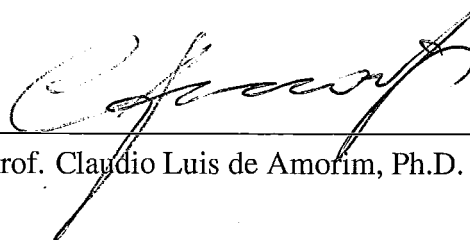


IMPLEMENTAÇÃO E AVALIAÇÃO DE UM SISTEMA DE VÍDEO SOB DEMANDA
BASEADO EM CACHE DE VÍDEO COOPERATIVA

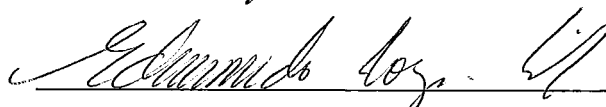
Leonardo Bidese de Pinho

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

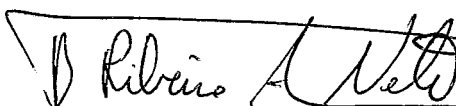
Aprovada por:



Prof. Claudio Luis de Amorim, Ph.D.



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



Prof. Berthier Ribeiro de Araujo Neto, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 2002

PINHO, LEONARDO BIDESE DE

Implementação e Avaliação de um Sistema
de Vídeo sob Demanda Baseado em Cache de
Vídeo Cooperativa [Rio de Janeiro] 2002

XII, 71 p. 29,7 cm (COPPE/UFRJ, M.Sc.,
Engenharia de Sistemas e Computação, 2002)

Tese – Universidade Federal do Rio de
Janeiro, COPPE

1 - Sistemas Distribuídos Multimídia

2 - Vídeo sob Demanda

3 - Cache Cooperativa

I. COPPE/UFRJ II. Título (série)

*Para todos aqueles que enxergam nos maiores obstáculos a oportunidade
de atingir as mais espetaculares vitórias*

Agradecimentos

Aos meus pais, Paulo e Mariza, por sempre me incentivarem a ampliar meus conhecimentos, fornecendo todo o suporte afetivo e financeiro durante esta longa jornada estudantil. Por serem o que são, tanto pessoal quanto profissionalmente, servirão sempre como os melhores de todos os exemplos a seguir.

A minha linda esposa Ana, por todos os seus sorrisos que alimentam o meu amor e me fortificam. Aproveito para me desculpar por todos os momentos em que o stress falou mais alto do que o bom senso.

A toda a minha família e a da Ana, pelo carinho e incentivo de sempre, e a todos meus verdadeiros amigos, que mesmo separados por tão grande distância, não deixaram de entrar em contato, ajudando a superar os momentos em que a saudade de Pelotas bateu mais forte.

Aos meus professores da UCPel, Adenauer Yamin e Cristiano Costa, que muito colaboraram para que a minha vinda para a COPPE se concretizasse, e aos do PESC por todo o conhecimento repassado.

Ao Thobias e ao Rodrigo, que aceitaram o desafio de vir morar comigo no Rio para estudar no curso mais bem conceituado do país, mesmo que fôssemos meros conhecidos. Principalmente ao primeiro, que mesmo com meu casamento, encarou a convivência até o fim, tendo sido ótimo parceiro em todos os momentos.

Ao meu orientador Claudio Amorim, por me incorporar ao LCP, onde recebi todas as condições necessárias para desenvolver minha tese, tendo servido como uma bússola nos momentos de indecisão, e ao meu colega de sala e mentor do tema da tese, Edison Ishikawa, por ter sempre feito o máximo para esclarecer minhas dúvidas, fossem elas técnicas ou sobre qualquer outro assunto.

A todos colegas e funcionários do LCP e do PESC, que nunca se negaram a ajudar quando requisitados, ao prof. Edmundo de Souza e Silva e sua orientada Adriane Cardozo por terem disponibilizado e dado suporte ao servidor RIO, a todas outras pessoas que de alguma forma colaboraram e a CAPES pelo suporte financeiro.

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

IMPLEMENTAÇÃO E AVALIAÇÃO DE UM SISTEMA DE VÍDEO SOB DEMANDA BASEADO EM CACHE DE VÍDEO COOPERATIVA

Leonardo Bidese de Pinho

Maior/2002

Orientador: Claudio Luis de Amorim

Programa: Engenharia de Sistemas e Computação

Esta tese propõe a utilização da técnica de reuso de fluxo denominada Cache de Vídeo Cooperativa (CVC) para agregar escalabilidade a servidores de Vídeo sob Demanda (VoD) de baixo custo. Nesta técnica, os buffers locais dos clientes integram uma memória global distribuída, usada como fonte prioritária de conteúdo, fazendo com que os clientes se tornem provedores de outros clientes, segundo um modelo *peer-to-peer* baseado em conceitos das técnicas *Chaining* e *Patching*. Para avaliar a proposta, implementamos o protótipo de sistema de VoD denominado Ambiente de Vídeo Global (*Global Video Environment* - GloVE), composto por extensões à CVC que são capazes de adaptá-la a servidores convencionais sem suporte a *multicast*, operando no modelo *pull*. O GloVE coordena a CVC através de um gerente centralizado com diferentes políticas de escolha do cliente responsável por responder a uma determinada requisição. Conduzimos vários experimentos com o GloVE, usando diferentes capacidades nos buffers locais dos clientes e diversas taxas de chegada, comparando o desempenho da CVC com versões do protótipo configuradas para funcionar como um sistema convencional e com combinações das técnicas de reuso de fluxo de vídeo: *Patching*, *Chaining* e *Batching*. Os resultados dos testes demonstram que CVC reduz significativamente o número de fluxos ativos no servidor, especialmente quando o intervalo médio entre chegadas de clientes é menor que a capacidade de armazenamento de vídeo, em unidades de tempo, do buffer local. Mostramos também que uma abordagem híbrida de CVC com *Batching* contribui para aumentar ainda mais a escalabilidade do servidor de VoD.

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

IMPLEMENTATION AND EVALUATION OF A VIDEO ON DEMAND SYSTEM BASED ON COOPERATIVE VIDEO CACHE

Leonardo Bidese de Pinho

May/2002

Advisor: Claudio Luis de Amorim

Department: Computing and Systems Engineering

This thesis proposes the utilization of the technique called Cooperative Video Cache (CVC), which is based on the reuse of the video stream to add scalability to low-cost Video-on-Demand (VoD) servers. In CVC, the local video buffers at the clients form a distributed global video cache that is used as a new source of video contents in addition to the VoD server. Through the use of techniques such as Chaining and Patching, CVC allows clients to become video providers for other clients, according to the peer-to-peer model. To evaluate this proposal, a prototype of scalable VoD server called Global Video Environment (GloVE) is implemented by adapting and extending CVC to support conventional servers that have no multicast capability and work in pull model. GloVE is based on a centralized CVC manager that supports different policies to dynamically select new video providers from existing clients. GloVE's scalable performance is evaluated using different local buffer capacities and various client arrival rates. Also, GloVE's performance is compared to that of VoD servers that combine techniques of video flow reuse such as Patching, Chaining, and Batching. The results demonstrate that CVC significantly reduces the amount of video streams that need to be active at the VoD server, provided that the average client's inter-arrival time is lower than the video buffer capacity, using time units. In addition it is shown that a hybrid VoD server approach that combines CVC with Batching improves even more the scalability of VoD servers.

Sumário

1	Introdução	1
1.1	Visão Geral	1
1.2	Contribuições da Tese	6
1.3	Organização da Tese	6
2	Conhecimentos Básicos	7
2.1	Mídia Contínua	7
2.1.1	Padrões de Compressão	8
2.1.2	Modelos de Transmissão	9
2.2	Modos de Distribuição de Mídia Contínua	11
2.3	Vídeo sob Demanda	12
2.3.1	Aplicações	12
2.3.2	Categorias	13
2.3.3	Abordagem Proativa X Reativa	13
2.3.4	Modelo <i>Push</i> X <i>Pull</i>	14
2.3.5	Componentes Básicos de Sistemas VoD	14
2.4	Paradigma cliente/servidor X <i>peer-to-peer</i>	16
2.5	Cache Cooperativa	17
2.6	Considerações Finais	18
3	Cache de Vídeo Cooperativa (CVC)	20
3.1	Granularidade	20
3.2	Controle do Buffer Local	20
3.3	Algoritmo Básico	21
3.4	Interatividade	21
3.5	Extensões Propostas	22
3.5.1	Independência de Formato da Mídia Contínua	22

3.5.2	Suporte a Servidores Convencionais	22
3.5.3	<i>Batching</i> no Cliente	23
4	GloVE	25
4.1	Aspectos Gerais de Implementação	25
4.1.1	Arquitetura Alvo	25
4.1.2	Linguagem e Modelo de Programação	25
4.1.3	Protocolo de Comunicação	26
4.1.4	Bibliotecas Gráficas	26
4.1.5	Funções de CVC Ausentes	26
4.2	Descrição dos Componentes do Sistema	26
4.2.1	Rede de Interconexão	26
4.2.2	Servidor de Vídeo (VS)	27
4.2.2.1	Servidor <i>Randomized I/O</i> (RIO)	27
4.2.3	Cliente da CVC (CVCC)	28
4.2.3.1	Buffer do Cliente	28
4.2.3.2	Threads	29
4.2.3.3	MpegTV Player (MTV)	30
4.2.3.4	Algoritmo de Execução do Cliente	31
4.2.3.5	Estados dos Clientes no Sistema	32
4.2.3.6	Transição de Estados dos Clientes	33
4.2.4	Gerente da CVC (CVCM)	34
4.2.4.1	Estrutura de Controle	34
4.2.4.2	Algoritmo de Execução do Gerente	34
4.2.4.3	Políticas de Escolha de Provedor	34
4.2.4.4	Modos de Operação do Gerente	36
5	Avaliação do GloVE	40
5.1	Ambiente Experimental	40
5.2	Carga de Trabalho	41
5.3	Análise dos Resultados	42
5.3.1	Experimento Base	42
5.3.2	Desempenho Comparativo dos Diferentes Modos de Operação do Gerente	43
5.3.3	Desempenho de CVC e CVC+Batching	51

5.3.4	Comparação do Desempenho de PEEC e PEEP	55
5.4	Discussão	56
6	Trabalhos Relacionados	59
6.1	Técnicas de Reuso de Fluxo	59
6.2	Sistemas P2P	60
7	Conclusões	62
	Referências Bibliográficas	64
	Apêndice	68
A	Campos da Estrutura do Gerente	68
B	Interface Gráfica	70
C	Procedimento de Geração de Carga	71

Lista de Figuras

1.1	Diferenças entre as técnicas	4
2.1	Classes de endereços IP	10
2.2	Necessidade de buffer no cliente	15
2.3	Distribuição de metadados e dados dentre os componentes do sistema . .	16
2.4	Hierarquia convencional de acesso a vídeo	17
2.5	Hierarquia de acesso a vídeo com cache cooperativa	18
3.1	Árvore de encadeamentos gerada pela CVC	21
3.2	Relação entre blocos e GoFs	22
3.3	Árvore de encadeamentos gerada pela abordagem estendida	23
4.1	Diagrama do GloVE	27
4.2	Diagrama do cliente	28
4.3	Transição de estados do cliente no sistema	33
4.4	Efeito de PEEC e PEEP na cooperação entre clientes	35
5.1	Tempos de consumo dos blocos do vídeo (Tempo Médio = 0,69 s/bloco) . .	41
5.2	Canais ocupados no servidor pelos diferentes modos de operação do gerente	44
5.3	Desempenho corrigido descartando fluxos de substituição do servidor . .	45
5.4	Latência de início de exibição	46
5.5	Porcentagem de clientes provendo	47
5.6	Largura de banda agregada ocupada	49
5.7	Clientes que solicitaram substituição de provedor	50
5.8	Taxa de ocupação de canais do servidor para buffers de 32, 64 e 128 pos .	51
5.9	Porcentagem de clientes provendo com buffers de 32, 64 e 128 posições .	52
5.10	Latência média com buffers de 32, 64 e 128 posições	53
5.11	Ocupação da largura de banda da rede com buffers de 32, 64 e 128 posições	54

5.12	Percentagem de clientes que requisitaram novo provedor para buffers de 32, 64 e 128 posições	55
5.13	Gráficos comparativos do GloVE no modo CVC+Batching com PEEC e PEEC	56
B.1	Interface gráfica do cliente	70

Lista de Tabelas

2.1	Padrões MPEG	8
2.2	Aplicações de VoD	12
4.1	Estados dos clientes no sistema	32
4.2	API do Gerente da CVC	35
5.1	Configuração das máquinas	40
5.2	Parâmetros dos experimentos	42
5.3	Percentagem de ocupação média de canais dos servidor	57

Capítulo 1

Introdução

1.1 Visão Geral

Nos últimos anos vem ocorrendo um crescente aumento no interesse das pessoas por conteúdo multimídia¹, sendo este caracterizado por muitos autores como mídia contínua (*Continuous Media - CM*)². Para atender esta demanda, diferentes estratégias de distribuição de CM têm sido investigadas buscando atender aos interesses dos usuários.

No caso ideal, quatro premissas básicas são consideradas ao se acessar um determinado conteúdo multimídia (seja ele de notícias, de entretenimento ou educacional): a) o conteúdo deve estar disponível na íntegra em qualquer momento; b) a exibição deve se iniciar prontamente, ou seja, com tempo de espera desprezível entre a requisição e o início da visualização; c) a exibição deve prosseguir sem interrupções; d) o fluxo da exibição pode ser modificado através de operações de videocassete (VCR), como pausa, avanço e etc. Buscando desenvolver sistemas com estas propriedades, vários esforços de pesquisas foram realizados na área denominada mídia contínua sob demanda (*CM on Demand - CMoD*), englobando técnicas de armazenamento, processamento, acesso e distribuição em tempo real deste tipo de mídia, caracterizada por apresentar tempo de exibição contínuo e grande volume de dados, freqüentemente armazenados de forma comprimida [25].

Um dos campos de CMoD que recebe maior atenção é vídeo sob demanda (*Video on Demand - VoD*). No escopo de sistemas de VoD, um dos principais obstáculos que impedem sua disseminação em ambientes onde a rede de distribuição não é fator limitante para a transmissão, está na inexistência de servidores escaláveis e interativos de baixo custo

¹A mídia é dita múltipla porque texto, sons e imagens são codificados e armazenados de formas distintas [34]

²No decorrer da tese, utilizamos termos na língua inglesa grafados em *italico*, sempre que não for possível uma tradução perfeita para a língua portuguesa ou que sejam de uso comum

capazes de atender às necessidades dos usuários, fato este determinado pelos diversos aspectos expostos a seguir.

De maneira geral, os trabalhos de VoD salientam o fato da largura de banda da rede (*Network Bandwidth* - NB) do servidor ser o principal fator que inibe o atendimento de grandes quantidades de clientes. Como este recurso é limitado e a transmissão de vídeo exige grande largura de banda (Ex.: Vídeo MPEG-1³ [20] usado em vídeo CD (VCD) possui uma taxa média de bits (*bitrate*) em torno de 1,5 Mbps), o servidor tende a oferecer poucos canais lógicos⁴, não conseguindo dar vazão a uma quantidade grande de fluxos simultâneos, sendo o total restrito a aproximadamente de $NB / bitrate$.

Nos sistemas de VoD convencionais, são utilizados fluxos *unicast*, de modo que cada cliente ocupa exclusivamente um canal lógico do servidor, ou seja, cada fluxo atende a um único cliente. Dentre suas principais características, as mais significativas são o suporte a operações de videocassete e o atendimento imediato da requisição do cliente, as quais fazem com que sejam categorizados como *Interactive VoD* (IVoD)⁵. Entretanto, para poder atender grandes quantidades de clientes, servidores especializados de alto custo, dotados de grande largura de banda, são requeridos.

Para contornar esta limitação dos sistemas convencionais, proporcionando um acesso mais eficiente ao servidor, foram desenvolvidas técnicas de *Broadcasting*, onde os fluxos, gerados em intervalos pré-estabelecidos, ficam disponíveis para todos clientes, fazendo com que o sistema apresente alto grau de escalabilidade. Esta abordagem é conhecida como *Near VoD* (NVoD), visto que ocorre uma latência significativa no começo da exibição do vídeo solicitado e que operações de *VCR* ficam comprometidas, modificando o conceito de VoD puro.

Com o intuito de agregar baixa latência e escalabilidade, diversos trabalhos foram propostos. Dentre eles, salientam-se *Batching* [9], *Stream Tapping/Patching* [19, 7] e *Chaining* [39].

No *Batching*, requisições relativas a um determinado conteúdo são enfileiradas até que uma certa quantidade seja atingida, a partir da qual a transmissão se dá através de *multicast* (único fluxo enviado a um grupo de clientes), economizando banda, mas gerando uma latência possivelmente grande nos primeiros clientes que solicitaram o conteúdo,

³Motion Picture Experts Group - 1 (International Standart 11172)

⁴Adotamos a expressão “canal lógico” para definir um conjunto de recursos reservados pelo servidor para empreender a transmissão de um fluxo, sendo o “total de canais lógicos” igual ao máximo de fluxos simultâneos que o servidor consegue transmitir

⁵As categorias de sistemas VoD são melhor explicadas no próximo capítulo

privilegiando os últimos.

Já *Stream Tapping* e *Patching*, duas técnicas semelhantes, partem do princípio de que os clientes têm capacidade de receber pelo menos o dobro do fluxo a ser exibido. Quando um cliente é o primeiro a pedir ao servidor um conteúdo, é criado um fluxo *multicast* completo, onde o conteúdo vai sendo enviado com uma taxa compatível a de consumo do cliente, de modo que um canal do servidor é ocupado por este fluxo durante todo o período da exibição. Já os clientes posteriores, ao solicitarem o mesmo conteúdo, passam a fazer parte do grupo de receptores do fluxo anteriormente criado, sendo o mesmo armazenado temporariamente em um buffer local. Como a parte inicial do vídeo foi perdida, o servidor estabelece um fluxo temporário, referente ao trecho compreendido entre o início do vídeo e a parte já armazenada no buffer, ocupando um canal adicional do servidor por um período relativamente curto.

Apesar de obterem bons resultados, essas técnicas desconsideram o fato do cliente estar retendo em sua memória - realmente ou potencialmente, dependendo da técnica - o conteúdo, devido à necessidade de existir pelo menos um buffer mínimo introduzido para eliminar o *jitter* da rede de transmissão [42]. Considerando esta característica, surge a oportunidade de migrar o sistema para o modelo *peer-to-peer*, solucionando o problema do gargalo existente no acesso ao servidor, intrínseco ao modelo cliente/servidor. A técnica *Chaining* aproveita esta característica, fazendo com que um cliente que possua em seu buffer o conteúdo requisitado por um novo cliente, fique responsável por provê-lo, gerando um encadeamento lógico de buffers no sistema, o que alivia a carga no servidor. A desvantagem de *Chaining* está na impossibilidade de novos clientes reutilizarem o fluxo de encadeamento, quando a parte inicial do conteúdo já tiver sido descartada.

Baseada em uma abordagem híbrida entre *Chaining* e *Patching*, com ênfase no modelo *peer-to-peer*, foi desenvolvida a técnica denominada Cache de Vídeo Cooperativa (CVC) [21, 22, 23]. Essencialmente, CVC consiste em tratar os buffers locais dos clientes como componentes de uma memória global passível de suprir conteúdo para os demais. Com isso, os clientes tornam-se provedores preferenciais de novos fluxos *multicast*, reduzindo a necessidade de acesso ao servidor, sendo que remendos, provenientes sempre que possível de outros clientes, e não do servidor, são enviados para repor seqüências ausentes (a Figura 1.1 apresenta diferenças entre as três últimas técnicas abordadas). A eficiência da CVC baseia-se em duas principais características referentes ao comportamento dos usuários no que tange o acesso à vídeos [2]: a) a grande maioria das requisições concentra-

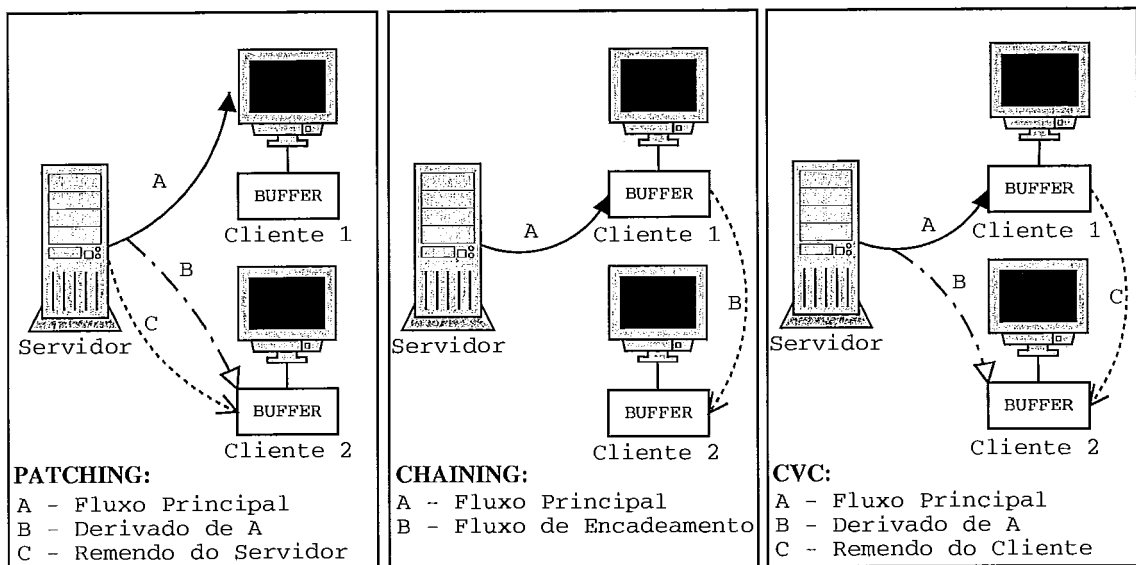


Figura 1.1: Diferenças entre as técnicas

se em poucos títulos; b) os acessos apresentam forte localidade temporal⁶. Resultados preliminares obtidos através de simulação indicam que a CVC tem um grande potencial para melhorar o desempenho e a escalabilidade de sistemas VoD.

Tendo como objetivo principal investigar aspectos práticos de implementação da CVC, bem como o comportamento resultante da introdução de seus conceitos em sistemas convencionais, esta tese implementa e avalia um protótipo de sistema de vídeo sob demanda baseado em cache de vídeo cooperativa.

O protótipo, denominado *Global Video Environment* (GloVE), foi desenvolvido tendo em vista ambientes de rede fechados, com estrutura homogênea, onde a probabilidade de ocorrer perdas de pacotes é desprezível. Isto ocorre em meios empresariais, acadêmicos e similares, onde existe largura de banda simétrica, e abundante o suficiente para que as técnicas de streaming de mídia contínua sejam eficazes.

