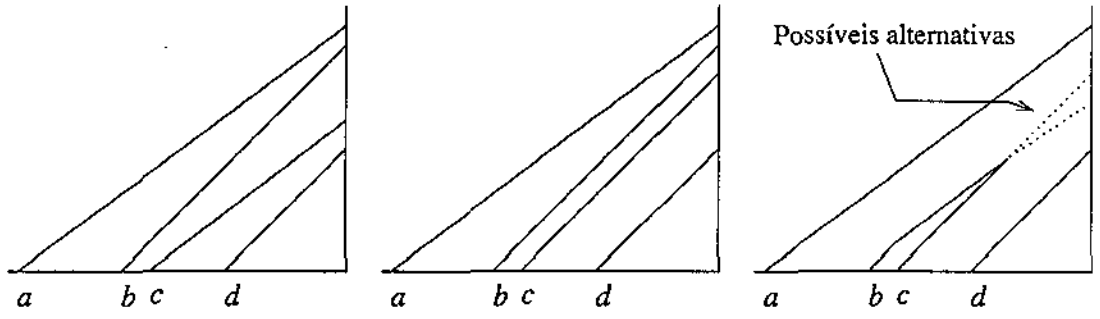
3.2 Otimizações no Intervalo I

As políticas de *Piggybacking* são caracterizadas pelo modo como as velocidades são atribuídas aos fluxos quando estes são iniciados. Nas políticas simples (mesclagem simples, par-ímpar e gulosa [33, 34]) uma vez atribuídas as velocidades, estas não são alteradas. Por outro lado, a política *Snapshot* pode alterar as taxas de exibição durante a construção da árvore ótima de mesclagem. Todas as políticas pressupõem intervalos básicos para as atribuições das velocidades dos fluxos. As políticas simples consideram a janela máxima de mesclagem, W_m, enquanto que as políticas de mesclagem simples generalizada, *Snapshot* e S^2 consideram a janela ótima, W. No entanto, nenhuma destas políticas considera o

tempo decorrido entre fluxos para atribuir as taxas de exibição aos fluxos no interior do intervalo *Snapshot*.

Como exemplo, considere as situações ilustradas nas Figura 3.5-a e 3.5-b. Pode-se facilmente perceber que é mais interessante mesclar os fluxos b e c e alterar a velocidade de sua resultante de modo que esta venha a se mesclar com um dos dois outros fluxos (a ou d) (Figura 3.5-c).

As políticas *Snapshot* e S^2 são as únicas que consideram a distância relativa entre os fluxos na busca da árvore ótima de mesclagem, porém este critério não é aplicado no interior do intervalo I. Este fato motivou a elaboração de uma heurística, denominada política Híbrida, cuja aplicação é limitada ao interior do intervalo I em substituição à política mesclagem simples. O funcionamento da política Híbrida é descrito a seguir.

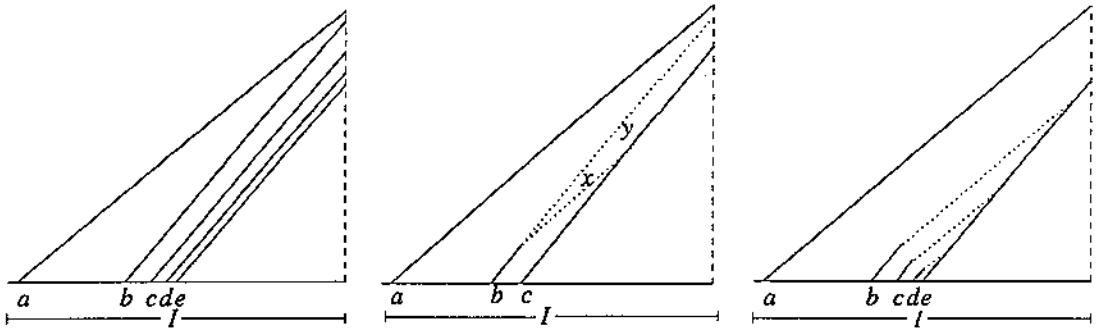


(a) Par-Ímpar (b) Mesclagem Simples (c) Possível Otimização

Figura 3.5: Cenários formados pelo modo de atribuição das velocidades aos fluxos nas políticas (a) par-ímpar, (b) mesclagem simples e (c) busca de otimização dinâmica.

Ao primeiro fluxo iniciado no intervalo I atribui-se a velocidade S_{min} e aos demais, S_{max}. A cada fluxo iniciado, um par de mesclagem é formado quando a distância para a posição de mesclagem entre este e seu antecessor é menor que a existente entre seus dois antecessores. Um novo fluxo é agregado a um par, constituindo um grupo de mesclagem composto por três fluxos, se o tempo decorrido entre o iniciação deste e de seu antecessor é inferior a um limite (*threshold*) predefinido. Agrega-se um novo fluxo ao grupo quando o tempo decorrido entre a iniciação de seu último componente (terceiro fluxo) e do novo fluxo é inferior a um intervalo cuja duração máxima correspondente à metade do *threshold*, de tal forma que os grupos de mesclagem formados têm cardinalidade máxima igual a quatro fluxos. Deve-se reaplicar este procedimento a cada novo fluxo disparado pelo sistema formando pares ou grupos de mesclagem, enquanto não se alcança o final do intervalo I. Quando um par de fluxos é mesclado, o fluxo resultante deve ser novamente analisado em conjunto com os fluxos vizinhos para verificar a formação de novos pares ou grupos de

mesclagem.



(a) Mesclagem Simples (b) Política Híbrida (c) Política Híbrida

Figura 3.6: Cenários ilustrando as políticas (a) mesclagem simples, (b) e (c) Híbrida.

A fim de ilustrar a política proposta, considere o cenário mostrado na Figura 3.6-a, o qual representa o comportamento da política mesclagem simples, ou seja, atribui-se S_{min} ao primeiro fluxo do intervalo I e S_{max} aos demais sem alterações posteriores até o final do intervalo. Descreve-se a seguir, a aplicação da política Híbrida sobre o mesmo conjunto de fluxos: seja TC, o vetor cuja componente representa o tempo decorrido desde o início do intervalo I até o começo da exibição de um fluxo. No instante em que o fluxo c é iniciado, verificam-se as distâncias x e y (segmentos pontilhados), se $x < y$, então desviar o fluxo b em direção ao c compondo o par de mesclagem b-c, ou seja, atribuir S_{min} ao fluxo b (Figura 3.6-b). Ao disparar o fluxo d, dado que o par b-c ainda não se mesclou, agrega-se d ao par se $TC[d] - TC[c] \leq threshold$, obtendo-se o grupo de mesclagem b-c-d em que d possui velocidade S_{max} e os demais S_{min}. Caso contrário, manter o fluxo c em S_{max} e atribuir S_{max} para d. O fluxo e é agregado ao grupo b-c-d se $TC[e] - TC[d] \leq threshold/2$, obtendo-se o grupo b-c-d-e em que a velocidade do fluxo e será S_{max} e a dos demais S_{min} (Figura 3.6-c).

Dentre as vantagens da política Híbrida pode-se citar: *i*) o descarte antecipado de fluxos para atendimento de outras requisições, *ii*) a redução do tempo de execução para geração da árvore ótima, em decorrência da redução do número de fluxos que atingem o final do intervalo I, e *iii*) a aproximação entre os fluxos que ultrapassam o final do intervalo I. Estes fatores em conjunto permitem a construção de uma árvore ótima de mesclagens com custo inferior que a gerada caso a política aplicada no intervalo I fosse mesclagem simples, na qual os fluxos $2, \ldots, n$ progridem "paralelamente".

3.2.1 Resultados Numéricos

Os resultados apresentados comparam a redução percentual média no número de quadros exibidos pelos fluxos de vídeo quando se emprega a política Híbrida em substituição à política mesclagem simples no intervalo I. Assume-se que as chegadas são modeladas por um processo de Poisson e utiliza-se $S_{min} = 28.5$ e $S_{max} = 31.5$ em quadros/segundo.

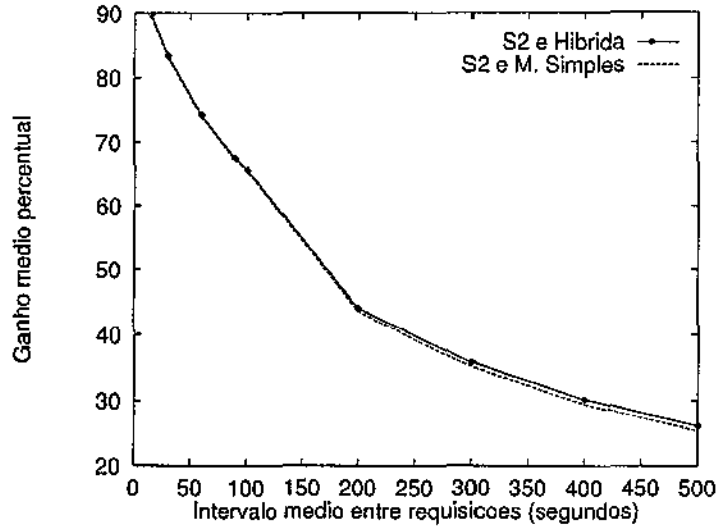


Figura 3.7: Efeito da variação do intervalo médio entre chegadas sobre a política S^2 com mesclagem simples e com Híbrida no intervalo I.

A Figura 3.7 ilustra o comportamento da política S^2 quando utiliza-se a política Híbrida em substituição à mesclagem simples no intervalo I. O intervalo médio entre chegadas assume valores entre 15 e 500 segundos, o valor do *threshold* foi fixado em 5 segundos e a duração do vídeo é de 2 horas. O ganho médio obtido reflete o menor número de quadros exibidos nos intervalos I e a redução no número de fluxos após o intervalo *Snapshot*.

A Figura 3.8 mostra o desempenho da política S^2 em conjunto com a política Híbrida e mesclagem simples quando fixa-se o intervalo médio de chegadas em 30 segundos e varia-se a duração do vídeo entre 30 minutos e 4 horas. Ambas as curvas refletem o ganho médio percentual obtido num período de 4 horas.

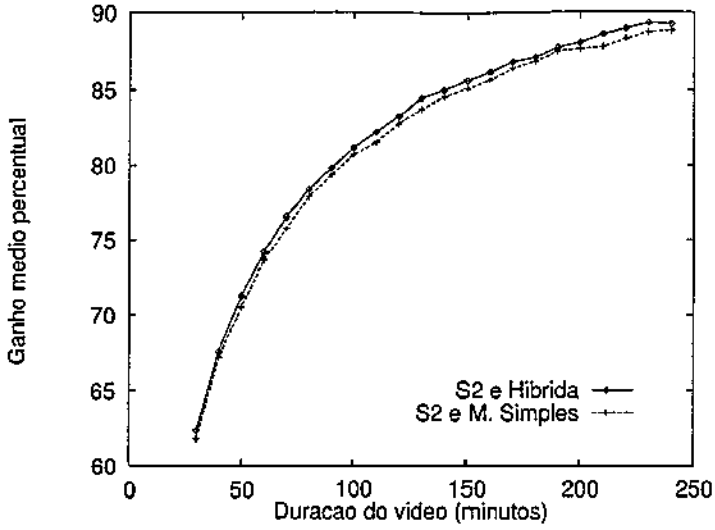


Figura 3.8: Efeito da variação da duração do vídeo sobre a política S^2 com mesclagem simples e com Híbrida no intervalo I.

3.3 Considerações sobre a Complexidade de Construção da Árvore de Mesclagens

Devido aos requisitos de tempo real das aplicações de vídeo e à intensa carga de processamento nos servidores (seção 2.3), a cpu pode transformar-se num recurso crítico num sistema de VoD. Portanto, uma característica desejável nos algoritmos empregados consiste no fato de que possuam baixo custo computacional.

Num sistema de VoD em que se emprega a técnica de *Piggybacking*, a política *Snapshot* constitui-se numa alternativa atrativa sob o aspecto de otimização da demanda de banda passante. Porém, a árvore ótima de mesclagens é construída por um algoritmo de programação dinâmica que possui complexidade $\Theta(n^3)$, em que n representa o número de fluxos que alcançam o final de um intervalo I. Adiciona-se a este fato o progresso contínuo dos fluxos paralelamente à execução do algoritmo, ou seja, deseja-se obter a árvore de mesclagens antes que o contexto caracterizado pelas posições dos fluxos seja alterado.

Assim sendo, busca-se uma alternativa que seja atrativa sob os aspectos de otimização dos recursos do sistema (minimização da demanda de banda passante) e de desempenho. Neste sentido, duas abordagens foram investigadas: *i*) utilização de algoritmos propostos para o problema da parentização ótima do produto de matrizes (solução através da triangulação de polígonos convexos), em virtude das analogias entre os problemas da parentização ótima e construção da árvore ótima de mesclagens; e *ii*) proposição de uma

heurística para a construção da árvore de mesclagens. Na seção 3.3.1 são introduzidos alguns conceitos para uma melhor compreensão da primeira abordagem, em seguida, cada uma delas é detalhada.

3.3.1 Definições Preliminares

A primeira abordagem seguida para a redução da complexidade de construção da árvore de mesclagens endereça as analogias existentes entre este problema e o problema da parentização ótima do produto de matrizes, conforme será explicado na seção 3.3.2. Em decorrência da redução deste problema ao problema da triangulação de polígonos convexos, faz-se necessária a introdução de algumas definições relacionadas, extraídas das referências [44, 45].

Polígono Simples

Polígono é uma curva fechada no plano finalizada em si mesma, formada por uma seqüência de segmentos de reta chamados **lados** do polígono. Um ponto unindo dois lados consecutivos é chamado de **vértice** do polígono. Se os lados de um polígono não se interceptam, diz-se que este é um **Polígono Simples**. O conjunto de pontos no plano envoltos pelo polígono formam o seu **interior** e o conjunto de pontos dos lados e vértices do polígono formam sua **fronteira**.

Polígono Convexo

Um polígono simples é **convexo** se, dados quaisquer dois pontos em sua fronteira ou em seu interior, todos os pontos no segmento de reta construído entre estes estão contidos no interior do polígono ou em sua fronteira.

Triangulação de Polígonos

Dados dois vértices não adjacentes v_i e v_j, o segmento $\overline{v_i v_j}$ é uma **corda** do polígono. Uma **triangulação** de um polígono é um conjunto $\mathcal{T}$ de cordas que dividem o polígono em **triângulos** disjuntos. Dado um polígono convexo $P = \langle v_0, v_1, \ldots, v_{n-1} \rangle$ e uma função de peso definida sobre os triângulos formados pelos lados e cordas de P, o problema da **triangulação ótima** consiste em encontrar uma triangulação que minimize a soma dos pesos dos triângulos na triangulação.

Vértice Mínimo Local

Em um polígono, dois vértices são chamados *vértices vizinhos* se são adjacentes ao mesmo lado, deste modo cada vértice possui dois vizinhos. Seja um vértice, v_i, e seu peso correspondente, w_i, diz-se que v_i é um **vértice mínimo local** se o seu peso é menor que os pesos de ambos os seus vizinhos: $w_i < \min(w_{i-1}, w_{i+1})$.

Vértice Máximo Local

Vértice máximo local é aquele cujo peso é maior que os pesos de seus vizinhos: $w_i > \max(w_{i-1}, w_{i+1})$. No caso de pesos iguais, define-se o menor vértice (vértice de menor peso) como o primeiro encontrado quando se percorre o polígono no sentido horário a partir do vértice de menor peso dentre todos no polígono.

Polígono Monótono Básico

Polígono Monótono Básico é aquele que possui apenas um vértice mínimo local e um vértice máximo local.

3.3.2 Analogias entre a Parentização Ótima do Produto de Matrizes e Geração da Árvore Ótima de Mesclagens

A multiplicação de matrizes é uma operação cujo custo é determinado pelo número de operações simples realizadas. Sejam duas matrizes, $M_{p,q}$ e $M_{q,r}$, o produto das matrizes, $M_{p,r} = M_{p,q} \times M_{q,r}$, possui custo dado por: $p \times q \times r$. Quando mais que duas matrizes precisam ser multiplicadas, o encadeamento para a realização da operação pode ter grande influência sobre o custo da mesma. O problema da parentização ótima do produto de matrizes consiste na determinação do encadeamento que minimiza o número de operações realizadas no produto de n matrizes ($M_{n+1} = M_1 \times M_2 \times M_3 \times \cdots \times M_n$) [44].

A parentização ótima do produto de matrizes pode ser vista como uma árvore binária em que as folhas correspondem às matrizes, os nós internos são as matrizes intermediárias e a raiz é o produto final gerando a matriz resultante (Figura 3.9). Portanto, o número de possíveis árvores formadas a partir de um conjunto de matrizes constitui-se no número de parentizações potencialmente ótimas e é dado pelo $(n-1)$-ésimo número de **Catalan** (Equação 2.1).

As árvores binárias obtidas como soluções de ambos os problemas (parentização do produto de matrizes e mesclagem de fluxos) podem ser mapeadas em triangulações de polígonos. No problema da árvore de mesclagens, cada fluxo representa um lado do polígono e a informação que caracteriza os lados são as posições dos fluxos ao final do intervalo *Snapshot*, (Figura 3.10-a). Se k fluxos alcançam o final do intervalo, o polígono a

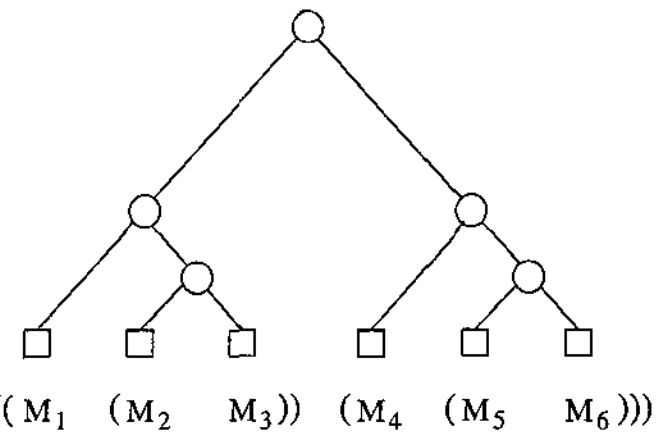


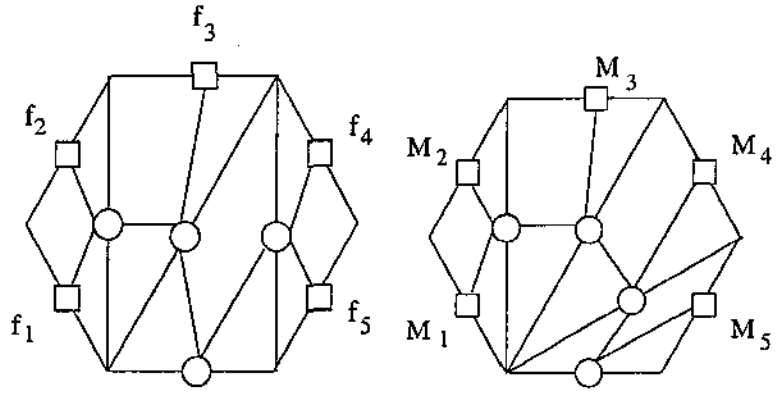
Figura 3.9: A parentização do produto de matrizes vista como uma árvore binária.

ser triangulado possui $k+1$ lados. No problema da parentização, as matrizes representam os lados do polígono a ser triangulado, porém a caracterização dos lados é realizada atribuindo-se "pesos" (dimensões das matrizes) aos vértices do polígono (Figura 3.10-b). Deste modo, uma matriz $M_{p,q}$ caracteriza um lado do polígono cujos vértices possuem pesos p e q. De forma similar ao problema anterior, o produto de k matrizes implica na triangulação de um polígono com $k+1$ lados.

