COPPE/UFRJ

AVALIAÇÃO EXPERIMENTAL DA MEMÓRIA COOPERATIVA COLAPSADA PARA SISTEMAS DE VÍDEO SOB DEMANDA

Arthur Cunha Granado

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

Orientador: Claudio Luis de Amorim

Rio de Janeiro
Setembro de 2010

AVALIAÇÃO EXPERIMENTAL DA MEMÓRIA COOPERATIVA COLAPSADA PARA SISTEMAS DE VÍDEO SOB DEMANDA

Arthur Cunha Granado

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

Examinada por:

Prof. Claudio Luis de Amorim, Ph.D.

Prof. Luís Felipe Magalhães de Moraes, Ph.D.

Prof. Célio Vinicius Neves de Albuquerque, Ph.D.

Prof. Leonardo Bidese de Pinho, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2010

Granado, Arthur Cunha

Avaliação Experimental da Memória Cooperativa Colapsada para Sistemas de Vídeo sob Demanda/Arthur Cunha Granado. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIV, 57 p.: il.; 29, 7cm.

Orientador: Claudio Luis de Amorim

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 54 – 57.

1. Video on Demand (VoD). 2. Streaming. 3. Gerência de Cache. 4. Proxy. 5. Memória Cooperativa Colapsada. 6. MCC. I. Amorim, Claudio Luis de *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus pais, que sempre
acreditaram em mim.

Agradecimentos

Primeiramente gostaria de agradecer a minha família, principalmente aos meus pais, que sempre acreditaram em mim e me deram condições para completar mais essa importante etapa da minha vida acadêmica e profissional. Agradecer aos meus revisores que tornaram o meu texto melhor. Obrigado Marcus, sei que o *mio nipóte* Lorenzo não é fácil! Obrigado Leonardo, sei que você estava passando por uma fase complicada e mesmo assim teve tempo para responder meus emails cheios de dúvidas.

Agradecer a família LCP, Claudio, Leonardo, Lauro, Hérbete, João, Bragato, Diego, Renato, Rodrigo e muitos outros. Que sempre me incentivaram e contribuíram de alguma forma para que eu conseguisse chegar até aqui. Ou com animadas conversas durante e após o almoço, ou dando suporte quando precisava, ou ainda com as intermináveis idas até a orla do Rio para fazermos teste.

E não poderia de deixar de agradecer aos meus amigos da linha de Arquitetura e Sistemas Operacionais, Boxeador de acapulco, Sr. Danis Day-Lewis, Fish Burn, Popov, Bruno, Tec, Luxa, Contê, Brebete e alguns outros que se listar vai dar outra dissertação. Que sempre contribuíram com o entretenimento, indo as chopadas comigo, fazendo importantes visitas ao mangue e a Lapa, e desbravando novos bares pela cidade onde lamentávamos sobre o enorme trabalho que nossas dissertações nos davam e onde também tentávamos contribuir para seleção natural de nossos neurônios, com a boa ajuda da cerveja!

Obrigado a todos! Contem comigo quando precisarem.

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

AVALIAÇÃO EXPERIMENTAL DA MEMÓRIA COOPERATIVA COLAPSADA PARA SISTEMAS DE VÍDEO SOB DEMANDA

Arthur Cunha Granado

Setembro/2010

Orientador: Claudio Luis de Amorim

Programa: Engenharia de Sistemas e Computação

Nessa dissertação implementa-se o modelo da Memória Cooperativa Colapsada (MCC) no servidor *proxy* do sistema GloVE para Vídeo sob Demanda. É realizada a validação entre os resultados das simulações já existentes e os resultados extraídos dos experimentos com o protótipo implementado. Também é realizada a avaliação dos resultados entre MCC e suas variações, a Memória Cooperativa Colapsada Local sem otimização (MCCL-) e a Memória Cooperativa Colapsada Local com otimização (MCCL+). Questões de qualidade de serviço como latência, taxa de bloqueio e número de fluxos oriundos do servidor de vídeo foram analisadas, e através desta análise foi possível concluir que a implementação da MCC se comporta da forma esperada e até superando alguns resultados obtidos nas simulações.

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

EXPERIMENTAL EVALUATION OF COLLAPSED COOPERATIVE VIDEO CACHE FOR VIDEO ON DEMAND SYSTEMS

Arthur Cunha Granado

September/2010

Advisor: Claudio Luis de Amorim

Department: Systems Engineering and Computer Science

In this dissertation, the Collapsed Cooperative Video Cache (C-CVC) model is implemented on the GloVE system's proxy server for Video on Demand. We compare previus simulations results and our new experimental results from the extended prototype. Furthermore, we compare C-CVC, Collapsed Local Cooperative Video Cache without optimization (C-LCVC-) and Collapsed Local Cooperative Video Cache with optimization (C-LCVC+), which were also implemented for this dissertation, to evaluate QoS aspects like latency, block rate and number of streams from the video server. The results demonstrated that the implementation behaves as expected, and in some cases it is better than simulations results.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

BS	*Begin Slot*, p. 21
CVC	*Cooperative Video Cache*, p. 10
DSLAM	*Digital Subscriber Line Access Multiplexer*, p. 2
ES	*End Slot*, p. 21
FIFO	*First-In-First-Out*, p. 12
GloVE-LAN	*Global Video Environment for Local Area Network*, p. 2
GloVE-Mix	*Global Video Environment for Wide and Local Area Network*, p. 2
GloVE-WAN	*Global Video Environment for Wide Area Network*, p. 2
GloVE	*Global Video Environment*, p. 2
HD	*High Definition*, p. 5
HTTP	*Hypertext Transfer Protocol*, p. 6
IP	*Internet Protocol*, p. 6
ISP	*Internet Service Providers*, p. 2
LAN	*Local Area Network*, p. 11
LCP	Laboratório de Computação Paralela, p. 36
LRU	*Least Recently Used*, p. 12
LS	*Link Slot*, p. 22
MCCL+	Memória Cooperativa Colapsada Local com Otimização, p. 2
MCCL-	Memória Cooperativa Colapsada Local sem Otimização, p. 2
MCC	Memória Cooperativa Colapsada, p. 1

MCD	Memória Cooperativa Distribuída, p. 1
P2P	*Peer-To-Peer*, p. 1
QoS	*Quality of Service*, p. 1
RS	*Read Slot*, p. 21
RTMP	*Real-Time Messaging Protocol*, p. 8
RTP	*Real-time Transport Protocol*, p. 8
RTT	*Round-Trip Time*, p. 36
SOCCER	*Self-Organizing Cooperative Caching Architecture*, p. 1
TCP	*Transport Control Protocol*, p. 6
UDP	*User Datagram Protocol*, p. 6
VoD	*Video on Demand*, p. 3
VsD	Vídeo sob Demanda, p. 7
WAN	*Wide Area Network*, p. 11
WS	Write Slot, p. 21

Capítulo 1

Introdução

A crescente popularidade de conteúdos multimídias em redes como a Internet, fez com que fosse necessário o desenvolvimento de técnicas que sejam capazes de utilizar com eficiência a rede para a distribuição desse conteúdo. Muitos são os desafios para se distribuir conteúdo multimídia na Internet. Em particular, o congestionamento das redes de distribuição, o atraso na entrega dos pacotes de mídia e a perda dos pacotes de mídia. Estes são alguns dos problemas que os sistemas de distribuição devem enfrentar para que consigam manter a qualidade de serviço (QoS).

Vários modelos para o gerenciamento da memória de servidores *proxy* para distribuição de conteúdo multimídia foram propostos com o intuito de se evitar ou minimizar esses problemas. Trabalhos como a Memória Cooperativa Distribuída (MCD)[1] utilizam técnicas *peer-to-peer* para agregar recursos ao sistema de distribuição. A MCD inova no sentido de utilizar o espaço de armazenagem dos *buffers* não apenas como uma linha de retardo, mas como uma área coletiva de memória usada de forma cooperativa pelo sistema de distribuição como um todo, visando a escalabilidade do sistema de distribuição. No entanto, clientes com acesso simétrico e com grande largura de banda na Internet são exceções e não a regra[1]. Deste modo a MCD é melhor utilizada em uma rede local onde o acesso simétrico dos clientes é a regra. Contudo os modelos de distribuição *peer-to-peer* estão fora do escopo desta dissertação, porque já foi avaliado nos trabalhos de Pinho[2] e Alves[3].

Outros modelos, como o SOCCER[4, 5] e Verscheure[6] propõem esquemas para o gerenciamento da memória de servidores *proxy*. Eles segmentam os vídeos e utilizam *ring buffers* para esconder a distância temporal entre os segmentos.

Já a Memória Cooperativa Colapsada (MCC)[1] é um modelo de distribuição de conteúdo multimídia com servidores *proxy* que introduz um novo conceito para o gerenciamento da cache de vídeo que será apresentado com detalhes no Capítulo 3. Um protótipo foi desenvolvido utilizando as características da MCC com o intuito de demonstrar o potencial que o modelo possui para a distribuição de vídeo em redes como a Internet.

O melhor aproveitamento dos recursos da rede no provimento de conteúdo multimídia traz benefícios para os clientes que poderiam usufruir de conteúdo de melhor qualidade, e para os distribuidores de conteúdo, que poderão disponibilizar mais e melhores conteúdos multimídias tais como Youtube[7] e Porta Curtas[8]. Também beneficiaria os ISPs (*Internet Service Providers*) e as companhias de telecomunicação que poderiam usufruir dessa eficiência na distribuição com o menor congestionamento de suas redes.

O primeiro protótipo que implementou a MCC, o GloVE-WAN, obteve êxito em experimentos dentro de uma DSLAM de uma empresa de telecomunicação[9]. Contudo, o GloVE-WAN não implementava todas as características da MCC apresentadas por Ishikawa[1], essa implementação não realizava o gerenciamento de recursos de forma tão eficiente. A ausência de algumas características da MCC no GloVE-WAN trouxe a motivação de se aperfeiçoar o protótipo, incluindo nessa nova implementação essas características que faltavam, e avaliando o desempenho das mesmas.

Desta forma, esta dissertação contribuir com: (i) a implementação do modelo da Memória Cooperativa Colapsada (MCC) no GloVE-Mix[3], segundo protótipo do Sistema GloVE que agrega tanto o GloVE-WAN[9] como o GloVE-LAN[2]; (ii) comparação dos resultados do GloVE-Mix utilizando a MCC, com as simulações realizadas por Ishikawa[1]; e (iii) comparação de desempenho da MCC e suas variantes (MCCL- e MCCL+).

A dissertação está organizada da seguinte forma. No Capítulo 2 se faz uma contextualização dos sistemas de distribuição de vídeo em larga escala e como os mesmo são inseridos em redes como a Internet.

No Capítulo 3 descreve-se o funcionamento da MCC e suas variações, para o melhor entendimento dos mecanismos que afetam o desempenho da gerência de cache. No Capítulo 4 são apresentados os resultados obtidos com os experimentos realizados com MCC, MCCL- e MCCL+ e as conclusões obtidas. E finalmente, no Capítulo 5 serão apresentados os principais trabalhos futuros identificados ao longo do trabalho realizado.

Capítulo 2

Distribuição de Vídeo em Larga Escala

Neste capítulo far-se-á a contextualização sobre o ambiente, técnicas e tecnologias que serão usadas de referência para o restante desta dissertação. Começa-se com a apresentação das classes de aplicações multimídias e seus desafios; passa-se para o entendimento dos sistemas de Vídeo sob Demanda (*Video on Demand* - VoD) e as técnicas de distribuição de vídeo; e finaliza-se descrevendo o GloVE-Mix, de forma sucinta, a técnica que implementa o modelo de distribuição de conteúdo multimídia estudado nesta dissertação.

2.1 Contexto

Nos últimos anos houve o surgimento de várias aplicações multimídias na Internet, como portais para distribuição de rádios virtuais, portais de distribuição de vídeos e aplicações de áudio/vídeo conferências. Essas aplicações começaram a se popularizar e a serem utilizadas por um crescente número de usuários para fins de entretenimento, trabalho e difusão de cultura.

Alguns exemplos de sistemas multimídias que são largamente utilizados são o Youtube[7] da Google[10], e o sistema de VoIP da Skype[11]. Esses tipos de aplicações multimídias transformaram o modo como os usuários e as empresas vêem a Internet.

Diferentemente das aplicações tradicionais da Internet, que vêem a integridade dos dados transferidos como crucial para o correto funcionamento das mesmas, as aplicações multimídias na Internet são, até certo ponto, tolerantes a perdas, mas não toleram bem os atrasos dos pacotes de multimídias[12].

Diante do crescente número de aplicações multimídia, tanto os ISP, quanto as companhias de telecomunicação estão enfrentando o problema do aumento do

tráfego dentro de suas redes prejudicando a qualidade de serviço (*Quality of Service* - QoS) oferecida, levando-as a procurarem soluções para serviços de distribuição com desempenho maior no reuso de fluxos de vídeos oriundos dos servidores de vídeo.

2.1.1 Classificação

As aplicações multimídias na Internet podem ser divididas em três classes de aplicações [12]:

Fluxo contínuo, áudio e vídeo armazenados: Nesta classe de sistemas multimídias estão as aplicações cujos usuários podem solicitar a qualquer momento arquivos de áudio e vídeo armazenados em servidores. Esses arquivos podem ser programas de rádio, músicas ou filmes.

As características principais dessa classe de aplicações são: (i) conteúdo armazenado: o conteúdo multimídia já está pré-gravado nos servidores, permitindo aos usuários realizarem operações de vídeo-cassete, como avanço, recuo, pausa, e a inicialização do conteúdo do ponto desejado; (ii) a reprodução da mídia contínua: enquanto o usuário está recebendo o arquivo, a parte já armazenada é reproduzida, evitando que o usuário tenha que esperar que todo o arquivo seja carregado antes da reprodução, o que poderia ser um tempo considerável e levar o usuário a desistir de assistir este conteúdo; (iii) a temporização: os dados devem ser recebidos do servidor a tempo de serem reproduzidos pelo cliente, caso contrário essa informação não será mais útil e será descartada – essa classe de aplicações não se comporta bem com as altas variações dos atrasos (*jitter*) dos pacotes de mídia na rede.

Áudio e vídeo de fluxo contínuo ao vivo: Esta classe de aplicações se assemelha mais com o atual modelo de rádio difusão (*broadcast*) com o qual estamos acostumados. Quando um usuário sintoniza um dos canais, ele irá receber o conteúdo ao vivo, do ponto em que o conteúdo está atualmente sendo transmitido. O usuário não poderá avançar no conteúdo transmitido, pois o mesmo ainda não foi produzido, mas pode realizar outras interações, dependendo do conteúdo, como recuar e pausar. Como a classe anterior, a temporização é um fator que também afeta esta classe de aplicações, já que a alta variação dos atrasos dos pacotes de mídia pode comprometer a qualidade do conteúdo exibido.

Vídeo e áudio interativos em tempo real: O uso de áudio e vídeo em tempo real e com interatividade permite que usuários se comuniquem entre si em tempo real, conhecidas como aplicações de vídeo conferência. A interatividade permite que os usuários criem conferências com múltiplos usuários. E através

da integração computador/telefone, provê, potencialmente, uma comunicação a um custo menor que o cobrado pelas companhias de telecomunicação. Esta classe de aplicações também sofre com o problema das altas variações dos atrasos dos pacotes de mídia.

2.1.2 Principais Desafios

Como observado na descrição das classes de aplicações multimídia na Internet, em todas há o problema dos pacotes de uma mídia terem que ser recebidos do servidor a tempo de serem reproduzidos de forma ordenada no equipamento do usuário (*set-top Box*, *player*, etc). O problema ocorre porque a Internet é uma rede de comutação de pacotes de melhor esforço, ou seja, ela não garante que um pacote enviado por um servidor será entregue a tempo no destino de forma ordenada, e nem que será entregue. O pacote ainda pode ser descartado, como veremos adiante.

Este é um paradigma diferente do encontrado nas redes de telefonia, que são redes de comutação de circuitos, onde todos os recursos necessários para a comunicação entre dois pontos são previamente reservados, garantindo a entrega ordenada dos pacotes e a QoS.

Já nas redes de comutação de pacotes, como a Internet, os recursos da rede - buffers, links, roteadores, etc - não são reservados para uma única comunicação, e sim compartilhados. Ao compartilhar esses recursos, pode-se ter problemas, como um enlace ou roteador congestionado, já que várias aplicações estarão utilizando os mesmos recursos. Isso pode fazer com que pacotes tenham que aguardar na fila até serem enviados, ou mesmo sendo descartados pela falta de espaço de armazenagem. O congestionamento da rede é uma das causas das altas variações dos atrasos (*jitter*) e das perdas dos pacotes de mídia.

Pode-se pensar que ao se aumentar a capacidade das linhas de comunicação e ao se colocar roteadores mais potentes, aumentando assim a largura de banda da rede, se é capaz de evitar o congestionamento. Entretanto mais e mais usuários irão utilizar a rede, e as aplicações irão cada vez usar mais e mais recursos da rede, como disponibilizar vídeos em alta definição (*High Definition* - HD), fazendo com que novamente a rede comece a sofrer congestionamentos. O aumento da capacidade da rede também gera altos custos para sua implantação e esta não é rápida.

Logo, tem-se que fazer com que as aplicações utilizem de forma mais racional os recursos oferecidos pela rede, ao invés de se esperar que os ISPs e as companhias de telecomunicação aumentem os recursos da rede. Por isso deve-se utilizar técnicas que otimizem o uso dos recursos da rede, como se verá mais adiante neste capítulo.

rtmp2009 Um exemplo que pode ser usado para ilustrar o que acabou de ser afirmado é a escolha do protocolo da camada de transporte que deve ser utilizado.

Duas possíveis opções são o uso de TCP (Transport Control Protocol)[13] ou UDP (User Datagram Protocol)[14], dois tradicionais protocolos de transporte para rede IP (Internet Protocol)[15].

O UDP é um protocolo de transporte simples, que apenas segmenta as mensagens da camada de aplicação e os repassa para a camada seguinte da rede IP, não se importando em saber se as mensagens irão chegar, ou não, a seu destino e nem em que ordem elas chegarão.

Já o TCP é um protocolo com mais recursos. Ele não se limita a segmentar as mensagens da camada de aplicação. Ele também garante a entrega dessas mensagens a seu destinatário e na ordem que foram enviadas. Para isso, o TCP necessita de mais recursos que o UDP: precisa de uma fila onde possa armazenar os segmentos, até que tenha certeza que os mesmos foram entregues; precisa de um sistema mais elaborado de comunicação, com mensagens que indiquem a chegada de cada pacote. O TCP também possui um mecanismo de controle de congestionamento, que ao detectar que a rede está se congestionando, diminui a taxa de envio de pacotes tentando minimizar o congestionamento e os efeitos do mesmo sobre os seus pacotes que estão trafegando.

Todas essas características do TCP o tornam um protocolo com uma sobrecarga maior que o UDP. Muitas aplicações multimídia preferem implementar o UDP no transporte de seus pacotes de mídia, ao invés do TCP, implementando a garantia de entrega e a ordenação dos pacotes na camada de aplicação.

Entretanto, algumas aplicações multimídia preferem o TCP, porque querem usar o protocolo de aplicação HTTP[16], que é o protocolo utilizado pela Web. O HTTP permite a inúmeras aplicações passarem pelos *firewalls*, já que a porta TCP 80 padrão do HTTP é mantida aberta na maioria dos *firewalls*. A maioria das portas TCP e UDP restantes são normalmente mantidas fechadas por questões de segurança. Devido a estas características, muitas aplicações multimídias utilizam o HTTP, conseqüentemente utilizando o TCP para o transporte das mensagens de mídia, mesmo ele sendo um protocolo que precise alocar uma quantidade maior de recursos e gere uma sobrecarga na rede, podendo assim, influenciar negativamente sobre a qualidade de serviço do sistema de distribuição de conteúdo multimídia.

Outro exemplo que pode ser usado para ilustrar, é considerar o uso, ou não, do *multicast* para a distribuição de vídeo na Internet. Com o *multicast* pode-se criar grupos de clientes do sistema que, ao invés de cada um receber um fluxo de vídeo do servidor de vídeo, um único fluxo seria mantido pelo servidor, as mensagens desse fluxo seriam replicadas pelo caminho, até que cada cliente recebesse a mensagem do servidor de vídeo. Porém, para que as aplicações que utilizem *multicast* sejam possíveis, todas as redes que estejam no caminho da distribuição do vídeo devem implementar a técnica *multicast* na rede IP, o que não ocorre na Internet.

2.2 Sistemas de Vídeo sob Demanda

Os sistemas de Vídeo sob Demanda (VsD) pertencem à classe de aplicações multimídia de fluxo contínuo com áudio e vídeo armazenados. Os sistemas VsD realizam a distribuição de áudio/vídeo em redes como a Internet. Quando um cliente na Internet deseja assistir a um vídeo, a qualquer momento que queira, e do ponto que queira, o sistema VsD irá prover este vídeo ao cliente. A melhor maneira de entender a lógica dos sistemas VsD é pensar em uma aplicação típica desses sistemas, o de uma vídeo locadora virtual.

2.2.1 Exemplo da Vídeo Locadora

Numa vídeo locadora tradicional, os clientes vão até a locadora, procuram os vídeos que eles acham interessantes e que gostariam de assistir, os alugam, e vão para suas casas, para que no momento mais oportuno daquele dia (ou noite) eles possam assisti-lo, devolvendo-os novamente a locadora. Contudo, esses clientes podem ter alguns problemas, como o vídeo desejado não estar disponível para alugar, ou os clientes não terem tempo de irem até a locadora, ou ainda, após realizarem o aluguel do vídeo, não poderem assisti-lo.

Todos esses problemas podem ser evitados, ou minimizados, se os clientes, ao invés de irem a uma vídeo locadora tradicional, alugassem os vídeos de uma vídeo locadora virtual. Essa vídeo locadora virtual teria uma página na Web com um catálogo dos vídeos disponíveis. Os clientes teriam apenas que escolher o vídeo no catálogo, pagar, e depois assistir ao vídeo quando desejassem, sem terem que se deslocar até a locadora, ou esperarem pela entrega do vídeo, ou correrem o risco do mesmo não estar disponível por falta de cópias, bastando apenas possuírem uma conexão com a Internet. Existe a possibilidade dos clientes sofrerem com alguma deficiência na infra-estrutura como falha de conexão com a Internet, servidor de vídeo indisponível, etc; mas os benefícios são maiores.

O dono da vídeo locadora virtual também pode se beneficiar dos sistemas VsD, pois ele não precisaria manter várias cópias dos vídeos, não correria o risco de não ter uma cópia disponível para um cliente e perder a locação, e poderia ampliar seu público consumidor, que antes eram apenas as pessoas que residiam no bairro onde a locadora estava instalada, para todas as pessoas no mundo que possuíssem uma conexão com a Internet.

2.2.2 Características Gerais

Os sistemas VsD são aplicações multimídia com fluxo contínuo e com áudio e vídeo armazenados, que possuem algumas características específicas[12]:

- Mídia armazenada: cada vídeo de um sistema VsD está armazenado no servidor de vídeo, permitindo que um cliente possa solicitar o vídeo a qualquer instante e realizar ações de avanço, recuo e pausa sobre o vídeo, que devem ser imediatamente realizadas após sua solicitação;

- Fluxo contínuo: após o cliente solicitar um vídeo, e enquanto o mesmo está recebendo este vídeo, a parte já armazenada do vídeo é reproduzida, evitando que todo o vídeo tenha que ser carregado antes de sua exibição, o que poderia causar perda de interesse de assistir ao vídeo;

- Reprodução contínua: depois que o vídeo iniciou sua exibição, ele deve respeitar sua temporização original, pois se um pacote de vídeo chegar depois do momento que deveria ser exibido, o mesmo não terá mais utilidade e deverá ser descartado, levando o sistema a uma baixa QoS. Isto impõe aos sistemas VsD uma intolerância a alta variação dos atrasos (*jitter*) dos pacotes de mídia.

Pode-se encontrar todas essas características nos vídeos armazenados em um servidor Web tradicional, que utilize o HTTP. Quando um cliente requisita um dos vídeos ao servidor Web, este começa a enviar o vídeo como um arquivo normal, em um fluxo contínuo. Esse vídeo será recebido pelo cliente e armazenado localmente. Enquanto o vídeo é carregado para a máquina local do cliente, a parte do arquivo que já foi carregada pode ser reproduzida por um tocador de vídeo (*video player*).