Buscando adequar o sistema às necessidades dos usuários de interagir com a exibição através de operações VCR, implementamos o recurso de pausa/recomeço⁷, visto que os clientes contam com pelo menos esta funcionalidade em um sistema VoD [3]. Além disso, o sistema foi criado de forma a apresentar baixa latência máxima de início de exibição. Somando estas características, nota-se que o GloVE se aproxima bastante de um sistema IVoD.

Em relação a proposta original de CVC, introduzimos extensões para possibilitar a

⁶Se um vídeo foi acessado recentemente, é provável que seja acessado novamente em breve

⁷Estão previstos como próximos passos do projeto as implementações de avanço e recuo discretos

utilização do RIO[38] como o servidor de vídeo do sistema, visto que este servidor não suporta *multicast*, opera no modelo *pull* e considera o vídeo como uma seqüência de blocos, o que contraria as premissas estabelecidas pela técnica CVC. Uma das alterações mais significativas, consiste na presença de conceitos da técnica *Batching* nos clientes para atingir melhor desempenho em altas taxas de chegada, onde a inexistência de *multicast* no servidor impossibilita que haja reutilização de fluxo até o instante em que o primeiro cliente seja capaz de prover.

Empreendemos também modificações no funcionamento do gerente da CVC, de modo que os clientes são categorizados dentro do sistema através de uma máquina de estados, permitindo simplificar o algoritmo de busca por um cliente (provedor) capaz de fornecer o conteúdo necessário para atender a uma determinada requisição. Além disso, criamos duas políticas distintas de escolha de provedor: PEEC e PEEP. Em PEEC (ênfase em *Chaining*), o gerente sempre tenta atender a uma requisição através da criação de um novo fluxo *multicast* completo a partir de um cliente que ainda não esteja provendo e que não tenha descartado o bloco inicial do vídeo. Caso isto não seja possível, tenta reutilizar um fluxo *multicast* já em curso, que seja capaz de receber um fluxo adicional de remendo de curta duração a partir do servidor. Já em PEEP (ênfase em *Patching*), ocorre o inverso, de modo que o gerente primeiro procura em sua base por um cliente que esteja provendo um fluxo *multicast* passível de receber remendo. Se não encontrar, busca um cliente capaz de prover um novo fluxo *multicast* completo.

A simplificação do algoritmo do gerente também permitiu que desenvolvêssemos diferentes modos de operação, representando combinações de características das três principais técnicas de reuso de fluxo de vídeo empregadas no protótipo: *Chaining*, *Patching* e *Batching*.

Para avaliar o sistema implementado, realizamos testes com diferentes tamanhos de buffers locais e diferentes intervalos entre chegadas de clientes ao sistema, em uma rede *fast ethernet* de 100 Mbps composta por 6 computadores interligados por um *switch* com suporte a *IP multicast*. Os resultados obtidos para o protótipo de CVC baseado nas técnicas *Patching* e *Chaining*, em comparação com a abordagem convencional, demonstram economia de até 94% na ocupação dos canais lógicos do servidor.

Já a versão de CVC estendida com conceitos de *Batching* nos clientes atinge economia ainda maior, tendendo a ocupar apenas um canal lógico do servidor com latência média de início de exibição ligeiramente superior a obtida com a versão anterior. Além disso, os resultados mostram que esta última abordagem alcança um grau de escalabilidade superior

em relação às demais combinações das técnicas que a constituem, considerando mesmo tamanho de buffer local nos clientes e mesma latência máxima, visto que é eficiente para uma faixa maior de taxas de chegada.

1.2 Contribuições da Tese

As principais contribuições da tese são:

- Implementação do GloVE - protótipo escalável e interativo de sistema de vídeo sob demanda baseado na técnica denominada Cache de Vídeo Cooperativa (CVC);
- Introdução de novo modelo de CVC para servidores convencionais, sem capacidade de transmissão *multicast* e baseados no modelo *pull*;
- Introdução de conceitos de *Batching* na técnica CVC original;
- Introdução de uma máquina de estados responsável por categorizar os clientes ativos no sistema;
- Introdução de diferentes políticas na escolha do provedor de conteúdo, denominadas PEEC (ênfase em *Chaining*) e PEEP (ênfase em *Patching*);
- Avaliação de variações do protótipo com diferentes tamanhos de buffer local nos clientes, para diferentes taxas de chegada de clientes ao sistema;
- Comparação entre diferentes versões do protótipo representando diferentes combinações de técnicas usadas por CVC: *Chaining*, *Patching* e *Batching*.

1.3 Organização da Tese

O restante da tese está organizado da seguinte forma. No capítulo 2 descrevemos os conhecimentos básicos pertencentes ao escopo de VoD. No capítulo 3 abordamos a técnica CVC original e extensões propostas. A implementação do protótipo é descrita no capítulo 4. No capítulo 5 apresentamos a metodologia experimental adotada, contendo a descrição do ambiente experimental e da carga de trabalho, seguida da análise dos resultados obtidos. Os trabalhos relacionados aparecem no capítulo 6. Por fim, no capítulo 7 são expostas as conclusões e trabalhos futuros.

Capítulo 2

Conhecimentos Básicos

Para facilitar o entendimento da tese, apresentamos neste capítulo conceitos básicos pertinentes ao seu escopo. Primeiramente, caracterizamos o tipo de conteúdo de nosso estudo: mídia contínua. A seguir, descrevemos aspectos relevantes e diferentes estratégias de projeto de nossa aplicação: vídeo sob demanda. Em seguida, comparamos os modelos cliente/servidor e *peer-to-peer* no que tange a escalabilidade. Por fim, discorremos sobre os benefícios de cache cooperativa, salientando as vantagens de sua aplicação em um sistema de VoD.

2.1 Mídia Contínua

O termo Mídia Contínua (*Continuous Media* - CM) abrange mídias que necessitam ser exibidas durante um determinado intervalo de tempo, normalmente com alguma interação do usuário [45]. A combinação de múltiplas CM forma um conteúdo multimídia¹, costumeiramente composto por áudio e vídeo, os quais demandam grande largura de banda para sua transmissão. Por exemplo, para efetuar uma transmissão digital *stereo* de um CD de áudio, são necessários 1.411 Mbps, o que representa uma quantidade suficiente para esgotar uma linha T1². Em relação a vídeo, tomando como exemplo a abordagem mais simples de transmissão digital que baseia-se na exibição de quadros formados por uma matriz de *pixels*, onde é necessário exibir pelo menos 25 quadros/s para evitar a impressão de congelamento nos movimentos mostrados nas imagens, um monitor configurado no padrão XGA (1024x768) com 24 bits por *pixel* demanda 472 Mbps, duas ordens de grandeza em relação a áudio. Desta forma, fica claro que técnicas capazes de compactar

¹Nesta tese, consideramos mídia contínua e conteúdo multimídia composto por áudio e vídeo como sinônimos, muitas vezes referenciados apenas pela palavra conteúdo

²Linha de transmissão com vazão de 1544Mbps

mídia contínua mantendo a qualidade de visualização e estratégias escaláveis de transmissão são aspectos fundamentais para a construção de sistemas de acesso à mídia contínua eficientes, os quais são abordados a seguir.

2.1.1 Padrões de Compressão

O formato amplamente adotado para transmissão de conteúdo multimídia comprimida segue os padrões MPEG, assim conhecidos por terem sido estabelecidos por um comitê técnico composto por membros da ISO e IEC denominado *Moving Picture Experts Group*, formado em 1988. Seu objetivo inicial era o de criar um padrão para codificação de vídeos com taxas de aproximadamente 1,5 Mbps, apropriado para o transporte através de circuitos de dados T1 e capaz de ser exibido a partir de um CD-ROM [44]. Note que, na prática, isto significa transmitir áudio e vídeo ocupando a mesma largura de banda necessária para transmitir um CD de áudio. No transcorrer dos anos, este grupo definiu vários padrões, buscando atender a determinados nichos de aplicações (Tabela 2.1).

Padrão	Aplicação
MPEG-1	CD-ROM e <i>links</i> T1 a 1.5 Mbps
MPEG-2	DVD e Transmissão de TV Digital entre 4 e 9 Mbps
MPEG-3	HDTV entre 20 e 40 Mbps (abandonado)
MPEG-4	Vídeofone, correio de vídeo, videoconferência e vídeogames
MPEG-6	Transmissão sem fio
MPEG-7	Padrão para informações sobre conteúdo a ser utilizado em buscas
MPEG-8	Descrição de objetos em quatro dimensões

Tabela 2.1: Padrões MPEG

Por apresentar maior suporte de ferramentas, qualidade aceitável de exibição e demanda de largura de banda compatível com nosso ambiente experimental, adotamos em nosso estudo o padrão MPEG-1³, descrito a seguir.

MPEG-1

O MPEG-1 (ISO/IEC 11172) foi o primeiro padrão liberado pelo grupo, tendo como objetivo produzir vídeo com qualidade semelhante a de um videocassete (352 x 240 para NTSC). Ele é composto por três partes: áudio, vídeo e sistema, sendo esta última responsável por multiplexar e sincronizar as duas primeiras.

³Cabe salientar que os conceitos introduzidos também são válidos para os demais padrões voltados ao transporte de mídia contínua com capacidade de acesso aleatório, diferenciando-se apenas por demandar diferente taxa de transmissão e tamanho de buffer

A compressão de áudio é organizada em três níveis: *layer I*, *layer II* e *layer III*⁴. Cada nível sucessivo compreende maior compactação com o custo de maior complexidade na codificação e decodificação [24]. O *stream* de áudio suporta um ou dois canais: único canal, dois canais independentes ou um sinal *stereo* multiplexado.

A compressão de vídeo aproveita a redundância espacial e temporal apresentada pelos filmes. A primeira forma é atacada usando um dos padrões mais empregados para compressão de imagens estáticas, denominado JPEG [46], com o objetivo de codificar os quadros do vídeo separadamente. A segunda, decorrente do fato de quadros consecutivos apresentarem pouca diferença entre si, é abordada com a definição de três tipos de quadros, assim caracterizados:

Intracodificados (I): são imagens autocontidas codificadas usando JPEG.

Preditivos (P): explora a redundância temporal entre as imagens, carregando as diferenças em relação ao último quadro. Devido a isso, atinge maior compressão em relação ao quadro I. Entretanto, só pode ser decodificado com a presença do quadro anterior.

Bidirecionais (B): necessita tanto o quadro anterior quanto o posterior para a decodificação. Atinge a maior compressão, porém exige que os dois quadros citados sejam retidos em um buffer para que o processo de decodificação seja realizado.

O padrão também introduz o conceito de grupo de quadros (*Group of Frames - GoF*⁵). O GoF é composto por uma seqüência de quadros consecutivos, sendo o primeiro do tipo I. Ele é autocontido pois os quadros que dele fazem parte dependem unicamente de outros quadros que também estão presentes no mesmo grupo, servindo como unidade para acesso aleatório e edição.

2.1.2 Modelos de Transmissão

Durante anos, as transmissões *unicast*, onde o fluxo de dados é recebido apenas por um cliente, dominaram os ambientes de rede. Desta forma, o número de clientes atendidos fica limitado pelo valor máximo de fluxos simultâneos imposto pela largura de banda da rede, podendo ser bastante baixo dependendo da taxa de bits dos fluxos. Devido ao crescente aumento no uso de aplicações que demandam grande largura de banda (exemplo:

⁴Amplamente usado na distribuição de músicas, sendo mais conhecido como MP3

⁵Também denominados por outros autores de *Group of Pictures - GoP*

transmissão de mídia contínua), tornou-se necessário adotar soluções de transmissão mais escaláveis, capazes de aproveitar o fato de muitos clientes buscarem o mesmo conteúdo ao mesmo tempo. A forma mais simples de compartilhar a transmissão é através de *broadcast*, onde o fluxo de dados é recebido por todos os clientes. Esta abordagem não é eficiente porque clientes passam a receber dados desnecessários, já que normalmente um mesmo conteúdo é desejado por um grupo de clientes, e não por todos eles. Devido a isso, tornou-se necessário desenvolver um serviço seletivo de transmissão, onde os fluxos sejam recebidos apenas por aqueles clientes que demonstrem interesse pelo conteúdo. Assim, surgiu em 1993 a primeira implementação de *multicast*, incorporada ao 4.4 BSD. Este tipo de transmissão apresenta um comportamento semelhante a rádio e televisão no sentido de que quando um cliente quer receber um determinado conteúdo, basta a ele selecionar um canal. A partir deste momento, passa a receber as informações sendo enviadas para tal canal pelo estação transmissora. Nesta analogia, um canal refere-se a um grupo de *multicast*, o qual possui um endereço especial pré-determinado usado tanto pelo provedor do conteúdo quanto pelo cliente.

Endereços Multicast

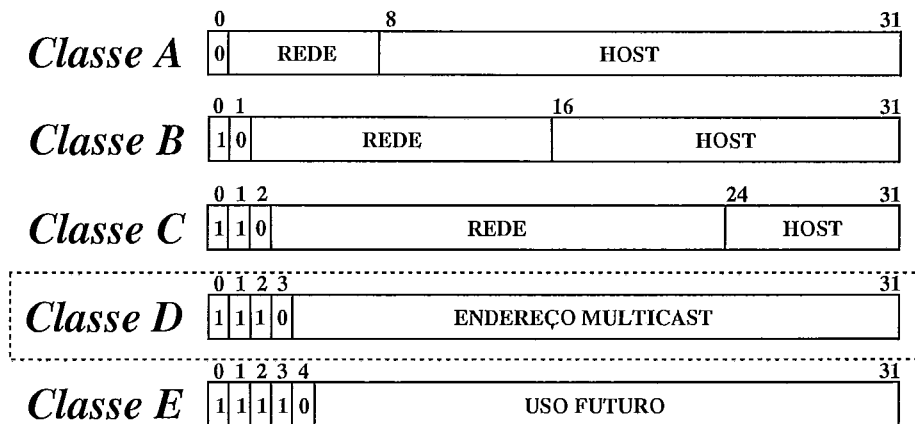


Figura 2.1: Classes de endereços IP

Como pode ser notado na Figura 2.1, os endereços *multicast* ocupam a classe D dos endereços IP, de forma que todo datagrama IP cujo endereço de destino comece com “1110” é um datagrama de *IP Multicast*. Os 28 bits adicionais determinam o grupo.

Gerenciamento de Grupos

Em redes IP, o protocolo mais difundido para gerenciamento de grupos *multicast* é chamado *Internet Group Management Protocol* (IGMP) [10]. No IGMP os roteadores mantêm listas dinâmicas contendo os *hosts* pertencentes aos grupos. Para que isto seja

possível, quando um *host* deseja entrar em um determinado grupo, ele envia uma mensagem IGMP *join* para o roteador, o qual introduz seu endereço na lista do grupo requisitado. Com isso, o *host* passa a receber os pacotes enviados para o endereço deste grupo. Além disso, o roteador periodicamente envia mensagens IGMP *query* para os *hosts* a fim de monitorar quais deles ainda pertencem ao grupo especificado. Na segunda versão do protocolo, IGMP V2 [12], foi introduzida a possibilidade do *host* solicitar explicitamente seu desvinculamento de um determinado grupo, através da mensagem IGMP *leave*.

Infraestrutura Exigida

O ambiente onde o *multicast* atinge maior eficiência é em redes constituídas por hierarquias de comutadores (*switches*).

Com a queda nos custos necessários para sua implantação, a tecnologia de *switch ethernet* vem sendo adotada em diversos ambientes empresariais, acadêmicos e similares. Isto oferece um grande impulso para aplicações multimídia, já que desaparece o problema de colisão inerente ao protocolo CSMA/CD, proporcionando tempos de resposta previsíveis e grande largura de banda. No caso de transmissão de dados através de *IP Multicast*, os *switches* se tornam ainda mais importantes, visto que eles passam a atuar como filtros, de modo que os pacotes sejam enviados apenas para aquelas máquinas que tenham clientes pertencentes ao grupo de recebimento dos mesmos.

Em *switches* de nível 2, o suporte a *multicast* é feito geralmente através da técnica chamada IGMP *Snooping*. Para isso, o *switch* deve monitorar suas portas de forma que, quando cheguem pacotes IGMP referentes a inclusão/exclusão do *host* em um grupo *multicast*, ele possa atualizar as entradas de sua tabela de *multicast*, que é usada no encaminhamento dos datagramas endereçados a tais grupos. Muitos dos equipamentos comercializados atualmente possuem esta funcionalidade.

2.2 Modos de Distribuição de Mídia Contínua

Existem duas possibilidades no envio de mídia contínua para o cliente: *download* e *streaming* [47].

No modo *download* o usuário recebe toda a mídia requisitada antes de proceder com a exibição. Esta abordagem tende a produzir uma latência inaceitável no início da exibição devido ao elevado tempo gasto na transferência, causado pelo grande tamanho deste tipo de mídia⁶.

⁶Vídeo MPEG-1 com duas horas ocupa em torno de 1350 MBytes

Contrastando com esta estratégia, o modo *streaming* permite que a exibição inicie tão logo uma porção mínima do vídeo chegue até o cliente. Desta forma, os demais trechos vão sendo recebidos, decodificados e exibidos em seqüência, caracterizando-se uma aplicação de tempo real. Apesar de praticamente eliminar a latência no início de exibição, esta abordagem introduz maior complexidade na transmissão já que os trechos do vídeo devem chegar no cliente a tempo de serem exibidos, de modo a não provocar interrupções na exibição.

2.3 Vídeo sob Demanda

A aplicação Vídeo sob Demanda (*Video on Demand - VoD*) pode ser caracterizada como uma vídeo locadora eletrônica [45], onde o cliente escolhe o título que deseja assistir a partir de um determinado catálogo. No caso ideal, o cliente começa a assistir o conteúdo requisitado imediatamente, sendo a transmissão realizada no modo *streaming* anteriormente exposto. Além disso, o cliente deve ser capaz de interagir com a exibição, podendo parar, avançar, recuar, e etc dentro da seqüência do vídeo. Implementar um sistema de VoD com todas estas funcionalidades, capaz de atender uma grande monta de clientes, não é tarefa trivial. Isto ocorre porque existe um compromisso entre a ocupação de largura de banda de rede do servidor, latência de início de exibição e o oferecimento de capacidade de interação ao usuário.

2.3.1 Aplicações

Sistemas de VoD encontram diversos nichos de aplicação. Podemos vislumbrar três áreas principais: aprendizado, entretenimento e negócios. Exemplos de aplicações específicas representando as áreas supra citadas são expostos na Tabela 2.2.

Área	Exemplo de Aplicação
Aprendizado	Ensino a distância, treinamento corporativo, ...
Entretenimento	Filmes/Notícias em casa, durante viagens,
Negócios	Propaganda, Catálogos multimídia, ...

Tabela 2.2: Aplicações de VoD

2.3.2 Categorias

Os sistemas de VoD tradicionais podem ser categorizados de acordo com o grau de interatividade que oferecem aos clientes, da seguinte forma:

No VoD (NoVoD): sistema similar ao oferecido pelas emissoras de televisão, baseado em *broadcast*, onde o usuário fica refém da grade de programação estática. Todos os usuários recebem o vídeo pelo mesmo canal.

Near VoD (NVoD): sistema onde múltiplos canais são usados para transmitir o mesmo vídeo com diferenças pré-definidas de início de transmissão, possibilitando ao usuário avançar ou recuar a exibição através da troca do canal. Cada canal geralmente atende a um grupo de usuários.

True VoD (TVoD): sistema em que o usuário solicita o vídeo e é atendido prontamente, sem ter que esperar por um instante de início de transmissão pré-definido. Usuários recebem fluxos por canais diferentes quando as requisições são feitas em momentos diferentes.

Interactive VoD (IVoD): extensão de TVoD onde os usuários possuem controle individual da exibição, sendo capazes de empreender operações de VCR em qualquer ponto do vídeo. Cada usuário ocupa um canal do servidor.

Analisando as características das categorias, nota-se que conforme vão sendo oferecidos mais recursos ao usuário, maior é a demanda por recursos do servidor, sendo que NoVoD e IVoD representam os casos extremos. A NoVoD é a que necessita menos recursos do servidor, podendo assim atender a inúmeros clientes, o que atesta sua escalabilidade, com o ponto negativo de não proporcionar ao usuário liberdade de horário e interatividade ao assistir o vídeo. Já IVoD permite que o usuário assista o vídeo em qualquer tempo e que interaja na exibição, ao custo de levar o servidor a saturação com poucos clientes. Encontrar um nível intermediário entre estas abordagens é a questão chave para obter um sistema de VoD eficiente, de forma que a oferta de interatividade e baixa latência de início de exibição não comprometam a escalabilidade do sistema.

2.3.3 Abordagem Proativa X Reativa

Existem duas abordagens distintas que podem ser adotadas na construção de sistemas de VoD quanto a transmissão dos fluxos de vídeo: proativa e reativa [18].

Na abordagem proativa, as transmissões ocorrem em tempos pré-determinados, sem que um usuário realmente requisite seu início. Analisando as categorias expostas acima, notamos que tanto NoVoD quanto NVoD adotam a abordagem proativa.

Em contrapartida, na abordagem reativa a transmissão se inicia apenas quando chegar ao servidor uma requisição proveniente de um cliente. Esta é a abordagem usada pelas categorias TVoD e IVoD anteriormente discriminadas.

2.3.4 Modelo *Push X Pull*

Além das possíveis abordagens de transmissão expostas acima, outro quesito importante em sistemas VoD é a forma como o vídeo é requisitado ao servidor, que pode ser feita segundo o modelo *push* ou *pull*.

No modelo *push* (ou *Server-push*), o cliente sinaliza o servidor para que a transmissão inicie. Após esta solicitação inicial, o cliente fica apenas recebendo sequencialmente os trechos do vídeo, ficando sob responsabilidade do servidor temporizar o envio de forma que a exibição por parte do cliente possa proceder de forma contínua. De maneira oposta, no modelo *pull* (ou *Client-pull*) o cliente solicita trechos do vídeo conforme vai necessitando, repetindo este procedimento até o final do vídeo. Segundo estudos realizados [31], o modelo *push* tende a suportar de 5% a 20% mais clientes do que o modelo *pull*.

2.3.5 Componentes Básicos de Sistemas VoD

Qualquer sistema de VoD é inerentemente formado por três componentes: servidor de vídeo, responsável por armazenar e transmitir mídia contínua; clientes, os quais solicitam e consomem conteúdos; e rede de comunicação, que serve como meio de interconexão entre os dois primeiramente citados. No que tange ao cliente, algumas considerações adicionais merecem ser expostas.