Os problemas da parentização ótima e da geração da árvore ótima de mesclagens possuem analogias através da correspondência natural de árvores binárias e triangulação de polígonos. Adicionalmente, suas fórmulas de recorrência apresentam grande semelhança, implicando em modificações mínimas nos algoritmos de programação dinâmica empregados na solução de ambos. A Equação 3.4 apresenta a fórmula de recorrência para o problema da parentização ótima, na qual p, q e r são as dimensões das matrizes a ser multiplicadas. A fórmula correspondente para o problema da mesclagem de fluxos está apresentada na Equação 2.5.

$$k^* = \text{argmin}_{i \leq k < j}\{C(i,k) + C(k+1,j) + p \cdot q \cdot r\}. \tag{3.4}$$

Nas próximas seções, descrevem-se detalhadamente as estratégias investigadas para o problema da redução do custo computacional do algoritmo de construção da árvore de mesclagens. Inicialmente, busca-se a solução através da adaptação de algoritmos propostos para o problema da parentização ótima (seção 3.3.3). Em seguida, na seção 3.3.4 investiga-se o problema sob a perspectiva da elaboração de uma nova heurística.



(a) Árvore de Mesclagem (b) Produto de Matrizes

Figura 3.10: Triangulação ilustrando os problemas da (a) mesclagem de fluxos e (b) parentização ótima.

3.3.3 Adaptação de Algoritmos do Problema da Parentização Ótima

Em virtude das analogias apresentadas, a primeira alternativa seguida para a redução da complexidade de construção da árvore de mesclagens consiste na utilização de algoritmos de triangulação de polígonos, aplicados ao problema da parentização (após sua redução ao problema da triangulação), os quais possuem custo computacional inferior ao algoritmo de programação dinâmica, ou da adaptação destes algoritmos ao problema da mesclagem de fluxos. T. C. Hu e M. T. Shing [45, 46, 47, 48] apresentam algoritmos, com complexidade de $O(n)$, para o caso específico de polígonos monótonos básicos, e $O(n \log n)$, para polígonos convexos em geral (definições da seção 3.3.1). Porém, em virtude das diferenças na caracterização dos polígonos, torna-se necessária a realização de uma redução entre os problemas, ou seja, é necessário transformar a posição dos fluxos em dados bidimensionais.

O primeiro esquema investigado consiste em caracterizar cada vértice do polígono através de divisores inteiros (fatoração) das posições dos fluxos que representam os lados do polígono. Porém, este esquema foi descartado devido ao fato de a posição de cada fluxo ser um valor dependente do instante de sua chegada no intervalo *Snapshot*, podendo assumir quaisquer valores (dentro da janela de mesclagem), isto é, não há garantias da existência de divisores inteiros para estes números.

Busca-se, então, um segundo esquema no qual explora-se o fato de que a posição de

cada fluxo é determinada por duas componentes: velocidade (taxa de exibição de quadros) × tempo. Deste modo, admite-se a possibilidade de caracterizar os fluxos através de valores reais (não naturais). Porém, a aplicação direta desta alternativa não se mostra plausível dado que dois lados adjacentes precisam definir o mesmo peso para o vértice comum. Assim sendo, utiliza-se o artifício de dividir a posição de um dos fluxos (primeiro fluxo, por exemplo) pela componente velocidade e utilizar a componente tempo como fator de interseção com o fluxo seguinte. Deste modo, as componentes encontradas são utilizadas como fatores de interseção entre fluxos vizinhos até que se obtenha todos os pesos caracterizadores dos vértices do polígono. Para uma melhor compreensão deste esquema, considere o exemplo a seguir.

Exemplo 3.1 Sejam seis fluxos ao final de um intervalo *Snapshot* e suas posições dadas por 3338, 3010, 2908, 2316, 1650 e 503, determinam-se as componentes de cada fluxo como mostrado na Tabela 3.1. Deste modo, as componentes velocidade × tempo caracterizam polígonos como na Figura 3.11-a. Empregando o artifício descrito, cujos dados estão contidos na Tabela 3.2, tem-se o polígono da Figura 3.11-b.

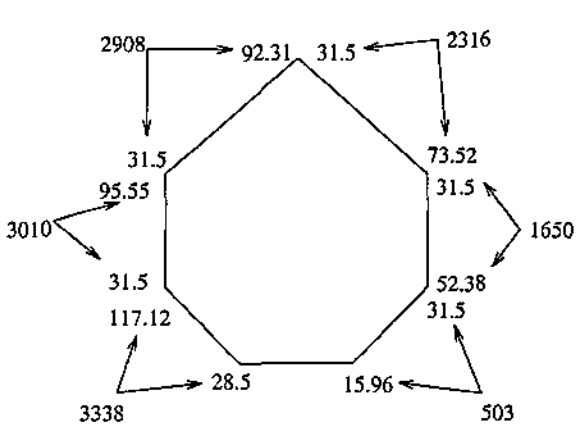
Fluxo	Posição	Velocidade × Tempo
1	3338	$S_{min}\times$ 117.12
2	3010	$S_{max}\times$ 95.55
3	2908	$S_{max}\times$ 92.31
4	2316	$S_{max}\times$ 73.52
5	1650	$S_{max}\times$ 52.38
6	503	$S_{max}\times$ 15.96

Tabela 3.1: Esquema de redução do problema de mesclagem de fluxos em triangulação de polígonos.

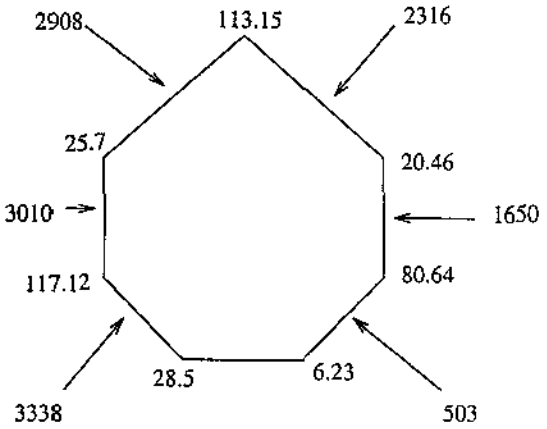
Fluxo	Posição	Velocidade × Tempo
1	3338	28.50 × 117.12
2	3010	117.12 × 25.70
3	2908	25.70 × 113.15
4	2316	113.15 × 20.46
5	1650	20.46 × 80.64
6	503	80.64 × 6.23

Tabela 3.2: Melhoramento do esquema de redução do problema de mesclagem de fluxos em triangulação de polígonos.

□



(a) Componentes tempo e velocidade



(b) Melhoramento da redução

Figura 3.11: Esquema de redução dos problemas: (a) polígono caracterizado pelas componentes velocidade x tempo e (b) melhoramento do esquema de redução.

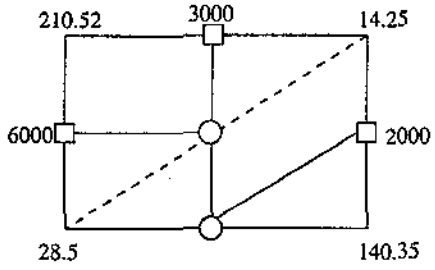
Porém, esta abordagem mostra-se inadequada, pois não há garantias de que a árvore gerada através deste método de redução seja a árvore ótima de mesclagens. Pode-se confirmar esta observação facilmente através de contra-exemplos. A Figura 3.12 ilustra uma situação em que se obtém uma árvore de mesclagem diferente da ótima para um conjunto de fluxos, cujas posições são 2000, 3000 e 6000, quando se utiliza o método de caracterização do polígono descrito.

Em virtude da inadequação das alternativas encontradas, busca-se, então, a solução do problema através da elaboração de uma heurística com complexidade inferior a do algoritmo de geração da árvore ótima de mesclagens. Na próxima seção, descreve-se a heurística proposta, avaliam-se requisitos de complexidade da heurística e qualidade das soluções geradas.

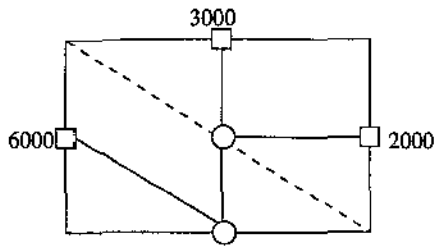
3.3.4 Heurística para Construção da Árvore de Mesclagens

A segunda abordagem investigada para a obtenção da árvore ótima de mesclagens com custo computacional inferior consiste na proposição de uma heurística. Observou-se que a árvore de mesclagens pode ser construída de forma *top down*, diferentemente do algoritmo de programação dinâmica que a constrói de forma *bottom up*. Com vistas à redução da complexidade, elaborou-se uma heurística, denominada *BuildTree*, que utiliza a estratégia de divisão e conquista na construção da árvore de mesclagens.

O princípio básico da heurística consiste em dividir sucessivamente o conjunto de



(a) Árvore de mesclagem



(b) Árvore ótima de mesclagem

Figura 3.12: Contra-exemplo da estratégia de redução do problema de mesclagem de fluxos em triangulação de polígonos.

fluxos a ser mesclado em duas partes computando-se os custos de cada subgrupo até que se obtenha a árvore de mesclagens. O critério de divisão utilizado leva em consideração os segmentos da árvore de mesclagem (número de quadros de vídeo dado pela posição de mesclagem) entre o primeiro fluxo e um fluxo intermediário (fluxo divisor) e entre este e o último fluxo. Em outras palavras, determina-se o fluxo que representa o ponto de divisão do conjunto em subárvores potencialmente ótimas. Deste modo, para um conjunto de fluxos $i, \ldots, j$, existe um fluxo k, com $i \leq k < j$, que minimiza o módulo da diferença dos comprimentos dos segmentos $P(i,k)$ e $P(k,j)$, dado por:

$$k^* = \text{argmin}_{i \leq k < j}\{|P(i,k) - P(k,j)|\} \tag{3.5}$$

Após a determinação de k^*, o conjunto original de fluxos $i, \ldots, j$ é particionado nos subconjuntos $i, \ldots, k^*$ e $k^*+1, \ldots, j$, se $P(i,k^*) < P(k^*,j)$, caso contrário, $i, \ldots, k^*-1$ e $k^*, \ldots, j$, reaplicando-se o critério de divisão em ambos. O algoritmo considera dois casos particulares em cujas entradas possuem somente dois e três fluxos. Nestes casos simplifica-se a análise para se obter melhor desempenho. No primeiro caso, cuja entrada é formada por 2 fluxos, o algoritmo calcula apenas a posição de mesclagem dos fluxos. No segundo, com entrada composta por 3 fluxos, comparam-se os segmentos $P(i,i+1)$ e $P(i+1,j)$ e descarta-se o de maior valor.

Deste modo, o custo para um conjunto de fluxos $i, \ldots, j$ é obtido através do somatório dos comprimentos dos segmentos da árvore de mesclagem construída, o qual representa o custo dos $n-1$ fluxos a menos do custo do fluxo resultante. A este valor deve-se adicionar o número de quadros do fluxo resultante.

No processo de determinação do fluxo divisor (Equação 3.5) são necessárias $O(n)$ operações. Pode-se utilizar o artifício descrito a seguir para se reduzir o número de

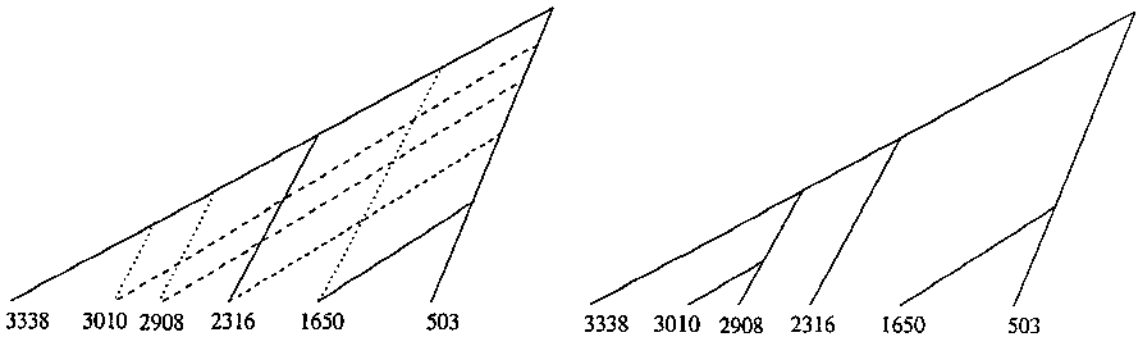
operações realizadas: *O fluxo divisor é o primeiro cujo segmento de mesclagem do fluxo inicial com o posterior do divisor é maior que o segmento de mesclagem do fluxo posterior do divisor com o último fluxo.* Para determinar o fluxo divisor, realiza-se uma busca no conjunto de fluxos partindo-se do primeiro em direção ao último; porém, no pior caso, $k = j - 1$, $O(n)$ operações são realizadas.

Teorema 3.1 *A heurística BuildTree possui complexidade* $O(n^2)$.

Prova: Apêndice A.

Para uma melhor compreensão da heurística, considere o exemplo a seguir.

Exemplo 3.2 Considera-se o mesmo cenário caracterizado no Exemplo 3.1, ou seja, seis fluxos são iniciados em um determinado intervalo e, ao final do mesmo, suas posições são: 3338, 3010, 2908, 2316, 1650 e 503. Sejam i, o primeiro fluxo; k, o fluxo divisor e j, o último fluxo do conjunto, particionam-se os seis fluxos nos conjuntos (1, ..., 4) e (5, ..., 6), ($i = 1$, $k = 4$, $j = 6$), dado que o quarto fluxo é o primeiro fluxo em que $P(i, k+1) \geq P(k+1, j)$, compondo-se a árvore da Figura 3.13-a. Em seguida, reaplica-se o algoritmo sobre os subconjuntos ($i = 1, \ldots, j = 4$) e ($i = 5$ e $j = 6$), em que os segmentos pontilhados representam os segmentos de mesclagem considerados pela heurística durante a determinação do fluxo divisor. A Figura 3.13-b ilustra a árvore de mesclagens final contruída pela heurística (neste caso, igual a árvore gerada pelo algoritmo de programação dinâmica).



(a) Primeiro Particionamento (b) Árvore Final

Figura 3.13: Construção da árvore de mesclagens pela heurística.

□

Em nossos experimentos a heurística foi implementada utilizando-se busca seqüencial do fluxo k através do conjunto $i, \ldots, j$; porém, substituindo-se busca seqüencial por

binária a heurística passa a ter complexidade $O(n \log n)$. No apêndice B, apresenta-se a heurística implementada em linguagem C.

Resultados Numéricos

A heurística descrita constitui-se numa aproximação da solução ótima, fazendo-se necessário avaliar a precisão em relação à esta. Com este objetivo, ambos algoritmos são aplicados ao final de intervalos *Snapshot* de modo a se comparar os custos das árvores de mesclagens geradas. Neste experimento as requisições foram modeladas através de um processo de Poisson e o intervalo médio entre requisições variou de 15 a 500 segundos.

A Figura 3.14 mostra as curvas do ganho médio percentual dos algoritmos de programação dinâmica e da heurística. Pode-se observar que a heurística *BuildTree* é bastante precisa quando comparada à solução ótima. Isto se deve ao fato de que, na maioria dos casos, a heurística obtém o resultado ótimo; de modo que, considerando-se casos individuais, a diferença percentual mostra-se muito pequena assumindo valores no máximo iguais a 5,6%. Porém, a diferença média amostral mostra-se desprezível, em alguns casos possuindo valores da ordem de 10^{-7} para taxas elevadas e nula para valores mais baixos de taxa, como pode-se constatar através da coincidências das curvas.

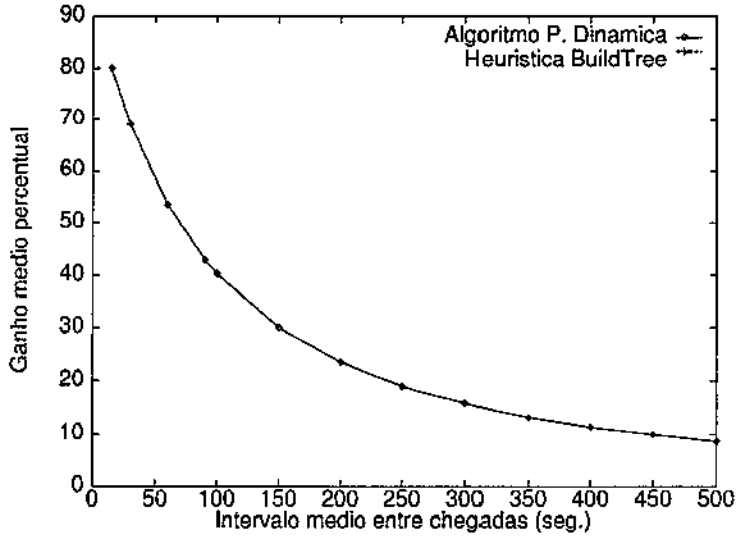


Figura 3.14: Comparação dos custos computados pela heurística e pelo algoritmo de programação dinâmica.

No gráfico da Figura 3.15 instâncias de tamanho 1 até 25 (fluxos), observadas em experimentos anteriores, foram fornecidas como entrada para a computação dos tempos de processamento de ambos os algoritmos. Observa-se em todos os casos que o tempo

de execução da heurística foi no máximo igual ao tempo do algoritmo de programação dinâmica, nunca excedendo-o. Este experimento foi realizado num microcomputador PC 486 Dx4-100MHz com Linux 2.0.0.