Outras aplicações de multimídia na Internet podem usar, ao invés do servidor Web, um servidor de mídia dedicado, que tenha como única função prover o vídeo para o cliente requisitante. Essa aplicação poderia utilizar outros protocolos, que não o HTTP, como o RTP[17] e RTMP[18]. O RTP e o RTMP são exemplos de protocolos do nível de aplicação especialmente implementados para a distribuição de áudio/vídeo na Internet, possuindo algumas funcionalidades como levar em consideração que alguns clientes gostariam de receber o áudio/vídeo em um formato diferente, como por exemplo, receberem vídeos HD por possuírem mais banda que os demais clientes. A Figura 2.1 mostra a qual camada da Internet alguns desses protocolos pertencem.

A arquitetura dos sistemas VsD, mostrada na Figura 2.2, seria a arquitetura cliente/servidor comumente utilizada na Internet, tendo em uma das bordas da rede o servidor de vídeo, e espalhados pelas bordas os clientes. Para cada cliente tem-se um fluxo de vídeo oriundo do servidor, que passaria pela rede, ou nuvem, e chegaria até o cliente solicitante.

Essa arquitetura parece atender às necessidades dos sistemas VsD, mas ela traz um problema se a rede começar a ficar congestionada. Se não for levado em consideração a banda do servidor para que o mesmo atenda aos clientes, e levar-se em consideração apenas os efeitos do congestionamento, tem-se um aumento da

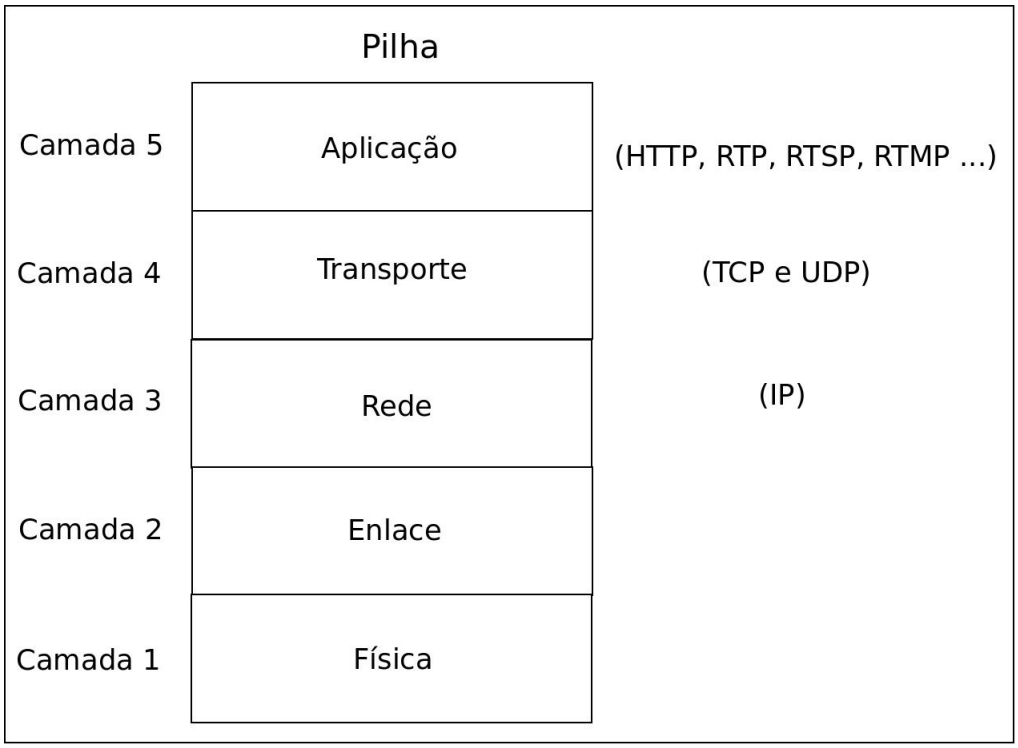

Figura 2.1: Pilha de Protocolos da Internet

variação dos atrasos na rede, o que, como já foi dito, não é tolerado por esse tipo de aplicação. Esse aumento da variação dos atrasos irá degradar a QoS do sistema, chegando a um ponto onde os clientes desistirão de assistir aos vídeos através dessa aplicação.

Esse problema pode se tornar pior se milhares, ou até milhões, de clientes quiserem assistir aos vídeos ao mesmo tempo, quando os mesmos se tornarem populares.

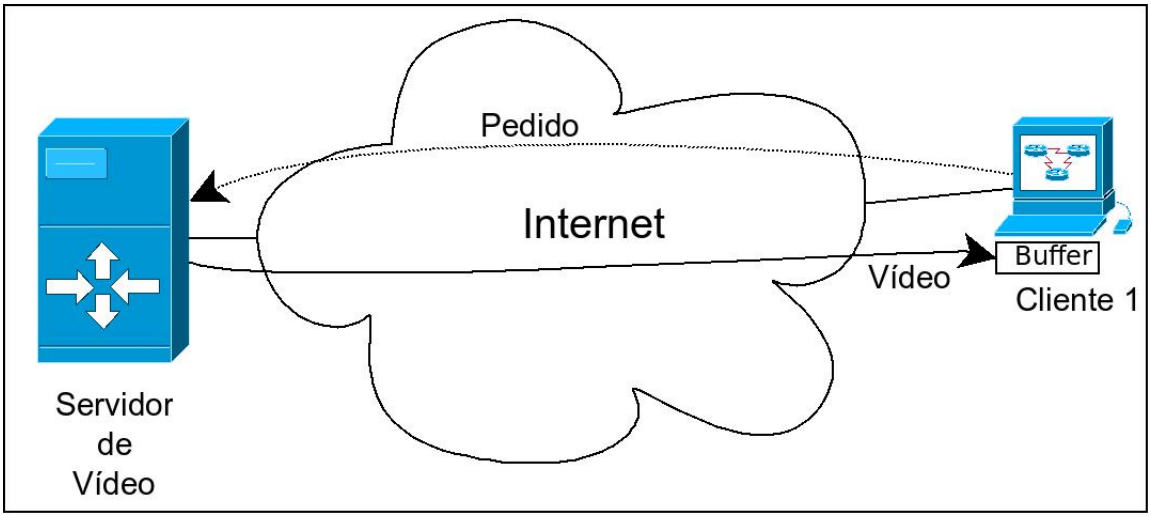

Figura 2.2: Arquitetura dos sistemas VsD

Outro problema a se considerar é o problema da banda que o servidor de vídeo deve possuir para atender a todos os clientes com QoS. Por exemplo, se assumir que o servidor de vídeo possui uma placa de rede de 1Gbps, que está ligada a um canal com a Internet de mesma largura de banda, e que o servidor de vídeo possui um vídeo com taxa de 512Kbps, que não é uma taxa alta para os padrões atuais, então o servidor de vídeo não será capaz de atender mais do que 2048 (1Gbps/512Kbps) clientes simultâneos com QoS (limite superior).

Há maneiras de se amenizar o problema da escalabilidade. Pode-se colocar 1 servidor com uma placa de rede de 10Gbps ligada em diversos canais com a Internet [19, 20], que fossem capazes de atender ao tráfego, elevando o limite de clientes atendidos com QoS para 20480 clientes, mas há um custo alto para manter todos esses canais e para a compra de todo o *hardware* necessário, o que torna essa solução de alto custo efetivo para ser implementada pelos ISPs e pelas companhias de telecomunicação. Esta solução também não seria eficiente diante do problema de congestionamento que a aplicação poderá enfrentar ao prover o vídeo a partir de um componente centralizado.

Outra maneira de amenizar o problema de escalabilidade para a distribuição de conteúdo multimídia e que também ajudaria com o problema do congestionamento, seria usar o *multicast* para a distribuição do vídeo. Com o *multicast* é possível criar grupos de clientes que estejam na mesma parte do vídeo para receberem o vídeo do servidor com apenas um fluxo, o *multicast* se encarregaria de replicar os pacotes quando for necessário para que todos os clientes recebam o vídeo. Mas, como foi apresentado na Subseção 2.1.2, para que a técnica funcione, todas as redes no caminho que os pacotes irão passar devem implementar o *multicast*, o que não acontece na Internet.

Deve-se usar outras técnicas para amenizar os problemas de escalabilidade e congestionamento. Na próxima subseção serão analisado três grupos dessas técnicas.

2.2.3 Técnicas escaláveis para sistemas VsD

Ao longo dos últimos anos, vários autores desenvolveram técnicas que permitiram aos sistemas VsD se tornarem mais escaláveis para a distribuição de conteúdo multimídia. Dentre essas, técnicas dois grupos que usam dois conceitos diferentes, o *peer-to-peer* (P2P) e o de servidores *proxy* (ou substitutos), serão descritos a seguir.

Sistemas baseados em P2P

Os sistemas VsD que utilizam técnicas P2P agregam os recursos dos clientes aos recursos do sistema. Várias técnicas foram desenvolvidas ao longo dos últimos anos, *Chaining* e *Cooperative Video Cache* (CVC)[21] são algumas dessas técnicas que se

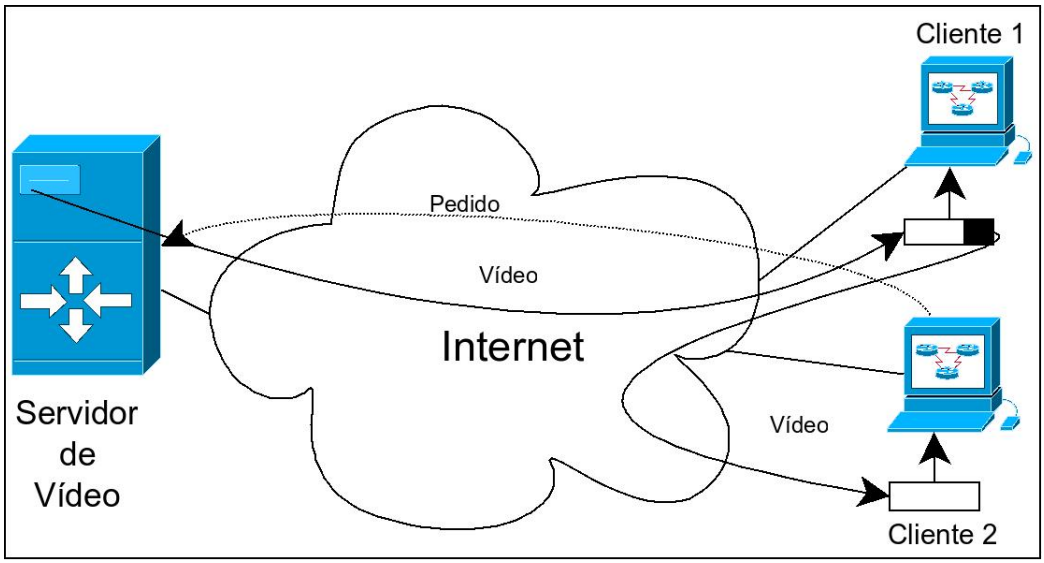

Figura 2.3: Arquitetura da técnica *Chaining*

utilizam do conceito P2P. A técnica *Chaining* será usada para ilustrar este modelo de distribuição.

Nesta técnica, o primeiro cliente do sistema irá requisitar o vídeo para o servidor, e receberá o vídeo do próprio servidor de vídeo. Já os próximos clientes irão requisitar o vídeo ao servidor, mas se os *buffers* dos clientes anteriores ainda possuírem o início do vídeo, ou outro ponto qualquer desejado do vídeo, os novos clientes não irão receber o vídeo do servidor, e sim dos demais clientes que ainda possuírem esse vídeo em memória, como mostra a figura 2.3. O encadeamento tenta utilizar os buffers dos clientes na rede como um único buffer, mas distribuído, agregando mais recursos ao sistema para cada cliente que entrar no sistema, desonerando o servidor de vídeo.

Para que o *Chaining* consiga ser bem aplicado, é necessário que cada cliente possua uma: largura de banda de subida suficiente para atender aos clientes que ele estiver provendo, por este motivo o *Chaining* é melhor aplicado em uma LAN (*Local Area Network*), apesar de nos últimos anos a banda de subida dos clientes em uma WAN (*Wide Area Network*) ter melhorado, permitindo o uso do *Chaining* na WAN também.

Sistemas baseados em Servidores *Proxy*

As abordagens que usam servidores *proxy* (ou servidores substitutos), a partir de agora denominados substitutos, tentam levar o tráfego de vídeo para as bordas da rede, para que o mesmo não contribua com o congestionamento, e para que o sistema aumente a sua escalabilidade, reduzindo os fluxos oriundos do servidor de vídeo e aumentando o número de clientes potenciais para o sistema.

Essa técnica consiste em colocar substitutos nas bordas da rede, mais próximas dos clientes em potencial, para que os mesmos, ao requisitarem um vídeo ao servidor de vídeo, sejam providos pelos substitutos ao invés do servidor, desonerando o mesmo, e evitando que o fluxo de vídeo tenha que atravessar toda a rede para

chegar até o cliente.

A idéia de se usar os substitutos é semelhante ao da memória cache, ou seja, tentar aumentar a velocidade de acesso dos clientes aos vídeos armazenados no servidor de vídeo, já que os substitutos estarão mais próximos dos clientes. Para isso, armazena-se os vídeos nos substitutos, ou nos seus discos, ou ainda em ambos.

Não necessariamente os substitutos serão capazes de armazenar todo o acervo do servidor de vídeo. Quando isto ocorrer, os substitutos serão obrigados a descartarem alguns vídeos, ou parte deles, para que os vídeos que estejam sendo requisitados pelos clientes passem a ser armazenados no substituto. Como essa substituição ocorre dependerá da política adotado pela cache. FIFO (*First-In-First-Out*) e LRU (*Least Recently Used*) são algumas opções. No Capítulo 4 se verá como a popularidade de um vídeo pode ser utilizada como política de substituição, levando o sistema a uma melhor QoS.

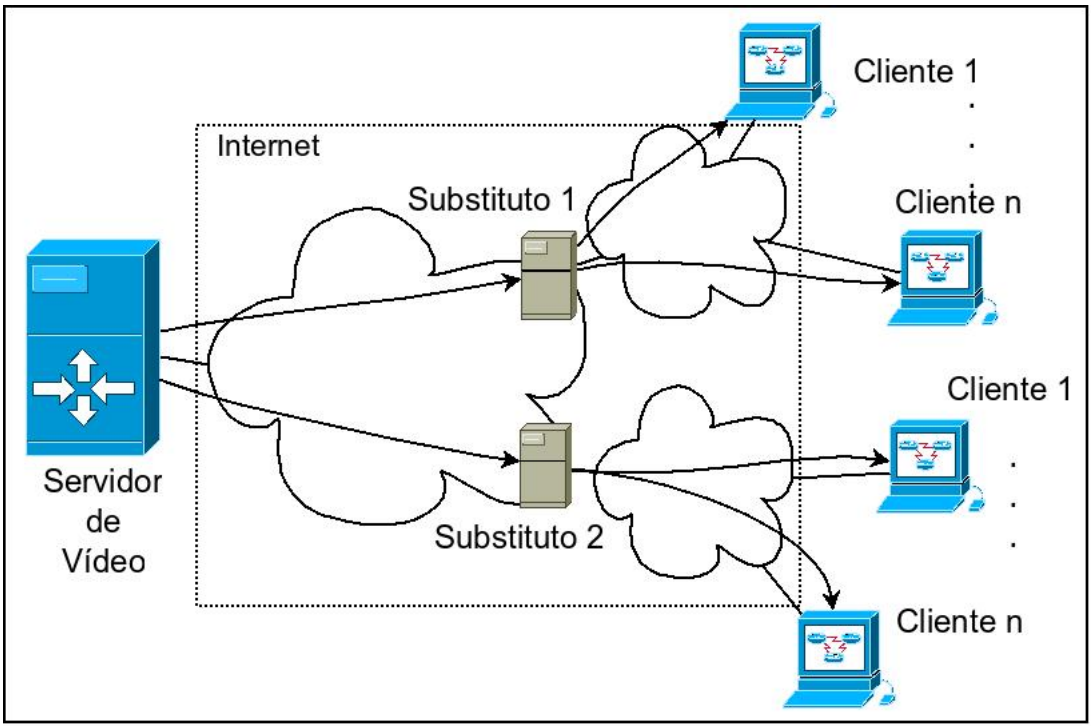

Figura 2.4: Técnica com substitutos

Na figura 2.4, podemos ver a arquitetura dos sistemas VsD com a inclusão de substitutos. Onde antes os clientes eram providos pelo servidor de vídeo, agora são providos pelos substitutos.

Sistemas híbridos

As duas técnicas anteriores permitem aos sistemas VsD alcançarem a escalabilidade desejada a um custo efetivo menor que seria muito bem aplicada nas redes dos ISPs e das companhias de telecomunicação. Mas também se pode combinar as duas

técnicas ortogonalmente em uma mesma solução. Pode-se ter um sistema VsD que utilize tanto a solução P2P, quanto a solução com substitutos. Essa solução irá colocar substitutos nas bordas da rede, para levar o tráfego para depois das bordas, e em cada uma dessas sub regiões criadas pelos substitutos, os clientes irão agregar recursos aos sistemas somando seus *buffers* por região, como podemos ver na figura 2.5.

A escolha de qual componente do sistema (servidor, substituto ou clientes) irá prover vídeo ao cliente dependerá da política adotada. Isto porque haverá situações em que tanto os clientes como o substituto terão a mesma parte do vídeo solicitado pelo novo cliente, e o sistema deverá escolher quem proverá. Uma opção é deixar sempre os clientes proverem, se estiverem próximos. Essas escolhas dependerão da política adotada pelo sistema.

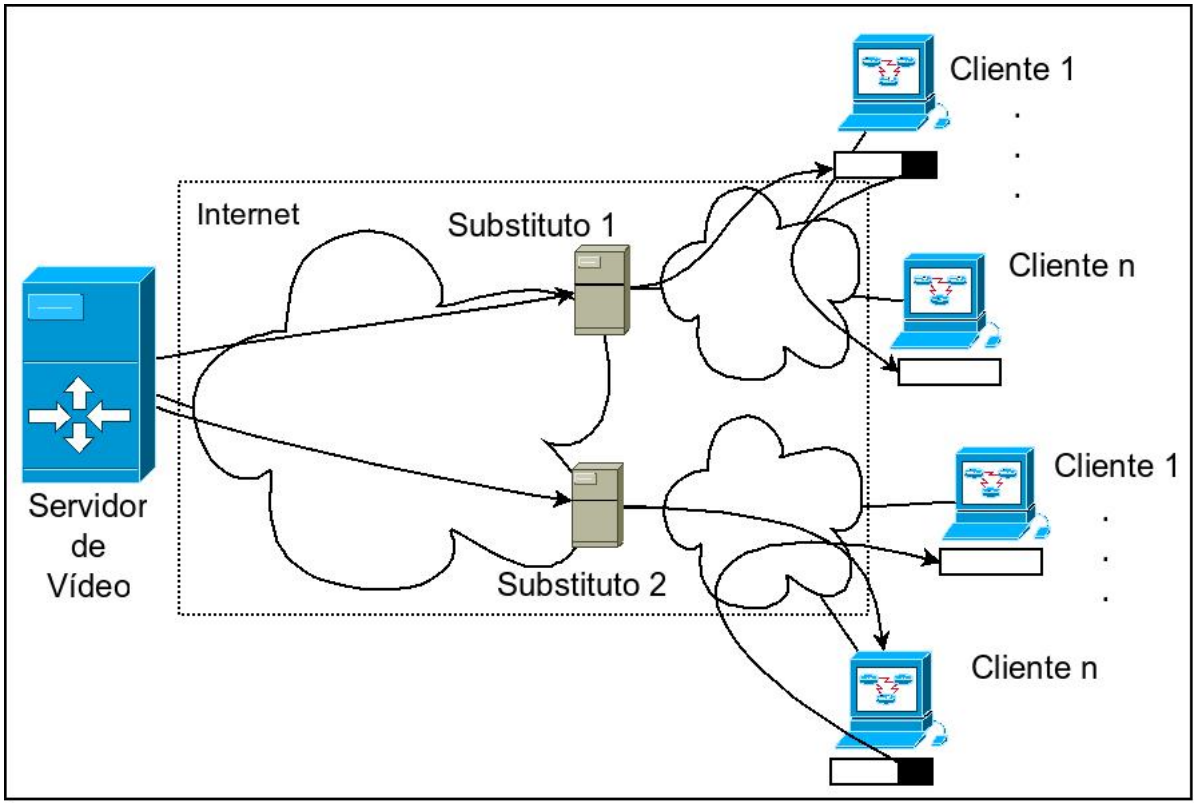

Figura 2.5: Técnica com substitutos e P2P

2.3 Estudo de Caso: GloVE-Mix

O sistema GloVE foi desenvolvido nas dissertações de mestrado de Pinho[2] e Bragato[9], que foram baseadas na tese de doutorado de Ishikawa[1]. A implementação GloVE-LAN usa técnicas P2P, e GloVE-WAN utiliza técnicas com substitutos. Posteriormente Alves[3] combina as duas implementações no sistema

GloVE-Mix, onde demonstra os ganhos de se usar as duas técnicas combinadas e além disso observa que poderiam ser adotadas outras políticas para a escolha do componente provedor, a partir das informações do grafo produzido pelos fluxos de vídeos entre os servidores e clientes do sistema VsD.

O GloVE-Mix é o sistema usado nesta dissertação para realização das medições. Apesar do GloVE-Mix usar técnicas dos dois modelos, MCC e MCD, o escopo desta dissertação se limita à técnica do modelo MCC que utiliza servidores substitutos, que no sistema GloVE são denominados distribuidores, onde se quer mostrar os desempenhos da MCC[22] e suas variações. A MCC é descrita com detalhes no Capítulo 3.

Algumas modificações foram realizadas no GloVE-Mix para se adequar às necessidades dos experimentos desejados:

1. Criação de um *pool* de memória para garantir que todo a memória utilizada pela cache fosse armazenada apenas na memória principal. Este *pool* de memória é estático; um bloco pré-definido é criado quando o distribuidor do sistema GloVE começa a sua execução;

2. Foi modificado no distribuidor GloVE o módulo de admissão de clientes, onde é realizado o cálculo de recurso de memória disponível para determinar se um novo cliente poderá ou não ser atendido pelo distribuidor;

3. Também foi modificado o módulo de gerência da cache para que fosse possível o compartilhamento dos recursos entre os arquivos multimídias presentes na cache do distribuidor do sistema GloVE, que também será explicado em detalhes no Capítulo 3.

2.3.1 Arquitetura

O sistema GloVE-Mix, como monstrado na Figura 2.6, possui a seguinte arquitetura:

Servidor de Vídeo: Componente responsável por armazenar o acervo de vídeo disponível para os clientes do sistema. O servidor de vídeo pode atender a pedidos de qualquer cliente ou de qualquer distribuidor;

SGG: Componente responsável por decidir qual componente do sistema - servidor de vídeo, distribuidor ou outro cliente - irá prover vídeo para o cliente requisitante. O SGG sabe qual parte do vídeo cada componente possui e qual componente está provendo para qual cliente;

Distribuidor: Os distribuidores ficam espalhados pelas bordas da Internet, onde realizam o cache dos vídeos mais populares com o objetivo de oferecer uma

melhor QoS para os clientes requisitantes desses vídeos. Para realizar esta tarefa com maior eficiência, foi implementado o modelo de gerenciamento da MCC para a memória cache dos distribuidores, que decidirá qual parte dos vídeos irá ou não permanecer na memória do distribuidor. Para facilitar a implementação, foi criado o sistema de giros, que a cada intervalo de tempo envia uma quantidade de bytes do vídeo correspondente ao intervalo de tempo, desta forma, mantêm-se a taxa de envio do vídeo para cada cliente que o requisitou;

Cliente: O cliente é o componente responsável por requisitar um vídeo ao SGG, e receber este vídeo do servidor de vídeo, do distribuidor, ou de outro cliente. Enquanto o cliente recebe o fluxo de vídeo de um dos possíveis provedores, o vídeo é exibido para o usuário.