Cliente

O cliente pode ser um computador ou um *set-top-box*, capaz de executar as tarefas básicas de receber, decodificar e exibir o vídeo. Estas tarefas bastam quando a transmissão do conteúdo for síncrona, ou seja, o trecho a ser decodificado chega sempre no instante exato. Para que isto ocorra, tanto o servidor quanto a rede devem oferecer alta qualidade de serviço (QoS). Na prática, onde redes de pacotes são largamente difundidas, torna-se necessário adotar um mecanismo auxiliar para garantir o correto funcionamento do sistema.

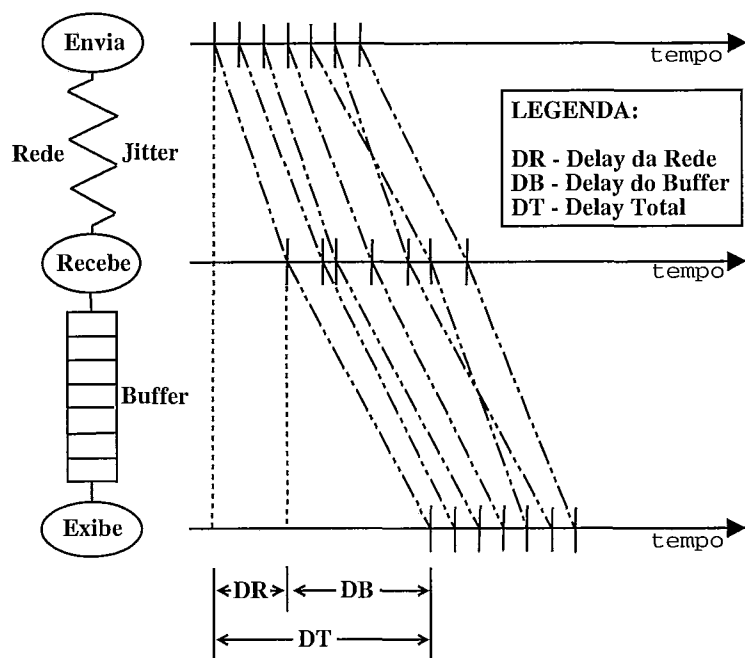


Figura 2.2: Necessidade de buffer no cliente

A Figura 2.2 apresenta o comportamento da transmissão em uma rede de pacotes, mostrando que ocorre um atraso (*delay*) entre o envio dos blocos e o seu recebimento, intrínseco ao tipo da rede, conhecido como *delay* da rede. Além disso, a rede tende a fazer com que os pacotes enviados com frequência constante sejam recebidos em intervalos heterogêneos, e em certas ocasiões em uma ordem diferente da estabelecida no envio, impondo uma variação no atraso denominada *jitter*. Para contornar este problema, é introduzido no cliente um buffer local. Desta forma, caso ocorra atraso (*delay*) na chegada de novos trechos do vídeo, o decodificador pode consumir os blocos já presentes no buffer até que a transmissão se normalize, evitando congelamentos na exibição. No caso oposto, onde trechos chegam antes do previsto, o buffer os acumula de modo que o decodificador possa consumi-los no momento futuro apropriado, evitando que ocorra descarte de conteúdo transmitido. Para que o buffer suporte estas variações, a exibição só é iniciada quando uma quantidade mínima de blocos proporcional ao *jitter* da rede é atingida, sendo o intervalo compreendido entre o recebimento do primeiro bloco e o início da exibição denominado *delay* do buffer, que somado ao atraso do recebimento do primeiro bloco, definem a latência de início de exibição, aqui chamada de *delay* total. É importante salientar que quanto maior o *jitter*, maior a capacidade necessária no buffer.

2.4 Paradigma cliente/servidor X *peer-to-peer*

O problema da escalabilidade no acesso a um conjunto de dados está diretamente ligado a forma como este é provido. Os modelos tradicionais baseiam-se no paradigma cliente/servidor, onde existe um cliente que deseja uma determinada informação e um servidor responsável por armazená-la e transmiti-la quando requisitado. Note que nesta abordagem o número de clientes simultaneamente atendidos fica limitado pelos diversos recursos finitos existentes no caminho crítico a ser percorrido pelos dados entre o local onde estão armazenados no servidor até a chegada no cliente.

Dependendo da aplicação, os próprios clientes podem compartilhar seus dados com os demais, transformando-se em possíveis provedores de conteúdo. A esta nova forma de troca de conteúdo estabelecida entre clientes, denominamos paradigma *peer-to-peer* (P2P), onde o desempenho tende a crescer com a chegada de novos clientes, visto que agregam novos recursos ao sistema. Para descobrir em que clientes os dados estão armazenados, torna-se necessário oferecer um serviço de meta informação⁷. Isto pode ser feito de forma centralizada ou distribuída. A Figura 2.3 resume os diferentes ambientes resultantes da forma como metadados e dados são distribuídos pelos componentes do sistema.

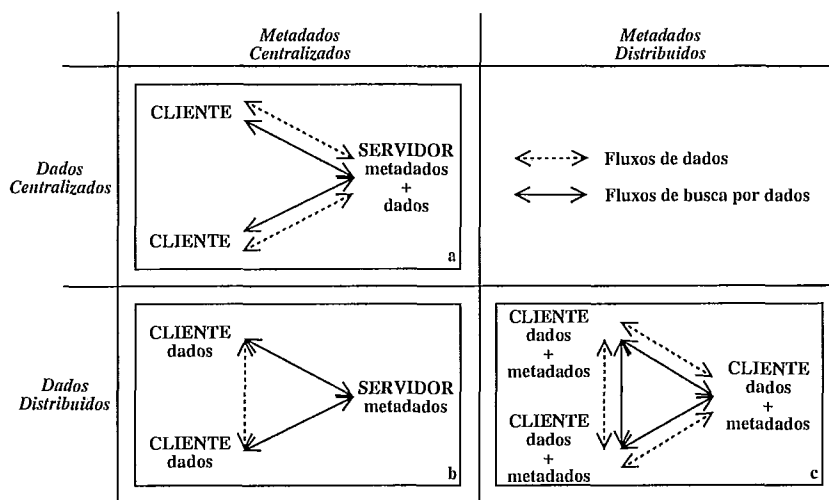


Figura 2.3: Distribuição de metadados e dados dentre os componentes do sistema

Analisando seu conteúdo, nota-se que a Figura 2.3-a reflete um sistema cliente/servidor tradicional. Já a Figura 2.3-b representa um sistema P2P com metadados centralizados. Por fim, a Figura 2.3-c mostra um sistema P2P totalmente distribuído.

⁷Informação sobre a informação

2.5 Cache Cooperativa

No que tange a arquitetura de computadores, o armazenamento de dados é feito segundo uma hierarquia de memórias com latência de acesso crescente. As mais próximas do processador, denominadas cache, apresentam baixa capacidade e alto custo. Sua função é guardar dados anteriormente buscados em níveis superiores da hierarquia (memória RAM, e depois disco), capazes de armazenar, em ordem crescente de grandeza, maior quantidade com menor custo. Como os dados apresentam localidade temporal⁸, muitas buscas subseqüentes são satisfeitas pela cache⁹, diminuindo o tempo médio de acesso.

Com o desenvolvimento de redes com latência inferior as apresentadas pelos discos, surgiu a oportunidade de gerenciar de forma global as memórias remotas [8, 11, 13] para armazenar temporariamente os dados, introduzindo um novo nível na hierarquia (entre RAM e disco), repercutindo em um menor tempo de acesso. Considerando que estas memórias remotas representam caches, e que existe troca de informações entre elas de modo a aumentar os *hits*, forma-se um ambiente de cache cooperativa.

O conceito de cache também é largamente usado por provedores de serviço e organizações conectadas a Internet para diminuir o uso da largura de banda e latência no acesso a documentos por seus usuários [35], sendo o *squid* [41] um dos produtos mais difundidos. Neste contexto, várias estratégias de cooperação de cache foram desenvolvidas [32].

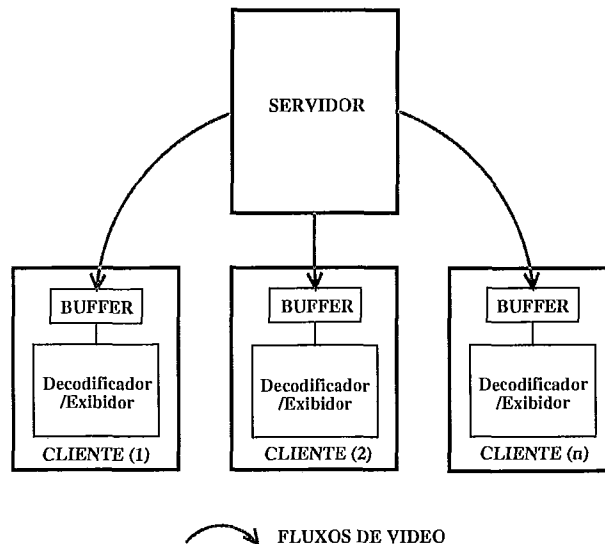


Figura 2.4: Hierarquia convencional de acesso a vídeo

⁸Se um ítem é referenciado, este tende a ser referenciado novamente em breve [29]

⁹Acertos (*hits*)

Em um sistema VoD convencional, a hierarquia de acesso a conteúdo segue o diagrama exposto na Figura 2.4, onde o fluxo provém do servidor de vídeo, sendo inicialmente armazenado em um buffer para eliminar *jitter* e então enviado para um módulo de decodificação e exibição.

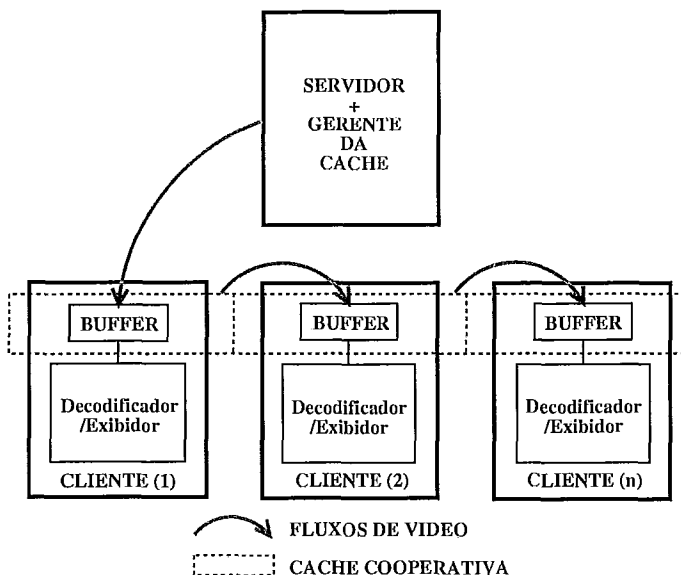


Figura 2.5: Hierarquia de acesso a vídeo com cache cooperativa

Através de um gerenciamento global dos buffers dos clientes, torna-se possível criar uma cache cooperativa, determinando a introdução de um novo nível na hierarquia de acesso (Figura 2.5).

Desta forma, o conteúdo passa a ser primeiramente buscado nesta cache cooperativa, acarretando em uma diminuição na carga do servidor, visto que este tipo de conteúdo também segue o princípio de localidade de referência [1]. Note também que conforme novos clientes chegam ao sistema, a capacidade da cache vai sendo incrementada, causando um aumento na probabilidade de ocorrer *hit*.

2.6 Considerações Finais

Como visto no decorrer deste capítulo, a eficiência da transmissão de mídia contínua depende da compressão do conteúdo. Neste quesito, o padrão amplamente adotado é o MPEG, o qual aproveita a redundância espacial e temporal apresentada pelos vídeos. Para isso, usa três tipos de quadros: I, B e P, os quais formam grupos autocontidos capazes de serem acessados aleatoriamente denominados GoF.

Além disso, foi salientada a necessidade de fazer com que diversos clientes compartilhem um mesmo fluxo através de *multicast*, tendo sido mostrados os protocolos básicos e infraestrutura exigida para sua utilização. Visto que o cliente deseja assistir o conteúdo imediatamente após a requisição, o modo *streaming* de distribuição mostra-se mais interessante para o usuário em relação ao modo *download*.

Vídeo sob Demanda aproveita o modo *streaming* para tender as requisições de vídeo em qualquer instante com baixa latência de início de exibição, sendo aplicável a diversas áreas de aprendizagem, entretenimento e negócios. Foram abordadas quatro categorias de sistemas VoD: NoVoD, NVoD, TVoD e IVoD, as quais possuem aspectos positivos e negativos, indicando a necessidade de encontrar um nível intermediário entre elas para conseguir uma implementação eficiente. Como alternativas de projeto, mostramos as abordagens proativa e reativa - onde a segunda tende a apresentar menor latência, e os modelos *pull* e *push* - onde o primeiro tende a suportar mais clientes simultâneos.

Expusemos também a necessidade de existir um buffer no cliente para eliminar o *jitter* da transmissão e a superior capacidade de atingir escalabilidade do modelo *peer-to-peer* em relação ao cliente/servidor tradicional. Por fim, explicamos a potencialidade de cache cooperativa em um sistema de VoD em diminuir o acesso no servidor de vídeo.

Capítulo 3

Cache de Vídeo Cooperativa (CVC)

A técnica CVC [21, 22, 23] alia, através de um gerenciamento global dos buffers dos clientes, a capacidade de encadeamento de buffers introduzida por *Chaining* [39] com a oportunidade de aplicar remendos em fluxos *multicast* em andamento desenvolvida por *Patching* [19], proporcionando a criação de sistemas VoD escaláveis e interativos. Na versão original, baseia-se na capacidade do servidor de vídeo de enviar fluxos *multicast* no modo *push*.

Neste capítulo, descrevemos as principais características da técnica CVC, as quais serviram como base para a criação do protótipo. Além disso, apresentamos extensões à proposta original, de modo a torná-la aplicável a servidores sem suporte a *multicast* operando em modo *push*, independente do formato da mídia contínua e com conceitos de *Batching* [9] a partir dos clientes.

3.1 Granularidade

A CVC gerencia os buffers dos clientes de forma global, tendo como unidade de acesso o Grupo de Quadros (GoF). Este grupo representa um conjunto autocontido de quadros definido no padrão MPEG, dotado de um *timestamp*, que permite um acesso aleatório aos mesmos.

3.2 Controle do Buffer Local

O controle do buffer local do cliente é feito com 5 ponteiros. Os ponteiros denominados *write* e *read*, indicam, respectivamente, as posições de armazenamento e exibição do próximo GoF. O *begin* aponta para o GoF de *timestamp* mais antigo, sendo atualizado quando ocorre descarte de GoF. Por fim, *low* e *high* determinam os níveis de ocupação

mínimo e máximo do buffer. Quando estes são atingidos, geram alertas para o gerente da CVC, para que este execute as ações necessárias para evitar *underflow/overflow*.

3.3 Algoritmo Básico

O algoritmo básico da CVC segue os seguintes passos. Ao chegar a primeira requisição, o servidor inicia um novo fluxo para o cliente. Tão logo o nível mínimo do buffer do cliente seja atingido, ele começa a exibição. Quando uma segunda requisição chegar, o gerente da CVC procura em sua tabela por um cliente, chamado provedor, que possua a parte inicial do vídeo em seu buffer local, o qual é capaz de armazenar tb segundos.

Caso a segunda requisição tenha sido feita até tb segundos depois de algum cliente ter recebido o primeiro GoF, este pode ser o provedor. Neste caso, o cliente é inserido no grupo de *multicast* servido por este provedor. Como a parte inicial se perdeu, outro cliente do grupo, denominado colaborador, proverá este trecho. Se nenhum provedor estiver disponível, então o último cliente que recebeu a parte inicial ao menos tb segundos antes implementa um encadeamento entre seu buffer e o do requisitante.

3.4 Interatividade

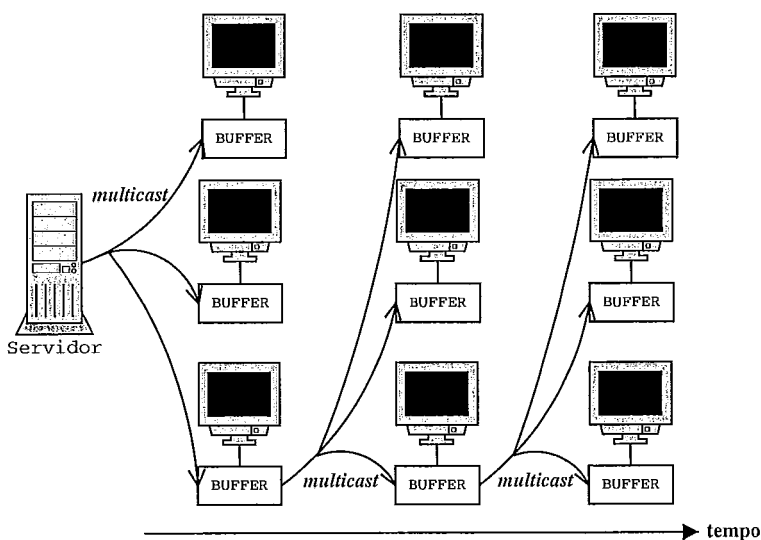


Figura 3.1: Árvore de encadeamentos gerada pela CVC

Conforme os clientes vão chegando, vai sendo gerada uma árvore de encadeamento (Figura 3.1). Como são estabelecidos fluxos em intervalos de tempos defasados de acordo com a capacidade do buffer, surge a oportunidade de oferecimento de funções VCR

aos clientes, da seguinte forma: a) Caso a operação solicitada seja de avanço, o gerente percorre a cadeia em direção ao servidor em busca do novo provedor que está transmitindo blocos adiantados; b) Se for requisitado um retrocesso, basta ao gerente percorrer o caminho inverso.

3.5 Extensões Propostas

Esta seção apresenta modificações efetuadas nos conceitos originais de CVC de modo a facilitar a implementação do protótipo, tornando-o capaz de operar com servidores de vídeo genéricos¹, limitados a transmissão *unicast* segundo o modelo *pull*.

3.5.1 Independência de Formato da Mídia Contínua

Servidores genéricos tratam o conteúdo multimídia como uma seqüência de bytes, agrupados em blocos [40]. Para compatibilizar as operações de transmissão provenientes do servidor e de clientes provedores, optamos por usar o bloco como unidade de transferência, em substituição ao GoF. Desta forma, qualquer tipo de mídia contínua pode se beneficiar de CVC. A desvantagem desta escolha está na limitação do acesso aleatório para MPEG-1. Como muitas vezes o início de um bloco não representa o início de um GoF (Figura 3.2), torna-se necessário um mecanismo para determinar a posição dos GoFs dentro dos blocos, com o intuito de permitir que o usuário acesse qualquer quadro do vídeo.

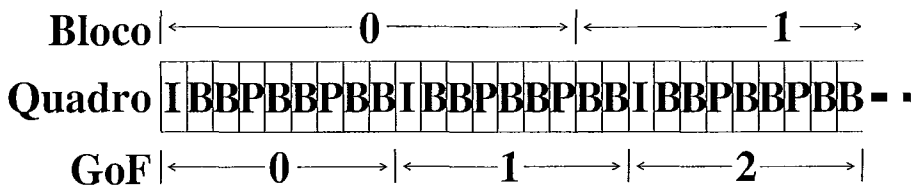


Figura 3.2: Relação entre blocos e GoFs

3.5.2 Suporte a Servidores Convencionais

O algoritmo original da CVC baseia-se na capacidade do servidor de vídeo de transmitir fluxos *multicast* e operar no modelo *push*. Existem diversos servidores que não suportam

¹Independentes do formato do conteúdo

tais funcionalidades. Para tornar a CVC efetiva para esta grande parcela de servidores, efetuamos as seguintes modificações no algoritmo:

1. Quando o cliente acessa o servidor, adota-se o modelo *pull*, solicitando blocos de vídeo de modo a manter seu buffer local sempre cheio;
2. Se no momento em que o segundo cliente requisitar o vídeo o primeiro ainda não tiver descartado o bloco inicial, este se torna provedor de um grupo de *multicast* ao qual o segundo cliente se insere. Note que a transmissão de blocos ocorre segundo o modelo *push*, de forma síncrona a exibição que está ocorrendo no provedor;
3. Conforme mais clientes chegam ao sistema, o gerente escolhe o provedor entre aqueles que já estão transmitindo fluxo capaz de receber remendo ou entre os que ainda não descartaram blocos².

Na prática o que acaba ocorrendo é um simples deslocamento da árvore de encadeamento de modo que sua origem passa do servidor para o primeiro cliente que chega ao sistema, o que pode ser constatado comparando as Figuras 3.1 e 3.3.

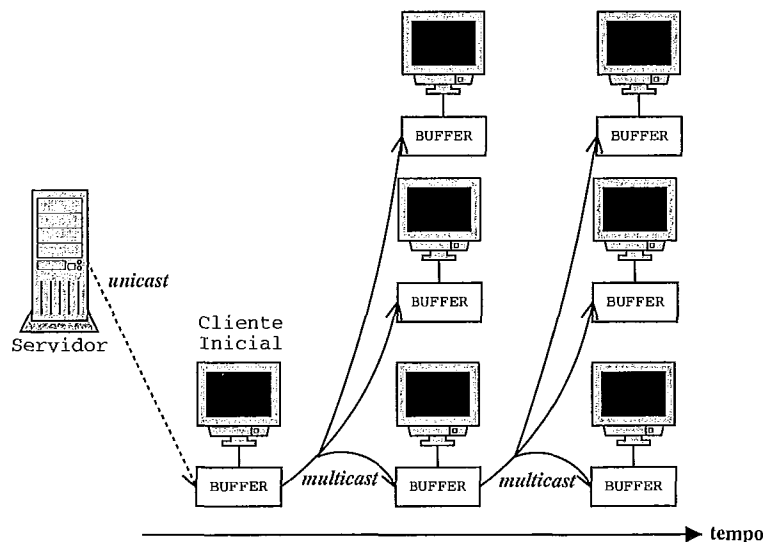


Figura 3.3: Árvore de encadeamentos gerada pela abordagem estendida

3.5.3 *Batching* no Cliente

Como optamos por realizar a transmissão de forma síncrona com a exibição, o cliente só pode começar a servir no momento em que o nível mínimo do buffer é atingido. Esta

²As políticas de escolha são descritas no próximo capítulo

limitação do sistema torna necessário que o servidor sem suporte a *multicast* estabeleça fluxos individuais para prover conteúdo a todos os clientes que cheguem ao sistema antes que o primeiro tenha começado a exibir. Caso a taxa de chegada seja grande, vários fluxos iniciais serão criados.

Para contornar este problema, aplicamos o conceito de *Batching* no sistema da seguinte forma. Conforme os clientes vão chegando, o gerente vai colocando-os no grupo de *multicast* do primeiro cliente (Figura 3.3). Desta forma, no momento em que começa a exibir também inicia a transmitir para seu grupo. Esta abordagem leva a uma duplicação na latência máxima teórica de início de exibição. No entanto, na prática, isto não é um problema significativo pois este valor não ultrapassa duas dezenas de segundos. Além disso, o aumento da latência média é inexpressivo.