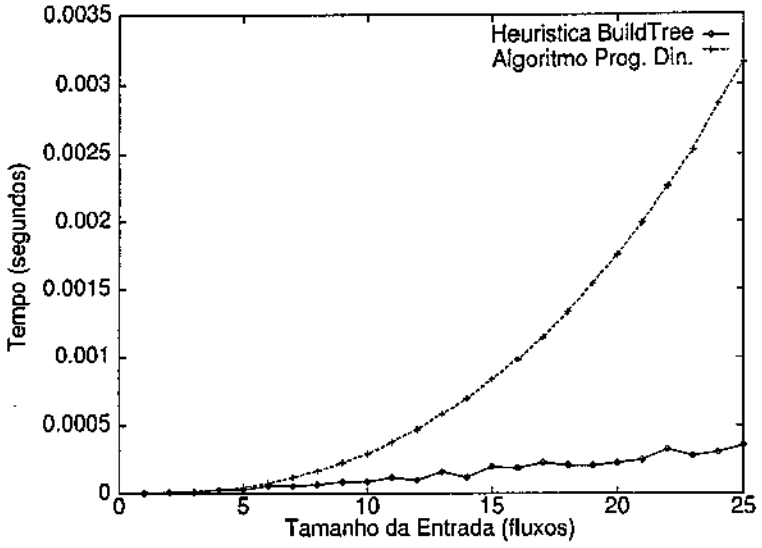


Figura 3.15: Comparação entre os tempos de execução da heurística e algoritmo de programação dinâmica.

3.4 Síntese do Capítulo

Políticas eficientes de *Piggybacking* precisam considerar diversos aspectos para uma efetiva redução da demanda de banda passante. As políticas S^2 e Híbrida, introduzidas neste capítulo, incorporam novas características, como o emprego de um segundo nível de otimização e a mesclagem de fluxos no interior do intervalo *Snapshot*, respectivamente; e, com isso, ampliam as vantagens da utilização de *Piggybacking* num sistema de VoD.

A determinação da estratégia ótima de mesclagem de um conjunto de fluxos de vídeo é realizada por um algoritmo de programação dinâmica, o qual possui custo computacional elevado. Duas estratégias para a redução desta complexidade foram investigados. Inicialmente, analisou-se a adaptação de algoritmos propostos para o problema da parentização ótima, em virtude de suas similaridades com o problema da mesclagem de fluxos. Na segunda abordagem, analisou-se a elaboração da uma heurística, resultando na proposição da heurística *BuildTree*.

Capítulo 4

Uma Nova Política de *Batching*

Batching constitui-se numa segunda alternativa para a ampliação do número de usuários suportados por um servidor, viabilizando o oferecimento de serviços de VoD em larga escala. Nesta técnica, requisições são retidas por intervalos de tempo com o objetivo de atendê-las com um único fluxo. *Batching* difere de *Piggybacking* basicamente em dois aspectos: *i*) no contexto de *Batching* o atendimento das requisições não ocorre de forma imediata e, *ii*) não há alterações nas taxas de exibição dos vídeos (característica que introduz complexidade adicional ao sistema).

Se por um lado *Batching* introduz um retardo inicial, por outro esta técnica constitui-se num mecanismo muito mais eficiente que *Piggybacking* para a redução da demanda de banda passante, uma vez que diversas requisições podem ser agrupadas antes do início da exibição do vídeo. *Batching* e *Piggybacking* possuem princípios diferentes, os quais não impedem a utilização simultânea das técnicas num servidor. De modo contrário, sua integração pode conduzir a um gerenciamento mais eficiente de recursos e ao balanceamento de fatores como o tempo de espera dos usuários e complexidade de implementação.

Este capítulo tem como objetivo apresentar um estudo direcionado a *Batching*. Inicialmente, introduzem-se alguns conceitos de otimização combinatória (seção 4.1). Na seção 4.2, algumas considerações preliminares sobre *Batching* são apresentadas. Parâmetros para avaliação de um sistema com *Batching* são apresentados na seção 4.3. Em seguida, introduz-se e avalia-se uma nova política de *Batching* (seção 4.4).

4.1 Conceitos de Otimização Combinatória

Nesta seção alguns conceitos de otimização combinatória, extraídos das referências [49, 50, 51], utilizados no decorrer do capítulo são introduzidos.

4.1.1 Problema de Programação Linear

Um **Problema de Programação Linear** — PPL é modelado através de expressões lineares. Para este problema objetiva-se a minimização (maximização) de uma função linear, denominada **Função Objetivo**, respeitando-se um sistema linear de igualdades ou desigualdades, ou seja, as **Restrições do Modelo**. As restrições determinam o **Conjunto de Soluções Viáveis**, de modo que a *melhor* das soluções, isto é, aquela que minimiza (maximiza) a função objetivo denomina-se **Solução Ótima**. Quando as variáveis de um PPL somente podem assumir valores inteiros, diz-se que este é um **PPL Inteiro**. Um **Problema de Programação Linear Inteiro 0-1** é uma classe particular de problemas de programação linear cujas variáveis do problema somente podem assumir valores 0 (zero) ou 1 (um).

A forma padrão de um PPL é apresentada a seguir:

Minimizar/Maximizar:

$$FO = c_1x_1 + c_2x_2 + c_3x_3 + \cdots + c_nx_n \tag{4.1}$$

sujeito a:

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_2 & + & a_{22}x_2 & + & a_{23}x_3 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & a_{m3}x_3 & + & \cdots & + & a_{mn}x_n & = & b_m \end{array} \tag{4.2}$$

$$x_1, x_2, \ldots, x_n \geq 0 \tag{4.3}$$

Na representação acima $x_1, x_2, \ldots, x_n$ são as **variáveis de decisão** não negativas e $c_1, c_2, \ldots, c_n$ são os **coeficientes de custo** associados às variáveis na função objetivo. Adicionalmente, $\sum_{j=1}^{n} a_{ij}x_j = b_1$ representa a i-ésima restrição com $i = 1, \ldots, m$ e a_{ij}, em que $i = 1, \ldots, m$ e $j = 1, \ldots, n$, são os **coeficientes das restrições**; finalmente, b_i, com $i = 1, \ldots, m$ são os **coeficientes do lado direito** da restrição. Um PPL também pode ser descrito utilizando-se a notação de matriz, ou seja:

Minimizar/Maximizar:

$$cx$$

sujeito a:

$$\mathbf{A}x = \mathbf{b}$$

$$x \geq 0$$

É interessante ressaltar que maximizar um problema é equivalente a minimizar o mesmo multiplicando-se os coeficientes de custo por -1, ou seja:

$$\text{Maximizar}\left(\sum_{j=1}^{n} c_j x_j\right) = \text{Minimizar}\left(\sum_{j=1}^{n} -c_j x_j\right) \tag{4.4}$$

4.1.2 Relaxação Linear

Rotinas computacionais utilizadas para a solução de um PPL Inteiro podem gerar soluções em que as variáveis podem assumir valores reais. Nestes casos, à solução obtida dá-se o nome de **Relaxação Linear**. Num problema de maximização (minimização), a relação linear pode definir um valor de solução maior (menor) que o valor da solução inteira.

4.1.3 Procedimentos de *Branch and Bound*

A solução de determinados problemas pode ser expressa através de uma n-upla $(x_1, \ldots, x_n)$ em que x_i é obtido a partir de um conjunto finito S_i. O problema a ser solucionado requer a determinação de uma dentre as possíveis n-uplas que satisfazem a uma função de critério (função objetivo). A busca da(s) solução(ões) pode ser facilitada quando organiza-se o espaço de soluções utilizando-se uma *organização de árvore*. Nestas representações, cada nó da árvore representa um **Estado do Problema** — EP. Todos os caminhos da raiz para outros nós definem o **Espaço de Estados** — EE do problema. **Estados da Solução** — ES são todos aqueles EP, S, para os quais o caminho da raiz pars os EP define um tupla no ES. **Estados de Resposta** — ER são todos aqueles ES, S, para os quais o caminho da raiz para S define uma tupla que compõe o conjunto das soluções do problema.

Cada elemento no espaço de soluções deve ser representado por pelo menos um nó, o qual pode ser particionado em subespaços de solução disjuntos. Espaços de estados organizados na forma de árvore podem assumir estrutura estática, quando a estrutura da árvore independe da instância do problema solucionado, ou dinâmica, caso contrário. Concebida a árvore de espaço de estados para um problema, sua solução consiste na geração sistemática dos EP, determinando quais destes são ES e, finalmente, quais ES são ER.

Os EP são gerados a partir da raiz prosseguindo-se com a geração dos demais nós. Ao nó cujos filhos ainda não foram gerados dá-se o nome de nó **vivo**. Um nó vivo cujos filhos estão sendo gerados chama-se **Nó-E**, ou seja, nó que está sendo expandido. Um **nó morto** é aquele que já teve todos os filhos gerados ou que sua expansão é interrompida. Aplicam-se **Funções de Limite** para a determinação de quais Nós-E devem ser eliminados antes da geração de todos os seus filhos.

Finalmente, pode-se definir procedimentos de *Branch and Bound* — B&B como toda classe de métodos de busca no EE nas quais todos os filhos de um Nó-E são gerados antes que qualquer outro nó possa tornar-se um Nó-E. Em virtude da interrupção da expansão de um *Nó-E*, isto é, de sua *poda*, procedimentos de B&B realizam uma enumeração implícita das soluções do problema.

4.2 Considerações Preliminares

Diversas são as políticas de *Batching* propostas na literatura e os critérios, por elas empregados, com o objetivo de realizar o agrupamento de requisições. Maximizar o efeito de *Batching* é uma atividade que envolve o balanceamento de requisitos opostos. Para se aumentar o número médio de usuários suportados por fluxo, torna-se necessária a utilização de janelas de *Batching* amplas, por outro lado, os usuários não estão dispostos a esperar por intervalos de tempo muito longos podendo, portanto, abandonar o sistema sem atendimento.

A modelagem do comportamento de abandono dos usuários permite a elaboração de esquemas baseados neste comportamento. Embora um dos objetivos de uma política de *Batching* seja a minimização da latência de atendimento, explorar o tempo de abandono estimado dos usuários constitui-se numa alternativa atrativa para a maximização do efeito de *Batching* em servidores de pequeno porte (em relação à população atendida).

A simulação de sistemas em que se modelam usuários totalmente tolerantes à latência, isto é, que não possuem a característica de abandono, não caracteriza com fidelidade o mundo real. Porém, modelar precisamente o comportamento de abandono de usuários é uma tarefa complexa, uma vez que a probabilidade de abandono aumenta com o tempo de espera. Este comportamento mais próximo à realidade varia de usuário para usuário, fato que torna difícil sua utilização numa abordagem de simulação.

Uma segunda alternativa consiste em analisar todo o conjunto de vídeos com requisições (considerando instantes futuros), e não somente aqueles cujas filas satisfaçam a um requisito de espera (pelo menos uma das requisições foi retida pelo intervalo de *Batching*) no momento da alocação de um fluxo. Deste modo, pode-se retardar a alocação de canais até que uma fila com um número maior de requisições esteja "habilitada" para receber um fluxo alocado. Porém, a quase totalidade das políticas não emprega este critério, ao con-

trário, determina-se o vídeo para o qual se vai alocar um novo fluxo de dados utilizando-se como base apenas o cenário composto pelos vídeos aptos no momento da alocação. Os esquemas *Batch* BML e *Batch* IML [36] constituem exceções, em que escalona-se o vídeo utilizando-se como critério as perdas (abandonos) estimadas até a próxima alocação de um fluxo.

Finalmente, pode-se dizer que o emprego de *Batching* somente é necessário em horários nos quais a carga no sistema ultrapassa sua capacidade de atendimento. Em horários de reduzida demanda, pode-se optar pela utilização exclusiva de *Piggybacking* para evitar que os usuários esperem desnecessariamente, dado que a probabilidade de outras requisições chegarem ao sistema é bastante baixa. Pode-se ainda oferecer serviços de VoD sem a utilização de *Batching* e *Piggybacking*, caso a carga seja muito baixa.

4.3 Parâmetros de Avaliação

Num sistema de VoD as chegadas de requisições compõem um processo $\mathcal{P}$, cujo tempo médio entre requisições é $1/\lambda$. Após conectar-se ao sistema, um usuário escolhe o vídeo que deseja assistir, k, com probabilidade p_k, $1 \leq k \leq V$, em que V é o número de programas armazenados no servidor, como pode ser visto na Figura 4.1. A requisição deste usuário é inserida na fila correspondente, onde permanece até que um fluxo seja alocado ou o usuário decida abandonar o sistema sem atendimento. Define-se o *tempo de abandono de um usuário* como o tempo máximo que este tolera na fila antes de deixar o sistema, denotado por uma variável aleatória T. O número máximo de fluxos que o sistema suporta é N_{max}, de modo que cada um destes fluxos pode apresentar qualquer um dos V vídeos armazenados. Considera-se um vídeo composto de L quadros.

Em um sistema com *Batching*, o principal parâmetro de avaliação constitui-se no número de usuários suportados. Os fatores seguintes podem ser utilizados na avaliação do modelo acima, constituindo-se em parâmetros secundários de avaliação:

- **Probabilidade de Abandono**: É definida como a razão entre o número de requisições que abandona o sistema pelo número total de requisições recebidas em um determinado intervalo de tempo. Deseja-se minimizar a probabilidade de abandono.

- **Tempo Médio de Atendimento** (*latência*): É definido como o tempo decorrido entre a chegada da requisição e o instante em que o usuário é atendido através da alocação de um fluxo. Objetiva-se a minimização da latência de atendimento.

- **Injustiça** (*Unfairness*): Seja $P_A(i)$ a probabilidade de abandono para o vídeo i e $\overline{P} = \sum_{i=1}^{V} P_A(i)/V$ a probabilidade média de abandono, define-se a injustiça no

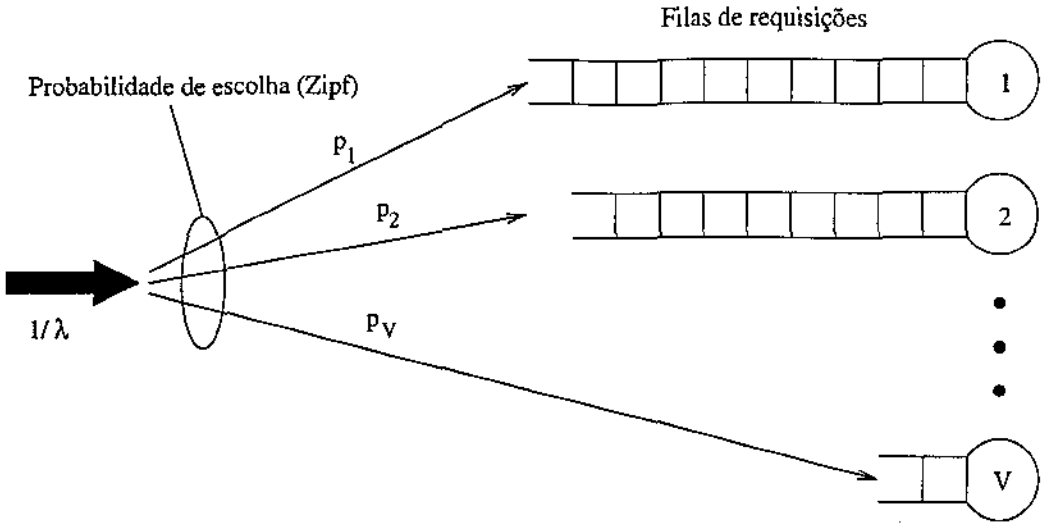


Figura 4.1: Processo de chegadas e escolha de vídeos conforme a distribuição de Zipf.

sistema como:

$$Unfairness = \sqrt{\frac{\sum_{i=1}^{V}(P_A(i) - \overline{P})^2}{V-1}}. \tag{4.5}$$

O sistema tende a ser justo quanto mais $P_A(i)$ aproxima-se de $\overline{P}$, para todo $1 \leq i \leq V$. A métrica de injustiça apresentada consiste da variância das probabilidades de abandono das requisições para cada uma das filas; de tal forma que, quanto maior a diferença entre as probabilidades de abandono entre os diversos vídeos, maior será a variância e, conseqüentemente, menor será a justiça no sistema. Em outras palavras, determinados vídeos são beneficiados em detrimento de outros, de modo que uma das causas possíveis deste beneficiamento relaciona-se à popularidade do vídeos.

4.4 A Política *Look-Ahead Optimize Batch*

Nesta seção, introduz-se um novo esquema de *Batching*, denominado *Look-Ahead Optimize Batch* — LAO-*Batch*. Neste esquema, utiliza-se algum conhecimento sobre o comportamento de abandono dos usuários com o objetivo de maximizar o efeito de *Batching*.

A política LAO-*Batch* assume que a janela de *Batching* possui duração determinada pelo tempo de abandono do usuário. Deste modo o intervalo de espera passa a ser dependente do usuário, não sendo mais um parâmetro global do sistema. A alocação de fluxos ocorre quando o tempo de abandono esgota-se, isto é, os *instantes de alocação* de fluxos

ocorrem quando um usuário está prestes a abandonar o sistema. Deste modo, os usuários devem ser inseridos nas filas obedecendo-se a ordenação temporal dos eventos de abandono, ou seja, a primeira requisição de uma fila é aquela cujo abandono ocorre primeiro, não obrigatoriamente é a primeira a chegar ao sistema. Esta característica impõe intervalos de espera distintos, nos quais espera-se a chegada de novas requisições, explorando-se ao máximo, o tempo de abandono dos usuários enquanto busca-se minimizar as desistências.

Seja F_k a fila de requisições do vídeo k, q_k o número de requisições atualmente contidas na fila F_k, com $q_k \geq 0$, e e_k um parâmetro (*status*) que assume valor $e_k = 1$ se $q_k > 0$ e $e_k = 0$ caso contrário, ou seja, e_k indica a existência de requisições na fila do vídeo k. Seja $S_R = \sum_{k=1}^{V} e_k$, o somatório dos *status* de todas as filas. A Equação 4.6 representa a seqüência de ordenação das requisições em suas respectivas filas.

$$S_R \text{ filas} \begin{cases} F_\alpha = \langle t_{\alpha 1} \quad t_{\alpha 2} \quad \cdots \quad t_{\alpha q_\alpha} \rangle \\ F_\beta = \langle t_{\beta 1} \quad t_{\beta 2} \quad \cdots \quad t_{\beta q_\beta} \rangle \\ F_\gamma = \langle t_{\gamma 1} \quad t_{\gamma 2} \quad \cdots \quad t_{\gamma q_\gamma} \rangle \\ \vdots \qquad \vdots \quad \vdots \quad \ddots \quad \vdots \\ F_\rho = \langle t_{\rho 1} \quad t_{\rho 2} \quad \cdots \quad t_{\rho q_\rho} \rangle \end{cases} \tag{4.6}$$

O critério de alocação de fluxos é o mesmo para todos os vídeos (independentemente de sua popularidade), o qual pode assumir dois modos de atuação. Seja N o número de fluxos disponíveis no sistema no instante de análise, pode-se definir o critério de alocação de fluxos como segue:

1) Se $S_R \leq N$ (o número de vídeos com requisições pendentes é menor ou igual ao número de canais atualmente disponíveis no sistema), aloca-se um fluxo para o vídeo cuja janela de *Batching* esgotou-se;

2) Caso contrário ($S_R > N$), define-se uma *Janela de Estudo* — JE ordenando-se os próximos eventos de alocação e liberação de fluxos que ocorrem em um determinado intervalo de tempo. Com base no cenário caracterizado pela JE, formula-se um problema de otimização que determina se o vídeo, cujo intervalo de espera esgotou-se, terá ou não direito à alocação de um fluxo.