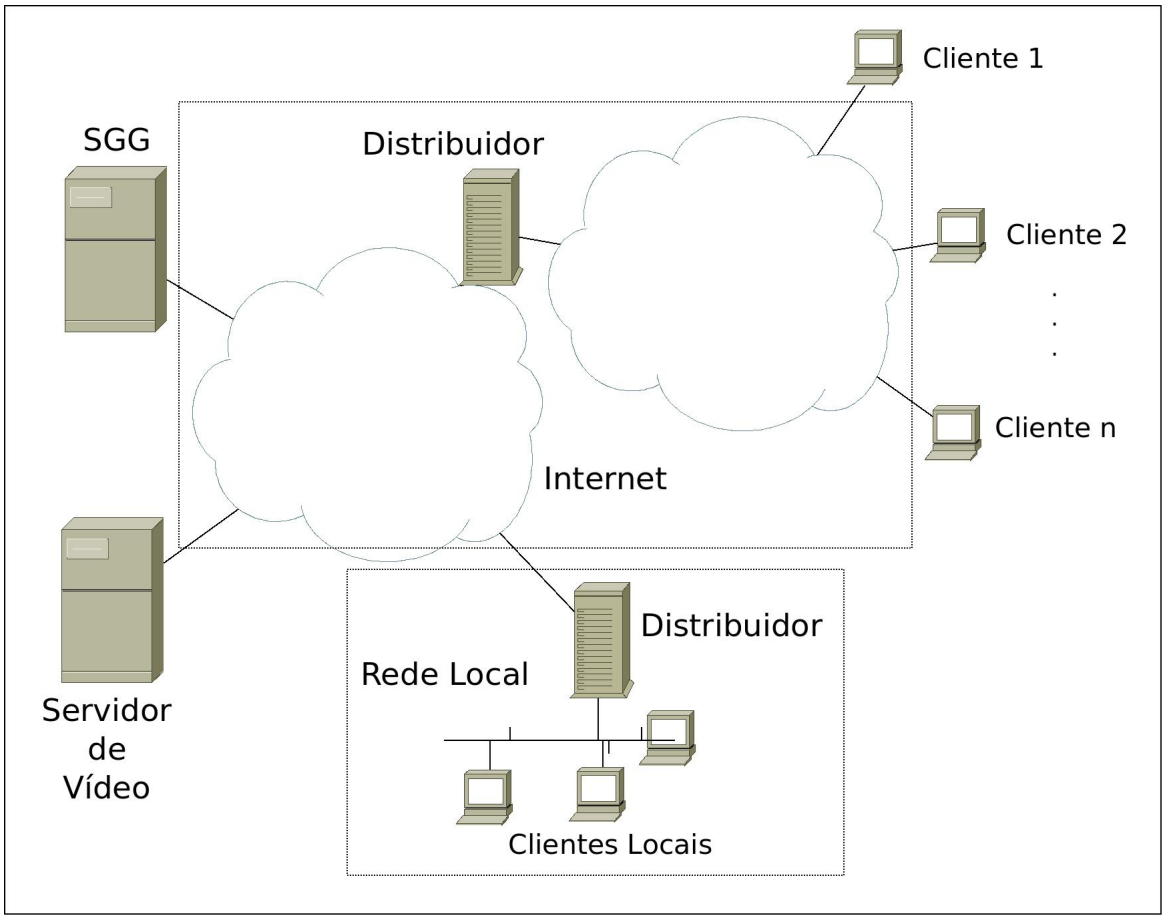

Figura 2.6: Arquitetura do sistema GloVE-Mix

Capítulo 3

Memória Cooperativa Colapsada

3.1 Memória cache

A diferença de largura de banda entre processadores e memórias vem aumentando cada vez mais ao longo dos anos [23]. A necessidade de aumentar o desempenho de acesso às informações contidas na memória dos computadores levou a hierarquização do acesso à memória, organizando-se vários níveis de cache perto dos processadores. Essa organização aumentou o desempenho do acesso aos dados disfarçando a diferença entre as larguras de banda do processador e da memória.

No entanto, devido à memória cache ser muito mais cara que a memória principal, a memória cache normalmente é bem menor que a memória principal[23], ou seja, apenas um pequeno subconjunto das informações contidas na memória principal poderá ser armazenada na memória cache.

Para que a taxa de acertos (*hit*) em memória cache seja alta, foram desenvolvidas estratégias de acesso aos dados contidos na memória cache aproveitando-se da localidade temporal e espacial dos dados requeridos pelo processador.

No contexto dos sistemas VsD, é instrutivo fazer uma analogia com o modelo memória/processador. Os clientes são os processadores, e o servidor de vídeo a memória principal, como na Figura 3.1. Os clientes precisam acessar os vídeos que estão no servidor, mas por causa da rede, a latência para o início da exibição do vídeo e a variação média dos atrasos dos pacotes de mídia podem ser altas, não oferecendo a QoS desejada.

Deste modo, pode-se introduzir um servidor *proxy*, ou substituto (Figura 3.2), equivalente a "memória cache"ao sistema para diminuir a latência dos clientes ao acesso dos vídeos armazenados pelo servidor de vídeo e a variação média dos atrasos dos pacotes de mídia. Similarmente à memória cache, o distribuidor de vídeo tem que utilizar estratégias para deixar a taxa de acertos alta o suficiente para manter a QoS.

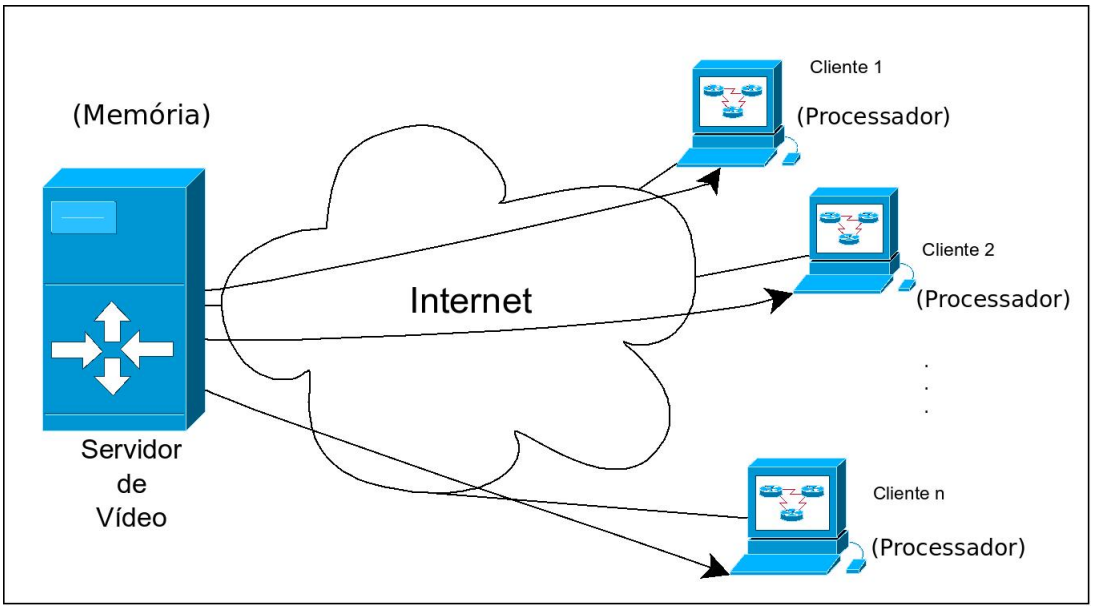

Figura 3.1: Analogia da arquitetura de memória com sistemas VsD.

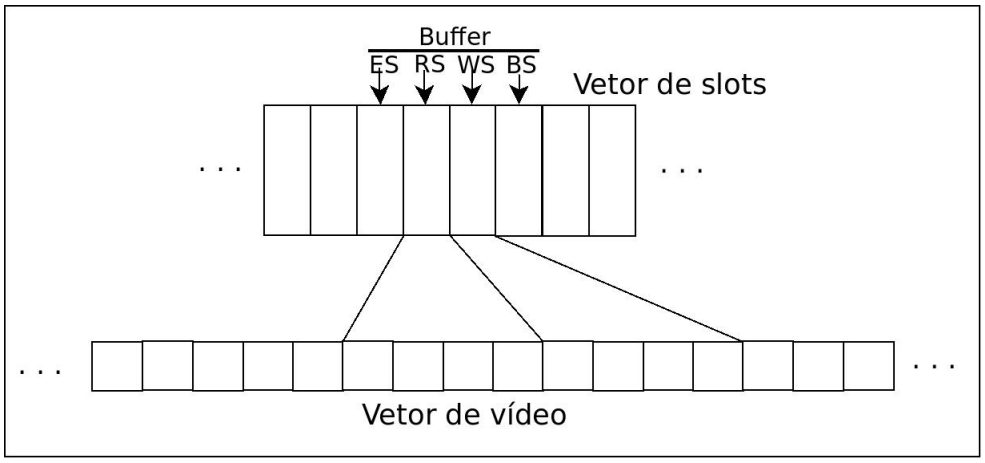

Figura 3.2: Analogia da arquitetura da memória cache com sistemas VsD.

3.2 Cache em Servidores *Proxy* de vídeo

Os servidores *proxies* de vídeo se diferem dos servidores proxies de conteúdo convencional por armazenarem arquivos de conteúdo multimídia que são muito maiores que as páginas HTML armazenadas pelos servidores de *proxy* de conteúdo convencional. Por esse motivo, os servidores *proxy*, ou substitutos, se utilizam de cache dinâmica, que leva em conta a relação temporal que existe entre os vários segmentos de um mesmo vídeo. O vídeo é dividido em vários segmentos, e esses segmentos se relacionam uns com os outros através da sua distância temporal[24–26]. Com isso, a distância temporal entre dois pedidos pode ser escondida se o conteúdo dos pedidos estiver na cache dinâmica.

Tradicionalmente, os distribuidores de vídeo com cache dinâmica utilizam

estruturas similares a vetores circulares, chamados de *ring buffers*[27]. Os *ring buffers* são alocados com um tamanho que corresponde a uma fração do vídeo. À medida que o *streaming* de vídeo vai chegando ao distribuidor, o *ring buffer* é preenchido com esse *streaming*. Quando o *ring buffe*r está completamente cheio, a primeira parte do vídeo que foi colocada no *ring buffer* é descartada para que uma nova parte oriunda do servidor de vídeo possa ser colocada no *ring buffer*. Essa política de FIFO (*First-In-First-Out*) dá a impressão que o *ring buffer* possui o comportamento de uma janela deslizante sobre o fluxo de vídeo.

Um vídeo pode ter mais de um *ring buffer* alocado, isso dependerá da ordem de chegada dos pedidos. Por exemplo, tem-se um servidor *proxy* onde o primeiro pedido chega em t_1 e um *ring buffer* é alocado para o cliente. Em t_2 chega um segundo pedido, se $t_2 - t_1 < x$, o mesmo *ring buffer* será usado para atender ao pedido, caso contrário, um novo *ring buffer* poderá ser alocado. Contudo esse dois *ring buffers* podem se unidos e formarem um único *ring buffer*, tendo apenas um fluxo de vídeo oriundo do servidor de vídeo. Para isso a distância entre os dois *ring buffers* tem que ser alocada também, havendo uma ocupação maior da cache. Essa escolha dependerá da política escolhida pelo servidor *proxy*.

3.3 Memória Cooperativa Colapsada (MCC)

O modelo da MCC[1, 22] surge da tentativa de se aproveitar das vantagens do modelo da Memória Cooperativa Distribuída (MCD)[1, 28–32], mas eliminando sua desvantagem. Na MCD os clientes do sistema armazenam partes do vídeo em seus *buffers* locais (Figura 3.3), e passam a serem potenciais provedores de vídeos. Deste modo, quando algum cliente requisitar alguma parte do vídeo, e este estiver armazenado em algum *buffer* local, o cliente requisitante será provido pelo cliente detentor da parte de vídeo, ao invés de ser provido pelo servidor de vídeo, como mostrado na Figura 3.4.

Contudo essa técnica *peer-to-peer* necessita que os clientes possuam uma banda de subida suficiente para atender os clientes requisitantes, o que nem sempre é possível na WAN, e existe o aumento do tráfego entre clientes, que pode levar a um congestionamento no *backbone*.

A MCC surgiu da idéia de colapsar os *buffers* dos clientes, que antes eram distribuídos, em um único componente do sistema, no servidor *proxy* (denominado distribuidor na termologia do sistema GloVE), que irá realizar a gerência desses *buffers*, como mostrado na Figura 3.5.

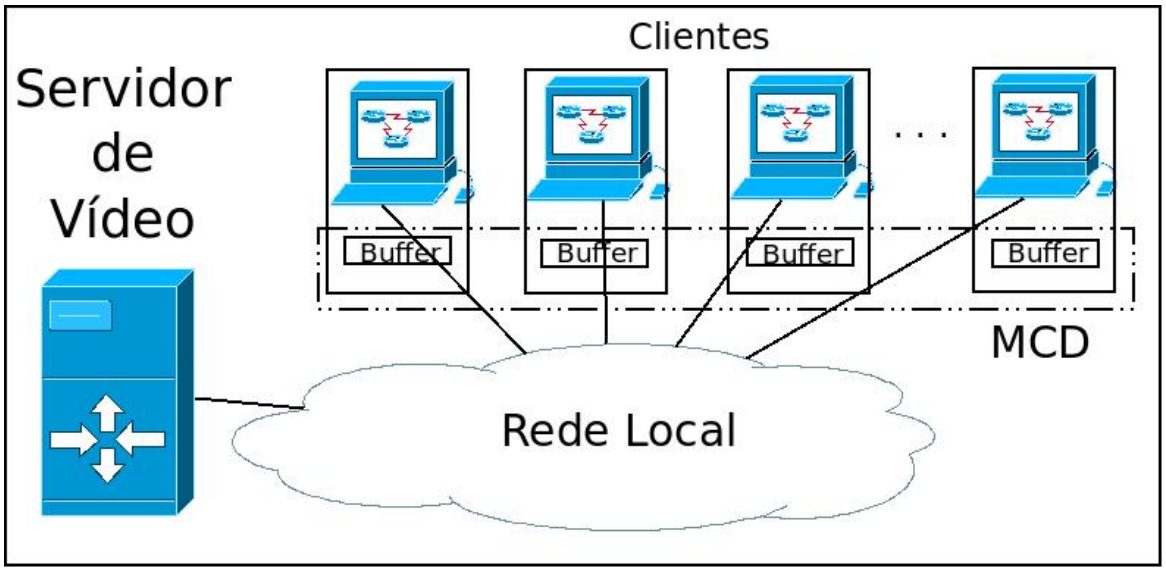

Figura 3.3: Memória Cooperativa Distribuída.

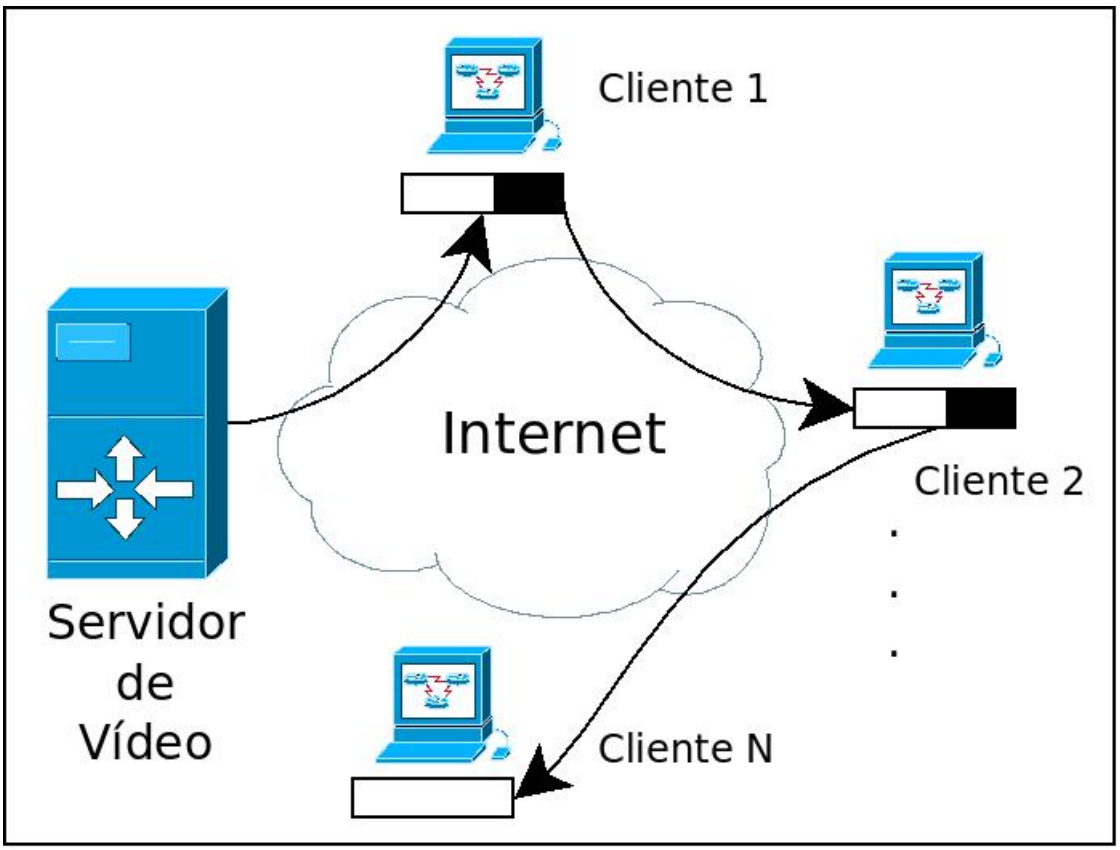

Figura 3.4: Arquitetura da Memória Cooperativa Distribuída.

3.3.1 Estruturas e conceitos da MCC

A estrutura da MCC é mais complexa que a dos *ring buffers*, sendo difícil de implementar. A MCC possui, para cada vídeo que será armazenado na cache, um anel duplo, onde o primeiro é um vetor de *slots* (metadados de controle) e o segundo

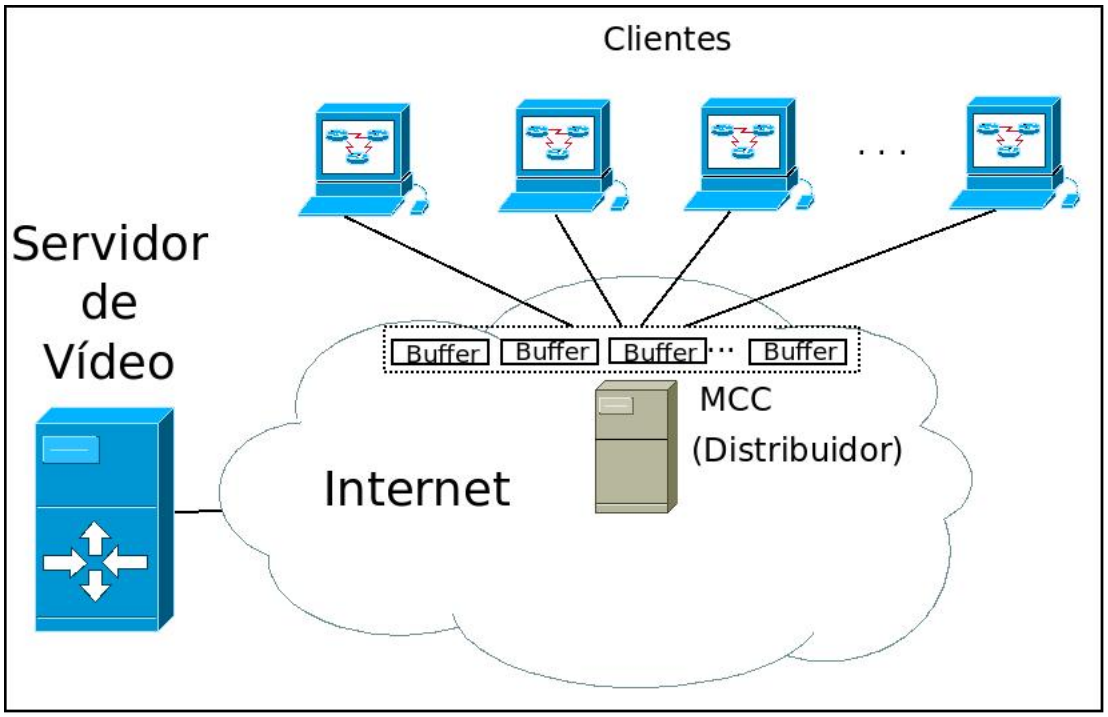

Figura 3.5: Memória Cooperativa Colapsada.

armazena o vídeo, como mostrado na Figura 3.6.

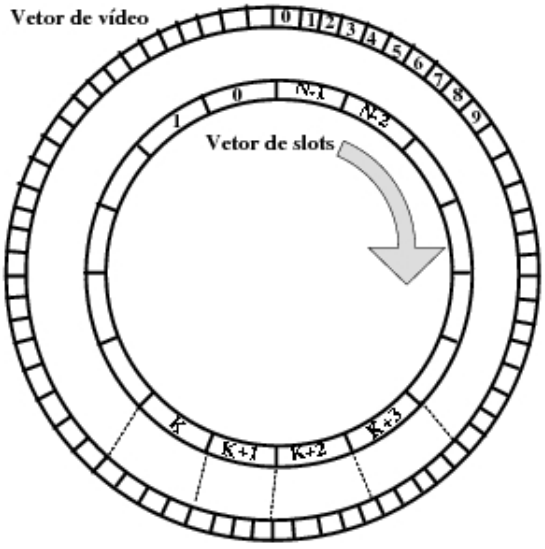

Figura 3.6: Vetor duplo em anel.

Esses *slots* simbolizam porções de vídeos que estão armazenados na cache, como mostrado na Figura 3.7, e toda a gerência da MCC é realizada em cima desse vetor de *slots*. Os *slots* podem receber marcas (*tags*), que indicam seu estado e função atual:

- OS - do tipo vazio ou sem prioridade;

- RS - do tipo de leitura (*Read Slot*);
- WS - do tipo de escrita (*Write Slot*);
- BS - do tipo início de *buffer* (*Begin Slot*);
- ES - do tipo fim de *buffer* (*End Slot*);

Com exceção do OS, que é o *slot* sem prioridade, os *slots* de todos os outros tipos possuem prioridade de permanecerem na cache, ou seja, não podem ter as porções de vídeo, por eles representadas, removidas da cache para liberar recursos. A MCC pode possuir um ou mais buffers por vetor de slots e os buffers podem ter um ou mais clientes. Cada buffer é representado no vetor de slots começando pelo tipo de slot BS (*Begin Slot*), e terminarem pelo tipo de slot ES (*End Slot*). Em geral, um buffer ocupa quatro slots do vetor tendo a seguinte ordem de tipos de slots: ES, RS, WS, e BS. O slot RS representa a porção temporal do vídeo que está sendo, ou será em breve, enviada para o cliente e o slot do tipo WS representa a porção do vídeo que está sendo recebida do servidor de vídeo por meio de um fluxo de vídeo (streaming). Os slots do tipo ES e BS limitam o buffer e realizam o controle para evitar *underflow* e *overflow*.

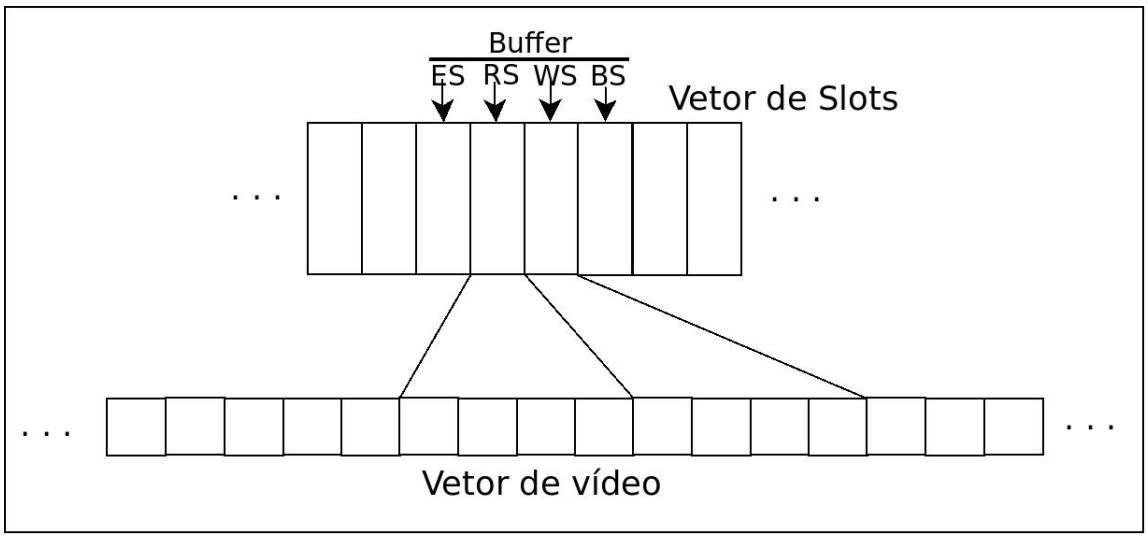

Figura 3.7: Projeção do anel de vetor de *slots* sobre o vetor de vídeo.