Capítulo 4

GloVE

Para avaliar o desempenho das principais idéias adotadas pela técnica cache de vídeo cooperativa foi implementado o protótipo denominado Ambiente de Vídeo Global (*Global Video Environment - GloVE*). O GloVE comporta-se como um sistema P2P com metadados centralizados. Isto se deve ao fato dos clientes realizarem solicitações de conteúdo a um componente centralizado - o gerente - que armazena apenas informações sobre os locais onde este conteúdo está distribuído, de forma que a transmissão real dos dados é sempre que possível realizada a partir de outro cliente. Neste capítulo, apresentamos os principais aspectos de implementação do sistema.

4.1 Aspectos Gerais de Implementação

Nesta seção são abordados itens do projeto do sistema GloVE relacionados à arquitetura de software e hardware para o quais o protótipo foi concebido, a linguagem e modelo de programação empregados e o protocolo de comunicação e bibliotecas gráficas adotadas.

4.1.1 Arquitetura Alvo

O sistema foi desenvolvido tendo como foco o sistema operacional Linux, usando ferramentas disponíveis na distribuição 6.2 da Redhat [33], com *kernel 2.2.X*, rodando sobre hardware compatível com X86.

4.1.2 Linguagem e Modelo de Programação

O sistema foi programado prioritariamente em linguagem C padrão (ANSI/ISO C). No entanto, foram necessários alguns trechos de código C++ no cliente, devido ao servidor adotado, descrito na próxima seção, possuir uma API nessa linguagem. Na criação do

cliente, adotamos o modelo de programação concorrente, utilizando *POSIX threads* [5], para permitir sobrepor computação com comunicação.

4.1.3 Protocolo de Comunicação

O protocolo UDP é usado em todas as comunicações, tanto de dados quanto de controle. Nas de dados, UDP foi adotado porque ele funciona diretamente sobre *IP Multicast* [10] e pelo seu pequeno *overhead*. Nas de controle, UDP facilitou a implementação do gerente da CVC, serializando os acessos à sua estrutura de forma a evitar os custos de criação de seções críticas.

4.1.4 Bibliotecas Gráficas

Uma interface gráfica simples para o cliente¹ foi moldada no ambiente de desenvolvimento de interfaces Glade [14], sendo posteriormente desenvolvida diretamente com a biblioteca GTK+ [17].

4.1.5 Funções de CVC Ausentes

Como o sistema foi desenvolvido com o intuito principal de provar a capacidade de CVC em atingir escalabilidade, postergamos para trabalhos futuros a implementação de funções que não alterassem significativamente as medições de desempenho do sistema. Devido a isso, estão ausentes nesta versão do sistema as seguintes funções definidas na proposta original de CVC: cliente colaborador, responsável por enviar fluxos de remendo; e as operações VCR de avanço/recuo discreto e contínuo.

4.2 Descrição dos Componentes do Sistema

Esta seção descreve os componentes do sistema, os quais aparecem de forma abstrata no diagrama exposto na Figura 4.1.

4.2.1 Rede de Interconexão

O sistema foi projetado tendo em vista sua utilização em ambientes de rede onde os componentes seguem uma topologia totalmente conectada [30] através de comutadores (*switches*) ou conjuntos dos mesmos distribuídos hierarquicamente de modo a criar um meio

¹No Apêndice 2 é apresentada a interface e seus componentes

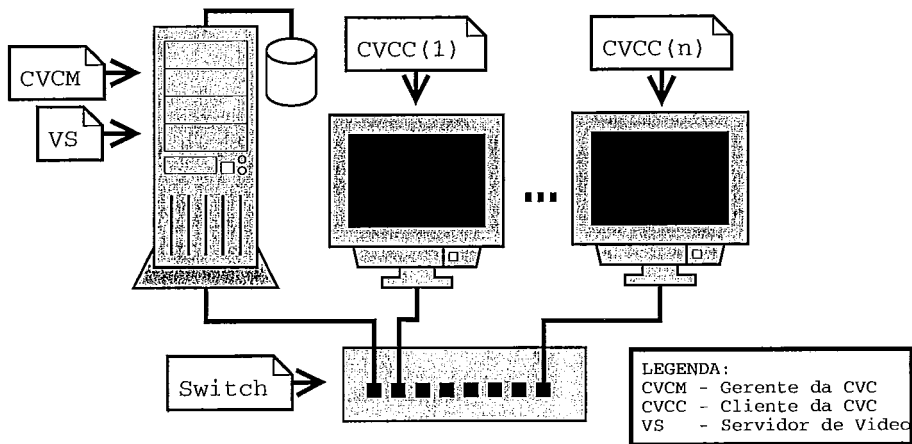


Figura 4.1: Diagrama do GloVE

com banda passante agregada suficientemente grande para não representar um gargalo para os fluxos transmitidos. Além disso, é crucial o suporte desta arquitetura a *multicast*, de modo que os pacotes enviado para um determinado endereço de grupo só cheguem aos clientes nele inscritos.

4.2.2 Servidor de Vídeo (VS)

A implementação do protótipo foi feita de forma que, com pequenas alterações, qualquer servidor de vídeo possa ser usado.

4.2.2.1 Servidor *Randomized I/O* (RIO)

O servidor adotado no desenvolvimento do protótipo é denominado *Randomized I/O* (RIO) [37]. O RIO serve como exemplo de sistema de vídeo convencional por trabalhar apenas com *unicast*. Dentre suas principais características, cabe destacar que ele trata os objetos (ex.: vídeo) como uma seqüência de blocos de 128KBytes. Para atingir recuperação de dados em tempo real, o RIO armazena os blocos aleatoriamente, de modo a garantir atrasos estatísticos. A comunicação do cliente com o servidor se dá através de uma API que permite criar sessões, onde a taxa de bits a ser empreendida na transmissão é definida na inicialização. Na prática, isto faz com que o servidor reserve recursos suficientes para garantir que os blocos sejam enviados com pelo menos a taxa configurada. Após aberta a sessão, o cliente passa a requisitar blocos, um por vez, de acordo com seu número na seqüência. Caso vários blocos sejam requisitados em uma taxa superior a definida na abertura da sessão, o servidor envia o conteúdo de forma *burst* dependendo da

disponibilidade de seus recursos. O controle de admissão do sistema é feito através das taxas fixadas na abertura das sessões. Quando a soma das taxas de sessões abertas atinge a largura de banda total, nenhuma outra pode ser iniciada até que outra seja encerrada.

4.2.3 Cliente da CVC (CVCC)

O CVCC funciona basicamente como controlador de um buffer no lado do cliente (Figura 4.2), tendo como tarefas requisitar blocos para o buffer, enviar os blocos para o software de decodificação/exibição, além de transmitir blocos por *multicast* quando sinalizado pelo gerente da CVC (CVCM).

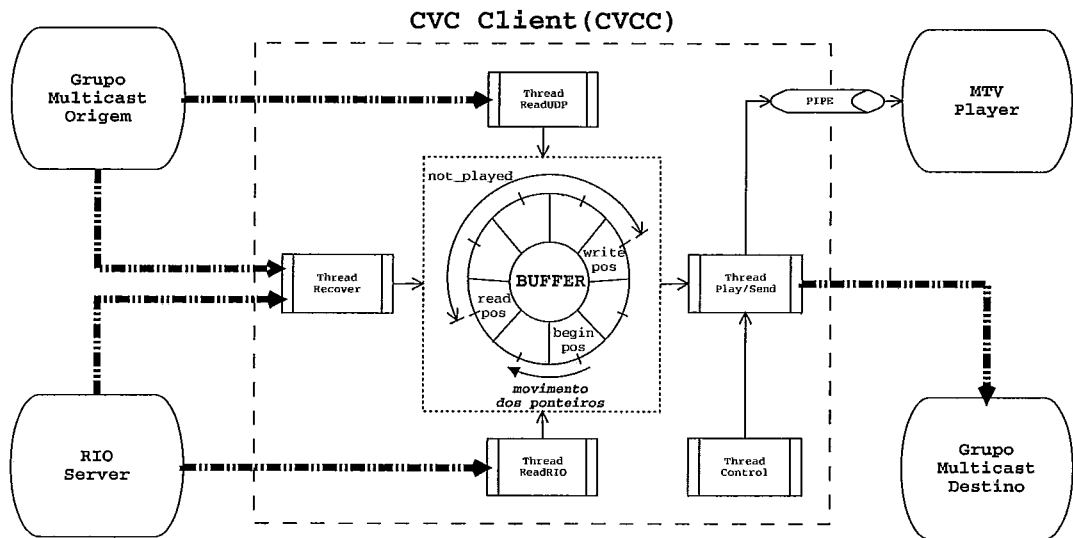


Figura 4.2: Diagrama do cliente

4.2.3.1 Buffer do Cliente

O buffer do cliente é circular e alocado de forma contígua como um vetor, sendo a sua capacidade total, medida em posições, denominada tamanho do buffer (BS). A forma como ele foi concebido difere, em alguns aspectos, da proposta original da CVC, principalmente no que se refere a granularidade de armazenamento e quantidade de ponteiros.

Enquanto na original cada posição do buffer contém um GoF, na implementada cada posição armazena uma estrutura composta pelos campos Block e Buffer. O primeiro funciona como identificador do bloco. O segundo guarda o bloco propriamente dito, cujo tamanho é de 128KBytes. Esta abordagem foi adotada para compatibilizar a granularidade da CVC com a do servidor. Em virtude disto, ocorre também um desvinculamento da CVC em relação ao tipo de vídeo, o que proporciona um suporte direto a qualquer padrão,

desde que exista um decodificador capaz de se comunicar com o CVCC através de algum mecanismo de IPC².

No que tange a ponteiros, foram implementados os de leitura (*read_pos*) - referente a posição do próximo bloco a ser enviado para exibição, de escrita (*write_pos*) - referente a posição que armazenará o próximo bloco a ser recebido, de envio (*begin_pos*) - aponta para a posição do bloco mais antigo e de remendo (*patch_pos*) - referente a posição onde o bloco de remendo é inserido. O ponteiro de underflow (*low*) foi substituído por um contador (*not_played*) que mantém a quantidade de blocos recebidos e não enviados para exibição.

Dois limites foram introduzidos: Limite de Pré-busca (PL) e Limite de Descarte (DL). O PL define o número mínimo de blocos que devem estar presentes no buffer para que se possa iniciar a exibição. Já o DL determina o momento em que blocos já enviados para exibição, mantidos no buffer para fins de cooperação, começam a ser eliminados, a fim de liberar espaço para o recebimento de novos. Como o pior caso ocorre quando o cliente recebe os blocos de um provedor, onde a taxa de chegada de blocos é, em média, igual a de exibição, a capacidade do buffer fica esgotada no momento em que é exibido o bloco (BS - PL). Devido a variação existente nos tempos de consumo de blocos contíguos e ao jitter da rede, foi introduzida uma margem de folga de 4 blocos (DL = BS - PL - 4). Com isso, no caso geral, o descarte inicia no momento em que o bloco DL é exibido, o que limita a janela de cooperação³ neste valor. Entretanto, no caso do cliente se tornar provedor, o descarte começa imediatamente.

Para controlar o acesso concorrente ao buffer, foi utilizado o algoritmo para buffers limitados usando semáforos descrito em Andrews [4], levemente modificado com a introdução da política de descarte exposta acima.

4.2.3.2 Threads

Para executar as tarefas, foram criadas cinco threads: ReadRIO, ReadUDP, Play/Send, Control e Recover.

ReadRIO: thread responsável por, através da respectiva API, abrir uma sessão com o servidor, requisitar os blocos e armazenar os mesmos no buffer.

ReadUDP: thread que tem por função aderir a um grupo de *multicast* determinado pelo CVCCM, controlando a chegada dos blocos através de *headers*. Isto ocorre da se-

²Comunicação entre processos

³Intervalo de tempo no qual o cliente pode criar novo *multicast*

guinte forma: antes de enviar os blocos, fragmentados em pacotes de 1KByte, o provedor envia uma mensagem contendo uma estrutura com os seguintes campos: magic number - número pré-definido que representa o início do *header*, video id - identificação do vídeo que está sendo transmitido, block id - identificador do bloco a ser transmitido e, novamente, magic number - representando fim do *header*. Com isso, a thread consegue saber se o bloco recebido é o esperado. Caso ocorra perda de bloco, desbloqueia a thread Recover.

Play/Send: thread que fica bloqueada até que um número mínimo de blocos (*minPrefetch*) estejam no buffer. Ao atingir este limite, Play/Send cumpre dupla função. A primeira é ler blocos do buffer e enviá-los, através de um *pipe*, para o decodificador. A outra é enviar blocos para um endereço *multicast* definido pelo CVCM conforme sinalização da thread Control. A transmissão se dá de forma síncrona em relação ao envio de blocos para o *pipe*. Desta forma, a taxa de envio é a mesma com que os dados são consumidos, simplificando o controle da transmissão.

Control: thread que fica a espera de uma sinalização do gerente, fazendo com que este cliente se torne provedor. A sinalização é feita com o envio, por parte do gerente, da identificação do primeiro bloco que o cliente deve transmitir. Ao receber o sinal, avisa a Play/Send para começar a enviar.

Recover: thread que funciona atrelada a ReadUDP. Quando ocorre algum problema no fluxo de recebimento de blocos, cabe a Recover contatar o gerente em busca de um novo provedor. Caso exista, a thread ReadUDP é reinicializada, passando a receber o fluxo de um novo grupo *multicast*. Se não existir, realiza novos contatos até que o contador *not_played* atinja um nível mínimo. Se isso ocorrer, uma nova sessão é aberta com o servidor.

4.2.3.3 MpegTV Player (MTV)

O decodificador usado no desenvolvimento chama-se MpegTV Player (MTV) [26]. Este software decodifica fluxos MPEG-1, suportando diversas formas de recebimento de fluxo, dentre elas a *stdin* (entrada padrão). Com isso, o CVCC cria, através da chamada de sistema *fork*, um novo processo que, a seguir, realiza a chamada de sistema *exec* para iniciar o MTV com o parâmetro que força a leitura a partir da entrada padrão. Após, o CVCC cria um *pipe*, redirecionando a saída do mesmo para a *stdin*, o que proporciona o mecanismo de comunicação entre os dois processos.

4.2.3.4 Algoritmo de Execução do Cliente

O cliente segue os seguintes passos:

1. Contata o gerente, requisitando um provedor e uma entrada na estrutura da CVC
2. Se existir provedor disponível, o gerente retorna três identificadores, referentes a: posição ocupada pelo provedor na estrutura (*pro_pos*), entrada alocada para o novo cliente (*cli_pos*) e primeiro bloco a ser lido (*TempBlock*). Se nenhum provedor puder ser usado, o gerente retorna um valor inválido em *TempBlock*
3. Se *TempBlock* for válido, inicializa as threads Recover e ReadUDP, usando *pro_pos* como índice para o cálculo do endereço IP do fluxo *multicast* a ser recebido, e criando posteriormente a thread Play/Send. Se *TempBlock* for inválido, recebe então o número total de blocos do vídeo
4. Caso *TempBlock* seja diferente de zero, torna-se necessário criar um fluxo do servidor, o que é feito através da inicialização da thread ReadRIO, passando *TempBlock* como parâmetro. Com isso, a thread fica encarregada de suprir o buffer com a seqüência de blocos compreendida entre zero e *TempBlock-1*, considerada um remendo no caso de estar apenas recuperando blocos iniciais perdidos ao aderir a um fluxo já em andamento
5. Quando o nível do buffer atinge *minPrefetch*, a thread Play/Send é desbloqueada, começando a exibição
6. Contata novamente o gerente, avisando que começou a exibir e que já pode prover
7. Envia os blocos em seqüência para o *pipe*, 1KB por escrita⁴
8. Caso o gerente sinalize a thread Control, passa a enviar blocos para o endereço IP determinado por *cli_pos*
9. Se o contador *not_played* atingir um nível mínimo, a thread Recover é desbloqueada

⁴Valor escolhido por ser divisor de 128KB e, incluindo os bytes usados pelos cabeçalhos IP e UDP, ser inferior a *maximum transmission unit* (MTU), evitando fragmentação [43]

4.2.3.5 Estados dos Clientes no Sistema

No GloVE, os clientes são categorizados de acordo com os blocos que estão armazenados em seu buffer local e por estar ou não provendo conteúdo através de um fluxo *multicast*.

Quando chega ao sistema, depois de requisitar um vídeo ao gerente, o cliente passa a receber os blocos a partir de outro cliente ou do servidor, caracterizado como estado 1. Cabe dizer que enquanto seu buffer local não atinge o limite de ocupação mínima para início de exibição, este cliente não pode prover fluxo *multicast* devido a transmissão estar atrelada a frequência com que os blocos são consumidos durante a reprodução.

Se não receber aviso do gerente para começar a prover tão logo possível, no momento em que o nível mínimo é atingido o cliente passa para o estado 2, onde apenas exibe os blocos. Se além de exibir também estiver provendo, passa para o estado 3, onde cada bloco antigo é descartado após ser enviado. Note que neste caso o fluxo *multicast* sendo provido pode ser reusado por um novo cliente enquanto o bloco atualmente em curso seja inferior ao nível máximo para recebimento de remendo, que equivale a metade do tamanho do buffer (BS).

Como o buffer possui uma capacidade limitada, quando a ocupação atinge um limite máximo ocorre a necessidade de descartar blocos antigos já exibidos com a intenção de liberar espaço para os novos blocos a serem recebidos. Caso este limite seja alcançado com o cliente apenas exibindo, ele passa ao estado 4, deixando de possuir o primeiro bloco do vídeo.

Quando o fluxo *multicast* sendo enviado não pode mais receber remendo, passa para o estado 5. Caso ocorra problemas no fluxo sendo provido, passa para o estado 6, procedendo apenas com a exibição, ficando impedido de continuar provendo.

A Tabela 4.1 resume os possíveis estados dos clientes no sistema.

Estado	Descrição
1	Recebendo blocos iniciais (pré-busca)
2	Exibindo sem descartar primeiro bloco
3	Provendo fluxo passível de receber remendo
4	Exibindo e descartando bloco mais antigo
5	Provendo fluxo não reutilizável
6	Exibindo mas impedido de continuar provendo

Tabela 4.1: Estados dos clientes no sistema

4.2.3.6 Transição de Estados dos Clientes

A transição de estados dos clientes no sistema ocorre segundo a Figura 4.3, determinada pelos eventos descritos a seguir:

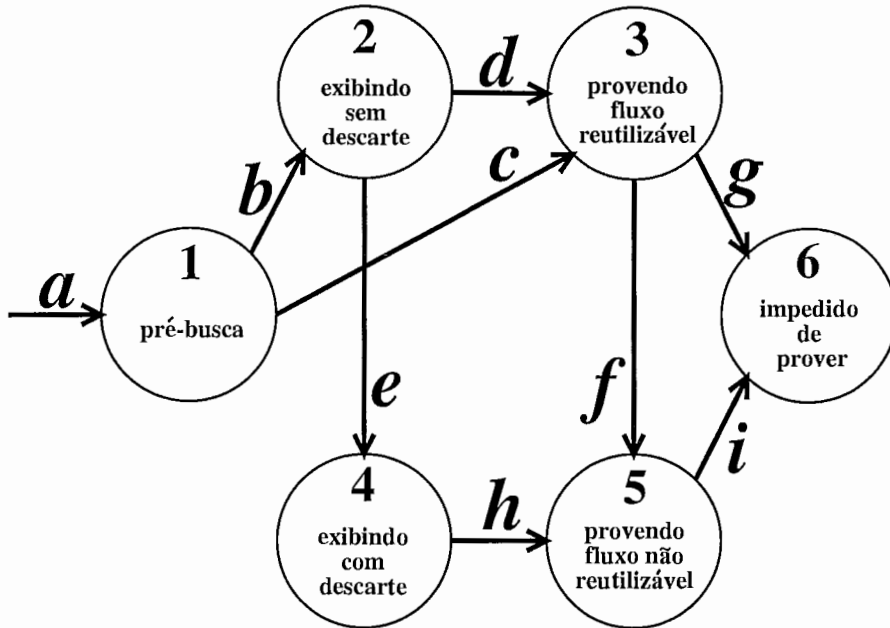


Figura 4.3: Transição de estados do cliente no sistema

- a) cliente requisita um vídeo
- b) o buffer atinge ocupação mínima para exibição sem receber ordem para prover
- c) o buffer atinge ocupação mínima, tendo clientes enfileirados em seu grupo de *multicast*
- d) o cliente recebe aviso para iniciar transmissão antes de descartar primeiro bloco
- e) o cliente começa a descartar blocos sem ordem para prover
- f) o fluxo sendo transmitido atinge nível máximo capaz de receber remendo
- g) todos os clientes providos por o seu grupo requisitam substituição de provedor devido a atrasos no fluxo ou perda de blocos
- h) o cliente recebe aviso para iniciar transmissão usada para prover outro cliente que requisitou substituição de provedor
- i) vide g

4.2.4 Gerente da CVC (CVCM)

O gerente mantém uma estrutura contendo informações sobre os clientes ativos no sistema. Baseado nestas informações, ele é capaz de incorporar novos clientes de forma a reutilizar ao máximo o conteúdo presente na cache cooperativa, evitando ao máximo o acesso ao servidor.

4.2.4.1 Estrutura de Controle

A estrutura de controle é um vetor cujas entradas vão sendo ocupadas pelos clientes que chegam no sistema. Nela são agrupadas todas as informações necessárias para que o gerente esteja a par do estado do cliente no sistema, dos blocos atualmente armazenados nos buffers, entre outras⁵.

4.2.4.2 Algoritmo de Execução do Gerente

O gerente funciona como um servidor que fica a espera de requisições em uma porta pré-definida (6554). Quando uma mensagem é recebida pelo socket UDP, é realizada a atualização das informações que dependem do fator tempo, através de cálculos baseados no tempo atual e na taxa média de consumo de blocos (*update_rate*). Por exemplo, o cálculo do bloco que um determinado cliente está exibindo (*block_playing*) é feito com a seguinte fórmula:

$$block_playing = \frac{tempo_atual - start_play}{update_rate} + first_block$$

Após isso, um *switch*⁶ faz o desvio da execução para a função correspondente a chamada recebida, dentre as expostas na Tabela 4.2, as quais constituem a API de interação entre os clientes e o gerente.

4.2.4.3 Políticas de Escolha de Provedor

O gerente tem como função coordenar o reuso dos buffers locais dos clientes. Logo, quando for requisitado um fluxo partindo de um determinado bloco inicial, o gerente consulta sua estrutura de informações sobre os conteúdos dos buffers e estados dos clientes para, baseado nas políticas apresentadas a seguir, escolher o cliente que ficará responsável por prover o conteúdo.