Pode-se perceber que, enquanto existe um número de fluxos disponíveis relativamente alto (considera-se que existe um número alto de fluxos disponíveis sempre que o número destes supera o número de filas com requisições), o sistema conserva todas as requisições inseridas nas filas. O critério descrito no item 1) é imparcial para todos os vídeos, no sentido de que não privilegia os vídeos populares em detrimento dos não populares. Por outro lado, o esquema introduzido no item 2) beneficia os vídeos populares quando o número de fluxos disponíveis é relativamente baixo.

Nas próximas seções, descreve-se o problema de otimização modelado (seção 4.4.1) e alternativas para a sua simplificação (seção 4.4.2). Em seguida, o modelo de simulação utilizado para avaliação da política é apresentado (seção 4.4.3) e os resultados numéricos são analisados (seção 4.4.4).

4.4.1 O Modelo de Otimização

O problema de otimização apresentado nesta seção constitui-se num problema de programação linear inteiro 0-1 (definido na seção 4.1.1). Dado que a JE constitui a base para a formulação do PPL, sua definição será apresentada inicialmente. Em seguida, formula-se o problema de otimização.

Seja t_{ki} o instante de alocação (*scheduling time*) da i-ésima requisição da fila k, ou seja, o momento em que esta abandona o sistema, e $V_t = \langle t_{\alpha 1}, t_{\beta 1}, \ldots, t_{\rho 1} \rangle$, em que $t_{\alpha 1} \leq t_{\beta 1} \leq \cdots \leq t_{\rho 1}$, o vetor (ordenado cronologicamente) contendo os instantes de alocação das primeiras requisições de cada fila, definem-se os limites da janela de estudo, $\Im$, como:

$$\Im = [\min\{V_t\}, \max\{V_t\}] \tag{4.7}$$

Em outras palavras, a JE consiste num intervalo de tempo cujos extremos são definidos pelos primeiro e último instantes de alocação das primeiras requisições nas diversas filas, ou seja, seu ponto inicial é determinado pelo instante de alocação atual (intervalo de *Batching* atualmente esgotado) e o ponto final constitui-se no último instante de alocação dentre as primeiras requisições de cada fila, considerando-se o cenário caracterizado no momento da análise. Adicionalmente, a JE inclui todos os eventos que representam instantes de alocação que ocorrem entre $\min\{V_t\}$ e $\max\{V_t\}$ e também os eventos de liberação de fluxos, que ocorrem em virtude do final da exibição de um vídeo.

A definição da JE permite a realização de uma análise em que se pode avaliar se é mais atrativo para o sistema alocar um fluxo para o vídeo cujo intervalo de *Batching* esgota-se, ou se este fluxo deve permanecer disponível até que ocorra o instante de alocação de uma fila com um número maior de requisições. Neste caso, enquanto aguarda-se a ocorrência deste evento, novas requisições podem chegar ao sistema, favorecendo ao aumento do efeito de *Batching*.

Concluída a apresentação da janela de estudo, pode-se prosseguir com a introdução do PPL. Em sua formulação, além de algumas das expressões previamente definidas, novos termos são utilizados. Utiliza-se a seguinte notação:

$\Im$	Janela de estudo.
A_{ki}	Número de canais liberados desde o início da janela de estudo até o instante t_{ki}.

V	Número total de vídeos armazenados no servidor. Considera-se uma constante, pois assume-se que o número de vídeos no servidor é fixo.
N	Número de fluxos disponíveis no instante $\min\{V_t\}$.
q_k	Número de requisições atualmente contidas na fila F_k.
r_k	Custo para assistir ao vídeo k (*revenue*), portanto possui valor positivo. r_k pode ser considerado uma constante para o problema pois seu valor é definido previamente.
t_{ki}	Instante de abandono da i-ésima requisição do vídeo k.
x_{ki}	Variável que assume valor $x_{ki} = 1$ caso ocorra a alocação de um fluxo no instante t_{ki}, caso contrário, $x_{ki} = 0$.

Com o objetivo de maximizar o efeito de *Batching*, e com isso, o número de usuários suportados pelo sistema, define-se a seguinte função objetivo:

$$\max \sum_{k=1}^{V} \sum_{i=1}^{q_k} r_k \times (q_k - i + 1) \times x_{ki} \tag{4.8}$$

sujeito às seguintes restrições:

Restrição 1: Número de fluxos alocados por fila;

$$\sum_{i=1}^{q_k} x_{ki} \leq 1, \quad k = 1, \ldots, V \tag{4.9}$$

Restrição 2: Número total de fluxos alocáveis;

$$x_{ki} \leq N + A_{ki} - \sum_{k'=1}^{V} \sum_{j=1}^{q_{k'x}} x_{k'j}, \quad k' \neq k; t_{k'j} < t_{ki}; k = 1, \ldots, V; i = 1, \ldots, q_k \text{ e } t_{ki} \in \Im \tag{4.10}$$

Restrição 3: Integridade da solução;

$$x_{ki} = 0, \quad \forall t_{ki} \notin \Im, k = 1, \ldots, V \text{ e } i = 1, \ldots, q_k \tag{4.11}$$

Restrição 4: Definição de limites;

$$\begin{array}{rcll} 0 \leq & N & \leq N_{max} & \\ 0 \leq & A_{ki} & \leq N_{max} & k = 1, \ldots, V \text{ e } i = 1, \ldots, q_k \\ 0 \leq & q_k & \leq \infty & k = 1, \ldots, V \\ 0 \leq & x_{ki} & \leq 1 & k = 1, \ldots, V \text{ e } i = 1, \ldots, q_k \end{array} \tag{4.12}$$

Restrição 5: Definição de tipos;

$$\text{Inteiros} \begin{cases} N & \\ q_k & k = 1, \ldots, V \\ A_{ki} & k = 1, \ldots, V \quad i = 1, \ldots, q_k \\ x_{ki} & k = 1, \ldots, V \quad i = 1, \ldots, q_k \end{cases} \tag{4.13}$$

A formulação do problema pode ser interpretada como segue:

A Equação 4.8 (função objetivo) traduz o ganho obtido com a configuração de alocação de fluxos determinada pela solução do problema. Pode-se perceber que para a maximização da função objetivo, a alocação dos fluxos deve ocorrer nas primeiras requisições de cada fila. Porém, em determinadas situações, tal fato pode não ocorrer em virtude da limitação no número de canais disponíveis.

A restrição representada pela Equação 4.9 determina que, para o conjunto de requisições atualmente inseridas numa fila, o número máximo de canais alocados é 1. Isto é, se na solução de um determinado problema a variável $x_{ki} = 1$, a solução obtida indica que as requisições $1, \ldots, i-1$ não são atendidas (abandonando o sistema) e que no instante t_{ki} ocorrerá a alocação de um fluxo para o vídeo k suportando as requisições $i, \ldots, q_k$.

O limite superior no número total de fluxos alocáveis é imposto pela restrição da Equação 4.10. Num determinado instante (t_{ki}), aloca-se um fluxo se o montante utilizado até então ($\sum_{k'=1}^{V} \sum_{j=1}^{q_{k'}} x_{k'j}$) é inferior à soma dos fluxos inicialmente disponíveis (N) com os liberados até este instante (A_{ki}). A restrição apresentada não está em conformidade com a representação padrão de um PPL (Equação 4.2), deste modo a mesma restrição pode ser redefinida como:

$$x_{ki} + \sum_{k'=1}^{V} \sum_{j=1}^{q_{k'x}} x_{k'j} \leq N + A_{ki}, \quad k' \neq k; t_{k'j} < t_{ki}; k = 1, \ldots, V, i = 1, \ldots, q_k \text{ e } t_{ki} \in \Im$$

Na Equação 4.11 determina-se que todas as variáveis que não estão contidas na JE devem ter seus valores ajustados para zero. A existência desta restrição previne a ocorrência de inconsistências na solução de problemas. Uma inconsistência acontece quando somente parte das variáveis (requisições) de uma fila encontra-se contida na JE e seus valores são definidos em zero pela solução do problema, ou seja, estas requisições não são atendidas. Para cada fila em que tais características são observadas, a primeira[1] das variáveis que está fora da janela de estudo tem seu valor ajustado em 1. Este fato ocorre porque as variáveis que encontram-se fora da JE não são afetadas pela restrição da Equação 4.10 e, uma vez

[1]Qualquer uma das variáveis pode ter seu valor ajustado para 1, porém, em virtude da natureza de maximização do problema, a primeira destas variáveis fora da JE tem seu valor definido em 1, conforme comentário para a função objetivo.

que as variáveis contidas na JE têm valor zero, a restrição definida pela Equação 4.9 ainda não foi imposta em tais filas. Pode-se interpretar estas inconsistências como a alocação de canais inexistentes, uma vez que a solução do problema determina a alocação de todos os canais disponíveis para as requisições na JE.

As demais restrições (Equações 4.12 e 4.13) são simples, e basicamente definem os limites (*bounds*) e tipos das variáveis do problema, respectivamente.

A solução dos problemas modelados leva em consideração todos os eventos que compõem a JE (eventos de alocação e liberação de canais) e indica para quais vídeos devem ser alocados os fluxos disponíveis no sistema e em que instantes. Mais precisamente, o problema de otimização formulado indica se é atrativo (ou não) para o sistema alocar um fluxo para o vídeo cuja requisição representa o primeiro evento da JE, isto é, a requisição cujo tempo de abandono esgotou-se. Através da análise realizada, pode-se decidir retardar a alocação para o instante de alocação de uma fila com um número maior de requisições. Em outras palavras, se o instante de alocação atual (intervalo de *Batching* esgotado) pertence à fila do vídeo k, t_{k1}, aloca-se um fluxo se $x_{k1} = 1$, caso contrário, a alocação é retida e a primeira requisição fila F_k abandona o sistema, de modo que para cada requisição cujo intervalo de *Batching* esgota-se, define-se uma nova janela de estudo e modela-se um problema.

A estratégia descrita privilegia os vídeos populares. No entanto, pode-se perceber que este é um mecanismo adaptativo, ou seja, as soluções dos problemas modelados tornam-se mais seletivas (favoráveis) em relação aos vídeos populares à medida em que o número de fluxos disponíveis decresce. No caso extremo, a alocação do único fluxo disponível é retida até o evento de alocação da primeira requisição da maior fila.

A JE é uma estrutura que claramente pode envolver todas as requisições que existem no sistema. No entanto, definir janelas de estudo muito amplas implica na delimitação de um intervalo de tempo muito extenso. Quanto maior o intervalo considerado, maior é o tamanho do problema modelado e, conseqüentemente, maior a carga de processamento no servidor. Considerou-se satisfatória a análise obtida com a definição da JE apresentada, uma vez que a mesma permite a inclusão de todas as filas com requisições na análise realizada.

Para uma melhor compreensão do esquema proposto, considere o exemplo a seguir.

Exemplo 4.1 Suponha que num determinado instante existe apenas um fluxo disponível ($N = 1$), cinco filas possuem requisições pendentes ($S_R = 5$), de modo que o número de requisições em cada uma destas filas é $q_1 = 5$, $q_2 = 3$, $q_3 = 2$, $q_4 = 2$ e $q_5 = 1$. Suponha também que $V_t = \langle t_{51}, t_{11}, t_{31}, t_{21}, t_{41} \rangle$ e que os eventos t_{12}, t_{13}, t_{22} e t_{32} ocorrem entre t_{51} e t_{41}. Finalmente, considere que ocorrem duas liberações de fluxos e que todos estes eventos que compõem a JE estão ordenados como na Figura 4.2. Deve-se ressaltar que o índice utilizado para as filas foi ordenado de forma ascendente apenas para efeito

do exemplo, não significando que em problemas reais a ordem seja exatamente como a ilustrada. Algumas filas podem não possuir requisições por terem sido atendidas a pouco tempo ou simplesmente por não ter ocorrido a chegada de requisições.

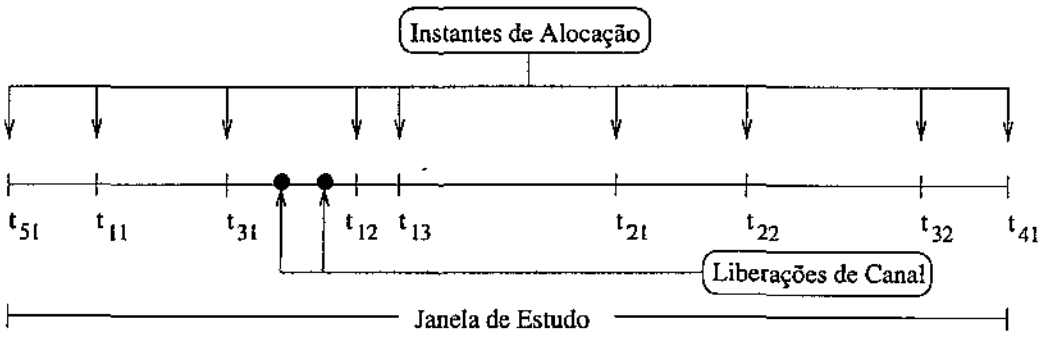


Figura 4.2: Exemplo de construção de uma JE.

Com base no cenário caracterizado, formula-se o seguinte PPL, no qual cada restrição está rotulada como R_i, em que i é o número da restrição no problema:

Maximizar:

$$5x_{11} + 4x_{12} + 3x_{13} + 2x_{14} + 1x_{15} + 3x_{21} + 2x_{22} + 1x_{23} + 2x_{31} + 1x_{32} + 2x_{41} + 1x_{42} + 1x_{51}$$

sujeito a:

$$\begin{aligned}
R_1 &: x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \leq 1 \\
R_2 &: x_{21} + x_{22} + x_{23} \leq 1 \\
R_3 &: x_{31} + x_{32} \leq 1 \\
R_4 &: x_{41} + x_{42} \leq 1 \\
R_5 &: x_{51} \leq 1
\end{aligned}$$

$$\begin{aligned}
R_6 &: x_{51} \leq 1 \\
R_7 &: x_{11} + x_{51} \leq 1 \\
R_8 &: x_{31} + x_{51} + x_{11} \leq 1 \\
R_9 &: x_{12} + x_{51} + x_{31} \leq 3 \\
R_{10} &: x_{13} + x_{51} + x_{31} \leq 3 \\
R_{11} &: x_{21} + x_{51} + x_{11} + x_{31} + x_{12} + x_{13} \leq 3 \\
R_{12} &: x_{22} + x_{51} + x_{11} + x_{31} + x_{12} + x_{13} \leq 3 \\
R_{13} &: x_{32} + x_{51} + x_{11} + x_{12} + x_{13} + x_{21} + x_{22} \leq 3 \\
R_{14} &: x_{41} + x_{51} + x_{11} + x_{31} + x_{12} + x_{13} + x_{21} + x_{22} + x_{32} \leq 3
\end{aligned}$$

Limites:

$$
\begin{array}{lrcl}
R_{15}: & 0 \leq & x_{11} & \leq 1 \\
R_{16}: & 0 \leq & x_{12} & \leq 1 \\
R_{17}: & 0 \leq & x_{13} & \leq 1 \\
R_{18}: & 0 \leq & x_{14} & \leq 0 \\
R_{19}: & 0 \leq & x_{15} & \leq 0 \\
R_{20}: & 0 \leq & x_{21} & \leq 1 \\
R_{21}: & 0 \leq & x_{22} & \leq 1 \\
R_{22}: & 0 \leq & x_{23} & \leq 0 \\
R_{23}: & 0 \leq & x_{31} & \leq 1 \\
R_{24}: & 0 \leq & x_{32} & \leq 1 \\
R_{25}: & 0 \leq & x_{41} & \leq 1 \\
R_{26}: & 0 \leq & x_{42} & \leq 0 \\
R_{27}: & 0 \leq & x_{51} & \leq 1
\end{array}
$$

A solução para o problema formulado acima é:

$$
\begin{array}{rcl}
x_{11} & = & 1 \\
x_{12} & = & 0 \\
x_{13} & = & 0 \\
x_{14} & = & 0 \\
x_{15} & = & 0 \\
x_{21} & = & 1 \\
x_{22} & = & 0 \\
x_{23} & = & 0 \\
x_{31} & = & 0 \\
x_{32} & = & 0 \\
x_{41} & = & 1 \\
x_{42} & = & 0 \\
x_{51} & = & 0
\end{array}
$$

□

4.4.2 Simplificações do Modelo

Nesta seção, são feitos alguns comentários sobre como simplificar o esquema proposto, uma vez que o modelo de otimização descrito permite a inclusão de restrições contendo informações redundantes para a solução dos problemas.

No Exemplo 4.1 existem 4 restrições que podem ser subtraídas do modelo sem prejuízo para a solução do problema, que são R_5, R_6, R_9 e R_{10}. Nota-se que existe uma

característica presente em todas elas, isto é, o número de variáveis com coeficientes não nulos nestas restrições é menor ou igual ao coeficiente do lado direito da restrição. Deste modo, estas restrições podem ser eliminadas da formulação do problema pois as demais introduzem as mesmas informações além de outras necessárias à solução do problema, isto é, envolvem todas as variáveis contidas nas restrições elimináveis. Em síntese, quaisquer restrições da forma:

$$\underbrace{x_\alpha + x_\beta + \cdots + x_\psi}_{n \text{ variáveis}} \leq n \tag{4.14}$$

podem ser excluídas da formulação do problema. Isto decorre do simples fato de se tratar de um PPL 0-1. Deste modo, cada variável assume valor máximo $x_i = 1$ e o valor do somatório das variáveis na restrição é sempre menor ou igual ao coeficiente do lado direito.

Pode-se perceber também que as restrições R_5 e R_6 são exatamente iguais, embora pertençam às classes de requisições definidas pelas Equações 4.9 e 4.10, respectivamente. Isto se deve ao fato de a requisição do vídeo 5 ser a única em sua fila e ser a primeira da JE. Adicionalmente, as restrições R_5 e R_6 enquadram-se na situação descrita no parágrafo anterior. Estas restrições podem ser substituídas por R_{27}, na qual determinam-se os limites da variável x_{51}.