Neste modelo, dois *buffers* se colapsam quando um *buffer* que está sendo criado irá utilizar parte dos *slots* de um *buffer* já existente, como mostrado na Figura 3.8. O *buffer* 1 que já existia ocupava os *slots* K, $K-1$, $K-2$ e $K-3$. Quando o *buffer* 2 é criado ele irá ocupar os *slots* $K-1$, $K-2$, $K-3$ e $K-4$. Os *slots* $K-1$, $K-2$ e $K-3$ formam uma intersecção entre o *buffer* 1 e 2, deste modo, os *buffers* são colapsados e apenas um único *buffer* existirá ocupando os *slots* K, $K-1$, $K-2$, $K-3$ e $K-4$. Esse *buffer* ocupará 5 *slots* e possuirá dois *slots* do tipo

RS, já que pelo menos dois clientes que estão pedindo vídeo em partes diferentes, caracterizando um *buffer* colapsado.

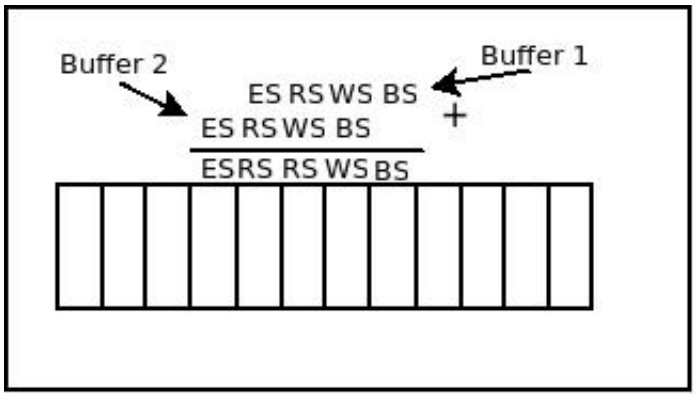

Figura 3.8: Colapsando dois *buffers*.

3.3.2 Os *Link Slots* (LS)

Existe ainda um tipo especial de *slot*, o *slot* de ligação, ou *link slot* (LS). Este *slot*, assim como os outros, possui prioridade na sua permanência na cache, mas com um grau a menos que os demais tipos – em alguns casos o *slot* do tipo LS poderá ser removido da cache.

O objetivo do LS é unir dois *buffers* que não foram colapsados, como se eles estivessem sendo colapsados em um único *buffer*, como mostrado na Figura 3.9-a. Desta forma os *buffers* colapsados se comportam como um único *buffer* e apenas um fluxo de vídeo (*streaming*) será necessário.

Contudo, diferentemente de quando são unidos os *ring buffers*, esses LS podem ser removidos separando os dois *buffers* e permitindo que a porção de vídeo onde antes estivesse a seqüência de LS possa ser removida, mas com o prejuízo de se ter mais um fluxo de vídeo, como mostrado na Figura 3.9-b.

Essa habilidade da MCC permite que a cache seja mais flexível, removendo porções de vídeos que atualmente não estão sendo usadas para atender a algum cliente do sistema, e permitindo que outros vídeos tenham a oportunidade de serem armazenados na cache.

Uma das características da MCC é o reuso de fluxos de vídeo oriundos do servidor de vídeo para atender a maior quantidade de clientes possível, ou seja, quanto mais clientes houver no sistema e menos fluxos de vídeos oriundos do servidor de vídeo, maior será a eficiência da MCC, ou maior será a taxa de acerto da cache.

É provável que a cache não seja grande o suficiente para armazenar todos os vídeos do acervo do servidor de vídeo. Neste caso, é necessário remover alguns *slots* de vídeo que estão armazenados na cache, e que não estão sendo utilizados por

nenhum cliente, para que novos *slots* de vídeos requisitados sejam armazenados na cache. Para manter a taxa de acerto da cache alta, a estratégia de escolha de qual *slot* de vídeo remover tem que ser eficiente.

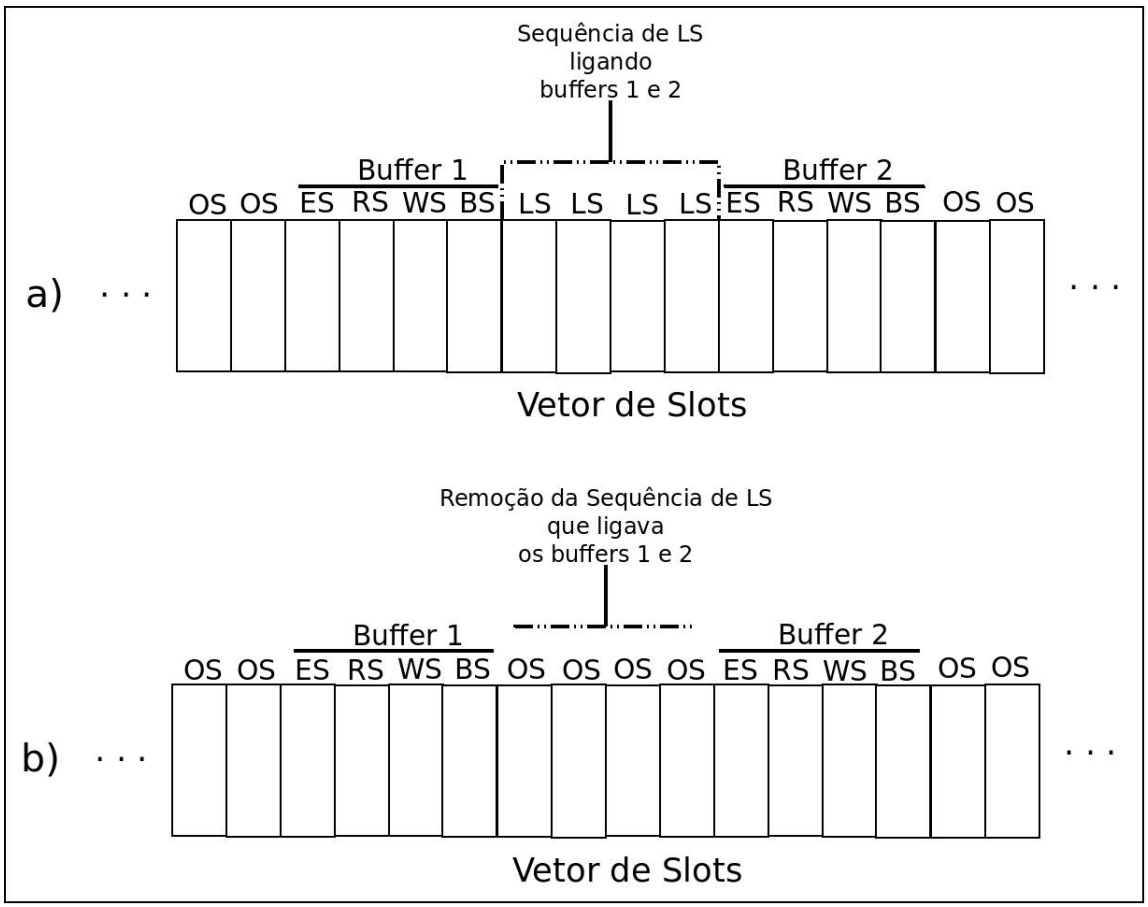

Figura 3.9: Remoção de seqüência de LS.

Algoritmo de remoção de LS

O problema de remoção de LS, como demonstrado por Ishikawa[1], é um problema NP - Completo, portanto muito custoso para se alcançar um resultado ótimo. Mas passível de uma solução próxima da ótima, a um custo menor.

O algoritmo consiste em organizar as seqüência de LS de vídeo levando-se em consideração o custo em removê-las. Para determinar o custo de remoção de cada seqüência de LS, é usada a seguinte equação:

$$F(d, f, t) = \frac{d}{minimo(t, f)}$$

Onde d é a distância do primeiro *slot* da seqüência de LS até o final do vetor de *slots* do vídeo, t é o tamanho da seqüência de LS em *slots*, e f é o número de *slots* que devem ser removidos para atender a demanda de recursos.

A intenção deste cálculo é priorizar a remoção de seqüência de LS que sejam maiores, em termos de número de *slots* que serão removidos, e que estejam mais próximos do final do vetor de *slots*, porque deste modo o novo fluxo de vídeo que será criado para atender os clientes que estiverem no(s) *buffer(s)* anterior(es) terá a menor duração possível, como mostrado na Figura 3.10.

O custo de remoção não é o único critério para a MCC escolher a seqüência de LS que será removida. A MCC também se utiliza da influência da popularidade de cada vídeo para tomar a decisão sobre qual seqüência de LS irá remover.

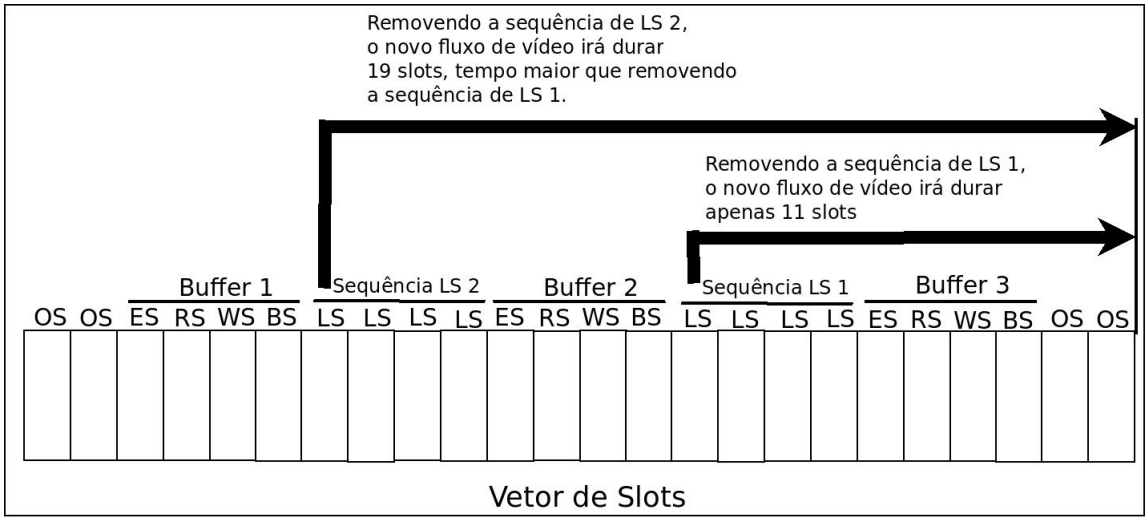

Figura 3.10: Escolha da seqüência de LS a ser removida.

Popularidade dos vídeos em sistemas VsD

Cada pessoa possui uma preferência, e é difícil adivinhar o que cada indivíduo irá preferir. No entanto, um conjunto suficientemente grande de pessoas possui tendências em comum que podem ser modeladas na distribuição Zipf.

Em síntese, a distribuição Zipf, dentro do contexto dos sistemas VsD, modela que as pessoas contidas em um grande subconjunto de clientes irão escolher os mesmos vídeos contidos em um pequeno subconjunto do acervo de vídeos. Esta característica de escolha das pessoas aponta para uma política de popularidade dos vídeos, priorizando a manutenção dos *slots* de vídeo mais populares (com o maior número de clientes) na cache, em detrimento dos vídeos menos populares. Mantendo os vídeos mais populares na memória, obteremos mais acertos (*hits*) no acesso a cache, e mantendo a QoS para um maior número de clientes do sistema.

3.3.3 Sistema de admissão de clientes

Antes que um cliente que realizou um pedido possa ser atendido pela MCC, uma verificação é realizada com objetivo de avaliar se há um mínimo de recursos capaz

de atender este novo cliente com QoS. Caso não haja recursos necessários o cliente será bloqueado, ou seja, ele não receberá vídeo desse distribuidor. A verificação é realizada da seguinte forma:

1. Primeiro verifica-se o número de *slots* necessários que terão que ser alocados para o novo cliente;
2. Verifica-se o número de *slots* disponíveis;
3. Se o número de *slots* disponíveis for maior ou igual que o número de *slots* necessários para aceitar, então o cliente é aceito, caso contrário ele será bloqueado.

O número de *slots* necessários irá depender se um novo *buffer* será criado ou não, e se o novo *buffer* criado irá se beneficiar de *slots* já com vídeos na cache ou não. Se o cliente pediu uma porção de vídeo onde já há um *buffer*, então nenhum *slot* será necessário e esse cliente se juntará ao *buffer* já existente, como mostrado na Figura 3.11-a. Se não há um *buffer* na posição onde o cliente pediu o vídeo, então um novo deverá ser criado, como mostrado na Figura 3.11-b. Mas se o *buffer* for criado onde já existe vídeo na cache, então não haverá a necessidade de se alocar todos os *slots* como mostrado na Figura 3.11-c.

O número de *slots* disponíveis dependerá do tamanho da cache. Se o acervo total do servidor de vídeo for de 100 *slots* e a cache representar apenas 15% do acervo, ou seja, 15 *slots*, a MCC irá verificar quantos desses *slots* já estão preenchidos; caso todos estejam, a MCC tentará liberar os *slots* que não estiverem reservados e irá considerar os com LS para calcular o número de *slots* disponíveis, ou seja, eles não serão considerados como *slots* alocados, como mostrando na Figura 3.12.

3.3.4 Funcionamento

A melhor maneira de se entender a MCC é seguir um exemplo de seu funcionamento, onde serão ilustrados os conceitos da seção anterior.

Um distribuidor irá realizar o armazenamento em memória dos três vídeos: V1 que tem o tamanho de 15 *slots*, V2 que tem o tamanho de 40 *slots* e V3 com 25 *slots*. A capacidade da cache é 30 *slots*. Um *slot* representa 10 segundos de vídeo, que pode ocupar um ou mais blocos de vídeo na memória cache, mas para simplificar o exemplo, a cache será divida em *slots* também. Essa divisão pode ser observada na Figura 3.13, onde cada vídeo já possui o primeiro *slot* em memória.

Em um dado tempo t_0, um cliente requisita V3 ao distribuidor e um *buffer* é criado, como mostra a Figura 3.14. A partir deste instante, um fluxo (seta para baixo) de vídeo é gerado pelo distribuidor, a partir do *slot* RS, com destino ao

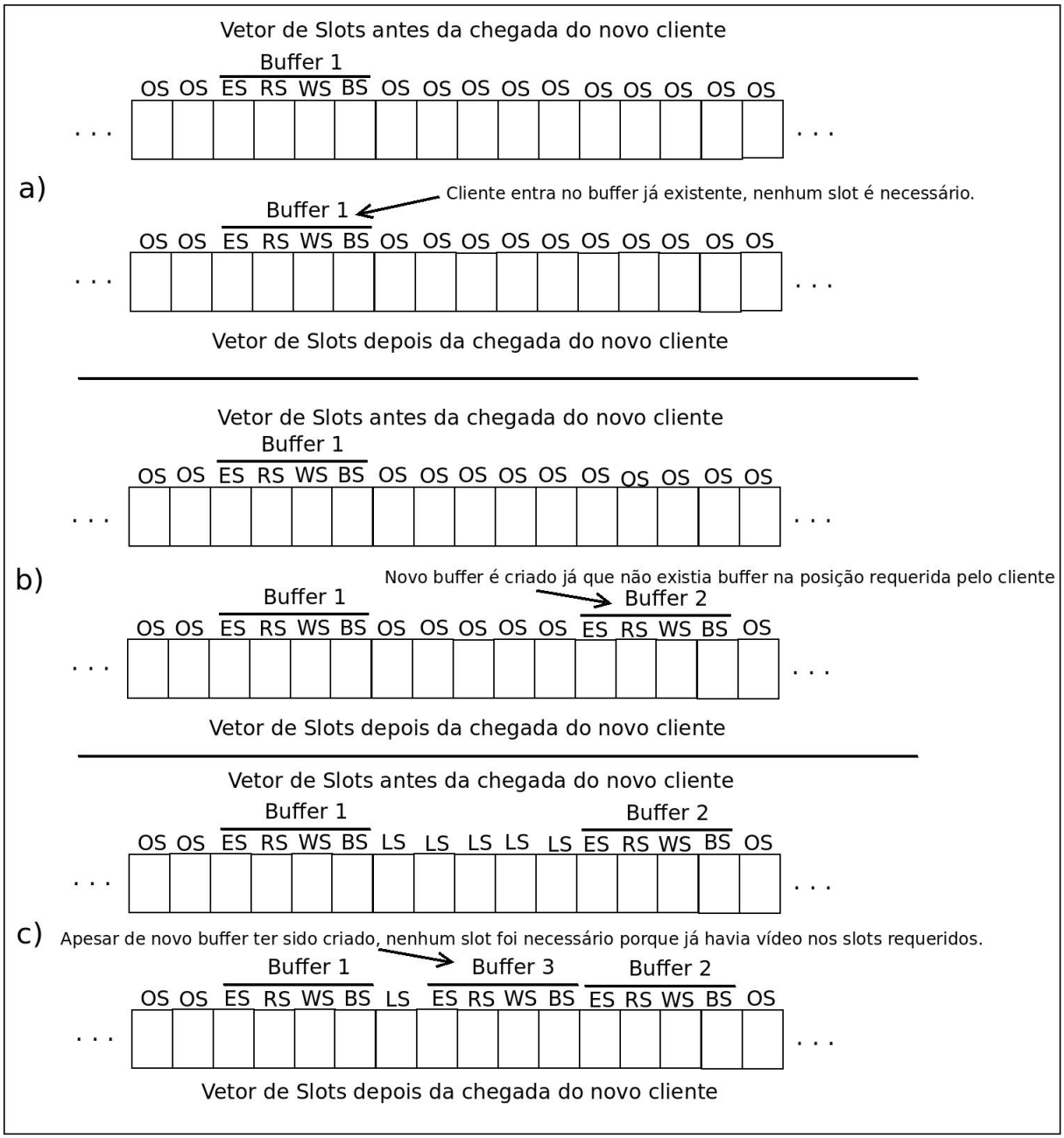

Figura 3.11: Exemplo de admissão de cliente.

cliente requisitante, e um segundo fluxo passa ser recebido pelo distribuidor, no *slot* WS, oriundo do servidor de vídeo (seta para cima).

Alguns instantes depois, em t_1, tem-se mais três clientes requisitando vídeo ao sistema, onde cada cliente requisita um dos vídeos disponíveis. Deste modo, três *buffers* são criados, um em cada vídeo, como mostrado na Figura 3.15. Como 7 *slots* já haviam sido decorrido desde que o primeiro cliente de V3 havia entrado no sistema, 8 *slots* de V3 já estão com vídeos na cache. Quando o segundo *buffer* é criado, uma seqüência de LS é criada para unir os dois *buffers* de V3. Com isto, irá priorizar a permanência do vídeo que está nos *slots* entre ambos os *buffers*. A

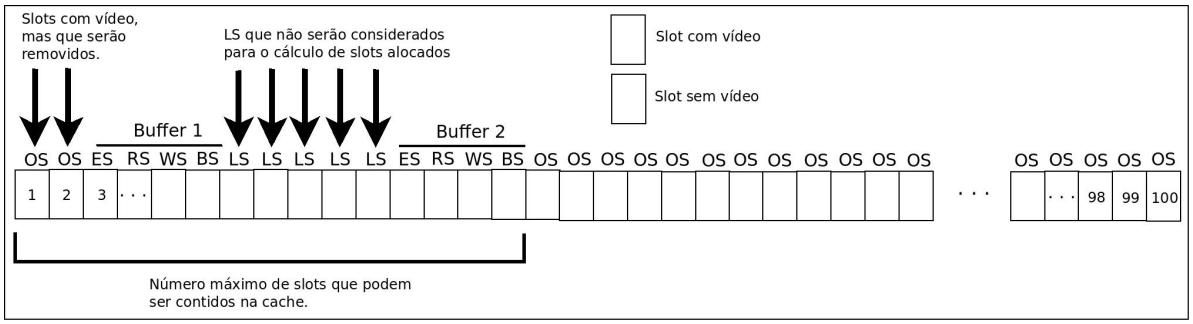

Figura 3.12: Ilustração de como os LS são considerados no cálculo de *slots* disponíveis.

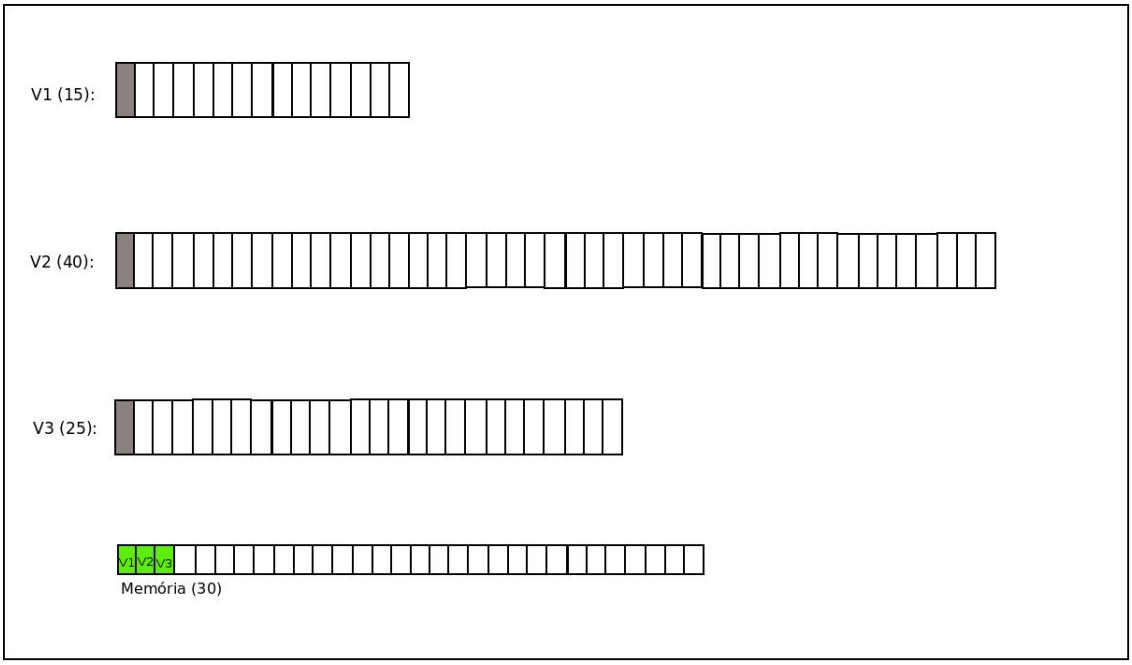

Figura 3.13: Estágio inicial da MCC com apenas um *slot* de cada vídeo em cache.

medida que o vetor de slot gira sobre o vetor de vídeo, os slots LS irão garantir que esses vídeos não serão removidos. Como é possível observar na Figura 3.15, V3 possui dois fluxos de vídeos saindo do distribuidor e indo em direção aos clientes, e apenas um fluxo de vídeos chegando oriundo do servidor de vídeo. Isto ocorre porque, apesar do novo *buffer* alocado possuir um *slot* WS, o mesmo já possui vídeo em memória, que foi previamente requisitado ao servidor de vídeo para atender ao(s) cliente(s) do primeiro *buffer* alocado. Deste modo, não será necessário pedir este *slot* de vídeo para o servidor de vídeo.

Já em V1 e V2, que antes não possuíam clientes, um buffer é criado em cada para atender aos novos clientes e um fluxo de vídeo chegando e outro saindo são também são criados. Neste momento, temos que V1 e V2 possuem um *slot* ocupado com vídeo cada, e V3 possui 8 *slots* ocupados, somando um total de 10 *slots* armazenados na cache, como mostrado na Figura 3.15.