Provedor de Novo Cliente

⁵No Apêndice A estão expostos todos os campos da estrutura de metadados sobre clientes do gerente

⁶Estrutura de desvio da linguagem C

Chamada	Funcionalidade
search	retorna o ip de um cliente que ainda não descartou blocos
update	atualiza as informações da estrutura da CVC
create_entry	aloca uma entrada para o cliente na estrutura
include_provider	avisa ao gerente que exibição começou e que já pode prover
include_client	busca um provedor e aloca entrada
replace_provider	busca um novo provedor
delete_provider	desaloca entrada na estrutura
pause	avisa o gerente que foi feito pause/resume
start_discard	avisa o gerente que começou a descartar

Tabela 4.2: API do Gerente da CVC

Baseado nas informações contidas na estrutura, diferentes políticas podem ser utilizadas na escolha de qual cliente se tornará provedor do novo cliente, recém chegado ao sistema.

Diferentemente do protocolo original, o protótipo oferece diferentes possibilidades que influenciam na escolha do provedor. Dentre elas, pode-se optar por sempre criar um novo fluxo *multicast* primeiro, ou seja, toda vez que houver um cliente que ainda não começou a descartar blocos, um novo fluxo *multicast* será criado para atender novos clientes, denominada PEEC (Política de Escolha com Ênfase em *Chaining*). Em contra partida, pode-se escolher sempre reutilizar fluxos que ainda sejam passíveis de receber remendo, nomeada PEEP (Política de Escolha com Ênfase em *Patching*). A Figura 4.4 compara o efeito destas políticas no funcionamento do sistema.

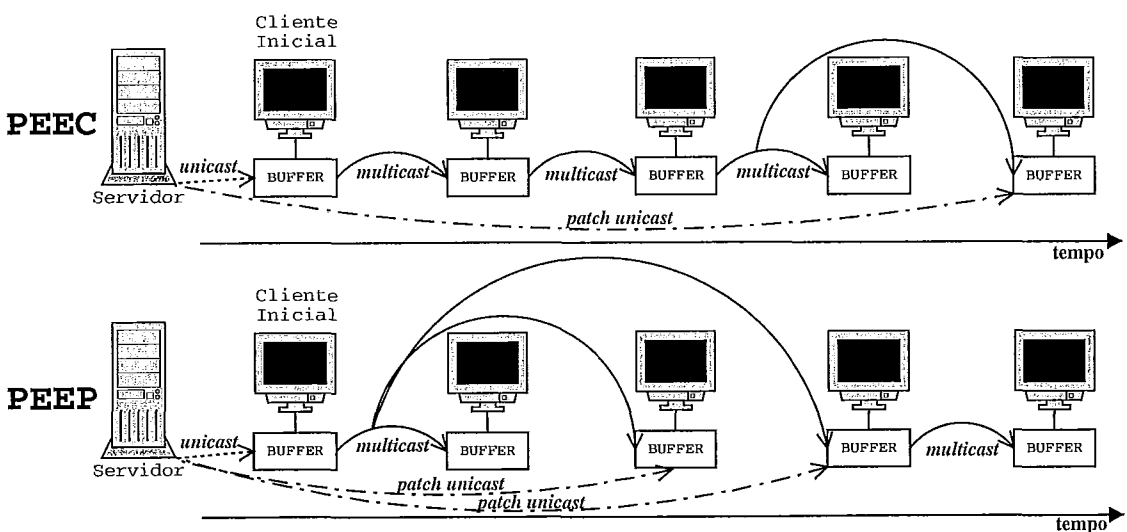


Figura 4.4: Efeito de PEEC e PEEP na cooperação entre clientes

Note que a adoção de PEEC ou PEEP representa um compromisso entre a redução na ocupação da largura de banda agregada da rede e o aumento na complexidade do gerente para tratar rompimentos no encadeamento dos buffers dos clientes. Quando PEEC é usada, são criadas longas cadeias de clientes, de forma que vários fluxos *multicast* transmitindo trechos diferentes do vídeo ficam ativos, ocupando grande largura de banda. Entretanto, como na maioria das vezes apenas um cliente é provido por outro cliente, a tarefa do gerente de recompor uma interrupção no encadeamento é facilitada, visto que ele terá que encontrar um único novo provedor. Já com PEEP, poucos fluxos *multicast* ficam ativos, ocupando pouca largura de banda. Em contra-partida, a tarefa do gerente torna-se mais complexa, visto que os fluxos atendem a vários clientes que precisarão de novo provedor caso o cliente que está transmitindo o fluxo não possa dar prosseguimento a esta operação. Além disto, em PEEP são necessários muitos fluxos de remendo que, devido a não estar implementada no GloVE a função de colaborador nos clientes, são requisitados ao servidor, ocupando seus recursos.

Outras opções se referem a, quando houver mais de um cliente com o mesmo status, qual deles será usado, entre o mais novo e o mais antigo, de acordo com seus tempos de chegada ao sistema.

Provedor Substituto

Quando ocorre algum problema no recebimento de blocos do provedor, o cliente gera alertas para o gerente requisitando novo provedor. Ao receber o pedido, o gerente busca em sua base de dados um cliente que satisfaça alguma das seguintes condições, pela ordem:

1. Esteja provendo o bloco imediatamente anterior ao último bloco recebido pelo cliente
2. Possua status igual a 2 e tenha o último bloco em seu buffer
3. Possua status igual a 4 e tenha o último bloco em seu buffer

Os alertas continuam a ser emitidos, em intervalos de um segundo, até que um novo provedor seja encontrado ou `not_played` atinja 3 blocos. Neste segundo caso, o gerente libera um novo fluxo do servidor.

4.2.4.4 Modos de Operação do Gerente

Como a versão estendida de CVC está fundamentada nas técnicas *Patching*, *Chaining* e *Batching*, implementamos o protótipo de forma que diferentes combinações das mesmas

possam ser escolhidas, possibilitando uma análise comparativa de desempenho entre elas.

Desta forma, definimos cinco modos de operação do gerente quanto a escolha do cliente que atuará como provedor de um cliente que chegue ao sistema, os quais são caracterizados a seguir:

Modo Convencional: Neste modo o sistema comporta-se como um sistema convencional, onde cada cliente é atendido pelo servidor através de um fluxo *unicast*. Para tanto, o gerente é configurado para sempre estabelecer novos fluxos a partir do servidor quando da chegada de novos clientes ao sistema.

Modo Chaining: Este modo é baseado na técnica *Chaining*, onde o reuso de fluxo se dá através do encadeamento dos buffers dos clientes. No GloVE, um novo cliente pode ser encadeado com o buffer de um cliente existente no sistema categorizado no estado 2, ou seja, esteja apenas exibindo sem ter descartado o bloco inicial. Caso não exista um cliente nesta condição, o gerente libera o servidor para prover este novo cliente. Desta forma, o algoritmo do gerente no modo Chaining possui os seguintes passos:

1. Escolhe o mais novo cliente entre os de status igual a 2
2. Libera um novo fluxo *unicast* do servidor

Modo Batching: Neste modo, a única oportunidade de reuso de fluxo ocorre através do enfileiramento de clientes, baseado em conceitos da técnica *Batching*, em um grupo *multicast* que começará a ser provido em um determinado instante. No GloVE, este enfileiramento ocorre quando um cliente está fazendo a pré-busca, ou seja, esteja no estado 1. Caso cheguem novos clientes ao sistema enquanto exista algum cliente nesta condição, o gerente efetua a introdução destes novos clientes no grupo atendido pelo cliente em pré-busca. No momento em que a pré-busca se encerra⁷, ele começa a exibir o vídeo e também a transmitir os blocos iniciais para seu grupo *multicast*. Note que o tempo de duração da pré-busca depende da origem dos blocos. Caso estejam vindo do servidor, chegam de forma *burst*, fazendo com que o cliente fique poucos segundos no estado 1. Já se estiverem sendo enviados por outro cliente, chegam na mesma taxa com que são exibidos, repercutindo em um pequeno aumento no tempo de pré-busca em relação a condição anterior. Como de

⁷Primeiros 16 blocos estão presentes no buffer

maneira geral o tempo no qual os clientes ficam no estado 1 é pequeno, o reuso de fluxo só ocorre quando o intervalo entre chegadas de clientes for pequeno, ou seja, a taxa de chegada de clientes deve ser grande para que esta abordagem torne-se efetiva. Resumindo, no modo *Batching* o gerente comporta-se da seguinte forma para escolher o provedor:

1. Escolhe o mais antigo entre os clientes de status igual a 1
2. Libera novo fluxo *unicast* do servidor

Modo Patching+Batching: Este modo é baseado principalmente na técnica denominada *Patching*, onde clientes reutilizam fluxos *multicast* em andamento através de fluxos complementares de curta duração, chamados de remendos, responsáveis por suprir os blocos anteriormente enviados. No GloVE, os que podem receber remendo são aqueles enviados por clientes que estejam no estado 3. Como um cliente só entra neste estado quando começa a prover, agregamos o enfileiramento anteriormente explicado no modo *Batching* para gerar fluxos *multicast* no sistema. Baseado nestas considerações, no modo *Patching+Batching* o gerente executa as seguintes tentativas para escolher o provedor:

1. Escolhe o mais novo cliente entre os de status igual a 3
2. Escolhe o mais antigo cliente entre os de status igual a 1
3. Libera novo fluxo *unicast* do servidor

Modo CVC: Este é o modo inicialmente concebido no sistema para avaliar a técnica CVC, que se baseia nas técnicas *Chaining* e *Patching*. Conforme descrito anteriormente, o GloVE suporta duas políticas de escolha de clientes: PEEC⁸ e PEEP.

Quando adota PEEC (ênfase na técnica *Chaining*), o gerente no modo CVC usa os seguintes passos para escolher o provedor:

1. Escolhe o mais novo cliente entre os de status igual a 2
2. Escolhe o mais novo cliente entre os de status igual a 3
3. Libera novo fluxo *unicast* do servidor

⁸Política usada em todos os experimentos, exceto nos testes comparativos de PEEP e PEEC

Quando é adotada PEEP (ênfase na técnica *Patching*), ocorre uma inversão de ordem entre o primeiro e o segundo passo exposto acima, fazendo com que o gerente no modo CVC comporte-se conforme o seguinte algoritmo:

1. Escolhe o mais novo cliente entre os de status igual a 3
2. Escolhe o mais novo cliente entre os de status igual a 2
3. Libera novo fluxo *unicast* do servidor

Modo CVC+Batching: Neste modo, agregamos ao modo CVC a capacidade de enfileiramento de clientes no grupo de *multicast* de clientes no estado 1, discutido no modo Batching. Desta forma, é introduzido um novo passo no algoritmo de escolha do gerente no modo CVC+Batching, fazendo com que ele opere da seguinte maneira:

1. Escolhe o mais novo cliente entre os de status igual a 2
2. Escolhe o mais novo cliente entre os de status igual a 3
3. Escolhe o mais antigo cliente entre os de status igual a 1
4. Libera novo fluxo *unicast* do servidor

Capítulo 5

Avaliação do GloVE

Este capítulo descreve os aspectos relacionados ao ambiente experimental e os testes efetuados para analisar o desempenho do protótipo.

5.1 Ambiente Experimental

O ambiente experimental é composto de 6 máquinas, configuradas de acordo com a Tabela 5.1, interconectadas por um *switch Fast Ethernet* 3COM SuperStack II 3300. Este equipamento trabalha em nível 2, entretanto, está apto a funcionar como filtro para aplicações *multicast*, pois suporta *IGMP Snooping*. Das máquinas citadas, uma executa o servidor de vídeo e o gerente da CVC. Esta máquina possui um disco rígido Ultra2 SCSI IBM Ultrastar 18ES, com capacidade de armazenar 9.1GBytes e taxa de transferência sustentável variando de 12,7 a 20,2 MBytes/s. Outra é utilizada para gerar a carga e monitorar o estado do sistema. As quatro restantes tem a função de executar instâncias dos clientes.

Ítem	Descrição
Processador	Intel Pentium III 650MHz
Memória RAM	512 MBytes
Interface de Rede	Intel EtherExpress Pro 10/100
Sistema Operacional	Linux 2.2.14-5.0

Tabela 5.1: Configuração das máquinas

5.2 Carga de Trabalho

A geração da carga de trabalho foi feita simulando a chegada de clientes ao sistema¹, baseada em diferentes Processos de Poisson [36], com tempos entre chegadas exponencialmente distribuídos com média $\frac{1}{\lambda}$, onde λ representa a taxa de chegada. Como o objetivo principal do sistema é atacar o problema da escalabilidade em vídeos com alta popularidade, adotamos nos testes taxas entre 1 e 120 chegadas/min. Estes valores foram escolhidos porque, para taxas inferiores a 1 chegada/min, consideradas pequenas, o servidor tende a ser suficiente para atender a maioria das requisições e para taxas superiores a 120 chegadas/min, o desempenho de CVC tende a seguir uma linha constante.

O vídeo adotado nos testes é formado pelos 3550 segundos iniciais do filme Star Wars IV, codificados em MPEG-1 segundo o padrão NTSC-SIF (352 x 240, 29,97 quadros/s). Como o objetivo principal é analisar o comportamento da CVC e sua contribuição no desempenho do sistema, a utilização de um único vídeo é suficiente.

Através de medições realizadas na exibição do vídeo, identificou-se um comportamento bastante homogêneo (Figura 5.1) nos tempos de consumo dos blocos por parte do decodificador, sendo a duração média igual a 0,69 s/bloco.

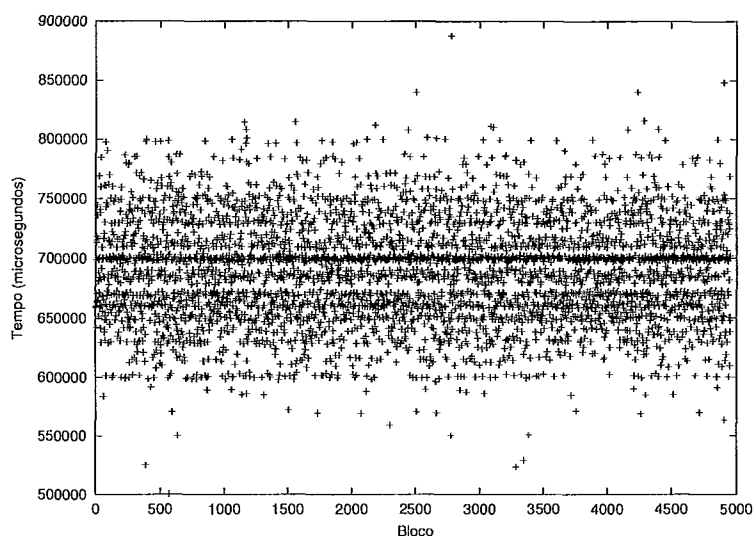


Figura 5.1: Tempos de consumo dos blocos do vídeo (Tempo Médio = 0,69 s/bloco)

Devido ao programa de decodificação MPEG consumir muita CPU e memória RAM, foi implementado um simulador de decodificador, a fim de liberar recursos indispensáveis para a execução de diversos clientes por máquina. Como o decodificador se comunica

¹No Apêndice 3 encontra-se o procedimento de geração de carga

com o cliente através de um *pipe*, de onde ele lê 4KBytes por vez, um simulador com comportamento bastante semelhante pôde ser desenvolvido. Com base no tempo médio de consumo dos blocos, o simulador calcula, descontando o *overhead* por ele introduzido, o tempo necessário para exibir 4KBytes de vídeo. Feito isso, ele simula o consumo através de chamadas regulares a *usleep*, passando o tempo calculado como parâmetro.

Os parâmetros utilizados nos experimentos estão resumidos na Tabela 5.2. Os tamanhos de buffer nela expostos equivalem, respectivamente, a 4/8/16 MBytes (suficientes para armazenar em torno de 20/40/80 segundos de MPEG-1).

Parâmetro	Valor(es)
Duração do Vídeo (s)	3550
Tempo Médio de Exibição de Bloco (s)	0,69
Tamanhos de Buffer (posições)	32/64/128
Taxas de Chegadas (clientes/min)	1/3/6/10/30/60/120

Tabela 5.2: Parâmetros dos experimentos

5.3 Análise dos Resultados

Nesta seção são discutidos os resultados de desempenho obtidos nos testes efetuados no GloVE², apresentados através de gráficos com valores médios.

5.3.1 Experimento Base

Como experimento base, buscou-se determinar a capacidade máxima de atendimento de clientes simultâneos do servidor de vídeo, isto é, o número de canais lógicos oferecidos pelo mesmo, segundo a abordagem convencional de um fluxo para cada cliente³. Inicialmente foi medida a vazão do servidor para atender a um único cliente. O valor obtido foi de 1,7Mbps, demonstrando que o servidor possui um *overhead* de 0,2 Mbps por fluxo MPEG-1.

Executando posteriormente os testes para definir o número total de canais, descobrimos que o servidor consegue sustentar 56 fluxos simultâneos, com uma ocupação de 96% da largura de banda nominal de 100Mbps. A partir de então, investigamos o desempenho dos diversos modos de operação do gerente anteriormente descritos para diferentes taxas

²Para facilitar a análise, a ocupação de recursos do servidor referente a fluxos de remendo foi desconsiderada

³Gerente no modo convencional

de chegada. Além disso, para entender melhor o efeito da agregação de *Batching* a CVC, realizamos medições no protótipo configurado nos modos CVC e CVC+Batching com diferentes tamanhos de buffer local nos clientes. Por fim, comparamos o desempenho das políticas PEEC e PEEP no modo CVC+Batching.

Para analisar quantitativamente o desempenho, foram utilizadas as seguintes métricas:

Taxa de Ocupação: é a percentagem de ocupação de canais lógicos do servidor;

Taxa de Provedimento: é a percentagem de clientes que estão funcionando como provedores;

Vazão da Rede: reflete a ocupação aproximada em Mbps da largura de banda agregada da rede, obtida pela soma do total ocupado por fluxos do servidor (fluxos do servidor * 1,7Mbps) com o total usado por fluxos de clientes provedores (provedores * 1,5Mbps);

Latência: é o intervalo médio em segundos compreendido entre a requisição do conteúdo e o início da exibição;

Taxa de Substituição: é percentagem de clientes que requisitaram substituição de provedor por experimentarem problemas no recebimento de blocos.

5.3.2 Desempenho Comparativo dos Diferentes Modos de Operação do Gerente

Nesta subsecção são analisadas as medições efetuadas com os diferentes modos de operação do gerente, considerando um buffer local no cliente com tamanho intermediário, capaz de armazenar 64 blocos (aproximadamente 40 segundos de vídeo MPEG-1).

Análise da Taxa de Ocupação

A Figura 5.2 demonstra o efeito de diferentes taxas de chegada de clientes (TC), na taxa de ocupação do servidor (TO), segundo os diferentes modos de operação do gerente descritos anteriormente.

No modo CVC, a ocupação do servidor possui duas tendências em intervalos distintos de taxa de chegada, tendo como ponto de transição $TC = 30$, onde apenas 4 canais são usados (aproximadamente 7%). Até este valor, TO cai drasticamente com pequenos aumentos da taxa de chegada, estacionando em torno de 10% de ocupação para TC entre 6 e 60. A partir de 30 clientes/min, TO passa a aumentar. Este efeito é decorrente do

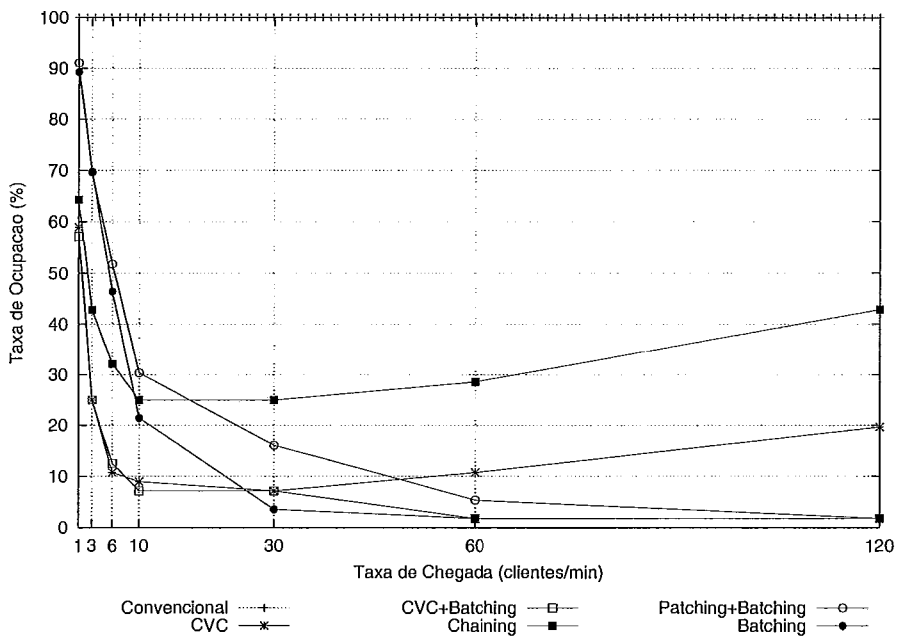


Figura 5.2: Canais ocupados no servidor pelos diferentes modos de operação do gerente

intervalo no qual o primeiro cliente realiza a pré-busca dos 16 primeiros blocos⁴, no qual está impossibilitado de prover. Com isso, clientes posteriores que chegam antes do seu início de exibição são liberados pelo gerente para receber fluxo direto do servidor. Como o servidor garante apenas a taxa mínima de envio, aproveitando a largura de banda livre para enviar os blocos requisitados em modo *burst*, este intervalo de pré-busca é baixo, não ultrapassando 5 segundos. Isto justifica a ocupação de aproximadamente 11 canais (TO = 20%) com taxa de chegada de 120 clientes/min, onde o intervalo médio entre chegadas de clientes é de 0,5 segundos, de modo que aproximadamente 10 clientes requisitam conteúdo antes que o primeiro possa começar a prover.

No modo CVC+Batching a influência da pré-busca na taxa de ocupação é removida, fazendo com que os clientes que requisitam conteúdo antes do primeiro começar a exibir sejam inseridos no seu grupo de recebimento, sendo atendidos tão logo a pré-busca seja concluída. Desta forma, a tendência de aumento de TO apresentada pelo modo CVC com TC superior a 30 é invertida, fazendo com que apenas um canal do servidor seja usado para taxas de chegada a partir de 60 clientes/min.

No modo Chaining, a TO é prejudicada pelo efeito da pré-busca dos blocos iniciais necessários para que o cliente possa começar a transmitir, atingindo nível mínimo de

⁴Este valor foi escolhido por ser equivalente a metade da capacidade do menor buffer utilizado nos testes, capaz de armazenar 32 blocos

25%. Diferente do modo CVC onde o efeito só aparece em grandes TC, o modo Chaining é influenciado pela latência média de início de exibição de todos os clientes, e não só do primeiro. Como esta média aproxima-se de 10 segundos, muitos clientes chegam ao sistema em instantes onde todos os demais clientes que ainda não estão provendo encontram-se fazendo pré-busca, obrigando o gerente a liberar novos fluxos do servidor.