4.4.3 Modelo de Simulação

Para a avaliação da política proposta utiliza-se o seguinte modelo de simulação: assume-se que as chegadas das requisições são modeladas por um processo de Poisson, cujo intervalo médio entre requisições é exponencialmente distribuído com média $1/\lambda$. A probabilidade de escolha de um vídeo é caracterizada através da distribuição **Zipf** com parâmetro $\theta = 0{,}271$ [38]. Nesta distribuição, a probabilidade de escolha por um vídeo k é dada por $p_k = \frac{c}{k^{(1-\theta)}}$, em que θ é o parâmetro para a distribuição e c é a constante de normalização, dada por:

$$c = \frac{1}{\sum_{k=1}^{V} \frac{1}{k^{(1-\theta)}}}. \tag{4.15}$$

No gráfico da Figura 4.3 mostra-se a caracterização das diversas probabilidades de escolha dos vídeos conforme a distribuição Zipf.

O tempo de abandono dos usuários foi modelado conforme o esquema proposto por Philip Yu *et al.* [36]. Neste esquema, assume-se que cada usuário espera durante um intervalo de tempo cuja duração é definida pela distribuição Normal (*Gaussiana*) com média μ e variância σ^2, ou seja, $T \sim N(\mu, \sigma^2)$. Deste modo, considera-se que após T minutos o usuário abandona o sistema sumariamente.

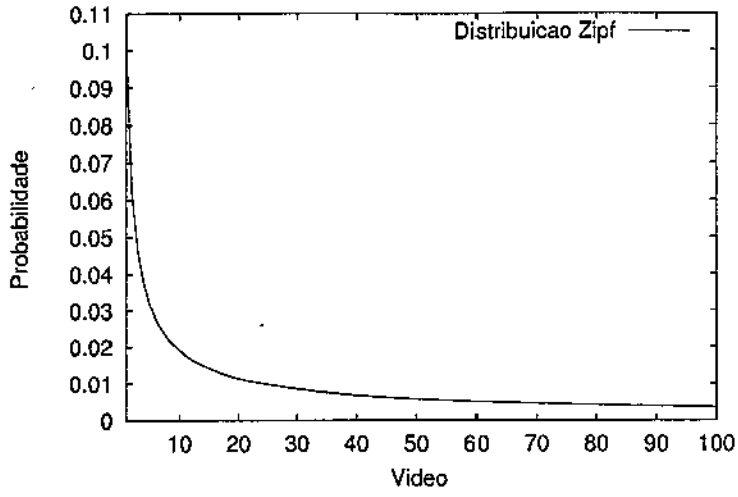


Figura 4.3: Probabilidade de escolha dos vídeos conforme a distribuição Zipf com parâmetro $\theta = 0{,}271$.

4.4.4 Resultados Numéricos

Os parâmetros utilizados na avaliação da política proposta são descritos a seguir. Assume-se que o número de vídeos armazenados no servidor é $V = 100$ e que a duração média de cada vídeo é de 2 horas (120 minutos). Duas taxas de chegada de requisições foram utilizadas, na primeira ocorrem 50 requisições por minuto e na segunda somente 10 requisições. Com isso objetiva-se observar o comportamento do sistema sob uma carga relativamente alta e uma carga moderada. A capacidade do servidor assume valores entre 100 e 600 fluxos e o tempo de abandono dos usuários foi modelado pela distribuição Normal com média $\mu = 10$ minutos e variância $\sigma^2 = 2{,}5$ minutos, $T \sim N(10,\ 2{,}5)$.

Para avaliar a política LAO-*Batch* foram desenvolvidos programas de simulação em linguagem C. Utilizou-se o método de *Box-Muller* [52, páginas 288–290] para a geração de números que obedecem à distribuição Normal. Para a solução dos PPLs modelados, utilizou-se o *software* de otimização *CPLEX® Linear Optimizer*[2] versão 3.0 e a *CPlex Callable Library*. Em alguns dos problemas de otimização, a solução obtida não é inteira, isto é, obtém-se a relaxação linear. Para situações como esta, foram implementados procedimentos de B&B. Nestes casos, realiza-se um *branching* sobre o nó cujo valor na solução obtida não é inteiro. Um *branching* consiste na geração de dois novos subproblemas, de modo que em um destes define-se valor 0 (zero) para o nó com valor não inteiro, no outro subproblema atribui-se valor 1 (um).

[2]CPLEX é marca registrada da *CPLEX Optimization, Inc.*

Nos experimentos realizados, comparou-se o esquema LAO-*Batch* com as políticas *Batch* MBQ, uma variação da política MQL original e S^2, que apresentou os melhores resultados dentre as políticas de *Piggybacking*. Para a política S^2, assume-se que se não existem fluxos disponíveis para o atendimento imediato de uma requisição, esta abandona o sistema sumariamente. O critério para a escolha de um vídeo para alocação na política MQL utilizada foi definido de forma semelhante ao primeiro caso da política LAO-*Batch*, isto é, as requisições são ordenadas conforme a ordem de abandono no sistema. Quando um intervalo de espera esgota-se, aloca-se um fluxo se houver algum disponível. Caso contrário, quando um fluxo torna-se disponível, seleciona-se a fila de maior tamanho dentre aquelas em que pelo menos um evento de abandono ocorreu. A política *Batch* MBQ possui a característica básica de definir um intervalo de espera mínimo, ω, através de um parâmetro de confiança, $\zeta = 0{,}75$, para a distribuição normal como mostrado na Equação 4.16:

$$\omega = \min\{s : \text{Prob}(T > s) \leq 1 - \zeta\} \tag{4.16}$$

São considerados eventos de alocação as chegadas de requisições, liberações de fluxos e eventos de alocação anteriormente retidos. Quando qualquer um destes eventos ocorre, selecionam-se dentre as filas com requisições aquelas que satisfazem ao requisito mínimo de espera (tempo de espera maior ou igual a ω minutos). Neste conjunto escolhe-se a fila de maior tamanho. Caso nenhuma fila satisfaça ao requisito de espera mínimo, um canal permanece livre.

Nas Figuras 4.4, 4.5 e 4.6 analisam-se fatores como o número de usuários suportados, probabilidade de abandono e injustiça no sistema em função da capacidade do servidor, respectivamente, considerando-se uma taxa de chegadas de 50 requisições por minuto.

No gráfico da Figura 4.4 pode-se analisar o desempenho das políticas quanto ao número de usuários suportados. Observa-se que a política LAO-*Batch* apresenta desempenho superior ao das demais políticas consideradas. Esta característica se deve, basicamente, ao fato da política ter a propriedade de "olhar" para momentos futuros enquanto realiza a análise para a alocação de um fluxo em um determinado instante. A variação da política MQL descrita apresenta desempenho semelhante, porém, dado que não tem a capacidade de avaliar um cenário futuro, seu desempenho também mostrou-se inferior ao da política LAO-*Batch*. Em outras palavras, quando o servidor encontra-se saturado, podem haver várias filas aptas para alocação. No entanto, quando um fluxo torna-se disponível sua realocação ocorre de forma imediata para tais filas, não havendo a retenção deste fluxo até um evento de alocação de uma fila com um número maior de requisições. Quanto à política *Batch* MBQ, o requisito de espera determinado, Equação 4.16, impõe um tempo mínimo para todas as filas, podendo haver desistências sumárias caso o tempo de abandono de um usuário seja menor que ω. Esta característica explica seu desempenho inferior ao

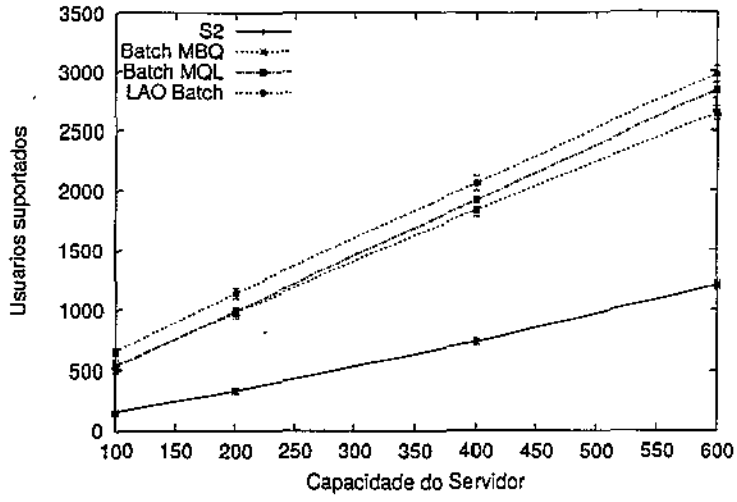


Figura 4.4: Usuários suportados × capacidade do servidor com taxa de 50 requisições por minuto.

das políticas LAO-*Batch* e MQL modificada.

Pode-se observar também que todas as políticas de *Batching* apresentam desempenho superior aos da política S^2. Os dados para a política S^2 podem parecer contraditórios aos do gráfico da Figura 3.2. No entanto, os experimentos diferem fundamentalmente no aspecto do número de vídeos e capacidades de servidor consideradas. No primeiro caso, analisa-se apenas um vídeo considerando-se que há fluxos suficientes para o atendimento de todas as suas requisições. Na análise atual aplica-se a política S^2 sobre um conjunto V de vídeos em um servidor com capacidade limitada, N_{max} fluxos. Deste modo, percebe-se que as políticas de *Piggybacking* são mais apropriadas para servidores de grande porte. O desempenho da política S^2 pode ser melhorado através de uma variação maior nas taxas de exibição dos fluxos. Porém, esta alternativa não é viável pois alterações superiores a 5% são captadas pela percepção humana [33].

A probabilidade de abandono dos usuários é apresentrada no gráfico da Figura 4.5. Observa-se que a política S^2 apresenta índice de abandonos superior aqueles obtidos pelas políticas de *Batching*. Tal fato decorre das características apresentadas durante a análise da figura anterior. Por outro lado, a política LAO-*Batch* possui os índices de probabilidade de abandono mais baixos dentre todas consideradas. Esta constatação é uma conseqüência direta da maximização do efeito de *Batching* no sistema. O comportamento das demais políticas, *Batch* MBQ e MQL modificada são próximos, porém, ocorre um diferenciamento entre as curvas à medida em que são levados em consideração servidores de maior porte. Em outras palavras, a política MQL busca conservar ao máximo as requisições definindo

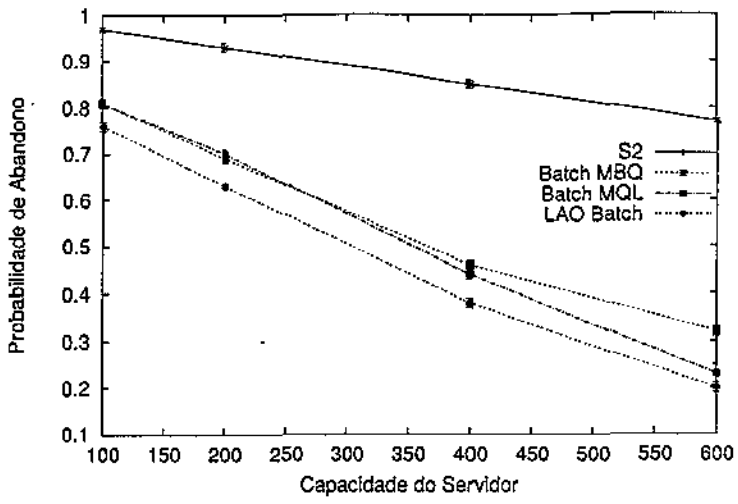


Figura 4.5: Probabilidade de abandono × capacidade do servidor com taxa de 50 requisições por minuto.

um intervalo de espera no máximo igual ao tempo de abandono do usuário. Deste modo, quando o servidor suporta maior número de fluxos, a política MQL conduz a um número menor de abandonos. Esta característica não é compartilhada pela política *Batch* MBQ devido à definição de um requisito de espera mínimo para todas as filas que pode ser maior que o tempo de abandono dos usuários.

O grau de injustiça é considerado no gráfico da Figura 4.6. Observa-se que a política S^2 proporciona os menores índices injustiça entre os vídeos. Isto se deve à alocação de fluxos obedecer a seqüência de chegada das requisições, ou seja, emprega-se o esquema FCFS para a alocação de fluxos, não havendo privilégios para qualquer um dos vídeos. Pode-se perceber um nível crescente de injustiça para a política S^2 com o aumento da capacidade do servidor. Isto ocorre porque, à medida em que aumenta-se o número de fluxos num servidor, aloca-se uma porcentagem maior destes para os vídeos populares devido à maior freqüência de chegadas de suas requisições, as quais têm maior chance de encontrar fluxos disponíveis para alocação.

Por outro lado, política LAO-*Batch* privilegia os vídeos populares em detrimento dos não populares apresentando, deste modo, os maiores níveis de injustiça; quanto menor a capacidade do servidor, mais cedo será explorada a diferença de popularidade dos vídeos. A política MQL somente privilegia os vídeos populares quando o servidor encontra-se saturado, ou seja, somente seleciona a maior fila quando o servidor encontra-se saturado e ocorre uma liberação de fluxo. Por isso, seus índices de injustiça encontram-se abaixo daqueles para as políticas LAO-*Batch* e *Batch* MBQ. A política *Batch* MBQ apresenta

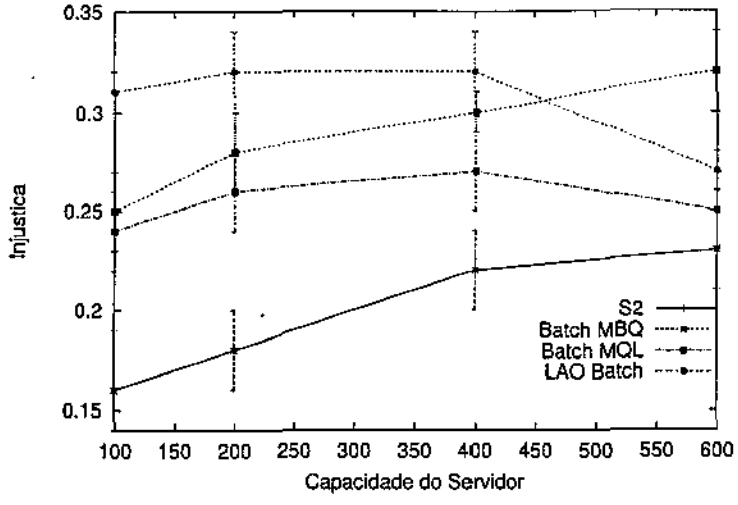


Figura 4.6: Injustiça × capacidade do servidor com taxa de 50 requisições por minuto.

índices crescentes de injustiça em virtude do requisito de espera mínimo. Quando se considera um servidor de pequeno porte, a probabilidade de abandono para as diversas filas é relativamente nivelada. Porém, para servidores de maior porte, os vídeos populares têm um número maior de usuários suportados, fato que não acontece com os vídeos não populares devido ao reduzido número de requisições, de modo que muitas destas requisições abandonam o sistema em virtude do requisito mínimo de espera.

Nas Figuras 4.7–4.8 analisam-se os mesmos experimentos das Figuras 4.5–4.6 considerando-se uma taxa de chegadas de 10 requisições por minuto.

Na Figura 4.7 são mostradas as curvas de probabilidade de abandono. Observa-se que a política LAO-*Batch* apresenta os níveis de abandono mais baixos, isso é possível através da análise realizada sobre a JE. A política *Batch* MBQ reduz o nível de perdas de requisições com o aumento da capacidade do servidor. No entanto, mesmo com servidores de maior porte, a determinação do requisito mínimo de espera impede uma redução maior no número de abandonos. Percebe-se que para servidores com capacidade de 600 fluxos, o nível de perdas mantém-se alto, inclusive maior que a taxa de perdas da política S^2, que para as demais capacidades de servidor consideradas apresenta os maiores índices de abandono.

Através do gráfico da Figura 4.8 pode-se analisar o nível de injustiça no sistema com a taxa de 10 requisições/minuto. Como conseqüência direta da queda da probabilidade de abandono, a política LAO-*Batch* transforma-se na mais justa de todas as políticas. De modo geral, a política S^2 novamente mostra-se menos injusta que as demais, com exceção das políticas LAO-*Batch* e MQL modificada, quando se considera um servidor

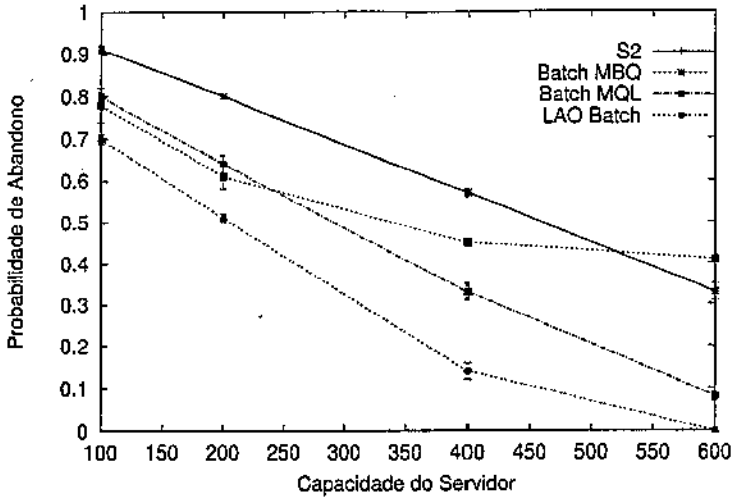


Figura 4.7: Probabilidade de abandono × capacidade do servidor com taxa de 10 requisições por minuto.

com capacidade de 600 fluxos. A política *Batch* MBQ é a que mais favorece aos vídeos populares, mais uma vez devido ao intervalo de espera definido, que sob baixa taxa de chegadas, conduz a um grande número de abandonos, em termos percentuais, entre os vídeos não populares.

4.5 Síntese do Capítulo

Batching constitui-se numa estratégia atrativa para o oferecimento de serviços de VoD em larga escala, pois permite aos servidores suportar um número de usuários maior que sua capacidade nominal. Sua característica básica é a retenção das requisições por intervalos de tempo com o objetivo de acumular o máximo número destas e atendê-las com um único fluxo de vídeo.