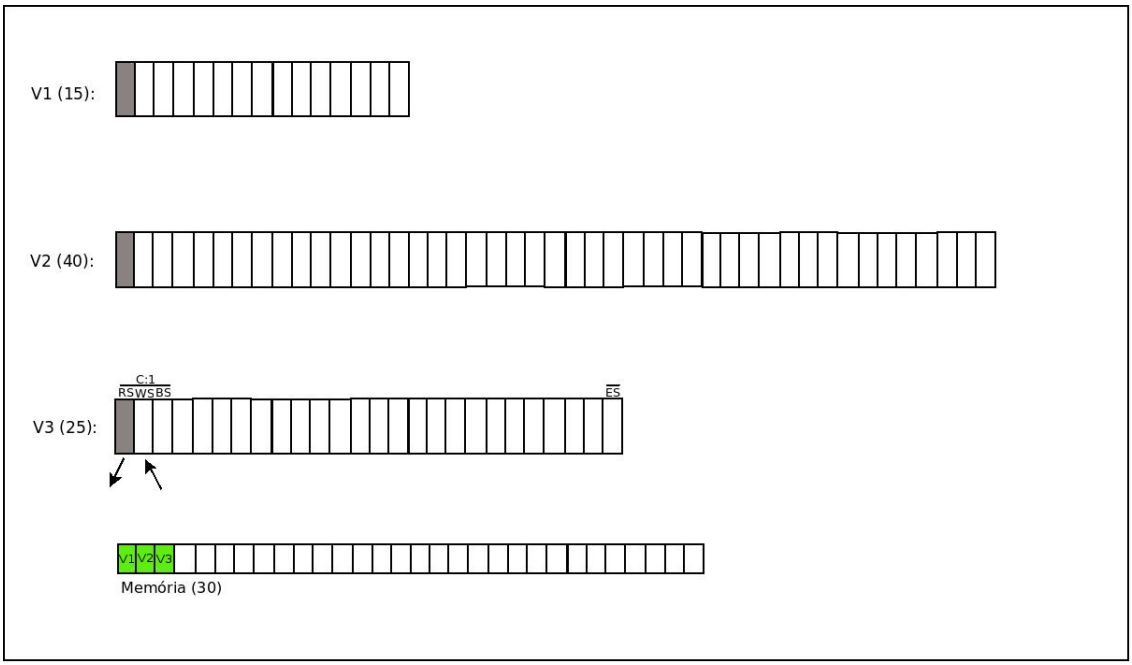

Figura 3.14: Vídeo V3 - primeiro *buffer* de cliente é criado com *slots* RS, WS e BS.

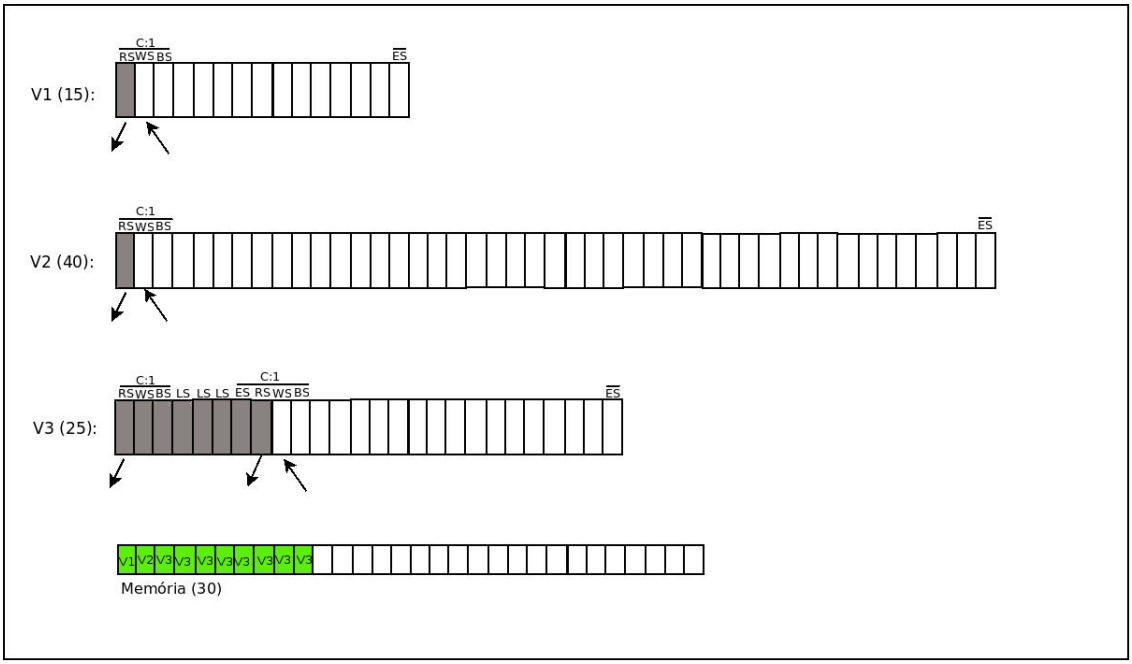

Figura 3.15: Criação dos primeiros LS para conectar dois *buffers*.

Em t_2 mais quatro clientes entram no sistema, três requisitando V2 e um V3. Mais um *buffer* é criado em cada vídeo, e novamente seqüências de LS são criadas para ligar os novos *buffers* com os *buffers* já presentes. O novo *buffer* de V2 conterá os três clientes, já que estes requisitam a mesma parte do vídeo e chegaram juntos. Contando os slots já ocupados, tem-se V1 com 7, V2 com 8 e V3 com 15, somando 30 slots ocupados, o que esgota o espaço da cache, como mostrado na Figura 3.16.

Quando em t_3 um novo cliente requisita V1, e um novo *buffer* é criado,

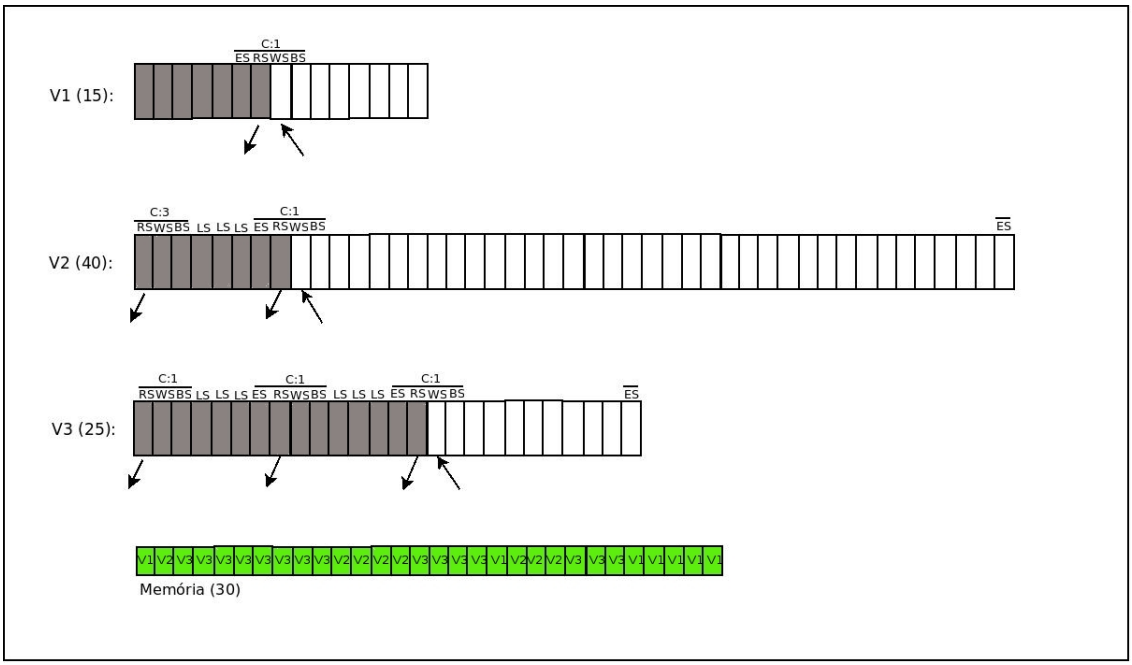

Figura 3.16: Criação de *buffer* com três clientes e cache saturada.

juntamente com a seqüência de LS para ligar os dois *buffers* de V1, tem-se que V1 passou a ter 8 *slots* ocupados, V2 9 *slots* e V3 16 *slots* tendo-se um total de 33 *slots*, que é maior que a capacidade da cache, ou seja, esse cenário não é sustentável pois há uma falta de recursos para mantê-lo, como mostrado na Figura 3.17.

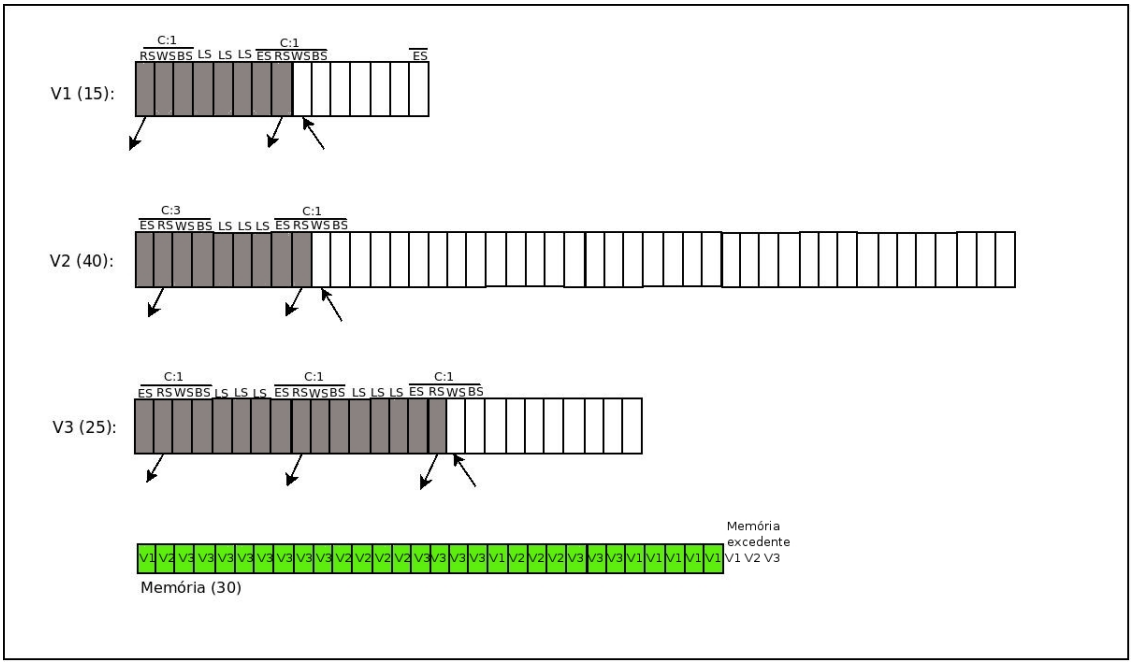

Figura 3.17: Primeiro cenário onde há mais *slots* alocados do que a capacidade da cache.

Contudo, quando em t_3 o novo cliente de V1 entrar no sistema, ao invés de se criar uma seqüência de LS, o que ocorre é que o distribuidor identifica que ao se

incluir o novo cliente em V1 mais recursos serão necessários, ou seja, algum vídeo deverá ser removido da cache. Seguindo a política de popularidade, o distribuidor irá procurar o vídeo menos popular, que possui menos clientes, e tentará remover o número de *slots*, não priorizados, suficiente para atender aos requisitos de memória, ou ainda, caso não haja *slots* não priorizados o suficiente, *slots* LS poderão ser removidos para atender a demanda de recursos. Como V1 é o menos popular, os *slots* removidos serão dele mesmo, como mostrado na Figura 3.18. Deste modo V1 passa a ter 5 *slots* ocupados, V2 9 e V3 16, somando-se 30 slots, que é a capacidade da cache.

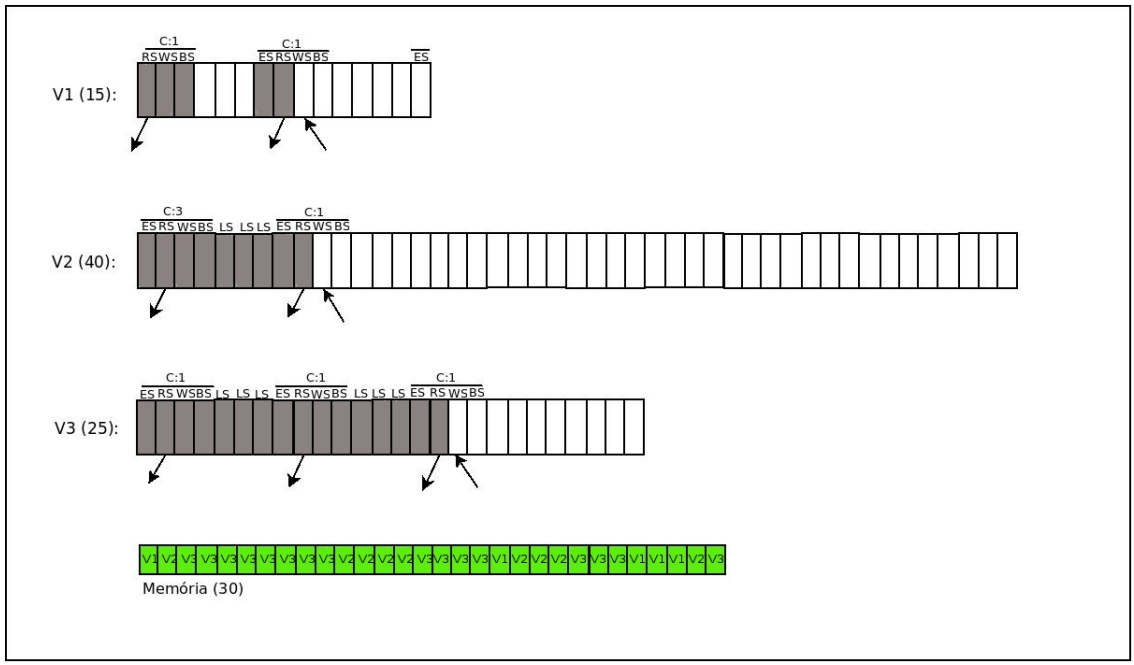

Figura 3.18: Remoção de LS para liberar recursos.

Em t_4 tem-se novamente uma situação onde mais recursos têm que ser liberados, como mostrado na Figura 3.19. Um novo cliente requisita V2, deste modo tem-se que V1 possui 5 slots ocupados, V2 tem 10 e V3 manteve seus 16, mas a soma resulta em 32 slots, o que ultrapassa o limite de 30 por 2. Para que o novo cliente possa ser atendido, será liberado um *slot* de V3 que não possui prioridade de permanência na memória. Mas como apenas este *slot* não é suficiente, a primeira seqüência de LS de V3 tem que ser liberada, já que V3 é o segundo vídeo menos popular que ainda possui recursos para serem liberados, como mostrado na Figura 3.20. A escolha da primeira seqüência de LS para ser removida se dá pelo fato desta seqüência estar mais próxima do final do vídeo, e como todas as seqüências de LS de V3 possuem o mesmo tamanho, de acordo com a equação mostrada na Seção 3.3.2, a última seqüência de LS possui o menor custo de remoção. E como apenas um *slot* é necessário para a soma ser 30, só um terá vídeo desalocado, mantendo os outros dois *slots* com vídeo.

No tempo t_5, observa-se o pedido de mais três clientes para V2, cada cliente

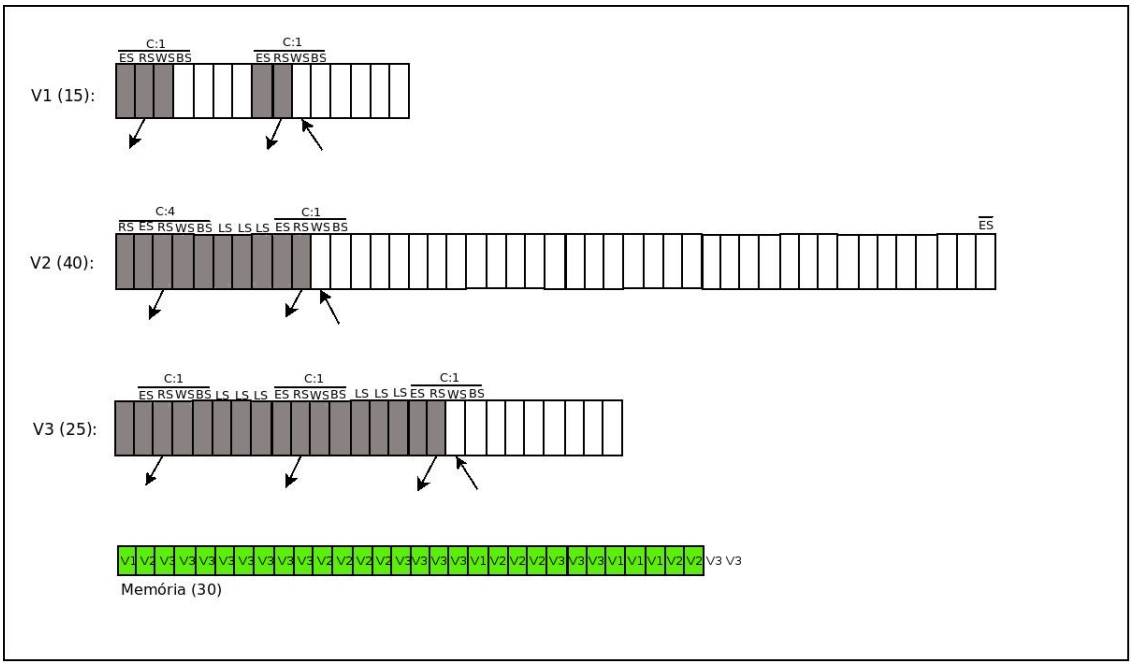

Figura 3.19: Segundo cenário onde há mais *slots* alocados do que a capacidade da cache.

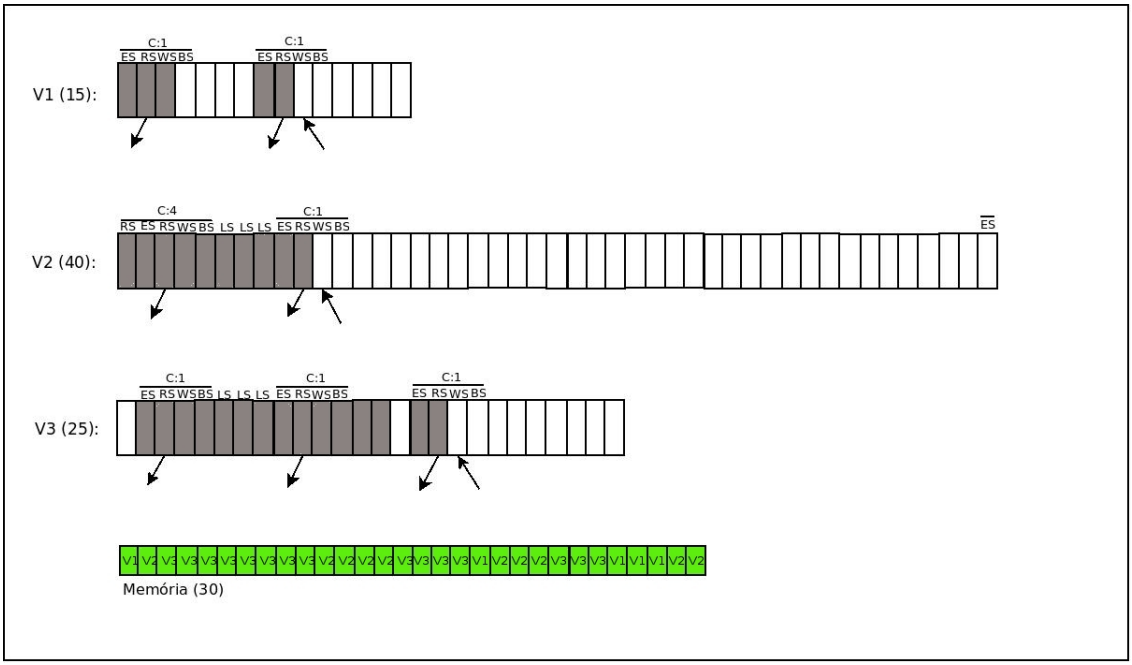

Figura 3.20: Remoção de mais LS para liberação de recursos.

pedindo em pontos diferentes do vídeo. Porém, nem todos os três clientes serão atendidos; o último cliente a ser atendido será bloqueado por falta de recursos, como mostrado na Figura 3.21. Como a memória cache está cheia, para que o primeiro dos três novos clientes possa entrar, mais memória terá que ser liberada. Para isso, além dos dois slots que estão ocupados, mas não com prioridade de remoção, mais uma seqüência de LS terá que ser removida, deste modo, o primeiro cliente poderá

entrar no sistema.

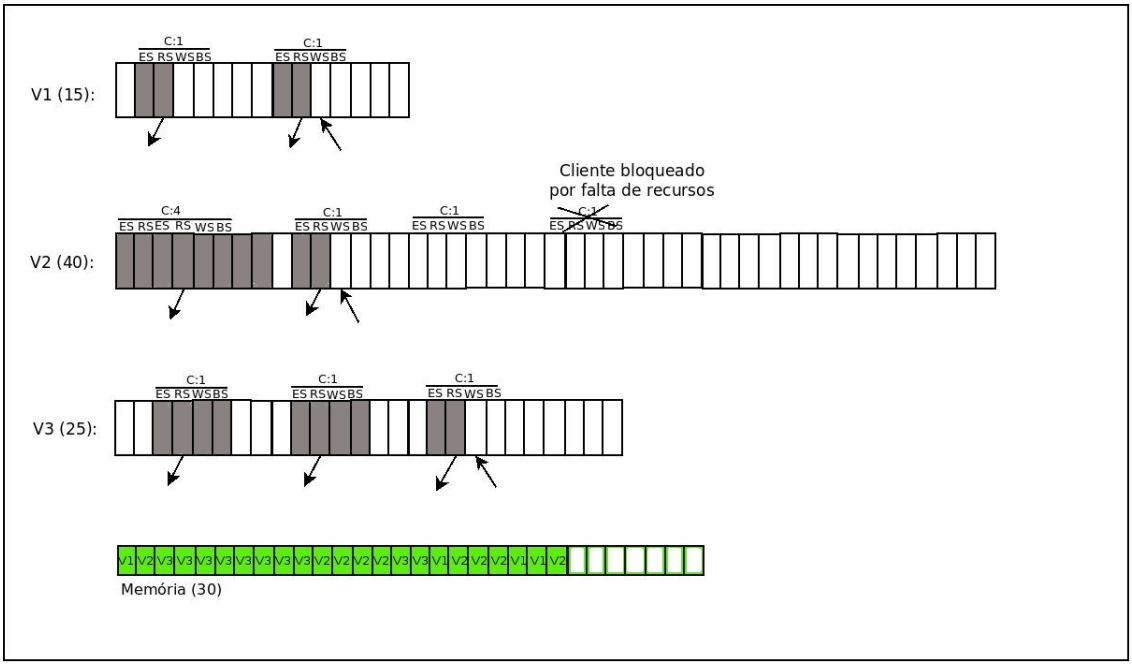

Figura 3.21: Cliente é bloqueado por falta de recursos.

Novamente, para que o segundo cliente possa entrar, mais *slots* ocupados terão que ser removidos e mais uma seqüência de LS também terá que ser removida, deste modo os recursos para este novo *buffer* estão garantidos. Contudo, quando o último dos três clientes requisita sua entrada no sistema, não há mais memória suficiente para ser liberada e garantir os recursos que este novo cliente irá precisar. Deste modo, o sistema irá bloquear este cliente. Para que este cliente possa receber vídeo, ele terá que pedir para outro distribuidor que possua recursos disponíveis, ou esperar até que o distribuidor volte a ter recursos disponíveis.

3.4 Memória Cooperativa Colapsada Local sem Otimização (MCCL-)

A MCC compartilha o espaço da cache com todos os vídeos, permitindo que os que necessitem de mais cache possam alocar mais memória. Contudo, a implementação do GloVE-WAN não compartilha a cache entre os vídeos, isto é, realiza a reserva de memória por vídeo, conforme detalhado a seguir.