No modo Patching+Batching, notamos um comportamento contrário ao esperado. Prevíamos que a curva apresentaria taxa de ocupação inferior ao modo Batching, exposto no próximo parágrafo, para TC inferior a 30 clientes/min. Buscando a origem desta anomalia, constatamos que é gerado por problemas de fluxos entre clientes que determinam muitos pedidos de substituição atendidos pelo gerente através de fluxos do servidor, fato este que ficará mais claro no decorrer deste capítulo.

No modo Batching, com taxas de chegada inferiores a 10 clientes/min, a TO é semelhante a obtida no modo Patching+Batching, sendo significativamente superior a TO alcançada com demais modos. Entretanto, para TC ≥ 30 , este é o modo que atinge a menor ocupação de canais do servidor.

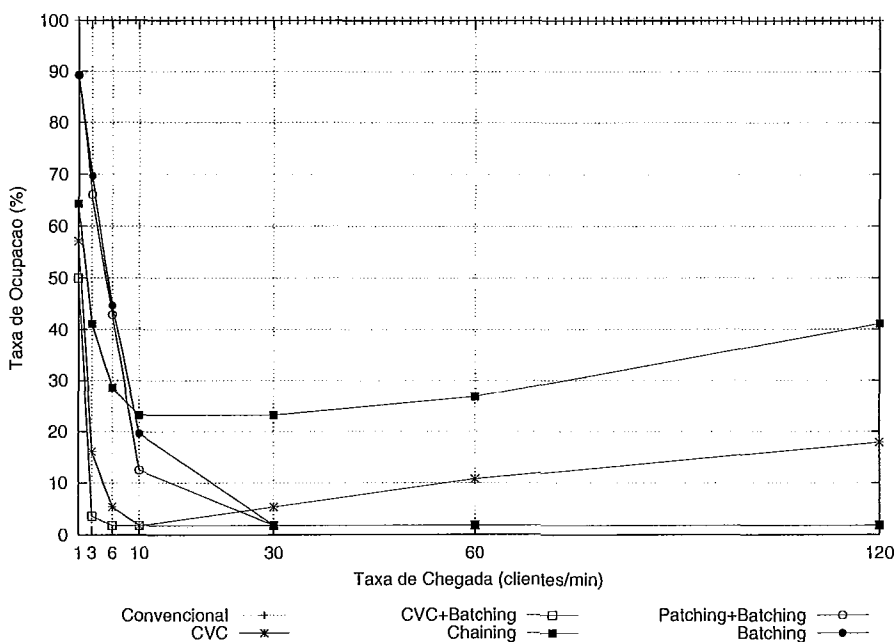


Figura 5.3: Desempenho corrigido descartando fluxos de substituição do servidor

Embora os resultados apresentados neste ítem sejam bastante significativos, não representam o desempenho ideal. Isto ocorre devido ao *overhead* imposto ao servidor devido a problemas de fluxo. A Figura 5.3 expõe a taxa de ocupação ideal possível de ser obtida com o protótipo, calculada desconsiderando os canais ocupados no servidor para transmi-

tir fluxos de substituição.

Pode ser notado nesta figura que os modos Chaining e Batching não são tão afetados por esta anomalia quanto os outros modos. A curva corrigida referente ao modo Patching+Batching confirma a sua potencialidade em obter menor TO em relação ao modo Batching para taxas de chegada inferiores a 30 clientes/min. O modo CVC é mais influenciado para TC entre 3 e 10. Enquanto que a TO mínima medida foi de 7% (4 canais) com TC igual a 30, a TO mínima corrigida equivale a apenas 1 canal, ocorrendo com TC igual a 10. Analisando o modo CVC+Batching, fica claro que ele é capaz de atingir taxa de ocupação mínima (1 canal) mesmo com baixa TC. Isto ocorre para TC superior a 3 clientes/min. No entanto, para TC inferior a este valor, TO atinge alto patamar.

Análise da Latência

A Figura 5.4 expõe a influência da variação na taxa de chegada de clientes (TC) na latência média (LT) existente entre a requisição do conteúdo por parte do cliente e o início da exibição, para os diferentes modos de operação do gerente.

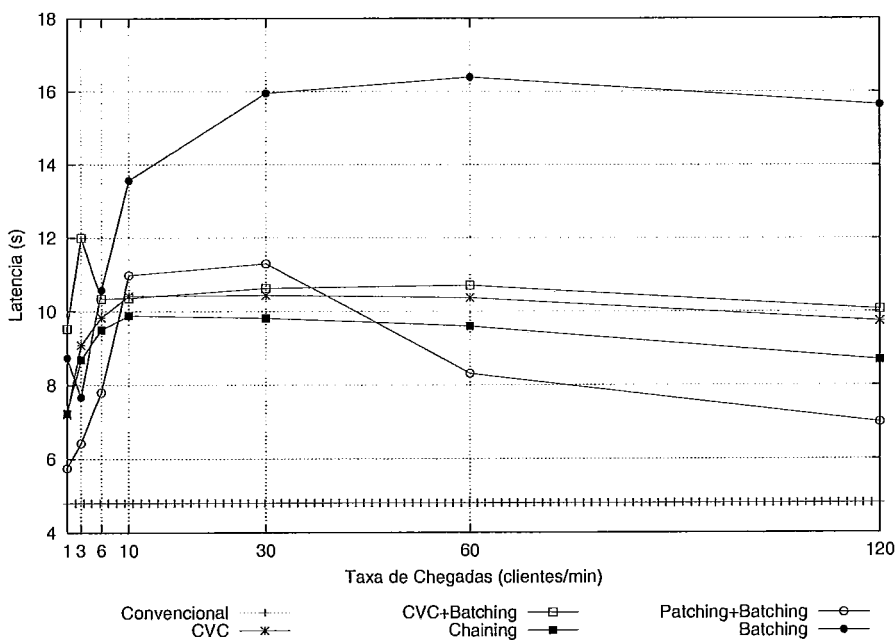


Figura 5.4: Latência de início de exibição

No modo Convencional, LT não se altera com a variação de TC entre 1 e 120, sendo seu valor médio igual a 4,8 segundos.

Nos modos CVC e CVC+Batching, LT tem valor aproximado de 10 segundos. Isto acontece porque na maioria dos casos, clientes são providos por outros clientes a uma taxa de bits equivalente a de exibição. Visto que cada bloco é consumido em um tempo

médio de 0,69 segundos, a pré-busca de clientes nesta condição tende a $16 * 0,69$, que gira em torno de 11 segundos. Entretanto, como alguns destes clientes recebem remendo ou fluxo completo do servidor de forma *burst*, atingem latência inicial inferior a este valor, contribuindo para que a latência média total seja inferior a 11.

No modo Chaining, a existência de um maior número de clientes sendo atendidos pelo servidor, em relação aos dois modos anteriores, faz com que a latência média seja ligeiramente inferior a medida com tais modos.

No modo Patching+Batching, com TC entre 1 e 10, a cooperação entre clientes é crescente, fazendo com que o número de clientes providos por outros clientes se eleve, fazendo com que a latência suba até 11. Para taxas de chegada maiores, LT entra em tendência de queda pois muitos dos clientes passam a receber remendo em modo *burst* da parte inicial do vídeo.

No modo Batching, ocorre efeito semelhante ao aferido no modo anterior, para TC entre 1 e 10. No entanto, a inexistência de Patching faz com que a latência continue a subir até TC igual 60, onde LT estabiliza-se em torno de 16 segundos.

Análise da Taxa de Provimento

A Figura 5.5 mostra a influência da taxa de chegada (TC) na taxa de provimento (TP) do sistema, que reflete a quantidade de clientes funcionando como provedores em relação ao total de clientes.

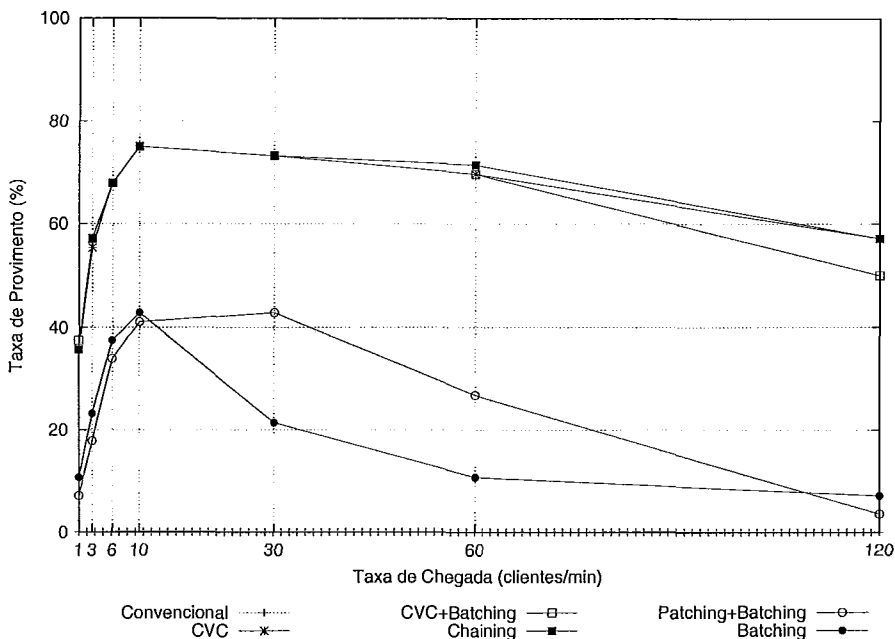


Figura 5.5: Percentagem de clientes provendo

Os modos CVC, CVC+Batching e Chaining apresentam curvas de TP semelhantes. Isto ocorre porque nestes três modos, o gerente escolhe como provedor, sempre que possível, um cliente que esteja exibindo e ainda não transmitindo. Analisando as taxas de chegada, notamos que para TC entre 1 e 10, TP é influenciada pela janela de cooperação do cliente, de modo que conforme aumenta a TC, mais clientes chegam no intervalo em que os clientes anteriores estão aptos a ser seus provedores. Após 10 clientes/min, TP mantém-se estável em aproximadamente 75% até TC igual a 60 clientes/min. A partir deste ponto, surge o efeito da pré-busca, levando a tendência de diminuição na TP. Isto ocorre porque nos modos CVC e Chaining, passa a crescer o número de clientes atendidos pelo servidor. Já no modo CVC+Batching, sua capacidade de esconder o efeito da pré-busca reflete na diminuição da TP devido a múltiplos clientes serem providos por um mesmo cliente.

No modo Patching+Batching, TP é influenciada pela quantidade significativa de clientes que pedem substituição de provedor com TC entre 6 e 60. Para atender a estas requisições, o gerente procura em sua estrutura por clientes habilitados a substituir os antigos provedores. Com isto, muitos clientes que estavam exibindo e não transmitindo⁵ tornam-se provedores dos clientes que sofreram problemas de recebimento, elevando TP para este intervalo de TC.

No modo Batching, TP tem três tendências distintas para diferentes intervalos de TC. Com TC entre 1 e 10, a diminuição crescente no intervalo médio entre chegadas faz com que mais clientes cheguem ao sistema nos períodos em que clientes anteriores estão fazendo pré-busca. Desta forma, TP tem tendência de subida já que mais clientes passam a prover, aliviando a carga do servidor. Com TC maior que 10, múltiplos clientes passam a ser atendidos por um mesmo cliente, fazendo com que TP passe a diminuir, tendendo a 2% (um único provedor).

Análise da Vazão da Rede

A Figura 5.6 apresenta a vazão aproximada da rede (VZ), definida pela largura de banda ocupada pelos fluxos provenientes do servidor (1,7 Mbps cada⁶) e dos clientes provedores (1,5 Mbps cada⁷), em relação as diferentes taxas de chegada (TC).

Como o sistema funcionando no modo convencional ocupa a mesma largura de banda (96 Mbps) para as taxas de chegadas usadas nos testes, introduzimos no gráfico uma curva representando a VZ deste modo de operação do gerente para servir como comparação.

⁵Categorizados nos estados 2 ou 4 explicados no capítulo anterior

⁶Como visto anteriormente neste capítulo, o RIO introduz um *overhead* de aproximadamente 0,2 Mbps

⁷Cujo *overhead* não é significativo

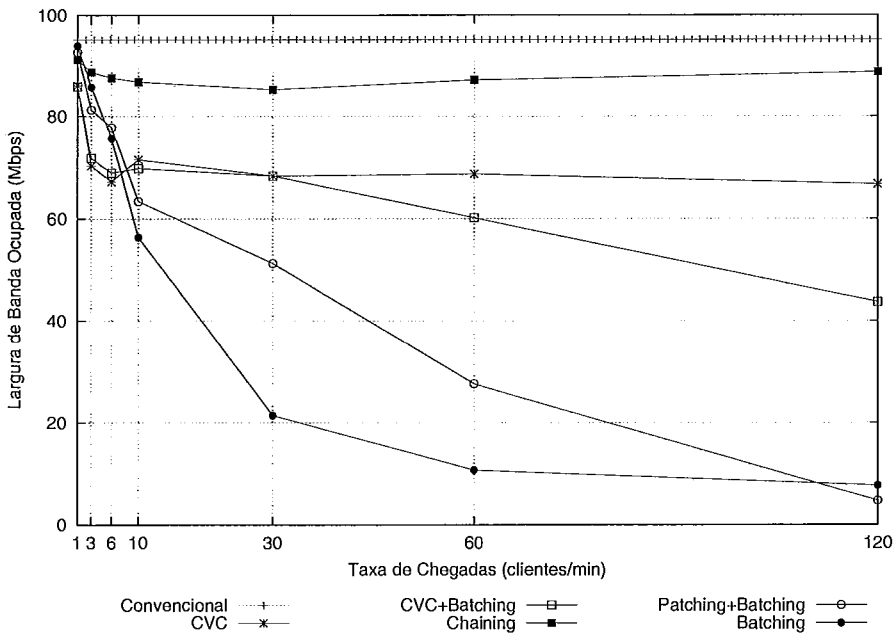


Figura 5.6: Largura de banda agregada ocupada

No modo CVC, com TC igual a 1 quase todos os clientes recebem fluxo do servidor devido ao intervalo médio entre chegadas ser superior a capacidade do buffer local dos clientes, impedindo cooperação e fazendo com que VZ aproxime-se da obtida no modo Convencional. No entanto, a partir de TC igual a 3, VZ estabiliza-se em torno de 70 Mbps, pois a maioria dos clientes é provida por outros clientes com fluxos de 1,5 Mbps, livres do *overhead* existente nos fluxos vindos do servidor.

No modo CVC+Batching, o comportamento de VZ é análogo ao modo anterior para TC entre 1 e 30. A partir deste valor, *Batching* passa a influenciar no total de clientes sendo providos por um mesmo cliente, levando a uma tendência de diminuição de VZ.

No modo Chaining, a curva de VZ é semelhante a do modo CVC, só que deslocada, representando ocupação de largura de banda em torno de 90Mbps. Este deslocamento se deve ao fato de que neste modo, quando o gerente não encontra um cliente⁸ no estado 2, libera novo fluxo do servidor. Já no modo CVC, o gerente pode introduzir o cliente em um grupo de *multicast* com fluxo capaz de receber remendo. Este reuso de fluxo alivia a carga do servidor e gera menor ocupação na largura de banda.

No modo Patching+Batching, VZ é influenciada pelo excesso de requisições de substituições de provedor ocorrido com TC entre 6 e 60. Desconsiderando estes fluxos de substituição, chegaríamos a uma curva de VZ inferior a do modo Batching.

⁸Exibindo e não provendo, sem estar descartando blocos

No modo *Batching*, a ocupação da largura de banda cai com o aumento da taxa de chegada por dois motivos. Para TC de até 10 clientes/min, conforme vai diminuindo o intervalo médio entre chegadas, mais provedores são criados para atender novos clientes que chegam ao sistema, de modo que poucos clientes são atendidos pelo mesmo provedor. Para valores superiores de TC, a tendência é de diminuição na criação de novos provedores devido a vários clientes passarem a ser atendidos por um mesmo provedor.

Análise da Taxa de Substituição

A Figura 5.7 mostra a influência da taxa de chegada de clientes (TC) na taxa de substituição (TS), definida pela percentagem de clientes que solicitam novo provedor devido a problemas na recepção de blocos.

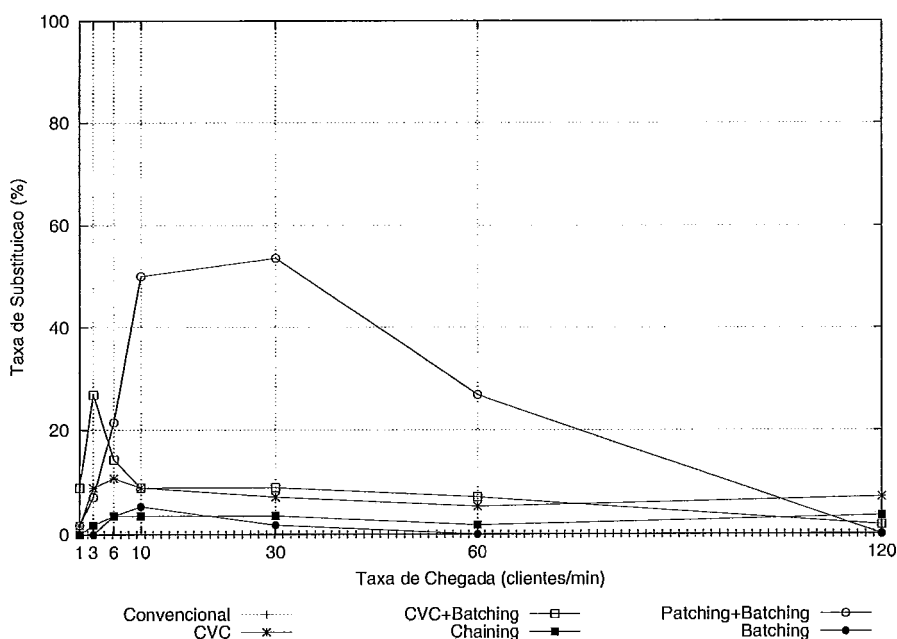


Figura 5.7: Clientes que solicitaram substituição de provedor

Desconsiderando o modo *Patching+Batching* que apresenta TS exagerada para TC entre 6 e 60, os demais modos atingem valores aceitáveis de requisição de substituição. Hipoteticamente, no modo citado, quando TC está dentro desta faixa mencionada, muitos clientes reusam fluxo através de *Patching*, de modo que são estabelecidos grupos *multicast* com vários clientes. Quando ocorre alguma perda no fluxo que alimenta um destes grupos, todos os clientes que dele fazem parte requisitam independentemente ao gerente um novo provedor. Desta forma, mesmo com poucas perdas, tendem a serem geradas muitas requisições. No entanto, este comportamento não pôde ser confirmado com as medições realizadas no escopo desta tese. Convém salientar que, como o caso ideal é

caracterizado por TS nula, investigar a origem destas anomalias de transmissão torna-se essencial para alcançar o desempenho ótimo do protótipo, sendo este um dos trabalhos futuros a serem desenvolvidos.

5.3.3 Desempenho de CVC e CVC+Batching

Apresentamos nesta seção os resultados obtidos pelo GloVE operando nos modos CVC e CVC+Batching, frente a diferentes tamanhos de buffer local nos clientes, com o intuito de verificar a influência da agregação de conceitos de *Batching* à CVC e do tamanho do buffer (32, 64 e 128 posições) no desempenho do sistema.

Análise da Taxa de Ocupação

A Figura 5.8 compara a taxa de ocupação de canais do servidor (TO) obtida por CVC e CVC+Batching com os três diferentes tamanhos de buffer (BS).

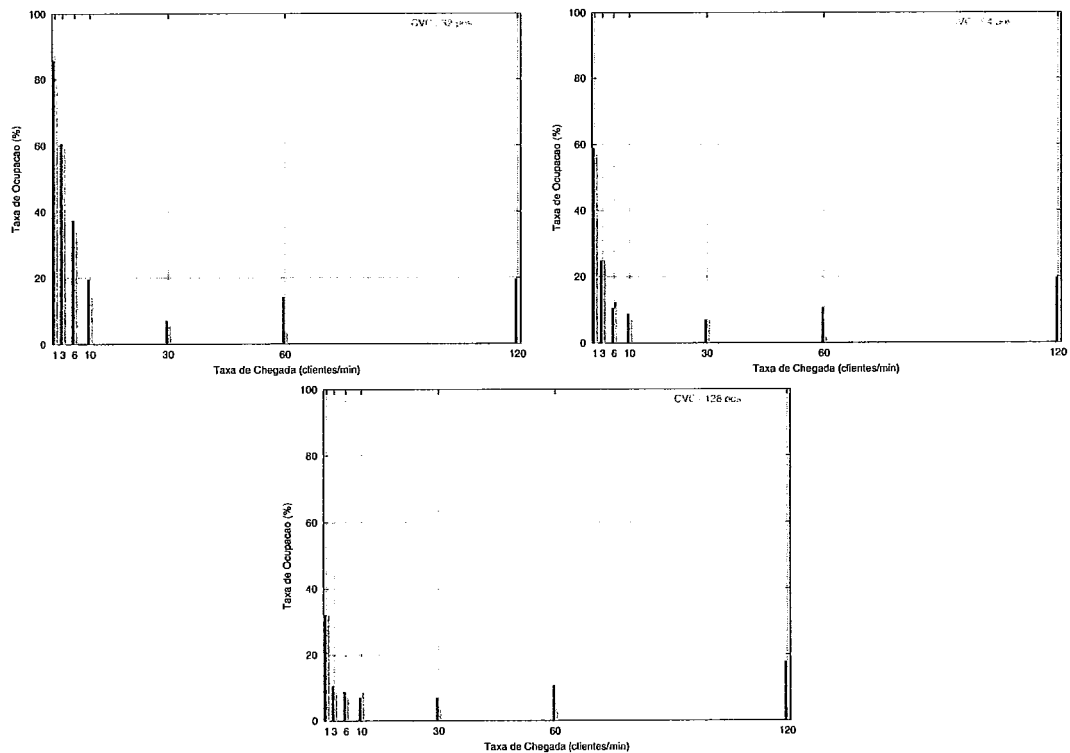


Figura 5.8: Taxa de ocupação de canais do servidor para buffers de 32, 64 e 128 pos

Em CVC, para TC a partir de 30, constata-se que o número de canais ocupados no servidor, independentemente do tamanho do buffer, converge para um certo valor, sendo este incrementado de forma proporcional ao aumento da TC. Isto acontece devido ao efeito da pré-busca explicado na subseção anterior.

Em CVC+Batching, para a mesma faixa de TC, o efeito da pré-busca da lugar ao efeito de *Batching*, de maneira que tende a ser ocupado apenas um canal do servidor.