Neste capítulo, propôs-se uma nova política de *Batching*, LAO-*Batch*, que busca maximizar o efeito de *Batching* retardando ao máximo a alocação de fluxos disponíveis. A política LAO-*Batch* define a janela de estudo, que consiste num intervalo que reúne eventos de alocação e liberação de fluxos. Baseado na janela de estudo, formula-se um problema de otimização que atua como mecanismo de decisão para a alocação de fluxos indicando o instante mais apropriado para a alocação destes. Deste modo, busca-se maximizar o número médio de usuários suportados por fluxo, otimizando-se a utilização dos recursos do sistema. Através de experimentos de simulação, observou-se que política LAO-*Batch*

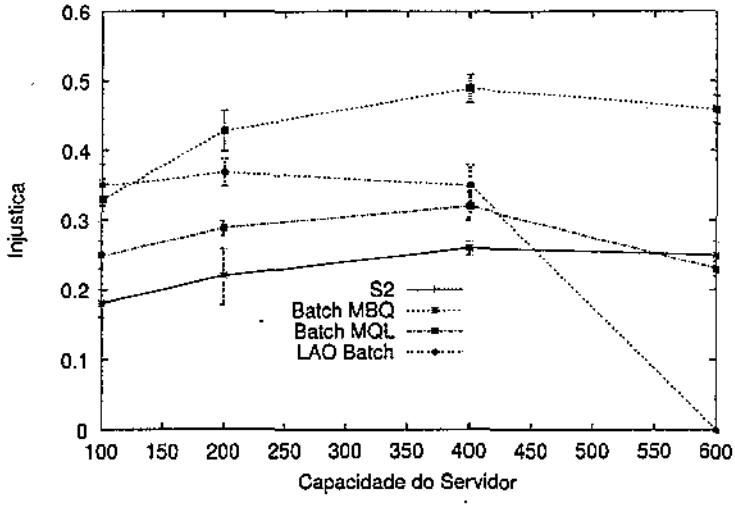


Figura 4.8: Injustiça × capacidade do servidor com taxa de 10 requisições por minuto.

apresenta resultados superiores aos de outras políticas de *Batching* e aos resultados da política de *Piggybacking* S^2.

Capítulo 5

Abordagem Integrada de *Batching* e *Piggybacking*

Analisou-se, em capítulos anteriores, como políticas de *Piggybacking* (capítulo 3) e de *Batching* (capítulo 4) podem ser aplicadas em um sistema de VoD. Ambas as técnicas foram abordadas em contextos isolados, levando-se em consideração apenas suas características próprias. No presente capítulo, investiga-se a integração de *Batching* com *Piggybacking* examinando-se quais fatores influenciam seu emprego conjunto, e como esta integração atua sobre aspectos como o número de usuários suportados, probabilidade de abandono e justiça do sistema.

O restante deste capítulo está organizado como segue. Na seção 5.1 apresentam-se algumas considerações preliminares sobre a integração das técnicas. A integração propriamente dita é apresentada na seção 5.2. Finalmente, na seção 5.3 apresentam-se os comentários finais do capítulo.

5.1 Considerações Preliminares

As técnicas de *Batching* e *Piggybacking* baseiam-se no pressuposto de que um conjunto de requisições pelos vídeos populares chega ao sistema num intervalo de tempo relativamente curto e ambas objetivam atender a estas requisições com um único fluxo. No entanto, estas são as únicas interseções existentes entre as técnicas.

Como observado no capítulo 3, os ganhos obtidos com *Piggybacking* são expandidos quanto mais próximas ocorrem as alocações de fluxos para um determinado vídeo. Por outro lado, a maximização dos ganhos oriundos do emprego de *Batching* ocorre aumentando-se a janela ou intervalo de *Batching*. Uma sistemática envolvendo ambas as técnicas pode ser concebida através do balanceamento destes fatores que, embora pareçam, não são mutuamente exclusivos.

A flexibilidade introduzida com a integração destas técnicas pode trazer diversas conseqüências para o sistema. Por um lado, pode-se optar por reduzir o tempo de espera dos usuários alocando-se fluxos para os vídeos populares mais freqüentemente (do que ocorre com a aplicação exclusiva de *Batching*) para sincronizá-los com *Piggybacking*. Por outro lado, esta redução do tempo de espera pode implicar na diminuição dos ganhos obtidos com *Batching*. Um outro aspecto, que decorre da presença de *Batching*, consiste no fato de que o número de fluxos a ser mesclado torna-se menor, reduzindo-se a complexidade introduzida no sistema com *Piggybacking*. Por fim, pode-se dizer que as políticas de *Batching* e *Piggybacking* utilizadas na abordagem integrada também podem influir no gerenciamento dos recursos do sistema, isto é, as políticas empregadas na integração podem conduzir à utilização mais (menos) eficiente dos recursos disponíveis.

O estudo integrado de *Batching* e *Piggybacking* requer a combinação de diversas das políticas mencionadas na seção 2.5, porém este trabalho envolve apenas um subconjunto deste.

5.2 Integrando *Batching* com *Piggybacking*

Um esquema integrado de *Batching* e *Piggybacking* sugere naturalmente uma estrutura composta por dois níveis, na qual a base consiste da estrutura que realiza o controle da estratégia de alocação e gerenciamento das filas de requisições, de modo que estas funções podem ser desempenhadas por uma política de *Batching*; enquanto que no segundo nível, implementa-se o controle dos fluxos em exibição no sistema, de tal forma que estas funções podem ser desempenhadas por uma política de *Piggybacking*. O esquema pode envolver uma "interface" para o intercâmbio de informações entre os níveis, Figura 5.1.

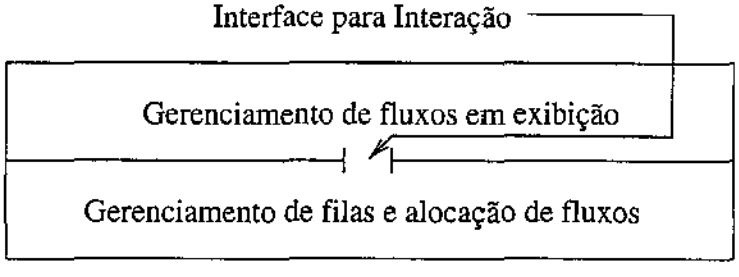


Figura 5.1: Esquema genérico para integração de *Batching* e *Piggybacking*.

Nas próximas seções, detalha-se a estratégia de integração investigada (seção 5.2.1) e os resultados numéricos obtidos com a implementação conjunta das técnicas (seção 5.2.2).

5.2.1 Mecanismos de Interação entre as Políticas de *Batching* e *Piggybacking*

Como mencionado anteriormente, as políticas de *Piggybacking* possuem a limitação de somente poder superpor fluxos se estes estiverem distantes no máximo W_m quadros. Mais precisamente, quando um fluxo é iniciado, este somente pode ser mesclado com um fluxo que está adiante se o mesmo ainda estiver dentro da janela máxima de mesclagem. Esta restrição deve ser especialmente observada, uma vez que na presença de *Batching*, fluxos de um mesmo vídeo normalmente apresentam grande distância entre si. Uma característica, que de certo modo agrava esta restrição, consiste no fato de que algumas políticas de *Batching*, através de sua própria definição, impõem intervalos mínimos entre a alocação de fluxos, como por exemplo, *Batch* MBQ. Neste esquema, os fluxos de um mesmo vídeo distam de pelo menos ω minutos. Com a determinação de intervalos muito amplos, a possibilidade de mesclagem de fluxos praticamente inexiste, ou seja, mesmo com a presença de políticas de *Piggybacking* tem-se a utilização quase que exclusiva de *Batching*.

Portanto, a determinação das políticas que serão utilizadas constitui-se num fator importante. Neste sentido a política LAO-*Batch* apresenta-se como uma opção atrativa, e sua escolha deve-se a dois motivos: *i*) seu desempenho superior em relação às demais políticas analisadas, e *ii*) inexistência de intervalos de duração fixa entre eventos de alocação de fluxo. Isto se deve ao fato do intervalo de *Batching* ser definido como o tempo de abandono dos usuários. Quanto às políticas de *Piggybacking*, foram selecionadas mesclagem simples e par-ímpar. A escolha destas políticas foi influenciada principalmente pelo escopo de atuação, isto é, a janela máxima de mesclagem, em virtude do distanciamento entre fluxos de um mesmo vídeo ocasionado pelo emprego de *Batching*.

As políticas mesclagem simples, Algoritmo *Snapshot* e S^2 não foram utilizadas porque pressupõem a determinação de uma taxa de chegadas para o cálculo da janela ótima de mesclagem. Neste contexto, pode-se determinar a taxa de requisição para um vídeo k como a taxa do processo global ($\mathcal{P}$) de chegadas no sistema multiplicada pela probabilidade de Zipf para o mesmo vídeo, $\lambda \times p_k$. Uma vez que as requisições são retidas durante a janela de *Batching* e esta não possui dimensionamento previsível, caracterizar a taxa de "alocação"[1] de fluxos para cada vídeo isoladamente torna-se uma atividade complexa. Adicionalmente, a política gulosa não foi considerada em virtude do distanciamento entre fluxos, fato que impede o emparelhamento sucessivo dos mesmos.

Em virtude dos fatores apresentados, o tempo médio de abandono torna-se o mecanismo principal de interação entre as técnicas. Em outras palavras, se o comportamento

[1] Utiliza-se o termo taxa de "alocação" porque considerando-se $\lambda \times p_k$ diretamente supõe-se que ocorre uma alocação para cada requisição que chega ao sistema, isto é, o emprego isolado de *Piggybacking*. Porém, a janela de *Batching* definida descaracteriza uma taxa de chegadas.

de abandono dos usuários é modelado definindo-se um tempo médio de abandono muito longo, fato que implica num maior distanciamento entre os eventos de alocação de fluxos, privilegia-se a política de *Batching* em detrimento das de *Piggybacking*; no outro extremo (tempo médio curto), beneficiam-se as políticas de *Piggybacking* decrescendo o ganho obtido com *Batching*. No primeiro caso, o sistema pode reduzir o tempo de atendimento e, deste modo, realizar um balanceamento entre as técnicas. Quanto às políticas de *Piggybacking*, estas somente podem direcionar seu funcionamento através das posições (quadros apresentados) dos fluxos em andamento, uma vez que a taxa de "alocação" é modificada pela política de *Batching*.

Durante a definição da política LAO-*Batch*, buscou-se examinar outros fatores que teriam algum tipo de influência sobre a interação das técnicas. Neste sentido, avaliou-se a estratégia de determinar o tempo de abandono de um usuário como o mínimo entre o tempo de abandono modelado e o tempo necessário para a mesclagem com o último fluxo (em exibição) do vídeo selecionado, caso exista. No entanto, esta abordagem não se mostrou adequada pois, para os vídeos populares, após a alocação de um fluxo, novas alocações tornam-se muito freqüentes em decorrência do tempo de mesclagem ser sempre muito baixo devido à chegada freqüente de requisições. Com isso, praticamente elimina-se a presença de *Batching* e, conseqüentemente, reduz-se o ganho proporcionado pela mesma.

5.2.2 Resultados Numéricos

A implementação conjunta de *Batching* e *Piggybacking* foi avaliada através de simulação. Basicamente avalia-se a influência da integração das técnicas sobre os aspectos de número de usuários suportados, probabilidade de abandono e justiça. Considerou-se uma taxa de 50 requisições por minuto em um servidor cuja capacidade variou deste 100 até 600 fluxos. O comportamento de abandono dos usuários foi modelado através da distribuição Normal. Inicialmente define-se o tempo médio de abandono dos usuários em $\mu = 10$ minutos, com o objetivo de avaliar a introdução de *Piggybacking* num cenário outrora definido exclusivamente para *Batching*. Em seguida, reduz-se o tempo médio de abandono dos usuários para $\mu = 8$ minutos. Deste modo, busca-se avaliar o novo comportamento do sistema quando a alocação de fluxos ocorre com maior freqüência. Mais precisamente, pretende-se quantificar a compensação da redução do ganho obtido com *Batching* através de *Piggybacking*.

Compara-se o desempenho da integração das políticas com o da política LAO-*Batch* apenas, uma vez que através da análise realizada no capítulo 4, observou-se sua superioridade em relação à melhor política de *Piggybacking*, S^2, considerando-se o critério de número médio de usuários suportados. Portanto, conclui-se que se a integração das técnicas é vantajosa em relação ao emprego isolado de *Batching*, também o é quando

comparada com a implementação isolada de *Piggybacking*.

As Figuras 5.2–5.4 ilustram o comportamento do sistema quando considera-se o tempo médio de abandono de 10 minutos.

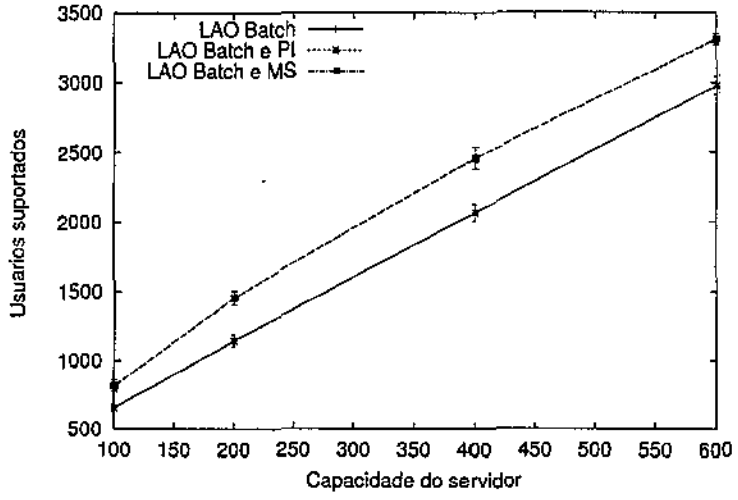


Figura 5.2: Usuários suportados × capacidade do servidor na presença de *Batching* e *Piggybacking*. $T \sim N(10,\ 2{,}5)$.

No gráfico da Figura 5.2 comparam-se as curvas do número de usuários suportados em função da capacidade do servidor para a política LAO-*Batch* isoladamente e de sua implementação conjunta com as políticas mesclagem simples e par-ímpar. Observa-se que com a integração das políticas, o número de usuários suportados é superior ao obtido com a utilização exclusiva de *Batching*. Isto se deve basicamente ao fato de que, com a introdução de *Piggybacking*, ocorrem liberações antecipadas de fluxos (antes do final da exibição dos vídeos). Nota-se também o comportamento idêntico nas implementações de LAO-*Batch* com as políticas mesclagem simples e par-ímpar, característica também presente nos gráficos das Figuras 5.3 e 5.4. A coincidência das curvas é proporcionada pelo tempo médio de abandono ($\mu = 10$ minutos), o qual impede o diferenciamento das políticas de *Piggybacking*, isto é, a política mesclagem simples comporta-se exatamente como a par-ímpar (realizando no máximo o emparelhamento de fluxos). Esta característica decorre do distanciamento entre fluxos, o qual impede a formação de grupos de mesclagem, mesmo considerando-se a janela máxima.

Na Figura 5.3 analisa-se a probabilidade de abandono. Nota-se que há uma queda no número de usuários que abandonam o sistema sem atendimento, isto se deve a dois fatores: (*i*) liberação de fluxos com as mesclagens, permitindo o atendimento de novas filas;

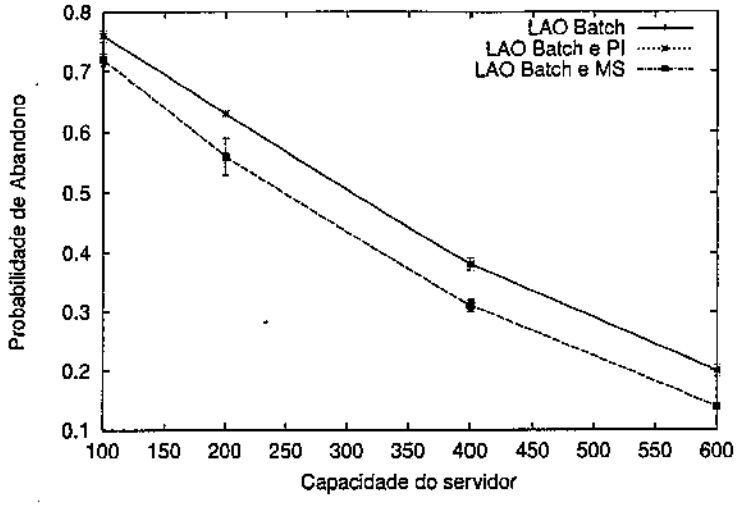


Figura 5.3: Probabilidade de abandono × capacidade do servidor na presença de *Batching* e *Piggybacking*. $T \sim N(10,\ 2{,}5)$.

e (*ii*) a realocação destes fluxos para o atendimento de vídeos populares, que claramente concentram um maior número de pedidos.

Os índices de injustiça são ilustrados no gráfico da Figura 5.4. Observa-se que a abordagem integrada de *Batching* e *Piggybacking* favorece ao aumento da injustiça no sistema, uma vez que, os fluxos liberados antecipadamente são realocados conforme os critérios da política LAO-*Batch*, que privilegia os vídeos populares. Em síntese, alocam-se mais fluxos para os vídeos populares reduzindo-se as desistências em suas filas, ao mesmo tempo em que os índices de abandono de usuários que requisitam vídeos de menor popularidade mantêm-se praticamente inalterados.

Nos gráficos das Figuras 5.5–5.7, analisam-se os mesmos requisitos de avaliação considerando-se um tempo de médio de abandono menor, $\mu = 8$ minutos.

No gráfico da Figura 5.5 comparam-se os índices de usuários suportados para política LAO-*Batch* isoladamente e em conjunto com as políticas mesclagem simples e par-ímpar, assumindo-se tempo médio de abandono de 8 e 10 minutos. Observa-se que com a determinação do tempo médio de abandono em 8 minutos, há uma redução no número de usuários suportados. Esta retração ocorre porque, quanto menor a janela de *Batching*, menor o número de requisições acumuladas, isto é, tem-se alocações mais freqüentes de fluxos privilegiando-se as políticas de *Piggybacking*. Uma vez que esta técnica apresenta menor eficiência em relação a *Batching*, os resultados de sua integração apresentam-se inferiores à medida em que se decrementa o valor de μ. Nota-se também que, mesmo com a redução do tempo médio de espera, praticamente não há diferenças entre as curvas das

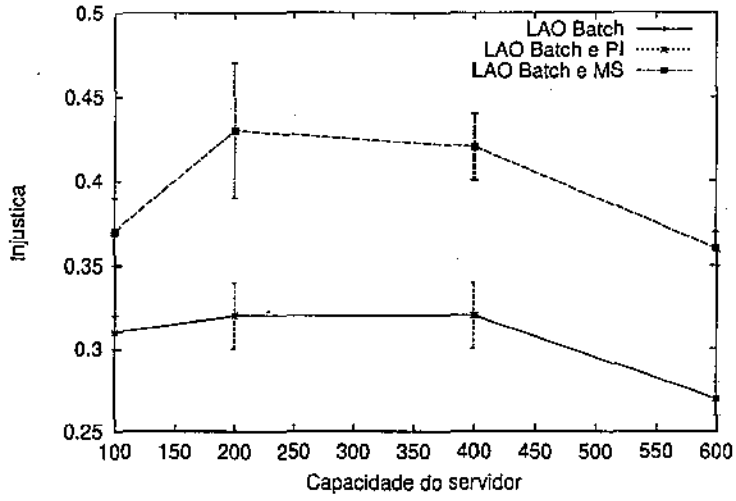


Figura 5.4: Injustiça × capacidade do servidor na presença de *Batching* e *Piggybacking*. $T \sim N(10,\ 2{,}5)$.

políticas mesclagem simples e par-ímpar[2] e que estas ainda posicionam-se acima da curva da política LAO-*Batch* isolada.