Outra característica do GloVE-WAN é seu sistema de admissão não considerar os *slots* LS na cache na contabilidade de recursos. Quando um cliente realiza um pedido de vídeo, o distribuidor GloVE-WAN considera os *slots* LS como *slots* ocupados, embora os mesmos pudessem ser removidos para a inclusão de um novo cliente. Essa política faz com que o desempenho do sistema seja menor como será visto no

Capítulo 5. Nessa dissertação essa implementação foi denominada como Memória Cooperativa Colapsada Local sem otimização, ou MCCL-.

Na MCCL- uma reserva da cache é feita para cada vídeo. Por exemplo, têm-se dois vídeos, V1 com tamanho de 50 *slots* e V2 com tamanho de 30 *slots*. Se o distribuidor tiver capacidade apenas para realizar o cache (armazenamento em memória) de 10% do tamanho total desses vídeos, existiria 8 slots (50 * 10% = 5, 30 * 10% = 3, 5 + 3 = 8) para armazenar vídeos. Na MCC esses 8 slots seriam compartilhados entre os vídeos, ou seja, se V2 precisasse de 5 slots para armazenar vídeo, V2 poderia requisitar os 5 *slots*, contanto que eles estivessem livres. Mas na MCCL-, V2 poderia apenas armazenar (no máximo) 3 *slots*, que equivale a 10% do seu tamanho. Mesmo que V1 não estivesse utilizando os seus 5 *slots*, V2 não poderia utilizar esse espaço livre na cache.

O seguinte exemplo ilustra como o sistema de admissão funciona na MCCL-. Na Figura 3.22 pode-se observar um vetor de *slots* do vídeo V1 com 40 *slots* de tamanho podendo armazenar até 12 *slots* na cache, o que equivale a 30% do vetor. Na Figura 3.22-a o primeiro cliente entra no sistema, utilizando apenas 4 *slots*, na Figura 3.22-b o segundo cliente do sistema; observa-se que entre o *buffer* do primeiro cliente e do segundo, há 4 *slots* que serão marcados como LS. No total 12 *slots* estão sendo usados para armazenar vídeo, sendo que 4 *slots* são LS. Na Figura 3.22-c um terceiro cliente pede o mesmo vídeo, na MCCL-, quando o sistema de admissão verifica se há recursos disponíveis, observa que os 12 *slots* estão ocupados e que não há recursos suficiente para o cliente e bloqueia-o. Na MCC o sistema de admissão verificaria que os 12 *slots* estão ocupados, mas que entre eles há 4 LS, deste modo, apenas 8 *slots* realmente estão ocupados, já que os LS podem ser removidos, e libera para que o novo cliente possa entrar no sistema.

Para isolar o efeito do sistema de admissão sobre o desempenho do sistema, uma nova implementação foi concebida, a Memória Cooperativa Colapsada com otimização, ou MCCL+.

3.5 Memória Cooperativa Colapsada Local com Otimização (MCCL+)

A MCCL+ foi concebida para otimizar o sistema de admissão da MCCL-, desta forma é possível isolar os efeitos da otimização do sistema de admissão entre a MCCL- e a MCCL+, e os efeitos do compartilhamento de recursos entre a MCCL+ e a MCC. A MCCL+ continua a não compartilhar a cache entre os vídeos, como a MCCL-. A MCCL+ equivale-se a MCC quando ambas possuem apenas um vídeo e essa comparação é realizada no Capítulo 5.

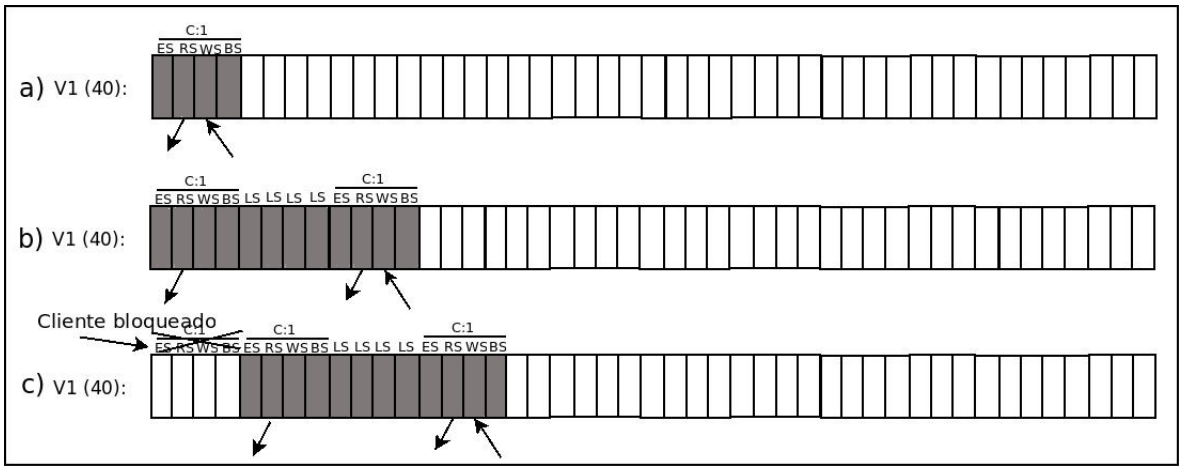

Figura 3.22: Estágio inicial da cache.

Na MCCL+, assim como na MCCL-, uma reserva da cache é feita para cada vídeo. Por exemplo, tem-se dois vídeos V1 com tamanho de 50 *slots* e V2 com tamanho de 30 *slots*. Se o distribuidor tiver capacidade apenas para realizar o cache de 10% do tamanho total desses vídeos, existirá 8 slots (50 * 10% = 5, 30 * 10% = 3, 5 + 3 = 8) para armazenar vídeo. Na MCC esses 8 slots seriam compartilhados como mencionado na Subseção 3.4.

Contundo, o sistema de admissão da MCCL+ é igual ao da MCC, no exemplo da Seção MCCL- na Figura 3.22-c, onde o terceiro cliente é bloqueado por falta de recursos. Na MCCL+, assim como na MCC, este cliente seria aceito pelo sistema, pois ao verificar o número de slots ocupados, que são 12, o sistema perceberia que 4 slots são do tipo LS e podem ser removidos do sistema, não os considerando no cálculo. Então, no final, o sistema irá verificar que apenas 8 slots estão ocupados e que ainda existem 4 slots que podem ser ocupados, permitindo assim que o novo cliente entrasse no sistema.

Os experimentos realizados no Capítulo 4 irão mostrar como o sistema de admissão e o compartilhamento de memória entre os vídeos influencia no desempenho do sistema GloVE-Mix, que foi implementado com as três gerências de cache; MCCL-, MCCL+ e MCC.

Capítulo 4

Análise Experimental

Este capítulo tem como objetivo apresentar a metodologia experimental e analisar os resultados obtidos nos experimentos realizados com o GloVE-Mix, e comparar: (i) os resultados da MCC com as simulações realizadas por Ishikawa[1]; (ii) desempenho entre MCC e MCCL-; (iii) o desempenho entre MCC e MCCL+; (iv) e o desempenho MCCL+ e MCCL-.

Este capítulo está organizado da seguinte forma: na Seção 4.1 são apresentadas as métricas de desempenho utilizadas, na Seção 4.2 é descrito o ambiente onde os experimentos foram realizados e a metodologia utilizada em cada experimento, na Seção 4.3 será realizada a validação das simulações realizadas por Ishikawa com os experimentos realizados com o GloVE-Mix, e finalmente na Seção 4.4 serão avaliados os desempenhos entre as três implementações de gerência da cache implementados no GloVE-Mix (MCCL-, MCCL+ e MCC).

4.1 Métricas

Para a análise de desempenho dos sistemas VsD com as três políticas de gerenciamento de cache - MCCL-, MCCL+ e MCC - foram escolhidas três métricas[1, 22]:

Latência: A latência é o tempo decorrido desde o momento que um cliente requisita um vídeo, até o momento que este mesmo cliente começa a assistir ao vídeo. A latência do sistema não pode ser grande ao ponto do cliente desistir de assistir ao vídeo, mas deve possuir um tamanho suficiente para armazenar o *buffer* inicial do cliente que irá esconder a variação dos atrasos dos pacotes de mídia. Para isso a latência deve estar dentro de um fator de paciência[33, 34] do cliente. A latência divide-se em três partes:

1. Latência de rede: tempo em que a mensagem de pedido leva para

percorrer a rede, desde o cliente até o servidor mais o tempo do primeiro pacote de vídeo chegar até o cliente (*Round-Trip Time - RTT*);

2. Latência de serviço: tempo que o distribuidor leva para processar o pedido do cliente e iniciar o processo de envio do vídeo;
3. Latência de *buffer*: tempo decorrente da chegada do primeiro pacote de vídeo, até que o *buffer* do cliente atinja o nível mínimo para o início da exibição. Os clientes utilizados para a realização dos pedidos de vídeo ao GloVE-Mix não possuem um *buffer* real, deste modo foi assumido um valor correspondente a 10 segundos de vídeo como tempo mínimo para inicio da exibição[1].

Deste modo,

$$Latência = Latência\,de\,rede + Latência\,da\,aplicacão + Latência\,de\,buffer$$

Taxa de bloqueio: É a relação entre o número de clientes que não tiveram o pedido atendido e o número total de clientes que pediram vídeo.

$$Taxa\,de\,Bloqueio = \frac{número\,de\,clientes\,não\,atendidos}{número\,total\,de\,clientes\,que\,pediram} x100$$

Quanto menor esta fração, melhor a eficiência na distribuição de vídeos.

Número de fluxos: É o número de fluxos de vídeos oriundos do servidor de vídeo para o distribuidor computado a cada intervalo de tempo equivalente ao tamanho de um *slot* (ver Subseção 4.2). Como um dos objetivos dos modelos de distribuição de conteúdo nos sistemas VsD é desonerar o servidor de vídeo, quanto menor o número de fluxos oriundos do servidor mais eficiente o sistema de distribuição será.

4.2 Metodologia

Todos os experimentos foram realizados dentro de um dos *clusters* do Laboratório de Computação Paralela (LCP)[35], Figura 4.1, com a seguinte configuração:

- 5 Intel Pentium D 3.0 GHz e 2GB de RAM, onde foram emulados os clientes;
- 1 Xeon Quad-Core 2.0 GHz e 16GB de RAM, como o servidor de vídeo e o GloVE-Mix;
- 1 Intel Pentium 4 2.4 GHz e 1GB de RAM para gerenciar os experimentos.;

- 1 3Com Baseline Switch 2924-SFP Plus Gigabit para interconectar as máquinas.

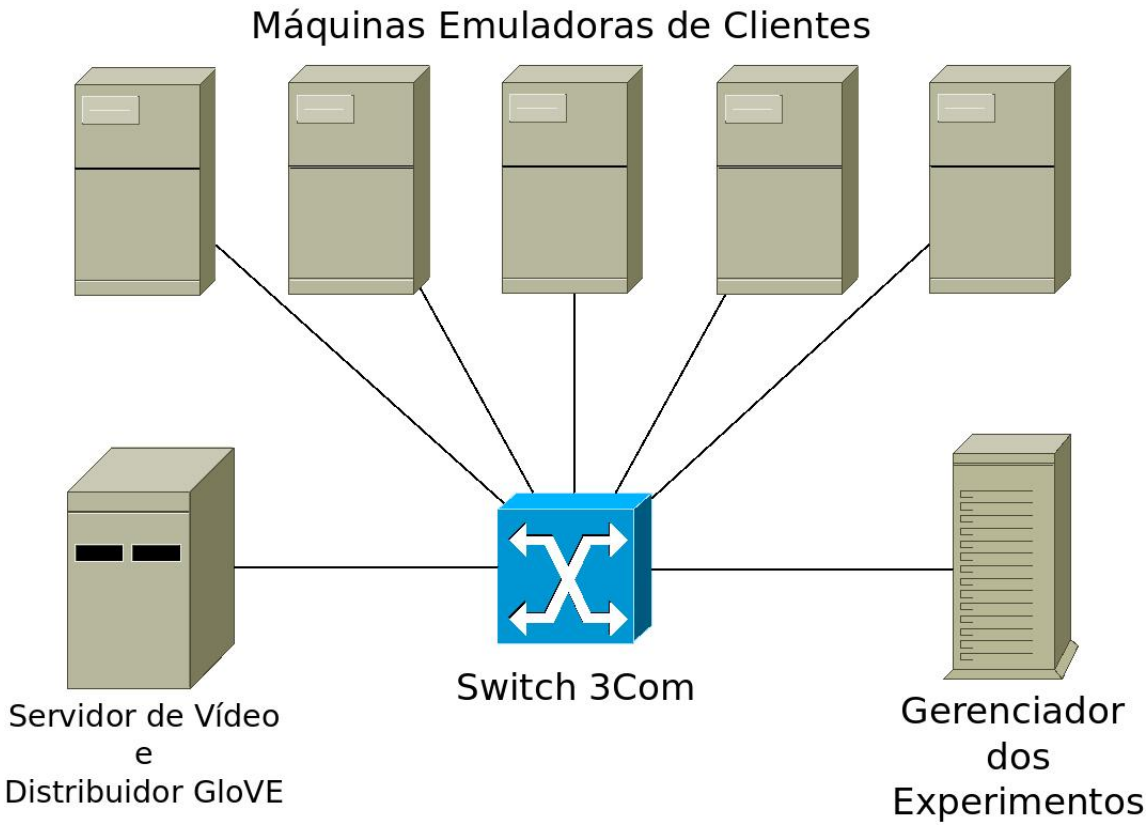

Figura 4.1: Ambiente LCP onde os experimentos foram realizados.

O servidor de vídeo e o distribuidor foram colocados na mesma máquina para evitar que o tráfego entre os dois prejudicasse a comunicação entre o distribuidor e os clientes. A rede possui banda nominal média de 950Mbps e *jitter* de 0,039 ms, medidos através do *Iperf* [36]. Em todos os experimentos foram executados com no máximo 900 clientes com vídeo de 1 Mbps de taxa de transmissão.

Para a realização dos experimentos foi implementado um cliente falso que realiza o pedido de vídeo para o distribuidor de vídeo e recebe o vídeo. Este cliente foi implementado de forma simples com a intenção de emular o funcionamento de um cliente real. O cliente apenas realiza um pedido de vídeo ao distribuidor GloVE, mas através do fluxo de vídeo que chega para o cliente pode-se verificar em qual taxa o vídeo está sendo transmitido e qual a latência para o início da exibição do vídeo.

O cliente irá realizar o pedido de um vídeo ao distribuidor que irá aceitar o cliente, se possuir recursos, e começará a enviar o vídeo até o final do experimento. Caso o distribuidor não possua recursos para aceitar o cliente, o mesmo será bloqueado e nenhum vídeo será enviado para o cliente.

Os pedidos dos clientes são realizados por um *script*, na máquina de gerência, seguindo uma distribuição de Poisson[37] que produz intervalos de tempo entre chegadas semelhantes aos observados em servidores de vídeos reais, e a escolha do vídeo que será pedido segue uma distribuição Zipf[37] que segue um padrão de escolha induzido pela popularidade, onde, aproximadamente, 80% dos clientes irão escolher 20% dos vídeos em média. Todos os pedidos requisitam o início do vídeo escolhido. Os parâmetros usados nos experimentos são descritos a seguir:

Intervalo de tempo entre chegadas: indica o intervalo médio de tempo entre as chegadas dos clientes ao sistema;

Cache: indica o tamanho da cache no experimento. O tamanho é indicado com porcentagem do tamanho do acervo contido no servidor de vídeo;

Número de vídeos: indica a quantidade disponível de vídeos que podem ser escolhidos pelos clientes;

Tamanho do *Slot*: indica a quantidade de vídeo, em segundos, que cada *slot* do vetor de *slot* está representando. Também indica o tempo em que cada giro dura;

Número Máximo de clientes: indica a quantidade máxima de clientes que podem requisitar vídeo durante o experimento;

Taxa dos vídeos: indica a taxa (em Mbps ou Kbps) na qual o vídeo é enviado para cada cliente;

Duração do vídeo: indica o tamanho do vídeo (em minutos) utilizado nos experimentos;

Duração do experimento: indica o tempo de duração (em minutos) de cada experimento;

Número de rodadas: indica o número de experimentos realizados com o mesmo conjunto dos demais parâmetros, com variação da semente para se obter o desvio padrão e a intervalo de confiança.

4.3 Avaliação Experimental x Simulação

O objetivo desta seção é validar as simulações realizadas por Ishikawa[1] com os resultados obtidos nos experimentos realizados para esta dissertação, objetivando demonstrar que a implementação da MCC no GloVE-Mix condiz com as simulações. Todas as simulações realizadas por Ishikawa possuem um intervalo de confiança de 95%.

4.3.1 Experimentos com 1 vídeo

Na Tabela 4.1 encontram-se os parâmetros utilizados tanto nas simulações [1], quanto nos experimentos com o GloVE-Mix.

Tabela 4.1: Parâmetros para a simulação e para os experimentos com 1 vídeo.

Parâmetro	Valor
Número de Vídeos	1
Cache	10; 15 e 20
Tempo entre chegadas(segundos)	3,1;7,2;10;15;30;45;60;90;120 e 150
Tamanho do *slot*(segundos)	8
Número Máximo de Clientes	900

Na Figura 4.2, retirada de Ishikawa[1], tem-se o resultado obtido na simulação para uma cache de 20%. Nesta figura existem três curvas indicadas como LS-, LS e LS+. Cada uma representa uma política de gerência dos LS adotadas nas simulações. LS- representa a política onde não existem LS's, ou seja, os *buffers* não podem ser ligados uns aos outros. A curva LS indica a política que onde se possui LS's, mas os mesmos, uma vez alocados, não podem ser retirados do vetor de *slots*, mantendo os *buffers* ligados. Já a curva LS+ indica a política com LS's, mas onde os mesmo podem ser removidos do vetor de *slots*, representando a política da MCC. Nesta dissertação, apenas a curva LS+ será relevante para a nossa validação.

Já a Figura 4.3 mostra o gráfico com os resultados obtidos nos experimentos com o GloVE-Mix para a MCC com cache de tamanhos de 10%, 15% e 20%, e apenas 1 vídeo.

Neste cenário, a MCC implementada no GloVE-Mix, com relação à taxa de bloqueio, obteve um resultado praticamente os mesmos apresentados nas simulações da MCC, mostrando que o GloVE-Mix conseguiu obter os resultados previstos pela simulações.

4.3.2 Experimentos com 100 vídeos

As Figuras 4.5 e 4.4 mostram os resultados obtidos nas simulações e experimentos para a taxa de bloqueio com tempo entre chegadas de 3,1 segundos. E as Figuras 4.6 e 4.7 mostram os resultados para a latência. A Tabela 4.2 mostra os valores dos parâmetros utilizados tanto nas simulações quanto nos experimentos.

Nas Figuras 4.5 e 4.4 observa-se que a taxa de bloqueio nas simulações e nos experimentos foram zero no momento de pico no acesso aos vídeos, tendo como referência a curva identificada como LS+. E nas Figuras 4.6 e 4.7 é possível observar que a latência do GloVE-Mix também esteve dentro do limite apresentado pelas simulações. Mais uma vez é possível mostrar que o GloVE-Mix correspondeu com

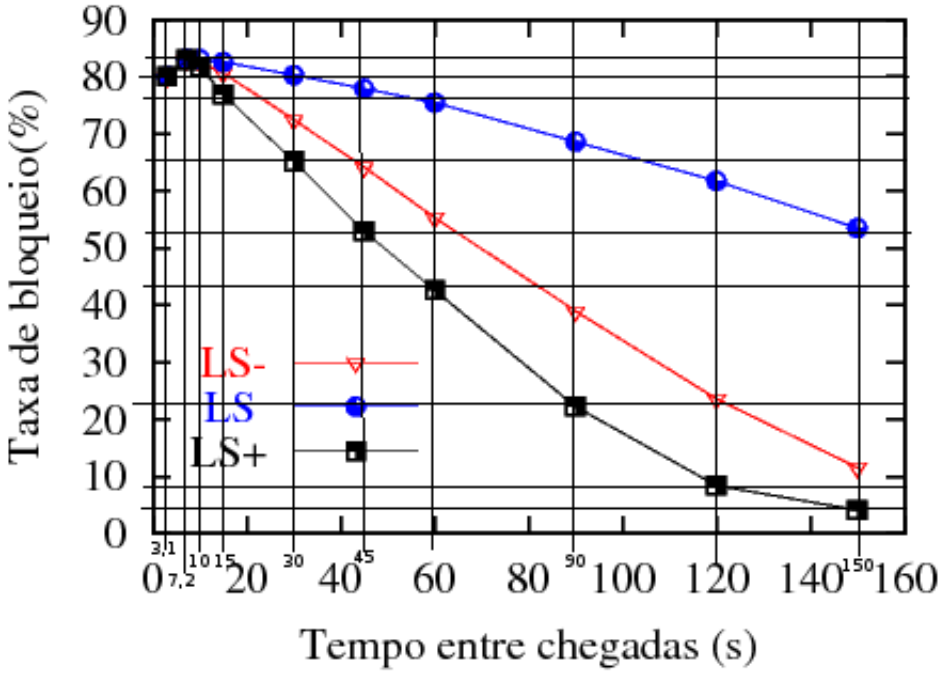

Figura 4.2: Simulação para 1 vídeo da MCC - Taxa de bloqueio x intervalo de tempo entre chegadas para cache de 20% (fonte: Ishikawa[1]).

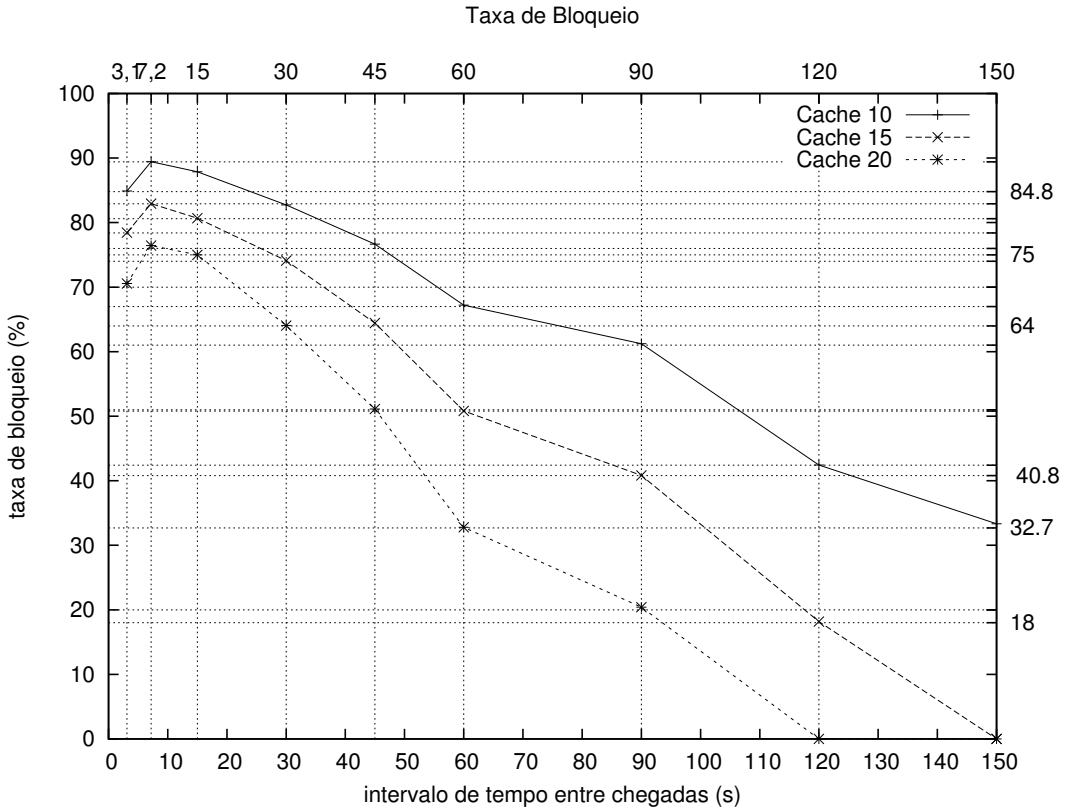

Figura 4.3: Experimento para 1 vídeo da MCC - Taxa de bloqueio x intervalo de tempo entre chegadas para cache de 10%, 15% e 20%.