Análise da Taxa de Provimento

A Figura 5.9 mostra a influência do tamanho do buffer (BS) na percentagem de clientes provendo nos modos CVC e CVC+Batching, denominada taxa de provimento (TP), frente a diferentes taxas de chegada.

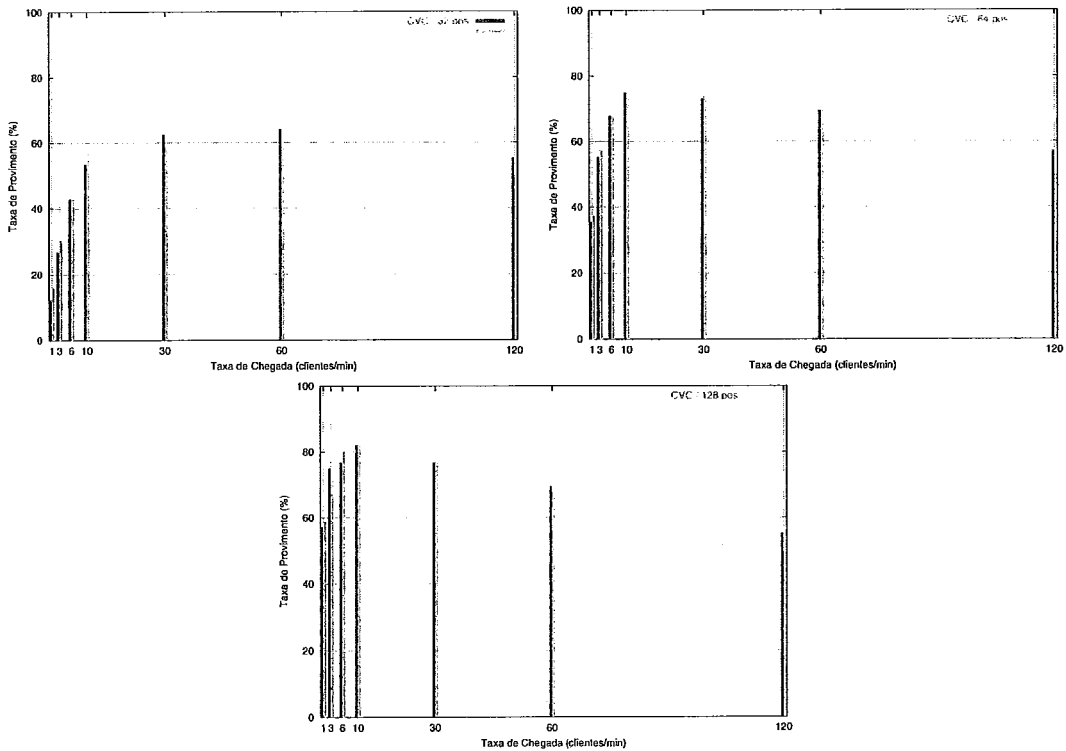


Figura 5.9: Percentagem de clientes provendo com buffers de 32, 64 e 128 posições

De maneira geral, o comportamento da TP é bastante similar nos dois modos estudados, sendo que para TC a partir de 30 os efeitos decorrentes da pré-busca no modo CVC e da introdução de *Batching* no modo CVC+Batching fazem com que TP entre em tendência de queda.

Outro ponto importante verificado é que para TC inferior a 60, ao analisar uma determinada TC, quanto maior é o buffer maior é a TP. Isto acontece devido a estar sendo adotada PEEC nestes experimentos, onde o gerente tenta primeiro escolher um cliente que esteja exibindo e ainda não transmitindo, sem ter descartado o primeiro bloco (*Chaining*). Como o intervalo no qual o cliente fica nessa condição⁹ é proporcional ao tamanho

⁹Denominado “janela de cooperação”

do buffer, a maioria das buscas do gerente são satisfeitas neste passo do seu algoritmo, fazendo com que vários clientes se tornem provedores.

Análise da Latência

A Figura 5.10 mostra a influência do tamanho do buffer na latência média de início de exibição (LT) dos modos CVC e CVC+Batching, com diferentes taxas de chegada.

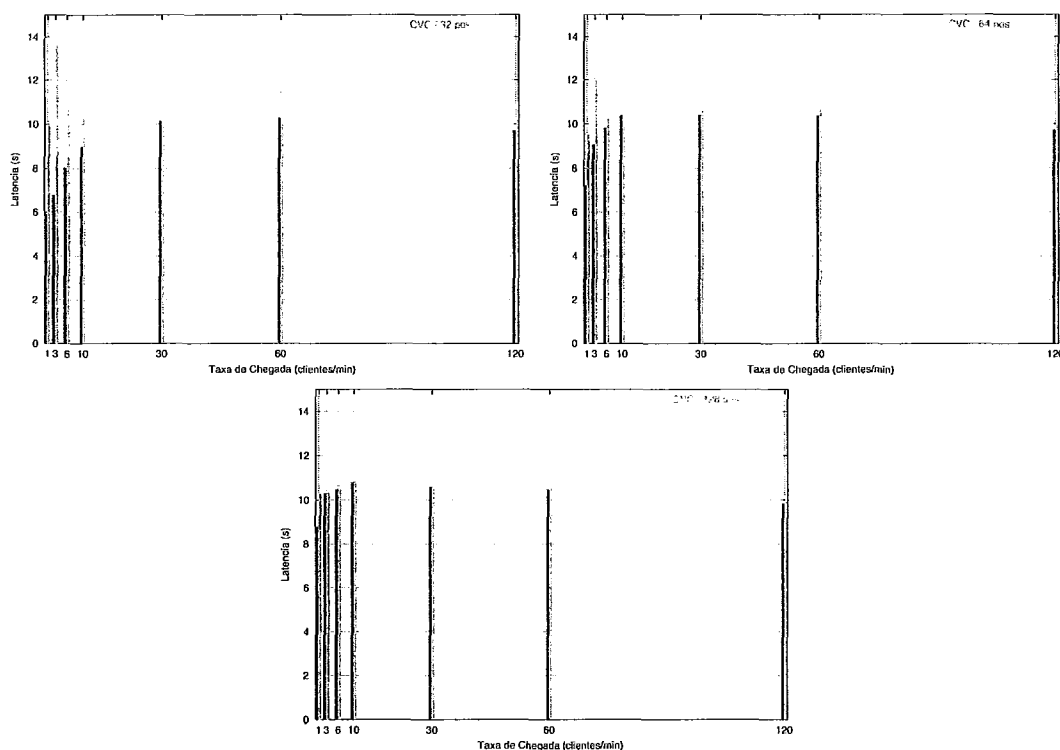


Figura 5.10: Latência média com buffers de 32, 64 e 128 posições

Os dois modos apresentam latência semelhante na maioria dos casos. Entretanto, para baixas TC, o modo CVC+Batching atinge maior LT, sendo que a diferença de LT entre os dois modos cresce com a diminuição do tamanho do buffer. Isto decorre de um maior número de requisições de substituição de provedor experimentado pelo modo CVC+Batching nestas condições, o que poderá ser visualizado no decorrer desta seção.

De maneira geral, desconsiderando o problema das requisições de substituição, a latência média só é influenciada pelo tamanho do buffer para pequenas taxas de chegada, onde a oportunidade de cooperação esta relacionada com a capacidade do buffer. Nesta condição, vários clientes são atendidos pelo servidor. Como este envia os blocos de forma *burst*, ocorre uma diminuição na latência média de início de exibição.

Análise da Vazão da Rede

A Figura 5.11 compara a vazão da rede (VZ) obtida com CVC e CVC+Batching

determinada pelos diferentes tamanhos de buffer e por diferentes taxas de chegada (TC).

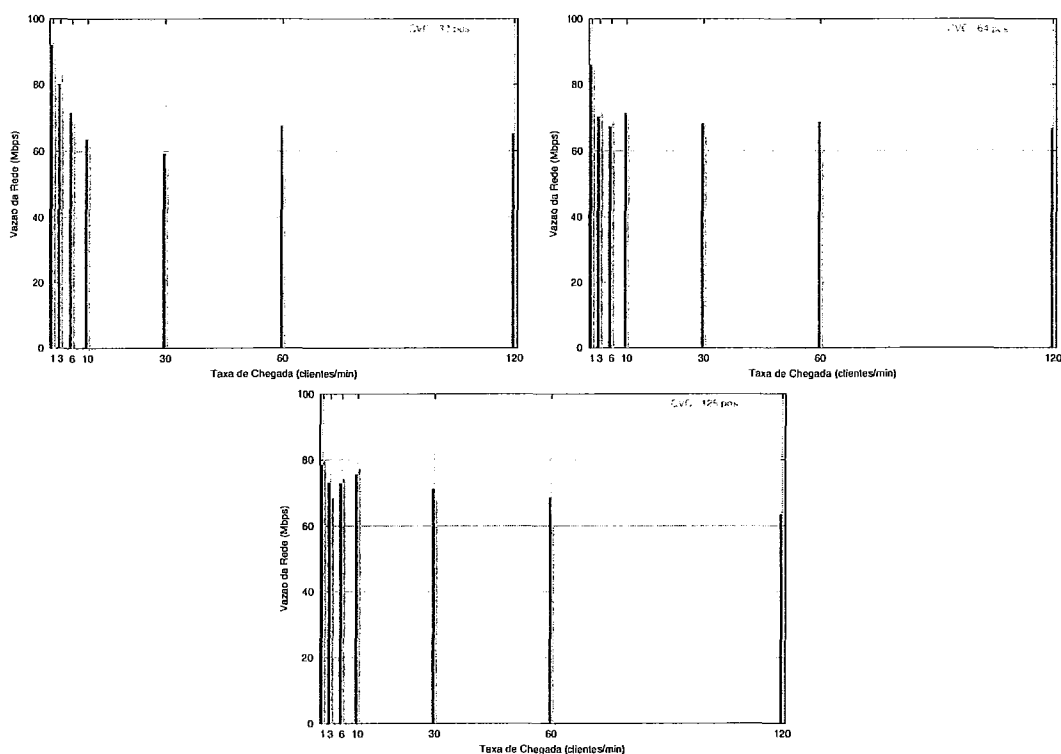


Figura 5.11: Ocupação da largura de banda da rede com buffers de 32, 64 e 128 posições

De maneira similar a outras métricas anteriormente analisadas, nota-se que o tamanho do buffer atua em VZ apenas para baixas TC, fazendo com que seja ocupada por fluxos de transmissão uma largura de banda de em torno de 80 Mbps. Conforme aumenta a TC, a VZ tende a estabilizar-se em torno de 60 Mbps, no modo CVC. Já em CVC+Batching, ela segue em tendência de queda determinada pelo enfileiramento de diversos clientes em mesmos grupos de *multicast*, que se acentua com grandes taxas de chegada.

Análise da Taxa de Substituição

A Figura 5.12 apresenta a taxa de substituição (TS) obtida com CVC e CVC+Batching determinada pelos diferentes tamanhos de buffer e por diferentes taxas de chegada (TC).

De maneira geral, pode-se dizer que para baixas TC, quanto menor o buffer maior a probabilidade de ocorrerem problemas no fluxos e posteriores requisições de novo provedor. Além disso, fica claro que o modo CVC+Batching é o mais afetado nesta situação. Como explicado na seção anterior, a discussão sobre a origem destes problemas de fluxo não faz parte do escopo desta tese, estando prevista como trabalho futuro.

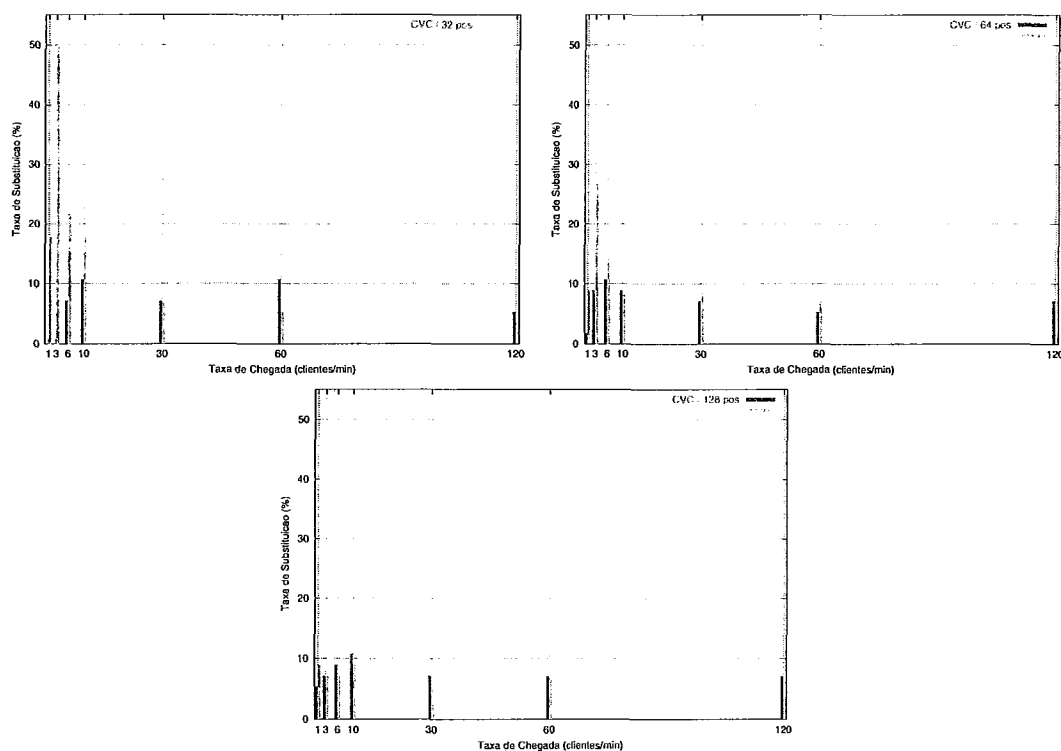


Figura 5.12: Percentagem de clientes que requisitaram novo provedor para buffers de 32, 64 e 128 posições

5.3.4 Comparação do Desempenho de PEEC e PEEP

A Figura 5.13 apresenta o desempenho comparativo das duas políticas de escolha de provedor - PEEC (ênfase em *Chaining*) e PEEP (ênfase em *Patching*) - propostas com o gerente configurado no modo CVC+Batching, usando dois tamanhos diferentes de buffer no cliente: 32 e 64 posições.

No gráfico que apresenta a taxa de ocupação, nota-se que ambas políticas demandam quantidades similares de canais do servidor. Quanto a taxa de provimento, fica claro no gráfico respectivo que PEEC cria maior quantidade de provedores, visto que nesta política o gerente sempre tenta criar primeiro um novo fluxo *multicast*, sendo que, quanto maior for o buffer, maior a oportunidade para que novos provedores sejam criados. Para PEEP, o aumento do buffer permite que mais clientes sejam incluídos em um mesmo fluxo.

Como PEEP usa remendos do servidor recebidos de forma *burst*, a latência média tende a ser mais baixa do que em PEEC, o que pode ser constatado no gráfico relacionado a esta métrica. A ênfase na reutilização de fluxos em curso de PEEP leva a uma diminuição no total de fluxos ativos na rede, ocupando menos largura de banda em relação a PEEC. Todavia, ocorre um aumento no número médio de requisições de substituição.

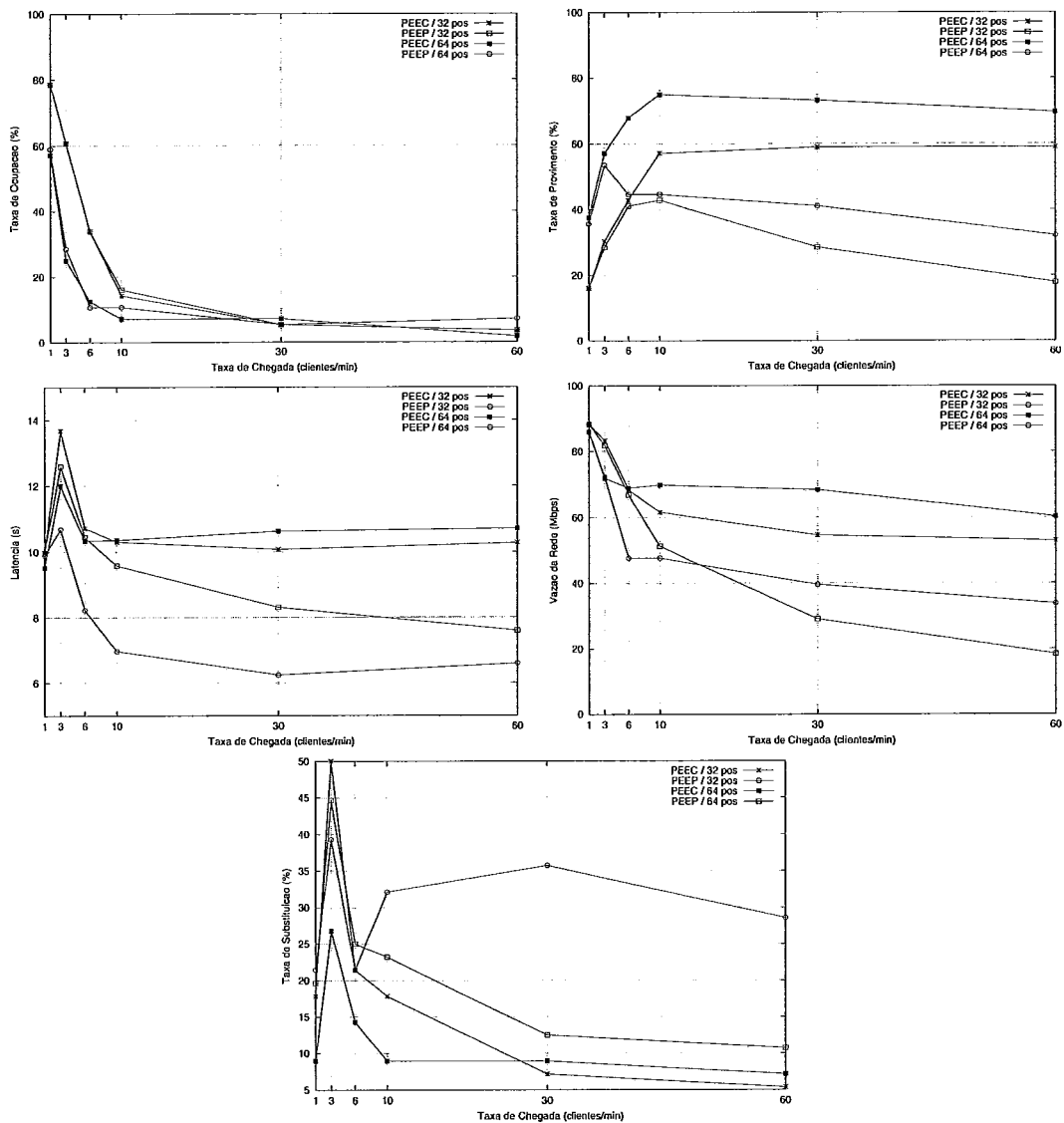


Figura 5.13: Gráficos comparativos do GloVE no modo CVC+Batching com PEEC e PEEP

5.4 Discussão

Os experimentos comprovam que CVC é uma técnica eficiente para a construção de sistemas de vídeo sob demanda escaláveis, principalmente quando o intervalo médio de tempo entre chegadas de clientes ao sistema é menor ou igual a capacidade de armazenamento de vídeo do buffer local dos clientes. Isto permite que múltiplos vídeos simultâneos sejam oferecidos pelo servidor, podendo estes serem acessados por um total de clientes proporcional a largura de banda agregada oferecida pela rede.

Como esperado, a introdução de *Batching* nos clientes contribuiu para eliminar o efeito colateral da pré-busca determinado pela inexistência de *multicast* a partir do servidor

e pela simplificação na implementação do módulo de transmissão dos clientes, onde o envio de blocos só pode ser feito após o início da exibição, fazendo com que a ocupação do servidor tenda a apenas um canal.

Analisando a Tabela 5.3 que apresenta a taxa média de ocupação de canais do servidor obtida com os diferentes modos de operação do gerente e buffer de 64 posições nos clientes, consideradas todas as taxas de chegada medidas (1, 3, 6, 10, 30, 60 e 120 clientes/min), CVC+Batching obteve o melhor desempenho, demandando apenas 16%.

Modo de Operação	Taxa de Ocupação Média (%)
Convencional	100
CVC	20
CVC+Batching	16
Chaining	38
Patching+Batching	38
Batching	34

Tabela 5.3: Percentagem de ocupação média de canais dos servidor

O tamanho do buffer influencia diretamente na taxa de chegada mínima onde a CVC começa a surtir efeito. Entretanto, fica claro que com um buffer intermediário de 64 posições, o que equivale a apenas 8 MBytes, conseguimos uma redução significativa no uso de canais do servidor para baixas taxas de chegada. Além disso, fica claro que caso um vídeo tenha grande procura, um buffer mínimo é suficiente para atingir uma ocupação mínima do servidor e alta escalabilidade.

A política de escolha influencia principalmente na ocupação da largura de banda agregada da rede. Como PEEP tenta reutilizar ao máximo os fluxos em andamento, gera menor ocupação. Visto que optamos por desconsiderar nos testes os recursos do servidor utilizados para enviar remendos, escolhemos PEEC para a realização da maioria dos testes por ela demandar menor uso deste tipo de fluxo.

Cabe dizer que os resultados obtidos servem como limite inferior de desempenho. Esta afirmação é motivada por existirem funções complementares as presentes no protótipo que contribuiriam para maior cooperação entre os clientes e menor uso de recursos do servidor. Uma delas é a implementação da função de colaborador nos clientes, como definida na versão original de CVC, evitando recorrer ao servidor para obter fluxos de remendo. Outra esta relacionada ao fato de que na versão atual, o gerente só escolhe como provedor de novo fluxo *multicast* um cliente que tenha o primeiro bloco em seu buffer. Esta abordagem desconsidera a possibilidade de um cliente que tenha descartado poucos

blocos prover um novo fluxo, sendo este completado através de remendo, aumentando a possibilidade de cooperação entre os clientes e diminuindo o acesso ao servidor.

A origem do grande número de requisições de substituição de provedor pode estar ligado a uma ineficiência do *switch* usado nos testes quanto ao controle dinâmico dos grupos *multicast*. O que nos leva a acreditar nesta hipótese é o fato de que o problema ocorre com maior frequência nos modos de operação que usam conceitos de *Patching*, onde vários clientes são introduzidos em fluxos *multicast* em curso e não naqueles em que os clientes aderem aos grupos antes do fluxo ser estabelecido (*Batching*). Entretanto, os testes realizados não permitem concluir a respeito, motivando um estudo mais aprofundado para confirmar ou não esta suspeita, capaz de apresentar soluções para o problema, o que contribuirá para atingir economias ainda maiores na ocupação dos recursos do sistema (largura de banda do servidor e da rede).