Na Figura 5.6 ilustram-se as probabilidades de abandono das políticas. Pode-se perceber que a redução no número de usuários suportados, ilustrada na Figura 5.5, é refletida através do aumento na probabilidade de abandono. De modo geral, a redução no tempo médio de abandono influencia de forma negativa a probabilidade de abandonos na sistemática integrada, porém esta ainda apresenta-se superior à implementação isolada de *Batching*. Adicionalmente, nota-se que as curvas para as políticas mesclagem simples e par-ímpar não mais coincidem. A menor probabilidade de abandono da política par-ímpar deve-se ao fato de a política mesclagem simples tentar formar grupos de mesclagem, nos quais alguns fluxos podem se superpor bem próximo do final da exibição do vídeo. Em outras palavras, em virtude de sincronizações tardias, a formação dos grupos de mesclagem não implica na mesma redução da probabilidade de abandonos obtida com o simples emparelhamento executado pela política par-ímpar.

No gráfico da Figura 5.7 analisa-se o critério de justiça. Nota-se que os índices de injustiça são mais altos quando o tempo médio de abandono é definido em 8 minutos. Esta característica decorre do aumento na freqüência de alocações de fluxos, de tal forma que os vídeos populares são ainda mais beneficiados. Em outras palavras, com a alocação mais freqüente para os vídeos populares, há uma redução no número de fluxos para os

[2]A diferença existente entre as curvas é mais notória no gráfico de probabilidade de abandono em virtude da menor abrangência das faixas de valores, fator que não propicia o achatamento das curvas.

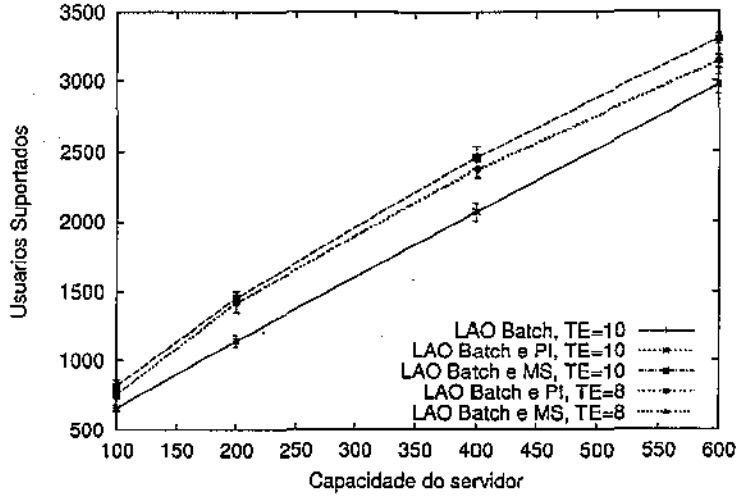


Figura 5.5: Usuários suportados × capacidade do servidor na presença de *Batching* e *Piggybacking*. $T \sim N(8,\ 2{,}5)$.

não populares. Conseqüentemente, tem-se índice maior de abandono em suas filas.

Através da análise dos resultados apresentados nesta seção, conclui-se que a integração de políticas de *Batching* e *Piggybacking* constitui-se numa estratégia atrativa para a maximização dos recursos do sistema, uma vez que permite a utilização mais eficiente dos mesmos. Um outro aspecto que deve ser enfatizado consiste no fato de que, mesmo utilizando-se o cenário definido para a política LAO-*Batch*, com a introdução de *Piggybacking* pode-se atender a uma maior população de usuários que a obtida com cenários nos quais o tempo médio de abandono modelado é menor.

5.3 Síntese do Capítulo

Este capítulo abordou a integração das técnicas de *Batching* e *Piggybacking*, as quais propõem o compartilhamento de fluxos de vídeo como mecanismo para a redução da demanda de banda passante. Alguns comentários foram feitos quanto às semelhanças e diferenças que envolvem ambas as técnicas, e como estes fatores, aparentemente antagônicos, podem ser organizados visando-se sua integração e intercâmbio.

Diversos aspectos relacionados ao emprego conjunto das políticas de *Batching* e *Piggybacking* foram analisados. Através deste estudo, conclui-se que uma sistemática integrada das técnicas é vantajosa em relação ao emprego isolado de ambas.

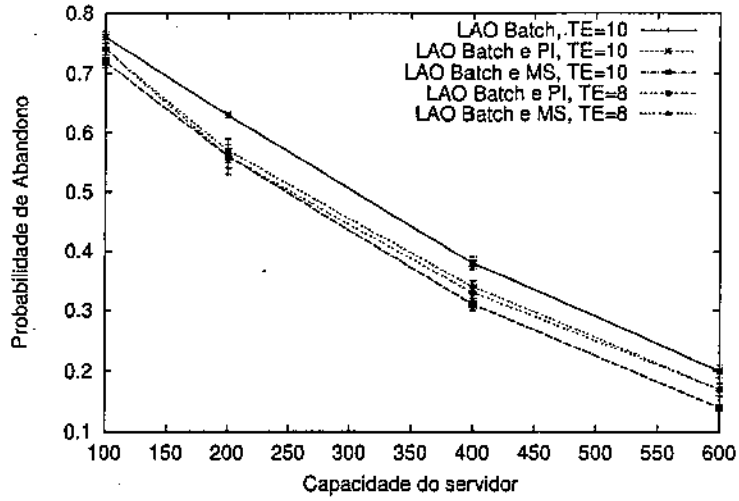


Figura 5.6: Probabilidade de abandono × capacidade do servidor na presença de *Batching* e *Piggybacking*. $T \sim N(8,\ 2{,}5)$.

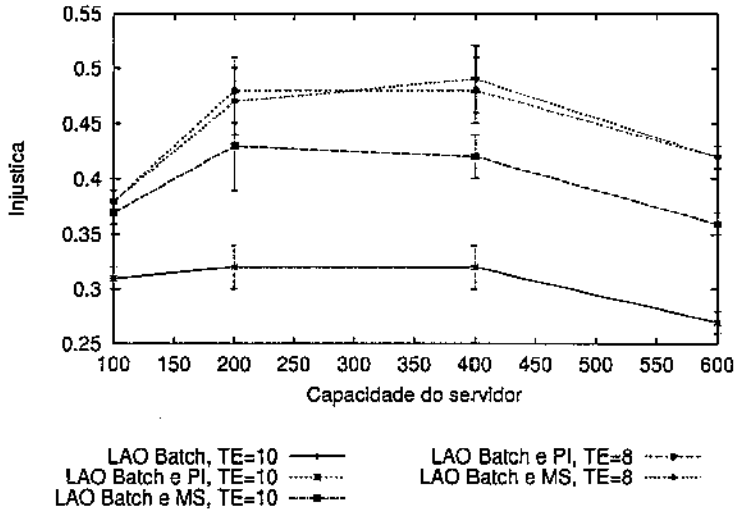


Figura 5.7: Injustiça × capacidade do servidor na presença de *Batching* e *Piggybacking*. $T \sim N(8,\ 2{,}5)$.

Capítulo 6

Conclusões

6.1 Consideração Finais

Com o advento do paradigma da integração de serviços no contexto das redes de telecomunicações, novos serviços têm sido criados e oferecidos aos usuários. Estes serviços, que se enquadram na classe das aplicações multimídia, despertam grande interesse em seus usuários em virtude de sua principal característica: o intercâmbio de dados através da utilização de diversas mídias de informação. Novas aplicações multimídia surgem continuamente, mantendo-se sempre crescente o número de usuários potenciais de seus serviços.

Nesta classe de aplicações destacam-se principalmente aquelas em que as mídias de áudio e, notadamente, de vídeo encontram-se presentes, de modo que as técnicas para a digitalização de vídeo, que surgiram na década de 80, possibilitaram o desenvolvimento destas aplicações. Destacando-se entre as demais encontra-se o Vídeo sob Demanda, esta aplicação permite a um usuário selecionar um vídeo dentre uma grande coleção e controlá-lo através de operações de VCR, como se o mesmo assistisse a um filme utilizando um aparelho de vídeo cassete em sua residência. O mercado para os serviços de VoD é extremamente abrangente, com isso seu potencial comercial situa-se em patamares muito elevados. No entanto, o oferecimento de um serviço de VoD em larga escala ainda não é possível em decorrência das limitações tecnológicas que existem atualmente. Diversas iniciativas envolvendo universidades e empresas têm sido implementadas com o objetivo de tornar operacional o oferecimento de serviços de VoD. Porém, estas possuem caráter experimental e não abrangem uma grande população de usuários. Em virtude destes motivos, VoD tem sido alvo de grande interesse e intensas pesquisas nos últimos anos.

Uma vez que VoD pressupõe a distribuição de vídeo de alta qualidade, muitas vezes no formato de HDTV, um dos pontos críticos nestes sistemas endereça a largura de banda requerida para a transmissão. O emprego da tecnologia de fibra ótica em redes suaviza esta

limitação nestes pontos de uma arquitetura de VoD. Deste modo, os servidores tornam-se o ponto crítico do sistema; mais precisamente, o subsistema de armazenamento (hierarquia de armazenamento) precisa lidar com um grande número de transações e suportar uma quantidade elevada de fluxos de dados.

As restrições existentes nos servidores, notadamente no subsistema de armazenamento, motivaram a proposição de um conjunto de técnicas que objetivam a redução do montante de banda passante requerido para a transmissão de fluxos de dados. Neste contexto, destacam-se as técnicas de *Batching* e *Piggybacking*, as quais propõem o compartilhamento de fluxos como mecanismo para o gerenciamento mais eficiente da demanda de banda passante em servidores de vídeo.

Nesta dissertação apresentou-se um estudo sobre o emprego de ambas as técnicas. O trabalho aborda problemas atuais e sua realização justifica-se pela importância da redução da demanda de banda passante para o atendimento de uma grande população de usuários. Neste estudo, caracterizado por sua natureza multidisciplinar, avaliaram-se aspectos de desempenho em servidores considerando-se diferentes cenários. O estudo envolveu a proposição e avaliação de novas políticas de *Piggybacking*, *Batching* e da avaliação da integração destas técnicas. Adicionalmente, investigaram-se estratégias para a redução do custo computacional para a geração da árvore de mesclagem de fluxos de vídeo.

Os experimentos realizados evidenciam a extensão dos resultados contidos na literatura contribuindo para o avanço do estado da arte da pesquisa sobre *Batching* e *Piggybacking*. Em alguns casos, algumas soluções originais foram encontradas para os problemas investigados (heurística *BuildTree* e política *Look-Ahead Optimize Batch*).

Finalmente, pode-se dizer que o oferecimento de serviços de VoD em larga escala requer uma maturação da tecnologia envolvida, isto é, alguma pesquisa precisa ser desenvolvida abordando assuntos como: técnicas de armazenamento de arquivos multimídia, suporte a aplicações multimídia em sistemas operacionais através de políticas para o escalonamento de processos e/ou *threads*, esquemas para o gerenciamento de *buffers*, balanceamento de carga em discos, suporte a operações de VCR, a *set top box*, além do próprio gerenciamento da demanda de banda passante, dentre outros.

6.2 Principais Contribuições

As principais contribuições deste trabalho são:

i) Proposição e análise de duas novas políticas de *Piggybacking*, denominadas S^2 e Híbrida, que conduzem ao gerenciamento mais eficiente da demanda de banda passante e, com isso, ampliam as vantagens do emprego da técnica de *Piggybacking* em um sistema de VoD.

ii) Proposição da heurística *BuildTree* que reduz a complexidade computacional de construção da árvore de mesclagens de fluxos de vídeo, constatando-se através de simulação, que a heurística mostrou-se bastante precisa quando comparada com a solução ótima gerada pelo algoritmo de programação dinâmica. Elaborou-se a demonstração formal de complexidade da heurística e constatou-se, também com o uso de simulações, a redução do tempo de construção da árvore de mesclagem.

iii) Proposição e análise de uma nova política de *Batching*, *Look-Ahead Optimize Batch*, que busca maximizar o número de usuários suportados pelo sistema solucionando o problema da alocação de fluxos através da modelagem de PPLs. Nestes problemas, o cenário corrente no sistema é analisado em conjunto com eventos que ocorrerão em momentos futuros, podendo-se decidir pela alocação ou retenção de fluxos disponíveis com o objetivo de maximizar o efeito de *Batching*.

iv) Realização de um estudo envolvendo a integração das técnicas de *Batching* e *Piggybacking* em que se pode constatar a vantagem desta integração. Neste contexto, comentários foram traçados envolvendo mecanismos de integração entre ambas as técnicas.

6.3 Extensões e Trabalhos Futuros

Este trabalho pode ser estendido através das seguintes atividades:

i) **Dimensionamento analítico de uma janela ótima para a política S^2:** A política S^2 emprega como intervalo para atribuição de velocidades aos fluxos a janela de mesclagem derivada analiticamente para a política Mesclagem Simples Generalizada. No entanto, estas políticas diferem, principalmente, no aspecto da topologia da árvore de mesclagem gerada, ou seja, a primeira pode construir árvores com qualquer topologia, enquanto que a última sempre gera árvores com topologia fixa. Em virtude desta característica, definir uma janela de mesclagem muito ampla para a política Mesclagem Simples Generalizada reduz sua eficiência em termos da diminuição da demanda de banda passante. Pode-se constatar esta afirmação analisando-se os resultados para a política Mesclagem Simples[1] apresentados em [41, 42]. Porém, se este comportamento inflexível ocorre somente nos intervalos *Snapshot*, tais conseqüências podem ser desprezíveis para a política S^2.

ii) **Análise da aplicação da heurística *BuildTree* sobre um número amplo de fluxos:** Com a redução da complexidade de construção da árvore de mesclagens,

[1]Neste caso, a política Mesclagem Simples constitui-se num sub-caso de sua versão generalizada, em que a janela de mesclagem assume o maior valor possível — Janela Máxima de Mesclagem.

obtida com a heurística *BuildTree*, torna-se viável a construção de árvores de mesclagens envolvendo um número amplo de fluxos de vídeo. Portanto, é interessante investigar sua aplicação neste contexto. Tal estudo, pode conduzir à proposição de nova(s) política(s) de *Piggybacking*.

iii) **Estudo da Política S^2 na presença de operações de VCR:** Pode-se realizar o mesmo estudo desenvolvido para a política S^2 adicionando-se o suporte a operações de VCR.

iv) **Proposição de uma heurística para a política *Look-Ahead Optimize Batch*:** A política LAO-*Batch* modela PPLs, cujas soluções determinam a qual vídeo alocar um fluxo quando o número destes torna-se escasso em relação às diversas filas com requisições pendentes. No entanto, a modelagem e solução destes PPLs introduz uma complexidade adicional ao sistema, aumentando-se a carga de processamento nos servidores. Deste modo, pode-se investigar a proposição de heurística(s) que represente(m) com fidelidade os problemas modelados e simplifique(m) a análise dos casos.

v) **Extensões aos estudos realizados sobre a política *Look-Ahead Optimize Batch*:** Nos experimentos realizados considerou-se o mesmo custo (*revenue*) para todos os vídeos no sistema. Em outras palavras, atribuiram-se "níveis de prioridade" iguais, com relação ao preço pago pelos usuários, a todos os vídeos. Com isso, pode-se desenvolver todo um estudo em cujos custos variem, segundo diversos critérios como taxas maiores para os vídeos não populares ou, de modo contrário, para os vídeos populares. A determinação destes critérios pode implicar em diferenças no comportamento do sistema com relação aos fatores de justiça, probabilidade de abandono e número de usuários suportados.

vi) **Suporte a operações de VCR na política LAO-*Batch*:** Pode-se realizar o mesmo estudo descrito no item anterior adicionando-se o suporte a operações de VCR.

vii) **Flexibilização da política LAO-*Batch* para reduzir o tempo de espera dos vídeos não populares:** A política LAO-*Batch* prioriza os vídeos populares em detrimento dos não populares, uma vez que aplica o mesmo critério de alocação para todos os vídeos indistintamente, o qual considera o número de requisições em cada fila como o aspecto mais importante para alocação. Assim sendo, pode-se investigar mecanismos de flexibilização da política de modo a torná-la mais justa e reduzir o tempo de espera das requisições por vídeos não populares, dado que a probabilidade de uma nova requisição chegar em um intervalo de tempo curto é muito pequena.

Uma possível alternativa para este problema seria o particionamento dos vídeos em conjuntos distintos, populares e não populares, e a aplicação do critério de alocação proposto somente sobre o conjunto dos vídeos populares.

viii) **Suporte a operações de VCR na integração de *Batching* e *Piggybacking*:** Pode-se avaliar o impacto do oferecimento de operações de VCR sobre a integração das técnicas de *Batching* e *Piggybacking*. Neste contexto, deve-se avaliar o tempo médio de retomada da exibição normal após a realização de uma operação de VCR e a probabilidade de aceitação/bloqueio para a realização destas operações, além dos aspectos de usuários suportados, probabilidade de abandono e justiça no sistema.