Tabela 4.2: Parâmetros para a simulação e para os experimentos com 100 vídeo.

Parâmetro	Valor
Número de Vídeos	100
Cache (%)	10
Tempo entre chegadas (s)	3.1
Número Máximo de Clientes	900

os resultados simulados. Estes resultados possuem um desvio padrão de 1 segundo e um intervalo de confiança de 95%.

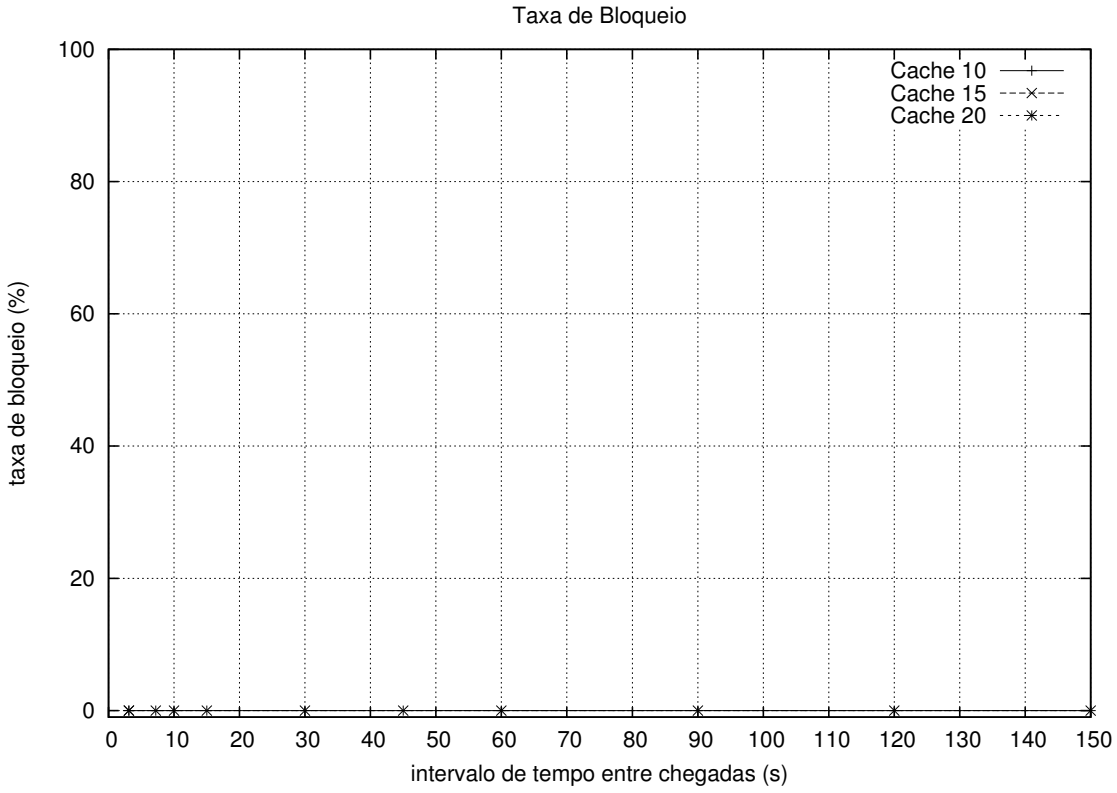

Figura 4.4: Experimento para 100 vídeo da MCC - Taxa de bloqueio x Cache para intervalo de tempo entre chegadas de 3,1 segundos.

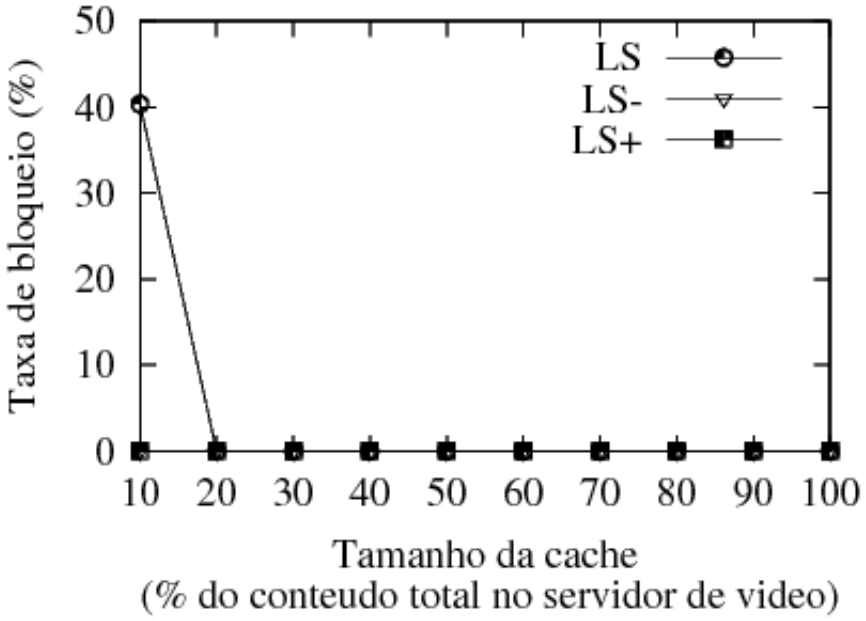

Figura 4.5: Simulação para 100 vídeo da MCC - Taxa de bloqueio x Cache para intervalo de tempo entre chegadas de 3,1 segundos (fonte: Ishikawa[1]).

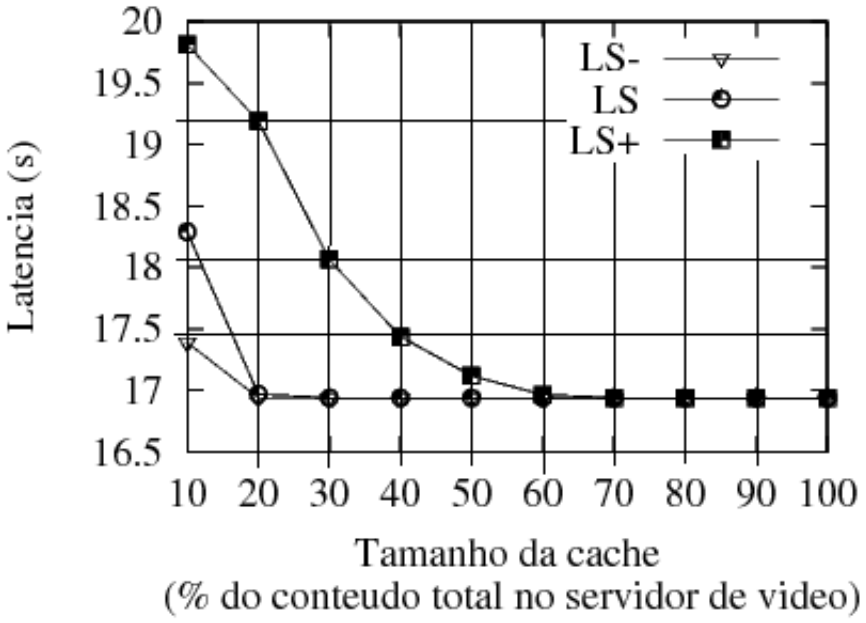

Figura 4.6: Simulação para 100 vídeo da MCC: Latência x Cache para intervalo médio de tempo entre chegadas de 3.1 segundos (fonte: Ishikawa[1]).

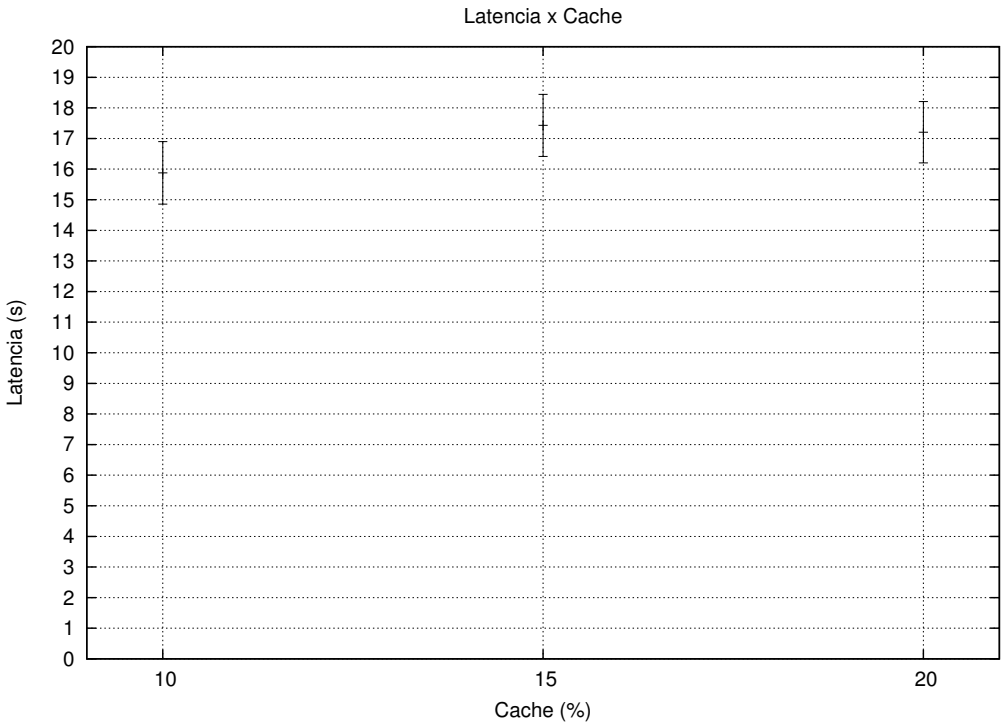

Figura 4.7: Experimento para 100 vídeo da MCC: Latência x Cache para intervalo médio de tempo entre chegadas de 3.1 segundos.

4.4 Avaliação do GloVE-Mix

Na Seção 4.3 mostrou-se que a implementação da MCC no GloVE-Mix correspondeu às expectativas levantadas pelas simulações realizadas por Ishikawa[1]. Nesta seção é mostrada a avaliação de desempenho entre os modelos da MCC, MCCL+ e MCCL-.

4.4.1 Experimentos com 1 vídeo

Os objetivos dos experimentos com apenas um único vídeo são:

Validação: Validar a equivalência entre as implementações quando colocadas no cenário onde possuem a mesma quantidade de recursos e o compartilhamento de recursos não irá influenciar nos resultados;

Otimização: Mostrar que havia mais uma característica da MCC que não tinha sido implementada na MCCL-, e que deu origem a MCCL+.

Os parâmetros de cada experimento são mostrados na Tabela 4.3.

Tabela 4.3: Parâmetros para os experimentos com 1 vídeo.

Parâmetro	Valor
Número de Vídeos	1
Cache(percentagem)	10; 15 e 20
Tempo entre chegadas(segundos)	3,1;7,2;10;15;30;45;60;90;120 e 150
Tamanho do *slot*(segundos)	8
Número Máximo de Clientes	900
Taxa do vídeo	1Mbps

MCC x MCCL-

A avaliação da MCC com a MCCL- no cenário com apenas um vídeo tem o objetivo de mostrar a equivalência das implementações para que, quando forem observados os resultados na Subseção 4.4.2 se possa analisar com clareza os resultados.

Contudo, quando se analisa o gráfico da Figura 4.8, pode-se observar que há uma diferença de resultados. Essa diferença deu-se porque, quando este experimento foi realizado, observou-se que a MCCL- também não estava levando em consideração os LS do vetor de *slots* no sistema de admissão dos clientes. Deste modo, fez-se necessário a implementação da MCCL+ no GloVE-Mix para que fosse possível avaliar a equivalência das implementações quando no cenário com apenas 1 vídeo.

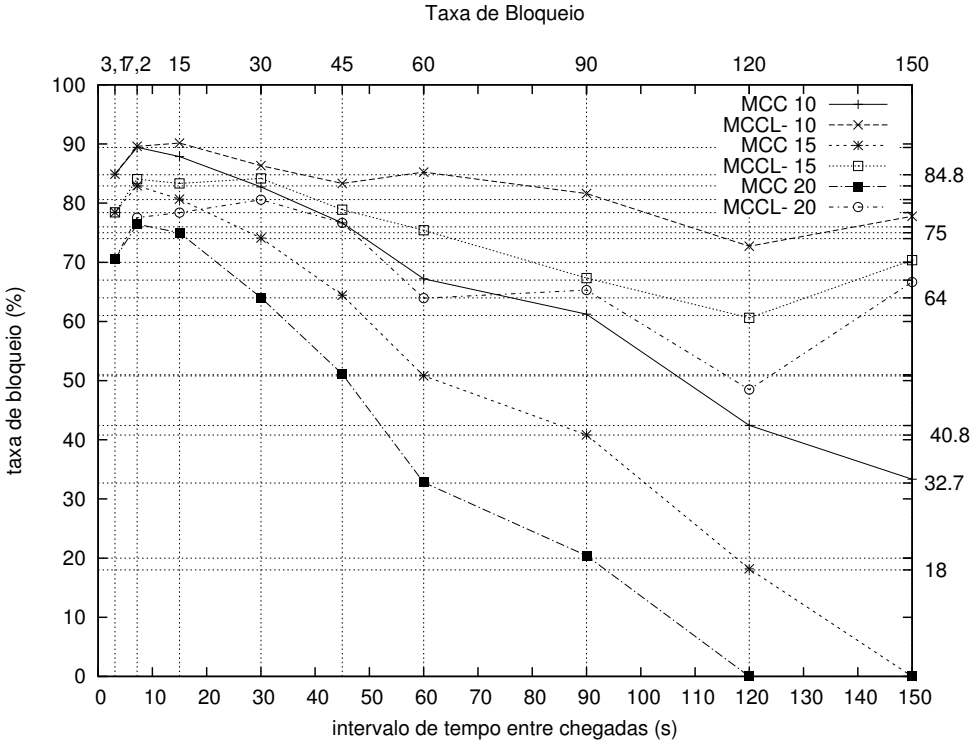

Figura 4.8: Experimento para 1 vídeo: Comparação da Taxa de Bloqueio x Intervalo de tempo entre Chegadas entre MCC e MCCL-.

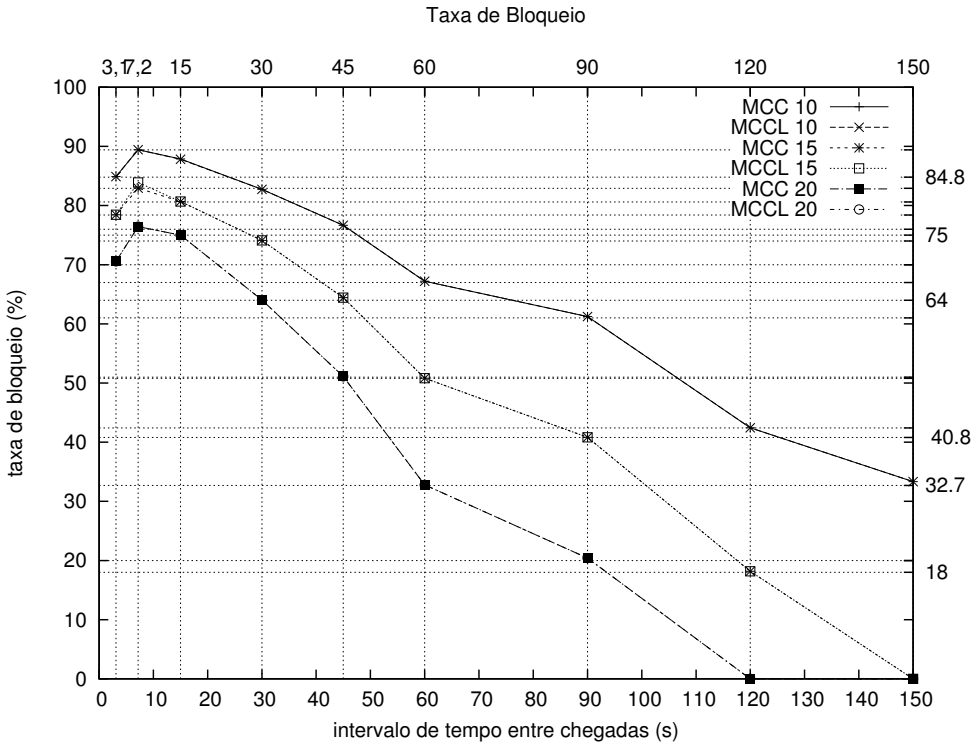

Figura 4.9: Experimento para 1 vídeo: Comparação da Taxa de Bloqueio x Intervalo de tempo entre Chegadas entre MCC e MCCL+.

MCC x MCCL+

Com a implementação da MCCL+ no GloVE-Mix, foi possível avaliação a equivalência do comportamento da MCC e MCCL+ quando no cenário com apenas 1 vídeo, como pode ser observado na Figura 4.9.

Discussão

Através desses resultados pôde-se considera a equivalência entre a MCC e a MCCL+ quando no cenário com apenas 1 vídeo, e o aumento do desempenho que a otimização incluída na MCCL+ agrega ao sistema, quando comparada com a MCCL-. Como a MCCL- não leva em consideração os LS do vetor de *slots* na admissão de clientes, ela se equivale aos modelos que utilizam *ring buffers* e permitem que esses *ring buffers* se colapsem, mas não que se separem, já que os mesmos não possuem os LS. Deste modo, a MCCL- deixa de implementar uma das principais características que Ishikawa[1] propôs para a MCC.

4.4.2 Experimentos com 100 vídeos

O objetivo desses experimentos é comparar o desempenho entre os três modelos: MCCL-, MCCL+ e MCC. Os parâmetros de cada experimento foram os seguintes:

Tabela 4.4: Parâmetros para os experimentos com 100 vídeos.

Parâmetro	Valor
Duração	1 hora
#Vídeos	100
Cache(percentagem)	10; 15 e 20
Tempo entre chegadas(segundos)	3,1;7,2;10;15;30;45;60;90;120 e 150
Tamanho do *slot*(segundos)	8
Número Máximo de Clientes	900
Taxa do vídeo	1Mbps
Zipf	0,271

MCC x MCCL-

Ao analisar os resultados encontrados nos gráficos das Figuras 4.10 e 4.11 se pode verificar que a MCC consegue um desempenho melhor que a MCCL- quanto à taxa de bloqueio dos clientes. Nenhum cliente foi bloqueado quando o GloVE-Mix se utilizou do modelo da MCC, enquanto houve um aumento significativo da taxa de bloqueio para MCCL- para os tempos entre chegadas menores que 15 segundos.

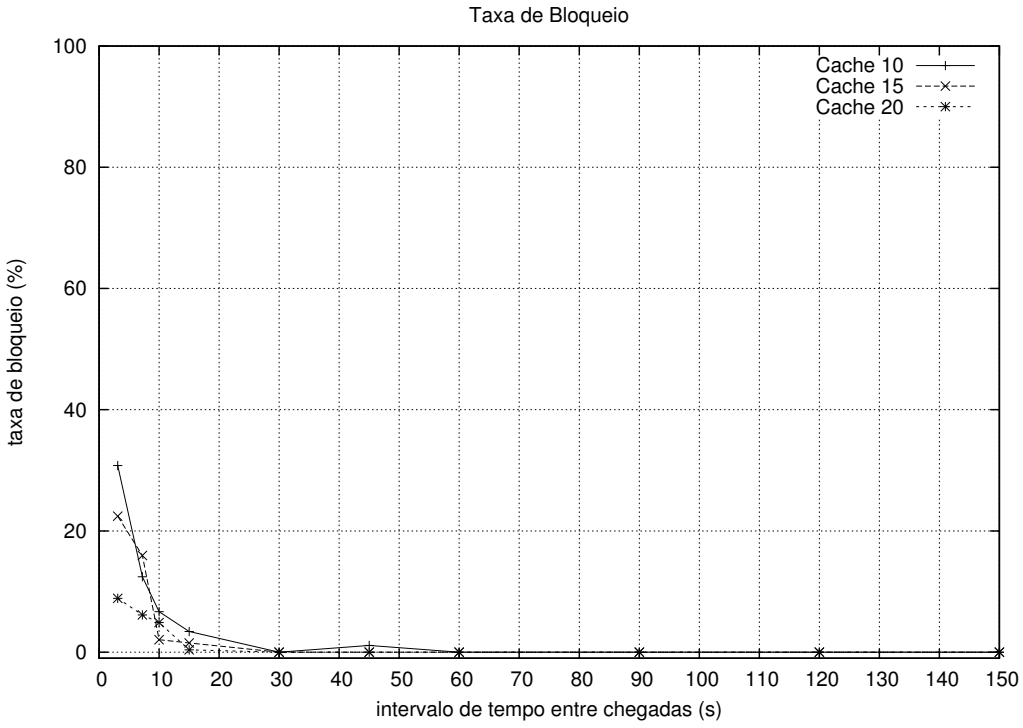

Figura 4.10: Experimento para 100 vídeos da MCCL- - Taxa de Bloqueio x Intervalo de tempo entre chegadas.

A MCC também levou vantagem sobre a MCCL- quanto ao número de fluxos para cache igual a 10%. No gráfico da Figura 4.12 tem-se o número de fluxos obtido a cada tempo de *slot* do GloVE-Mix ao longo de todo o experimento para

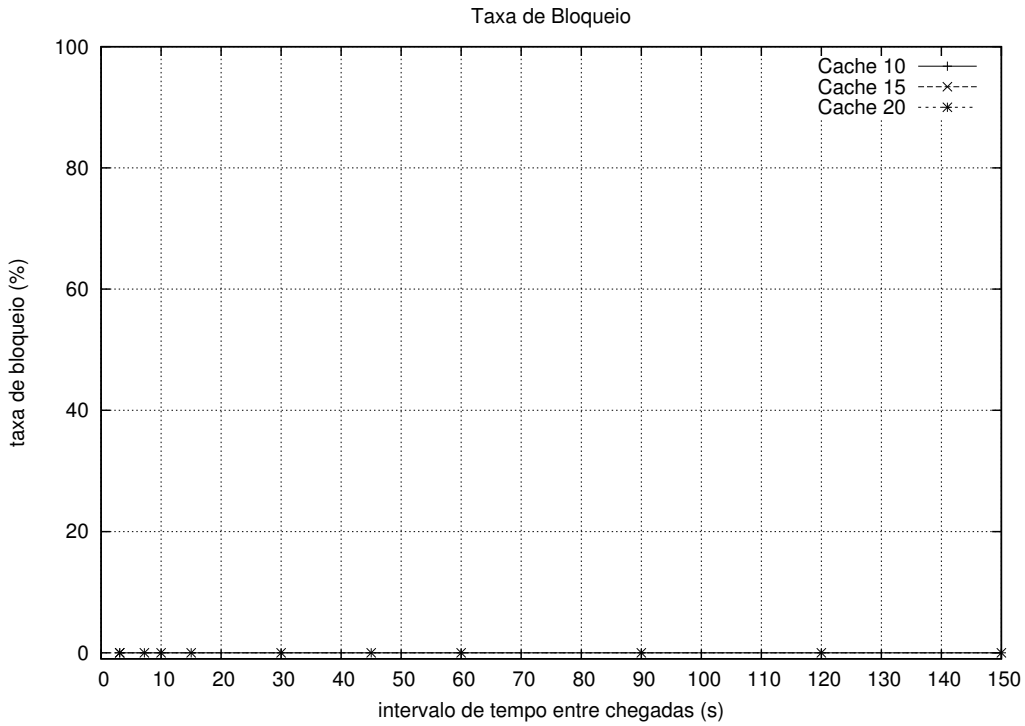

Figura 4.11: Experimento para 100 vídeos da MCC - Taxa de Bloqueio x Intervalo de tempo entre chegadas.