Outra questão importante que ficou de fora do escopo da tese, é a avaliação do comportamento do sistema frente a operações de pausa/recomeço implementadas. Como o foco da tese é a escalabilidade e não a interatividade, adotamos uma estratégia simples para tratar estas operações. Caso um provedor pare de exibir e por conseguinte pare de transmitir, seus clientes passam a consumir os blocos de seu buffer local até chegar no limite de ocupação mínima. Neste momento, requisitam ao servidor substituição de provedor. Note que isto tende a degradar o sistema pois o gerente tentará encontrar clientes diferentes para prover estes clientes que estão sincronizados na espera do mesmo bloco. Logo, diferentes estratégias para tratar estas limitações merecem ser também investigadas.

Capítulo 6

Trabalhos Relacionados

Várias alternativas de reuso de fluxos têm sido desenvolvidas com o intuito de atenuar os problemas intrínsecos a disponibilização de Vídeo sob Demanda. Além disso, diversos sistemas escaláveis de distribuição de arquivos vêm sendo implementados segundo o paradigma *peer-to-peer*. Neste capítulo abordaremos trabalhos pertencentes a estas duas áreas, relacionando-os com a proposta apresentada nesta tese. Na primeira seção descrevemos trabalhos referentes a VoD com ênfase no reuso de fluxo. Por fim, abordamos sistemas baseados no modelo *peer-to-peer* representando o estado-da-arte nesta categoria, atentando para similaridades em relação ao nosso trabalho.

6.1 Técnicas de Reuso de Fluxo

Dentre as propostas existentes de reuso de fluxo que atingem alta escalabilidade, estão as técnicas de *Broadcasting*. A idéia básica dos diversos esquemas de *Broadcasting* desenvolvidos [18] está na divisão do vídeo em segmentos, sendo estes enviados periodicamente em canais dedicados do servidor, de modo que enquanto um segmento está sendo exibido é garantido que o próximo seja recebido a tempo de ser exibido sem que ocorra descontinuidade na visualização. A principal desvantagem destas técnicas está na ocupação de vários canais do servidor e na latência significativa de início de exibição. Note que o GloVE não usa conceitos de *Broadcasting*.

Outra técnica que pode atingir alta escalabilidade é *Batching* [9], onde requisições para um mesmo vídeo são enfileiradas de modo a serem atendidas por um mesmo fluxo. Duas estratégias usadas para controlar este enfileiramento são: por tamanho, onde o fluxo inicia quando uma determinada quantidade de requisições é atingida; e por tempo, onde é determinado um limite máximo de tempo entre o instante da primeira requisição e o início do envio. A limitação de *Batching* esta na necessidade de impor grande latência aos

primeiros clientes que requisitarem um determinado vídeo para que poucos fluxos sejam usados. Como pôde ser visto nos capítulos anteriores, o GloVE adota esta técnica para eliminar o efeito da pré-busca explicado previamente, seguindo a estratégia de controle de enfileiramento por tempo (poucos segundos).

Adotando uma abordagem distinta, a técnica *PiggyBacking* [16] consiste em fazer com que dois fluxos existentes com um mesmo conteúdo, porém defasados, sejam sincronizados, possibilitando a eliminação de um deles. Isto é feito acelerando a taxa de transmissão do mais novo em até 5% (fato imperceptível ao ser humano) e retardando o mais antigo na mesma proporção, fazendo-os entrar em sincronia. Atualmente, o GloVE não usa esta técnica, contudo ela poderá ser empregada futuramente para liberar recursos extras, ao custo de maior complexidade no gerente.

Uma das técnicas fundamentais para o desenvolvimento do GloVE é *Patching* [19], que permite através de fluxos curtos de remendo reutilizar fluxos completos em andamento. *Transition Patching* [6] estende a proposta original com a possibilidade de empreender remendos em fluxos de remendo, levando a uma diminuição na largura de banda ocupada por fluxos deste tipo. O GloVE implementa a proposta original, podendo ser estendido futuramente com *Transition Patching*.

A outra técnica básica usada no GloVE é *Chaining* [39] que cria encadeamentos entre os buffers dos clientes. Desta forma nem todos os clientes recebem conteúdo do servidor e sim de outros clientes, sendo altamente escalável. Estudos de desempenho apresentados mostram que o custo extra de armazenamento necessário para o funcionamento da técnica é mínimo, sendo isto também válido para o GloVE.

A técnica Cache de Vídeo Cooperativa (CVC) [21, 22, 23] engloba as duas anteriores, *Patching* e *Chaining*. Nos trabalhos de CVC, é enfatizada a capacidade da técnica de oferecer interatividade, fato este que não é abordado pelos trabalhos de *Patching* e *Chaining*. Para avaliar a técnica, foram feitas simulações que demonstram sua alta escalabilidade. O GloVE adota em parte a CVC, de modo que algumas de suas características não estão presentes no protótipo: GoF como unidade de transferência, *multicast* a partir do servidor, cliente funcionando como colaborador para enviar remendos, entre outras.

6.2 Sistemas P2P

Recentemente, os benefícios do modelo P2P impulsionaram a criação de diversos sistemas de troca direta de arquivos entre clientes, sendo o Napster [27] o mais difundido.

O Napster é um sistema P2P projetado para operar com um servidor centralizado responsável por manter informações sobre os arquivos disponíveis nos clientes ativos no sistema. Desta forma, quando um cliente deseja um determinado arquivo, ele contata este servidor central que responde informando os clientes capazes de prover o conteúdo selecionado.

Representando uma evolução do conceito usado pelo Napster, o sistema Gnutella [15] empreende a troca de arquivos direta entre os clientes sem a existência de um servidor de metadados centralizado. Para tanto, cada cliente mantém localmente uma base de dados dinâmica contendo os endereços de outros clientes que participam do sistema. Para atualizar esta base, o cliente requisita aos demais clientes cujos endereços estão presentes em sua base, informações sobre novos clientes. Quando deseja um determinado arquivo, envia uma requisição para todos os endereços de sua base. Desta forma, os clientes capacitados a enviar tal arquivo respondem a requisição, formando uma lista de possíveis provedores de tal conteúdo.

Na versão atual, o GloVE assemelha-se ao Napster na existência de um servidor de metadados centralizado, responsável por coordenar a reutilização do conteúdo existente nos buffers locais dos clientes.

Outro exemplo interessante de sistema P2P é denominado *Cooperative Networking* (CoopNet) [28], onde os clientes cooperam uns com os outros para melhorar o desempenho da rede. O CoopNet funciona como um complemento para o modelo cliente/servidor de comunicação, atuando apenas quando a quantidade de requisições ao servidor for superior a sua capacidade de atendimento. Quando isto ocorre, o servidor redireciona requisições para os clientes que já fizeram o *download* do conteúdo, os quais passam a servi-lo, liberando o servidor desta tarefa, agregando alta disponibilidade ao sistema. A principal diferença entre a idéia de CoopNet e do GloVE está no fato de que o GloVE tentar sempre reutilizar os conteúdos presentes nas caches dos clientes de modo a manter uma ocupação mínima dos recursos do servidor, enquanto CoopNet preocupa-se apenas em evitar a saturação do servidor.

Capítulo 7

Conclusões

Nesta tese propusemos, implementamos e avaliamos um novo tipo de sistema de VoD, denominado *Global Vídeo Environment* (GloVE), baseado na técnica Cache de Video Cooperativa (CVC) que empreende uma coordenação global dos buffers locais dos clientes para, através de um modelo *peer-to-peer* e de conceitos das técnicas *Chaining* e *Patching*, alcançar escalabilidade e interatividade.

Foram descritas extensões ao protocolo original de CVC, desenvolvidas com o intuito de adaptar esta técnica para ambientes compostos por servidores sem suporte a *multicast* e funcionando no modelo *pull*, capazes de operar com qualquer tipo de mídia contínua. Introduzimos também conceitos da técnica *Batching* nos clientes para atingir melhor desempenho em altas taxas de chegada, onde a inexistência de *multicast* no servidor impossibilita que haja reutilização de fluxo até o instante em que o primeiro cliente seja capaz de prover. Criamos também uma máquina de estados no gerente com a responsabilidade de categorizar os clientes dentro do sistema, tornando mais intuitiva a escolha do cliente que exercerá a função de provedor, de acordo com as políticas PEEC (ênfase em *Chaining*) e PEEP (ênfase em *Patching*) estabelecidas. Além disto, a existência desta máquina proporcionou um desenvolvimento simplificado de diversos modos de operação no gerente, representando combinações de características das três principais técnicas empregadas no protótipo: *Chaining*, *Patching* e *Batching*.

A análise do GloVE serviu para corroborar resultados de simulação anteriormente realizados, mostrando que CVC é capaz de reduzir significativamente o acesso ao servidor, tendendo a apresentar alto grau de escalabilidade. A contribuição de CVC nesta redução depende da taxa de chegada de clientes ao sistema, de forma que só é possível estabelecer cooperação entre clientes se o intervalo entre duas chegadas consecutivas for condizente com a capacidade do buffer local do cliente. Como o objetivo da CVC é escalar o serviço

de VoD para vídeos com alta popularidade em horários de pico, buffers de poucos MBytes são suficientes.

Os resultados quantitativos obtidos com buffer de tamanho intermediário (8 MBytes, capaz de armazenar em torno de 40 segundos de vídeo MPEG-1) demonstram a capacidade da CVC em economizar recursos do servidor mesmo com intervalo médio entre chegadas de clientes de 1 minuto, atingindo percentual máximo superior a 94% para a versão baseada em *Chaining* e *Patching* com taxa de chegada de 30 clientes/min. Com a introdução de *Batching* nos clientes, a ocupação tende a manter-se em apenas um canal para taxas de chegadas superiores a este valor. Além disto, foi mostrado que esta abordagem é a que se adapta melhor a variação de taxa de chegada empreendida nos testes, obtendo ocupação média de 16% dos canais do servidor com 56 clientes ativos.

Os principais problemas encontrados foram relacionados a transmissão *multicast*, fazendo com que em condições intermitentes vários clientes sofressem interrupção no recebimento de fluxo de provedores, motivando uma quantidade significativa de requisições de substituição de provedor.

Além destes problemas citados que merecem ser estudados em trabalhos futuros, outras questões que ficaram em aberto devem ser investigadas, como o comportamento resultante das operações de pausa/recomeço e diferentes estratégias para tratar o reposicionamento dos clientes afetados por tais operações dentro da árvore de encadeamento de clientes da CVC. Outro trabalho a ser desenvolvido é a implementação da função de colaborador nos gerentes, operações discretas de recuo/avanço e criação de fluxos *multicast* incompletos, passíveis de receber remendo, seguida da avaliação dos impactos dessas extensões no desempenho do sistema.

Referências Bibliográficas

- [1] Acharya, S., Smith, B. “Middleman: A Video Caching Proxy Server”. In: *Proceedings of the Workshop on Network and Operating System Support for Digital Audio and Video*, June 2000.
- [2] Acharya, S., Smith, B., Parnes, P. “Characterizing User Access To Videos On The World Wide Web”. In: *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, San Jose, CA, Jan 2000.
- [3] Allen, J. R., Heltai, B. L., Koeniga, A. H., et al. “VCTV: A Video-On-Demand Market Test”. *AT&T Technical Journal*, v. 72, n. 1, pp. 7–14, 1993.
- [4] Andrews, G. R. *Concurrent programming: principles and practice*, chapter 4. Benjamin/Cummings, 1991.
- [5] Butenhof, D. R. *Programming with POSIX Threads*. Addison-Wesley, 1997.
- [6] Cai, Y., Hua, K. A. “An Efficient Bandwidth-Sharing Technique for True Video on Demand Systems”. In: *Proceedings of the ACM Multimedia*, pp. 211–214, 1999.
- [7] Carter, S., Long, D. “Improving video-on-demand server efficiency through stream tapping”. In: *Proceedings of the Sixth International Conference on Computer Communications and Networks*, pp. 200–207, 1997.
- [8] Dahlin, M., Wang, R., Anderson, T. E., et al. “Cooperative Caching: Using Remote Client Memory to Improve File System Performance”. In: *Proceedings of the Operating Systems Design and Implementation*, pp. 267–280, 1994.
- [9] Dan, A., Sitaram, D., Shahabuddin, P. “Dynamic Batching Policies for an On-Demand Video Server”. *Multimedia Systems*, v. 4, n. 3, pp. 112–121, 1996.
- [10] Deering, S. “Host Extensions for IP Multicasting”. *RFC 1112, Network Working Group*, August, 1989.

- [11] Feeley, M. J., Morgan, W. E., Pighin, F. H., et al. “Implementing Global Memory Management in a Workstation Cluster”. In: *Proceedings of the Symposium on Operating Systems Principles*, pp. 201–212, 1995.
- [12] Fenner, W. C. “Internet Group Management Protocol, Version 2”. *RFC 2236, Network Working Group*, November, 1997.
- [13] Flouris, M., Markatos, E. P. “The Network RamDisk: Using remote memory on heterogeneous NOWs”. *Cluster Computing*, v. 2, n. 4, pp. 281–293, 1999.
- [14] “GLADE GTK+ User Interface Builder”. <http://glade.gnome.org>; accessed April 24, 2002.
- [15] “Gnutella”. <http://www.gnutella.com>; accessed April 24, 2002.
- [16] Golubchik, L., Lui, J. C. S., Muntz, R. “Adaptive Piggybacking: A Novel Technique for Data Sharing in Video-on-Demand Storage Servers”. *Multimedia Systems*, v. 4, n. 3, pp. 140–155, 1996.
- [17] “GTK+ the gimp toolkit”. <http://www.gtk.org>; accessed April 24, 2002.
- [18] Hu, A. “Video-on-Demand Broadcasting Protocols: A Comprehensive Study”. In: *Proceedings of the IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.
- [19] Hua, K. A., Cai, Y., Sheu, S. “Patching: A Multicast Technique for True Video-on-Demand Services”. In: *Proceedings of the ACM Multimedia*, pp. 191–200, 1998.
- [20] IEC, I. “Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbps”, 1991.
- [21] Ishikawa, E., Amorim, C. *Memória Cooperativa Distribuída: Uma Solução Escalável para Sistemas de Vídeo sob Demanda Interativos*. Technical Report ES-542/00, COPPE/UFRJ Programa de Engenharia de Sistemas e Computação, Outubro 2000.
- [22] Ishikawa, E., Amorim, C. “Cooperative Video Caching for Interactive and Scalable VoD Systems”. In: *Proceedings of the First International Conference on Networking, Part 2*, Lecture Notes in Computer Science 2094, pp. 776–785, 2001.

- [23] Ishikawa, E., Amorim, C. “Memória Cooperativa Distribuída para Sistemas de VoD peer-to-peer”. In: *Anais do 19º Simpósio Brasileiro de Redes de Computadores*, 2001. <http://www.sbrc2001.ufsc.br/artigos/8184-1834.pdf>; accessed April 24, 2002.
- [24] Kuo, F., Effelsberg, W., Garcia-Luna-Aceves, J. *Multimedia communications : protocols and applications*, chapter 3. Prentice-Hall, 1998.
- [25] Liu, J. C. L., Du, D. H. C. “Continuous Media on Demand”. *IEEE Computer*, v. 34, n. 9, pp. 37–39, 2001.
- [26] “MpegTV Player”. <http://www.mpegTV.org>; accessed April 24, 2002.
- [27] “Napster”. <http://www.napster.com>; accessed April 24, 2002.
- [28] Padmanabhan, V. N., Sripanidkulchai, K. “The Case for Cooperative Networking”. In: *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, Cambridge, MA, March 2002.
- [29] Patterson, D. A., Hennessy, J. L. *Computer Architecture - A Quantitative Approach*, chapter 1. Morgan-Kaufmann, 2 ed., 1996.
- [30] Patterson, D. A., Hennessy, J. L. *Computer Architecture - A Quantitative Approach*, chapter 7. Morgan-Kaufmann, 2 ed., 1996.
- [31] Rao, S., Vin, H., Tarafdar, A. “Comparative Evaluation of Server-push and Client-pull Architectures for Multimedia Servers”. In: *Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1996.
- [32] Raunak, M. S. “A Survey of Cooperative Caching”. <http://citeseer.nj.nec.com/432816.html>; accessed April 24, 2002.
- [33] “Red Hat Homepage”. <http://www.redhat.com>; accessed April 24, 2002.
- [34] Ribeiro-Neto, B. “VÍdeo sob Demanda no Lar: Ficção ou Realidade?”. <http://www.vod.dcc.ufmg.br/vod/docs/descricao/jornal.jsp>; accessed April 24, 2002.

- [35] Rodriguez, P., Ross, K., Biersack, E. W. “Improving the WWW: caching or multicast?”. *Computer Networks and ISDN Systems*, v. 30, n. 22–23, pp. 2223–2243, 1998.
- [36] Ross, S. M. *A course in simulation*. Macmillan Publishing Company, 1990.
- [37] Santos, J. R., Muntz, R. *Design of the RIO (Randomized I/O) Storage Server*. Technical Report 970032, UCLA Computer Science Department, 1997.
- [38] Santos, J. R., Muntz, R. “Performance Analysis of the RIO Multimedia Storage System with Heterogeneous Disk Configurations”. In: *Proceedings of the ACM Multimedia*, pp. 303–308, 1998.
- [39] Sheu, S., Hua, K. A., Tavanapong, W. “Chaining: A Generalized Batching Technique for Video-On-Demand”. In: *Proceedings of the International Conference on Multimedia Computing and Systems*, pp. 110–117, 1997.
- [40] Sitaram, D., Dan, A. *Multimedia Servers: Applications, Environments, and Design*, chapter 7. Morgan-Kaufmann, 2000.
- [41] “Squid Web Proxy Cache”. <http://www.squid-cache.org>; accessed April 24, 2002.
- [42] Sreenan, C. J., Chen, J., Agrawal, P., et al. “Delay Reduction Techniques for Playout Buffering”. *IEEE Transactions on Multimedia*, v. 2, n. 2, pp. 88–100, 2000.
- [43] Stevens, W. R. *UNIX network programming*. Prentice Hall Software Series, 1990.
- [44] Symes, P. *Video Compression Demystified*. McGraw-Hill, 2001.
- [45] Tanenbaum, A. S. *Computer Networks*, chapter 7, pp. 723–760. Prentice-Hall, 3 ed., 1996.
- [46] Wallace, G. K. “The JPEG still picture compression standard”. *Communications of the ACM*, v. 34, n. 4, pp. 30–44, 1991.
- [47] Wu, D., Hou, Y. T., Zhu, W., et al. “Streaming Video over the Internet: Approaches and Directions”. *IEEE Trans. Circuits and Systems for Video Technology*, v. 11, n. 3, pp. 282–300, March 2001.

Apêndice A

Campos da Estrutura do Gerente

Cada entrada da estrutura de metadados de clientes mantida pelo gerente está especificada da seguinte forma:

```
struct cvc_entry
{
    /* Endereço IP do Cliente detentor da entrada */
    struct in_addr self_addr;

    /* Endereço IP do seu Provedor */
    struct in_addr provider_addr;

    /* Endereço IP do último cliente sendo provido */
    struct in_addr client_addr;

    /* Estado deste cliente no sistema */
    int status;

    /* Bloco sendo exibido por este cliente */
    unsigned long block_playing;

    /* Bloco sendo enviado por este cliente */
    unsigned long block_sending;

    /* Identificador do vídeo */
    int video_id;
}
```



```

/* Tamanho do vídeo em Bytes */
unsigned long size;

/* Flag indicando pausa na exibição */
int paused;

/* Total de clientes sendo providos */
int counter;

/* Flag indicando o recebimento de fluxo do servidor */
int RIO_flow;

/* Flag indicando que requisitou novo provedor */
int Replace;

/* Tempo de início de exibição ou reinício após pausa */
struct timeval start_play;

/* Instante em que começou a prover */
struct timeval start_send;

/* Primeiro bloco enviado */
unsigned long first_send;

/* Primeiro bloco exibido ou bloco exibido após pausa */
unsigned long first_play;

/* Tempo médio de consumo de blocos */
unsigned long update_rate;
};

```

Apêndice B

Interface Gráfica

A interface gráfica do cliente é mostrada na Figura B.1.

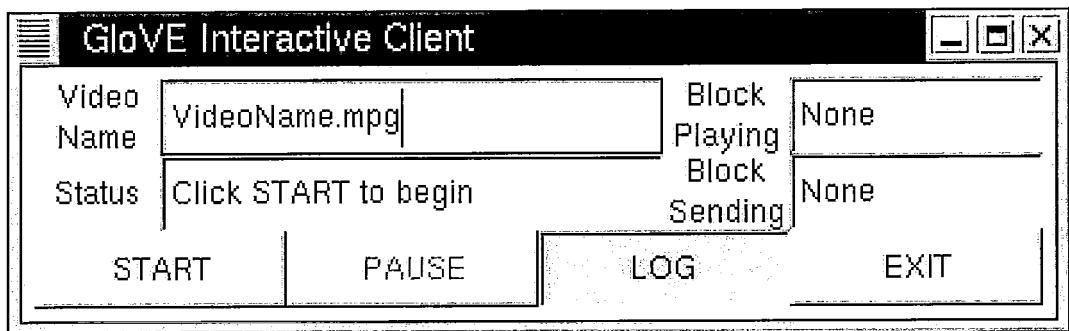


Figura B.1: Interface gráfica do cliente

Nela podem ser identificados os campos:

Video Name: nome do vídeo;

Status: Buffering - recebendo blocos iniciais; Playing - exibindo o vídeo; Paused - pausado; Playing/Providing - exibindo e provendo fluxo *multicast*;

Block Playing: bloco sendo exibido;

Block Sending: bloco enviado por *multicast*.

Aparecem também os botões:

START: usado para iniciar a exibição (fica desabilitado durante a mesma);

PAUSE: usado para efetuar pausa na exibição (só é habilitado após o início da exibição);

LOG: se ativo, exibe no terminal o log do cliente;

EXIT: termina o cliente.

Apêndice C

Procedimento de Geração de Carga

Para realizar a simulação, foi desenvolvido um programa gerador de intervalos entre chegadas (em microsegundos) de n clientes segundo tal processo, o qual escreve os tempos respectivos em linhas consecutivas de um arquivo denominado `times.dat`. Além deste, foi criado o arquivo `hosts.dat` contendo os endereços das máquinas configuradas para executar os clientes. A partir disto, a máquina geradora de carga segue o seguinte procedimento:

1. Lê um tempo de `times.dat`
2. Chama *usleep*, passando o valor lido como parâmetro, a fim de gerar o intervalo necessário
3. Passado o tempo, lê de `hosts.dat`, segundo uma política round-robin, o endereço da máquina onde o cliente deve ser inicializado
4. Faz uma chamada `rsh` para o endereço lido, iniciando de fato o cliente
5. Volta para o primeiro passo se todos os intervalos ainda não foram lidos