Apêndice A

Complexidade da Heurística *BuildTree*

$$
\begin{aligned}
T(n) &= k + T(k) + T(n-k) \\
T(n) &= (n-1) + T(n-1) + T(1), \qquad \text{fazendo } T(1) = c_1 \\
&= (n-1) + [(n-2) + T(n-2) + c_1] + c_1 = \\
&\quad 2 \times n - 3 + 2 \times c_1 + T(n-2) \\
&= 2 \times n - 3 + 2 \times c_1 + (n-3) + T(n-3) + c_1 = \\
&\quad 3 \times n - 6 + 3 \times c_1 + T(n-3) \\
&= 3 \times n - 6 + 3 \times c_1 + (n-4) + T(n-4) + c_1 \\
&= 4 \times n - 10 + 4 \times c_1 + T(n-4) \\
&= x \times n - \sum_{i=1}^{x} i + x \times c_1 + T(n-x), \qquad \text{no pior caso } x = n \\
&= n^2 - \sum_{i=1}^{n} i + n \times c_1 + c_2, \qquad \text{fazendo } T(n-n) = T(0) = c_2 \\
&= n^2 - \frac{n \times (n-1)}{2} + n \times c_1 + c_2 \\
&= n^2 - \frac{n^2}{2} + \frac{n}{2} + n \times c_1 + c_2 \\
&= \frac{n^2}{2} + \left(c_1 + \frac{1}{2}\right) \times n + c_2 \\
T(n) &= O(n^2)
\end{aligned}
$$

■

Apêndice B

Código da Heurística *BuildTree*

Algoritmo 1 Heurística de Construção da Árvore de Mesclagem.

```
int BuildTree(int Posicao[], int i, int j) {
   int n = j - i + 1, // Número de fluxos.
       k = i,         // Fluxo divisor.
       Custo = 0,     // Custo da árvore.
       MPik,          // Posição de Mesclagem dos fluxos i e k.
       MPkj;          // Posição de Mesclagem dos fluxos k + 1 e j.

    switch(n) {
        case 0:
        case 1:  return(0);
        case 2:  return(MergePosition(Posicao[i], Posicao[j]));
        case 3:  MPik = MergePosition(Posicao[i], Posicao[i + 1]);
                 MPkj = MergePosition(Posicao[i + 1], Posicao[j]);
                 return(((MPik  > MPkj) ? MPkj : MPik) +
                        MergePosition(Posicao[i], Posicao[j]));
        default: while (MergePosition(Posicao[i], Posicao[k + 1]) <
                        MergePosition(Posicao[k + 1], Posicao[j]))
                   k++;
                 Custo  = BuildTree(Posicao, i, k);
                 Custo += BuildTree(Posicao, k + 1, j);
                 Custo += MergePosition(Posicao[i], Posicao[j]);
                 return(Custo);
     }
}
```

Apêndice C

Lista de Acrônimos

Acrônimo	Descrição
AAL	*ATM Adaptation Layer*
ADSL	*Asymmetric Digital Subscriber Line*
ANSI	*American National Standards Institute*
ATM	*Asynchronous Transfer Mode*
B&B	*Branch and Bound*
BML	*Max Batch with Minimal Loss*
CAC	*Connection Admission Control*
CCIR	*Consultative Committee on International Radio*
CD-ROM	*Compact Disk - Read Only Memory*
DAVOD	*Dynamically Allocated Video on Demand*
EE	*Espaço de Estados*
EP	*Estado do Problema*
ER	*Estado de Resposta*
ES	*Estado da Solução*
FCFS	*First Come First Served*
FDMA	*Frequency Division Multiple Access*
FTTC	*Fiber To The Curb*
FTTH	*Fiber To The Home*
GOP	*Group of Pictures*
HDTV	*High Definition TV*
HFC	*Hybrid Fiber/Coax*
IML	*Min Idle with Minimal Loss*
	Continua na próxima página...

Tabela C.1: Tabela de Acrônimos.

Acrônimo	Descrição
IMQ	*Min Idle with MQL*
JE	*Janela de Estudo*
JPEG	*Join Photographic Experts Group*
LAO-Batch	*Look Ahead Optimize Batch*
MBQ	*Max Batch with MQL*
MFQL	*Maximum Factored Queue Length*
MOD	*Movie on Demand*
MPEG	*Motion Picture Expert Group*
MQL	*Maximun Queue Length*
NTSC	*National Television System Committee*
NVOD	*Near Video on Demand*
ONU	*Optical Network Unit*
PPL	*Problema de Programação Linear*
PVOD	*Partitioned Video on Demand*
QOS	*Quality of Service*
RAID	*Redundant Arrays of Inexpensive Disks*
RAM	*Random Access Memory*
RDSI-FL	*Rede Digital de Serviços Integrados de Faixa Larga*
RISC	*Reduced Instruction Set Computer*
SONET	*Synchronous Optical Networks*
STS-1	*Sinal de Transporte Síncrono de Nível 1*
TDMA	*Time Division Multiple Access*
TVOD	*True Video on Demand*
VCR	*Videocassete Recorder*
VOD	*Video on Demand*

Tabela C.1: Tabela de Acrônimos.

Apêndice D

Tabelas de Símbolos

D.1 Letras Gregas

Símbolo	Descrição
λ	Taxa de chegadas de requisições.
μ	Média para a distribuição normal dos tempos de abandono dos usuários.
σ^2	Variância para a distribuição normal dos tempos de abandono dos usuários.
θ	Parâmetro para a distribuição Zipf.
ω	Tempo mínimo de espera da política *Batch* MBQ.
ζ	Parâmetro de confiança para a determinação de ω na política *Batch* MBQ.
Δt	Intervalo entre alocações de fluxos de vídeo no esquema NVOD.
$\Im$	Janela de Estudo.

Tabela D.1: Letras Gregas.

D.2 Letras Maiúsculas

Símbolo	Descrição
$\mathcal{A}(i,j)$	Árvore de mesclagem dos fluxos $i,\ldots,j$.
A_{ki}	Número de fluxos que se tornaram disponíveis até o instante de alocação da i-ésima requisição do vídeo k (política LAO-*Batch*).
$\mathbf{A}$	Matriz com os coeficientes das restrições de um PPL.
$C(i,j)$	Custo para mesclagem dos fluxos $i,\ldots,j$.
$\mathcal{C}$	Conjunto de vídeos não populares no esquema de *Batch Min_Idle*.
F_k	Fila de requisições do vídeo k.
$\mathcal{H}$	Conjunto de vídeos populares no esquema de *Batch Min_Idle*.
I	Intervalo *Snapshot*.
L	Número de quadros de um vídeo.
$M_{p,q}$	Matriz de dimensões p e q.
N	Número de fluxos de vídeo disponíveis no instante de definição de uma janela de estudo (intervalo de *Batching* esgotado).
N_{max}	Número máximo de fluxos de vídeo suportado por um servidor.
$\mathcal{P}$	Processo geral de chegada de requisições por vídeos.
P	Polígono simples.
$\overline{P}$	Probabilidade global de abandono num sistema com V vídeos.
$P_A(k)$	Probabilidade de abandono de usuários do vídeo k.
$P(i,j)$	Posição de mesclagem dos fluxos de vídeo i e j.
$Q(\ell)$	Número de quadros exibidos por um vídeo dentro de um intervalo *Snapshot*.
$Quadros_I$	Somatório de todos os $Q(\ell)$ de um intervalo *Snapshot*.
R_i	i-ésima requisição de um PPL (modelado pela política LAO-*Batch*).
S_{max}	Taxa máxima de exibição de um fluxo de vídeo no contexto de *Piggybacking*, $S_{max} = 1{,}05 \times S_n$.
S_{min}	Taxa mínima de exibição de um fluxo de vídeo no contexto de *Piggybacking*, $S_{min} = 0{,}95 \times S_n$.
S_n	Taxa normal de exibição de um fluxo de vídeo, $S_n = 30$ quadros/segundo.
S_R	Número de filas com requisições pendentes.
$\mathcal{T}$	Conjunto de cordas de um polígono (problema da triangulação de polígonos convexos).
	Continua na próxima página...

Tabela D.2: Letras Maiúsculas.

Símbolo	Descrição
T	Tempo de abandono dos usuários, modelado através da distribuição normal para o estudo de *Batching*, $T \sim N(\mu, \sigma^2)$.
TC	Vetor contendo o tempo decorrido desde o início de um intervalo *Snapshot* até a chegada de uma requisição (política Híbrida).
V	Número de vídeos armazenados no sistema.
V_t	Vetor contendo os instantes de alocação das primeiras requisições das filas dos vídeos ordenados cronologicamente.
W	Janela ótima de mesclagem.
W_m	Janela máxima de mesclagem.
W'_m	Janela máxima alterada de mesclagem (política S^2).
$W_{Pol}(f_i)$	Janela de mesclagem para a política *Pol* calculada em relação à posição f_i.

Tabela D.2: Letras Maiúsculas.

D.3 Letras Minúsculas

Símbolo	Descrição
b	Matriz com os coeficientes do lado direito de um PPL.
c	Matriz com os coeficientes de custo de um PPL.
c	Constante de normalização da distribuição Zipf.
e_k	*status* que indica a presença de requisições na fila do vídeo k.
f_i	Quadro atualmente exibido pelo fluxo de vídeo i.
p_k	Probabilidade de escolha do vídeo k conforme a distribuição Zipf.
q_k	Número de requisições da fila do vídeo k.
r_k	Custo (taxa monetária) para assistir ao vídeo k.
t_{ki}	Instante de alocação (abandono) da i-ésima requisição da fila do vídeo k.
v_i	i-ésimo vértice de um polígono.
x_{ki}	Variável que representa a i-ésima requisição da fila do vídeo k nos PPLs formulados pela política LAO-*Batch*.
w_i	Peso de um vértice no problema da triangulação de polígonos.

Tabela D.3: Letras Minúsculas.

Bibliografia

[1] Jean M. McManus e Keith W. Ross. Video on Demand over ATM: Constant-Rate Transmission and Transport. Em *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies: Networking the Next Generation* [53], páginas 1357–1362.

[2] Jean M. McManus e Keith W. Ross. Video on Demand over ATM: Constant-Rate Transmission and Transport. *IEEE Journal on Selected Areas in Communications*, 14(6):1087–1098, Agosto 1996.

[3] Daniel Minoli. *Video Dialtone Technology: Digital Video over ADSL, HFC, FTTC & ATM.* McGraw Hill, 1995.

[4] François Fluckiger. *Understanding Networked Multimedia: Applications and Technology.* Prentice Hall, 1995.

[5] Luiz Fernando Gomes Soares, Guido Lemos, e Sérgio Colcher. *Redes de Computadores: Das Lans, Mans e Wans às Redes ATM.* Redes e Conectividade. Editora Campus, 1995.

[6] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):47–58, Abril 1991.

[7] Martin de Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN.* Prentice Hall, 3ª edição, 1995.

[8] Andrew S. Tanembaum. *Computer Networks.* Computer Networking. Prentice Hall, 3ª edição, 1996.

[9] Ralf Steinmetz. Multimedia File Systems Survey: Approaches for Continuous Media Disk Scheduling. *Computer Communications*, 18(3):133–144, Março 1995.

[10] C. L. Jonathan Liu, Jenwei Hsieh, David H. C. Du, e Mengjou Lin. Performance of A Storage System for Supporting Different Video Types and Qualities. Em

Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies: Networking the Next Generation [53], páginas 2–9.

[11] Banu Özden e Rajee Rastogi e Avi Silberschatz. On the Design of a Low-Cost Video-on-Demand Storage Server. *Multimedia Systems*, 4(1):40–54, Fevereiro 1996.

[12] Asit Dan, Martin Kienzle, e Dinkar Sitaram. A Dynamic Policy of Segment Replication for Load-Balancing in Video-on-Demand Servers. *Multimedia Systems*, 3(3):93–103, 1995.

[13] Kun-Lung Wu e Philip S. Yu. Consumption-Based Buffer Management for Maximizing System Throughputs of News-on-Demand Multimedia Systems. Technical Report RC 20143 (98156), IBM Research Report, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, Janeiro 1995.

[14] Antoine N. Mourad. Issues in the Design of a Storage Server for Video-on-Demand. *Multimedia Systems*, 4(2):70–86, Abril 1996.

[15] Harrick M. Vin, Alok Goyal, e Pawan Goyal. Algorithms for Designing Multimedia Servers. *Computer Communications*, 18(3):192–203, Março 1995.

[16] Chiung-Shien Wu, Gin-Kou Ma, e Bao-Shuh P. Lin. On Scalable Design of an ATM-based Video Server. Em *Proceedings of the ICC'96: Converging Technologies for Tomorrow's Applications*, volume 1, páginas 1335–1340, Dallas, Junho 1996. IEEE Communications Society.

[17] Y. B. Lee e P. C. Wong. A Server Array Approach for Video-on-Demand Service on Local Area Networks. Em *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies: Networking the Next Generation* [53], páginas 27–34.

[18] Victor O. K. Li, Wanjiun Liao, Xiaoxin Qiu, e W. M. Wong. Performance Model of Interactive Video-on-Demand Systems. *IEEE Journal on Selected Areas in Communications*, 14(6):1099–1109, Agosto 1996.

[19] Kelvin C. Almeroth e Mostafa H. Ammar. The Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Servide. *IEEE Journal on Selected Areas in Communication*, 14(6):1110–1122, Agosto 1996.

[20] Philip S. Yu, Joel L. Wolf, e Hadas Shachnai. Design and Analysis of a Look-Ahead Scheduling Scheme to Support Pause-Resume for Video-on-Demand Applications. Research Report RC 19683 (87236), IBM Research Division, Agosto 1994.

[21] Phikip S. Yu, Joel L. Wolf, e Hadas Shachnai. Design and Analysis of a Look-Ahead Scheduling Scheme to Support Pause-Resume for Video-on-Demand Applications. *Multimedia Systems*, 3(4):137–149, Setembro 1995.

[22] Asit Dan, Perwez Shahabuddin, Dinkar Sitaram, e Don Towsley. Channel Allocation under Batching and VCR Control in Video-on-Demand Systems. *Journal of Parallel and Distributed Computing*, 30:168–179, 1995.

[23] Jayanta K. Dey-Sircar, James D. Salehi, James F. Kurose, e Don Towsley. Providing VCR Capabilities in Large-Scale Video Servers. Em *Proceedings of the 2nd ACM International Conference on Multimedia*, páginas 25–36, San Francisco, CA, 1994. ACM.

[24] Victor O. K. Li e Wanjiun Liao. The Split and Merge Protocol for Interactive Video-on-Demand. *IEEE Multimedia*, 4(4):51–62, Dezembro 1997.

[25] Jean-Paul Nussbaummer e Frank Schaffa. Capacity Analysis of CATV for on-Demand Multimedia Distribution. Em *Proceedings of the First ISMM International Conference on Distributed Multimedia Systems Applications*, Agosto 1994.

[26] Jean-Paul Nussbaummer e Frank Schaffa. Impact of Channel Allocation Policies on Video on Demand Over CATV. Private Communication.

[27] J. P. Nussbaummer, B. V. Patel, F. Schaffa, e J.P.G. Sterbenz. Networking Requirements for Interactive Video on Demand. *IEEE Journal on Selected Areas in Communications*, 13(No. 5):779–787, Junho 1995.

[28] Chatschik C. Bisdikian e Baiju V. Patel. Issues on Movie Allocation in Distributed Video-on-Demand Systems. Em *Proceedings of the ICC'95*, volume 3, páginas 250–255, Seattle, Junho 1995. IEEE Communications Society: Communications — Gateway to Globalization.

[29] Frank Schaffa e Jean Paul Nussbaumer. On Bandwidth and Storage Tradeoffs in Multimedia Distribution Networks. Em *Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies: Bringing Information to People* [54], páginas 1020–1026.

[30] Christos Papadimitriou, Srinivas Ramanathan, P. Venkat Rangan, e Srihari SampathKumar. Multimedia Information Caching for Personalized Video-on-Demand. *Computer Communications*, 18(3), Março 1995.

[31] Asit Dan e Dinkar Sitaram. A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads. Technical Report RC 20206 (89404), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598, Setembro 1995.

[32] Asit Dan e Dinkar Sitaram. Multimedia Caching Strategies for Heterogeneous Application and Server Environments. Technical Report RC 20670 (91482), IBM Research Division, T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598, Dezembro 1996.

[33] Leana Golubchik, John C. S. Lui, e Richard Muntz. Reducing I/O Demand in Video-on-Demand Storage Servers. *ACM Sigmetrics*, páginas 25–36, 1995. Ottawa, Canada.

[34] Leana Golubchik, John C. S. Lui, e Richard Muntz. Adaptive Piggybacking: A Novel Technique for Data Sharing in Video-on-Demand Storage Servers. *Multimedia Systems*, 4(3):140–155, 1996.

[35] Kevin C. Almeroth, Asit Dan, Dinkar Sitaram, e William H. Tetzlaff. Long Term Channel Allocation Strategies for Video Applications. Research Report RC 20249 (89566), Networking and Telecommunications Group-Georgia Institute of Technology e IBM Research Division, Outubro 1995.

[36] Hadas Shachnai e Philip S. Yu. The Role of Wait Tolerance in Effective Batching: A Paradigm for Multimedia Scheduling Schemes. Technical Report RC 20038 (88607), IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, Abril 1995.

[37] Charu C. Aggarwal, Joel L. Wolf, e Philip S. Yu. The Maximun Factor Queue Length Batching Scheme for Video-on-Demand Systems. Technical Report RC 20261 (91305), IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, Novembro 1996.

[38] Asit Dan Dinkar Sitaram e Perwez Shahabuddin. Dynamic Batching Policies for an on-Demand Video Server. *Multimedia Systems*, 4:112–121, 1996.

[39] Hadas Shachnai e Philip S. Yu. An Analytical Study of Multimedia Batching Schemes. Technical Report RC 20662 (91420), IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, Dezembro 1996.

[40] Asit Dan, Perwez Shahabuddin, Dinkar Sitaram, e William H. Tetzlaff. Anticipatory Scheduling with Batching for Video Servers. Technical Report RC 19640 (89313),

IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, 1994.

[41] Charu C. Aggarwal, Joel L. Wolf, e Philip S. Yu. Adaptive Piggybacking Schemes for Video-on-Demand Systems. Technical Report RC 20635 (91350), IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, Novembro 1996.

[42] Charu C. Aggarwal, Joel L. Wolf, e Philip S. Yu. On Optimal Piggyback Merging Policies for Video-on-Demand Systems. Technical Report RC 20337 (90078), IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, Fevereiro 1996.

[43] Martin Gardner. Catalan numbers. *Scientific American*, páginas 120–124, Junho 1976.

[44] Thomas H. Cormen, Charles E. Leiserson, e Ronald L. Rivest. *Introduction to Algorithms.* The MIT electrical engineering and computer science series. The MIT Press, 3ª edição, 1991.

[45] T. C. Hu e M. T. Shing. Some theorems about Matrix Multiplication. Em *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, páginas 28–35. IEEE Computer Society, 1980.

[46] T. C. Hu e M. T. Shing. Computation of Matrix Chain Products. Part I. *SIAM Journal on Computing*, 11(2):362–373, Maio 1982.

[47] T. C. Hu e M. T. Shing. Computation of Matrix Chain Products. Part II. *SIAM Journal on Computing*, 13(2):228–251, Maio 1984.

[48] T. C. Hu e M. T. Shing. An $O(n)$ Algorithm to Find a Near-Optimum Partition of a Convex Polygon. *Journal of Algorithms*, 2:122–138, 1981.

[49] Paulo F. Bregalda, Antonio A. F. de Oliveira, e Cláudio T. Bornstein. *Introdução à Programação Linear.* Editora Campus, 3ª edição, 1988.

[50] Shu-Cherng Fang e Sarat Puthenpura. *Linear Optimization and Extensions.* Prentice Hall, 1993.

[51] Ellis Horowitz e Sartaj Sahni. *Fundamentals of Computer Algorithms.* Computer software engineering series. Computer Science Press, 1984.

[52] Willian H. Press, Saul A. Teukolsky, Willian T. Vetterling, e Brian P. Flannery. *Numerical Recipes in C: the Art of Scientifc Computing.* Cambridge University Press, 2ª edição, 1994.

[53] IEEE Computer Society e IEEE Communications Society. *Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies: Networking the Next Generation*, San Francisco, California, Março 1996.

[54] IEEE Computer Society e IEEE Communications Society. *Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies: Bringing Information to People*, Boston, Massachusetts, Abril 1995.

Índice