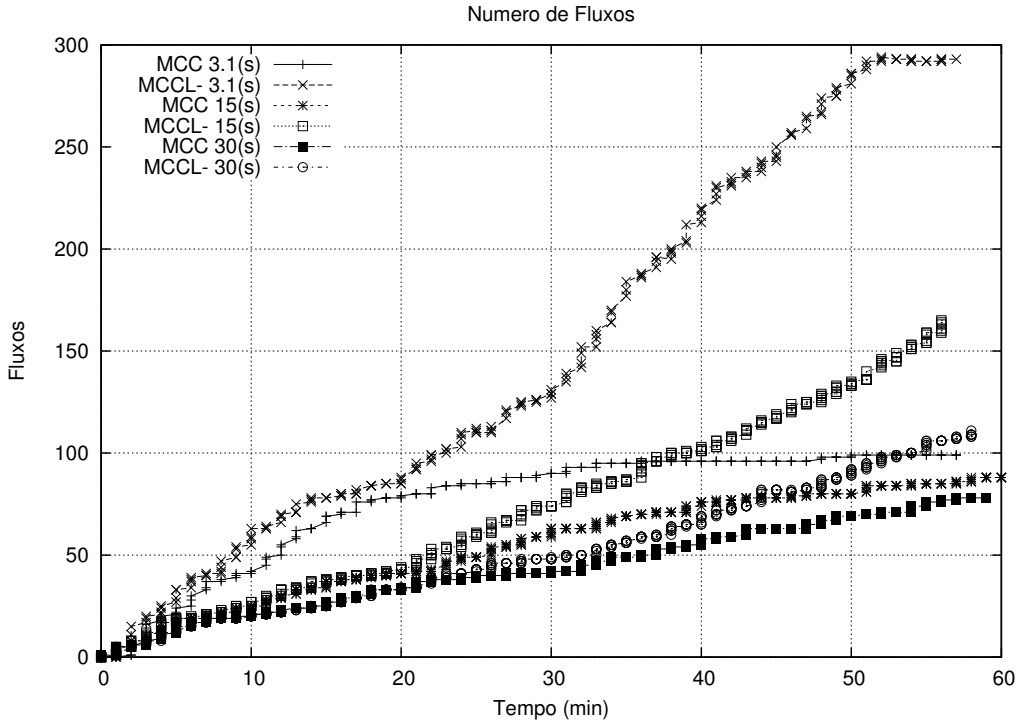

Figura 4.12: Experimento para 100 vídeos da MCC e MCCL- - Número de Fluxo x Tempo do experimento para Intervalo de tempo entre Chegadas de 3,1, 15 e 30 segundos e cache de 10%.

os intervalos de tempo entre chegadas de 3,1, 10 e 30 segundos. Como se pode observar, em todos a MCC possui um número de fluxos menor, principalmente para o tempo entre chegadas de 3,1 segundos que é quando o sistema se encontra com a maior carga. Isto porque, como a MCC compartilha recursos da cache, não existe a necessidade de se desalocar os LS entre os *buffers* dos vídeos mais requisitados. Deste modo, nenhum novo fluxo oriundo do servidor de vídeo é necessário.

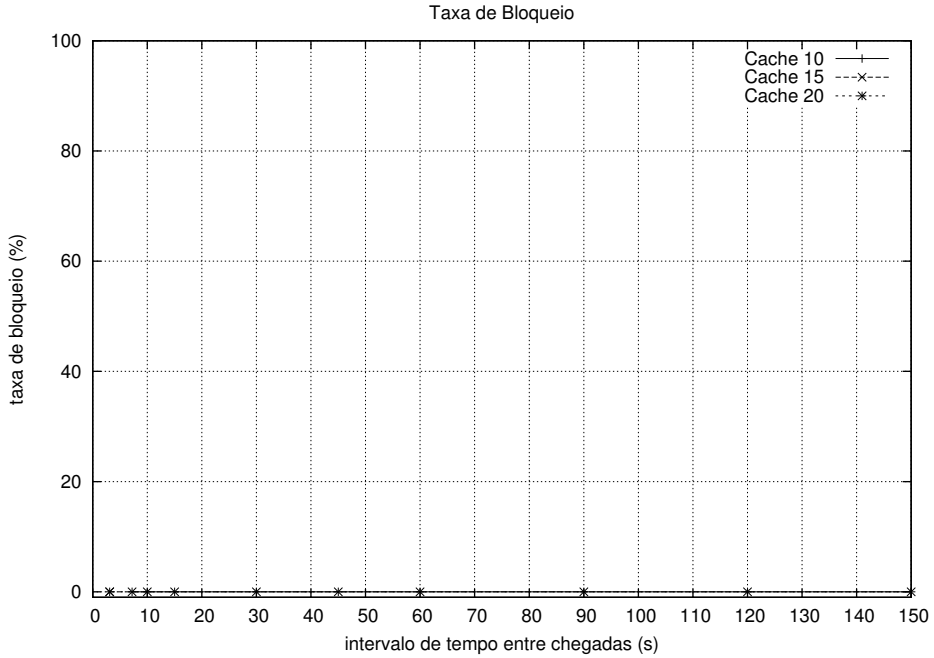

Figura 4.13: Experimento para 100 vídeos da MCCL+ - Taxa de Bloqueio x Intervalo de tempo entre chegadas.

MCC x MCCL+

Ao se analisar os resultados do gráfico da Figura 4.13, pode-se observar que a taxa de bloqueio dos clientes também foi zero para a MCCL+, assim como os da MCC apresentada na Figura 4.11.

Entretanto, a MCC ganha da MCCL+ quando se observa o número de fluxos de ambas, mostrado no gráfico da Figura 4.14.

MCCL+ x MCCL-

Com já apresentado nas Figuras 4.10 e 4.13; a taxa de bloqueio dos clientes da MCCL+ é inferior à da MCCL-. Porém, o número de fluxos apresentado no gráfico da Figura 4.15 mostra que a MCCL- consegue ganhar da MCCL+. Isso acontece porque a MCCL+ remove LS para que mais clientes possam ser admitidos no sistema, o que reflete na sua taxa de bloqueio inferior quando comparada com a MCCL-. Contudo, ao se remover o LS, uma lacuna no vetor de *slot* é deixada, e o

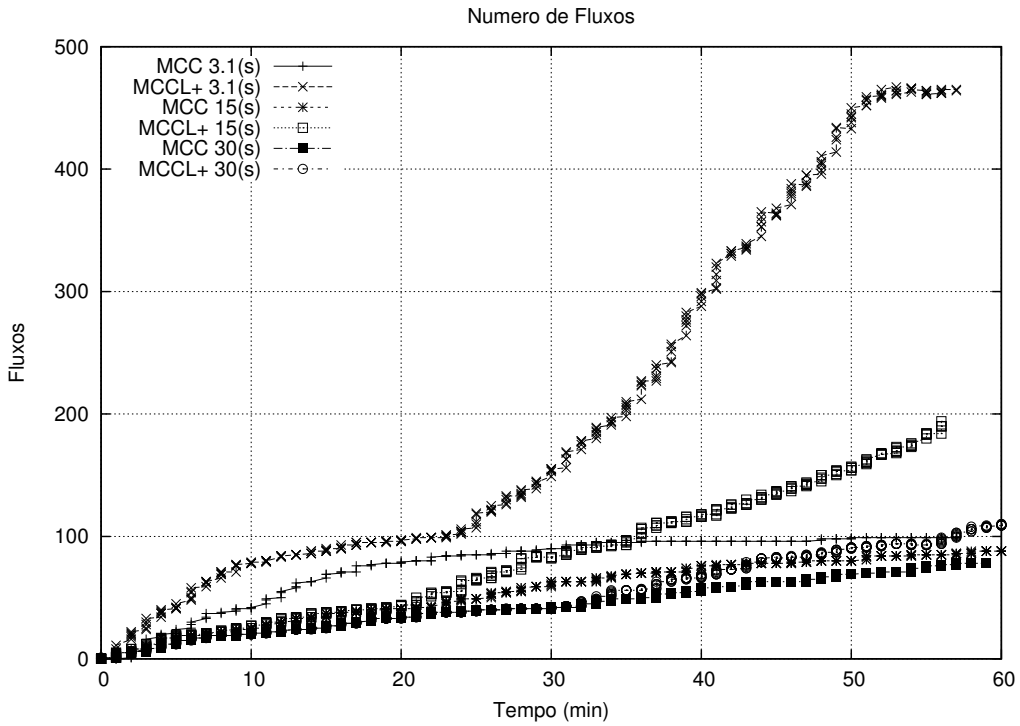

Figura 4.14: Experimento para 100 vídeos da MCC e MCCL+ - Número de Fluxos x Tempo do experimento para Intervalos de tempo entre Chegadas de 3,1, 15 e 30 segundos e cache de 10%.

buffer do cliente que vier em seguida a esta lacuna precisará pedir um fluxo de vídeo para o servidor, pois esta parte do vídeo não estará mais na cache do GloVE-Mix.

Discussão

Como foi possível observar, as taxas de bloqueio da MCCL+ e da MCC foram zero para todos os cenários, possuindo um resultado melhor do que a MCCL-, principalmente nos momentos de pico de acessos aos vídeos. Contudo, a MCC obteve um desempenho melhor do que a MCCL+, pois nos momentos de pico de acessos aos vídeos, o número de fluxos da MCC foi menor do que na MCCL+, como se pôde ver na Figura 4.14. A MCC também ganha da MCCL-, Figura 4.12, apesar da MCCL- ganhar da MCCL+, quando se considera apenas a taxa de bloqueio (Figura 4.15). Na Figura 4.16 tem-se o número de fluxos de vídeo para os três modelos no momento de pico no acesso aos vídeos (intervalo de tempo entre chegadas de 3.1 segundos).

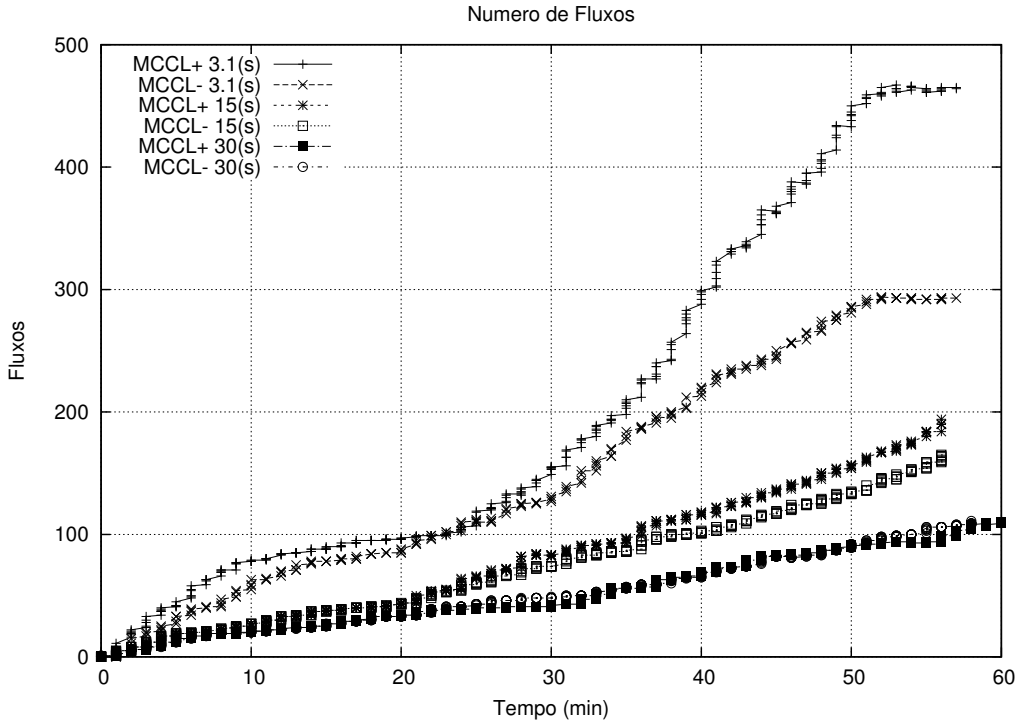

Figura 4.15: Experimento para 100 vídeos da MCC e MCCL+ - Número de Fluxos x Tempo do experimento para Intervalos de tempo entre Chegadas de 3,1, 10 e 30 segundos e cache de 10%.

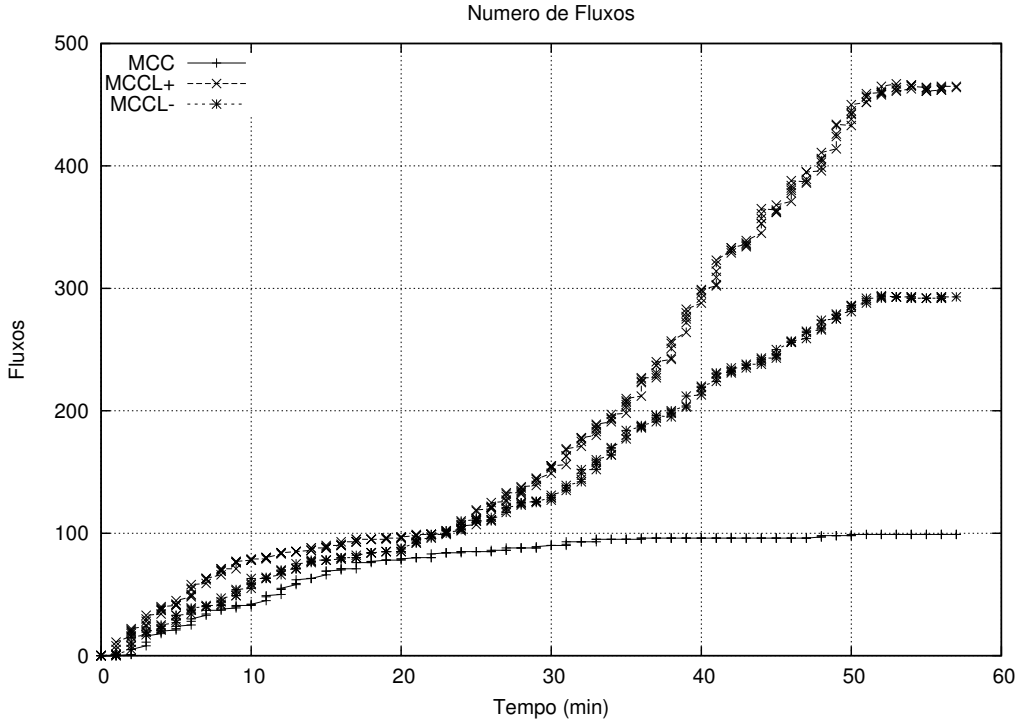

Figura 4.16: Experimento para 100 vídeos da MCC, MCCL+ e MCCL- - Número de fluxos para cache de 10% e intervalo de tempo entre chegadas de 3,1 segundos.

Capítulo 5

Conclusões e Trabalhos Futuros

Ao longo desta dissertação foram contextualizadas os desafios que os sistemas de distribuição de conteúdo multimídia na Internet possuem, como o congestionamento da rede, os atrasos dos pacotes de vídeo e a perda dos mesmos. Também foram apresentados alguns modelos de distribuição que têm como objetivo superar os desafios, como as técnicas que utilizam o modelo *peer-to-peer* para distribuição de conteúdo multimídia e as técnicas que utilizam os servidores *proxy* (substitutos).

Em particular no Capítulo 3 foi apresentada a Memória Cooperativa Colapsada (MCC) que é um modelo de gerência da memória do sistemas de distribuição de conteúdo multimídia que utiliza servidores substitutos. Neste modelo foi introduzido o conceito do vetor duplo de anéis, onde um é o vetor de *slots* (metadados) e o segundo é o vetor de vídeos. Também foi introduzido o conceito dos *slots* de ligação (*Link Slot* - LS) que são responsáveis pelo ganho de eficiência do modelo permitindo que os buffers de clientes no mesmo vetor de *slots* possam ser lidados para economizarem fluxos de vídeo oriundos do servidor de vídeo, e se separarem para liberar recursos (memória), quando necessários, para que novos clientes possam entrar no sistema.

No Capítulo 4 foram discutidos os resultados experimentais obtidos com o protótipo GloVE-Mix, que foi extendido nesta dissertação para implementar os modelos da MCC, MCCL+ e MCCL-. A partir da metodologia adotada foi possível observar o desempenho de cada modelo. Os resultados dos experimentos com o GloVE-Mix confirmaram os resultados obtidos nas simulações realizadas por Ishikawa validando-as.

5.1 Trabalhos Futuros

Após as análises apresentadas nesta dissertação, algumas questões ainda ficam em aberto e podem ser exploradas em trabalhos futuros. Entre elas, temos o uso da memória secundária como cache e a implementação da MCC em um servidor de

vídeo largamente utilizado na Internet como LightTPD[38].

5.1.1 Uso de Memória Secundária como Cache

Nem nas simulações de Ishikawa[1], e nem nos experimentos realizados nesta dissertação com o GloVE-Mix, a memória secundária do *proxy* foi utilizada para armazenar vídeos. A questão seria como a memória secundária poderia ser usada. Pode-se levantar dois possíveis modelos para a sua utilização. Ambos os modelos poderiam ser implementados no GloVE-Mix para avaliação de desempenho:

1. Como continuação da memória principal. Neste modelo a memória secundária poderia ser utilizada como uma continuidade da memória principal, sendo gerenciada da mesmo forma, o vetor de *slots* não faria distinção de qual memória ele estaria usando, apenas o vetor de vídeo teria parte dele sendo alocado na memória principal e parte dele sendo alocado na memória secundária. Claro que, como a memória secundária possui uma largura de banda menor que a memória principal, uma política que priorizasse a popularidade do vídeo também poderia ser usada para beneficiar os vídeos mais populares e estes teriam seu vetor de vídeo alocado apenas na memória principal;

2. Hierarquizar as memórias. Neste modelo apenas os vídeos mais populares seriam armazenados na memória principal, os restantes seriam armazenados na memória secundária. Como poucos clientes estariam requisitando os vídeos pouco populares, a largura de banda da memória secundária seria suficiente para atender com QoS estes clientes, enquanto os clientes dos vídeos mais populares usufruiriam da memória principal, já que os mesmos demandariam mais largura de banda.

5.1.2 Memória Cooperativa Colapsada em Servidores de Vídeo Tradicionais

Ainda não há trabalhos que demonstrem o real ganho da MCC em servidores de vídeos que utilizados na Internet, como LightTPD[38] utilizado pelo YouTube[7].

Ao desenvolver um módulo do LightTPD, por exemplo, que implemente uma cache que seja gerenciada pela MCC, seria possível medir o desempenho que a MCC traria a um sistema real utilizado na Internet.

Portanto, poderiam ser criados dois cenários: (i) o primeiro onde o servidor de vídeo utilizasse seu próprio módulo de cache, e requisições de vídeos seriam realizadas até que a QoS do sistema começasse a se degradar; (ii) no segundo cenário seria

utilizado o módulo de cache com a MCC e novamente requisições de vídeo seriam realizadas até que o QoS do sistema começasse a se degradar. Deste modo, seria possível analisar o ganho que a MCC conseguiria agregar ao sistema de distribuição de conteúdo.

Referências Bibliográficas

[1] ISHIKAWA, E. *Memória Cooperativa para Distritribuição de Vídeo sob Demanda*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, out. 2003.

[2] PINHO, L. B. *Implementação e Avaliação de um Sistema de Vídeo sob Demanda Baseado em Cache de Vídeo Cooperativa*. Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, maio 2002. (in Portuguese).

[3] ALVES, J. M. O. *Implementação de uma Solução Integrada para Transmissão de Fluxos em Sistemas de Vídeo Sob Demanda*. Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, set. 2007. (in Portuguese).

[4] HOFMANN, M., NG, T. S. E., GUO, K., et al. *Caching Techniques for Streaming Multimedia over the Internet*. Relatório técnico, 1999.

[5] KATHERINE, E. B., GUO, K., HOFMANN, M., et al. "Design and Implementation of a Caching System for Streaming Media over the Internet". In: *in IEEE Real Time Technology and Applications Symposium*, pp. 111–121, 1999.

[6] VERSCHEURE, O., VENKATRAMANI, C., FROSSARD, P., et al. "Joint Server Scheduling and Proxy Caching for Video Delivery". In: *in Proc. 60 th International Workshop on Web Caching and Content Distribution*, pp. 413–423, 2001.

[7] "Youtube". , fev. 2005. Disponível em: ¡www.youtube.com¿. www.youtube.com - YouTube is the world's most popular online video community, allowing millions of people to discover, watch and share originally-created videos.

[8] "Porta Curtas Petrobras". . www.portacurtas.com.br/.

[9] BRAGATO, L. L. S. *Implementação e Avaliação de um Sistema de Vídeo sob Demanda Baseado em Cache Cooperativa Colapsada de Vídeo*. Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, set. 2006. (in Portuguese).

[10] "Google". , 1997. Disponível em: ¡www.google.com¿. www.google.com.

[11] "Skype". , 2003. Disponível em: ¡www.skype.com¿. www.skype.com - Skype is software that enables the world's conversations.

[12] KUROSE, J. F., ROSS, K. W. *Redes de Computadores e a Internet: Uma Nova Abordagem.* Pearson, Addison Wesley, 2003.

[13] "RFC793 - Transmission Control Protocol". , . http://www.faqs.org/rfcs/rfc793.html.

[14] "RFC768 - User Datagram Protocol". , . http://www.faqs.org/rfcs/rfc768.html.

[15] "RFC791 - INTERNET PROTOCOL". , . http://www.ietf.org/rfc/rfc791.txt.

[16] "RFC2068 - Hypertext Transfer Protocol – HTTP/1.1". , . http://www.ietf.org/rfc/rfc2068.txt.

[17] "RFC1889 - RTP: A Transport Protocol for Real-Time Applications". , . http://www.ietf.org/rfc/rfc1889.txt.

[18] "Real Time Messaging Protocol Chunk Stream". , 1990. www.adobe.com/devnet/rtmp/pdf/rtmp_specification_1.0.pdf.

[19] FRAZIE, H., DOORN, S. V., HAYS, R., et al. "IEEE 802.3ad - Link Aggregation (LAG)". , 2007. www.ieee802.org/3/hssg/public/apr07/frazier_01_0407.pdf.

[20] "CISCO - Link Aggregation". . www.cisco.com/en/US/products/ps9967/products_qanda_item09186a0080a36439.shtml.

[21] PINHO, L. B., AMORIM, C. L. "Assessing the efficiency of stream reuse techniques in P2P video-on-demand systems", *Journal of Network and Computer Applications (JNCA)*, v. 29, n. 1, pp. 25–45, jan. 2006. ISSN: 1084-8045, Academic Press - Elsevier.

[22] ISHIKAWA, E., AMORIM, C. L. "Collapsed Cooperative Video Cache for Content Distribution Network". In: *Proceedings of the Brazilian Simposium on Computer Networks (SBRC)*, pp. 249–264, Natal, RN, Brazil, maio 2003.

[23] PATTERSON, D. A., HENNESSY, J. L. "Computer Organization and Design - The hardware / software interface". 4 ed., cap. 5, ELSEVIER, 2009.

[24] GOLUBCHIK, L., LUI, J. C. S., MUNTZ, R. R. "Adaptive Piggybacking: A Novel Technique for Data Sharing in Video-On-Demand Storage Servers", *ACM Multimedia Systems Journal*, v. 4, pp. 140–155, 1996.

[25] EAGER, D., VERNON, M., ZAHORJAN, J. "Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand". In: *Proc. IS&T/SPIE Conf. On Multimedia Computing and Networking 2000 (MMCN 2000)*, pp. 206–215, 2000.

[26] CHEN, S., SHEN, B., WEE, S., et al. "Adaptive and lazy segmentation based proxy caching for streaming media delivery". In: *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pp. 22–31, New York, NY, USA, 2003. ACM. ISBN: 1-58113-694-3. doi: http://doi.acm.org/10.1145/776322.776328.

[27] SZWARCFITER, J. L., MARKENZON, L. "Estrutura de Dados e seus Algoritmos". 2 ed., cap. 2, LTC, 1994.

[28] ISHIKAWA, E., AMORIM, C. L. "Cooperative Video Caching for Interactive and Scalable VoD Systems". In: Lorenz, P. (Ed.), *Networking - ICN 2001, First International Conference, Colmar, France, July 9-13, 2001 Proceedings, Part 2*, v. 2094, *Lecture Notes in Computer Science*, Springer, pp. 776–785, 2001. ISBN: 3-540-42303-6.

[29] ISHIKAWA, E., AMORIM, C. L. "Memória Cooperativa Distribuída para Sistemas de VoD peer-to-peer". In: *Proceedings of the Brazilian Simposium on Computer Networks (SBRC)*, pp. 822–837, Florianópolis, SC, Brazil, May 2001. (in Portuguese).

[30] PINHO, L. B., ISHIKAWA, E., AMORIM, C. L. "GloVE: A Distributed Environment for Low Cost Scalable VoD Systems". In: *Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 117–124, Vitória, ES, Brazil, October 2002. IEEE Computer Society Press.

[31] PINHO, L. B., ISHIKAWA, E., AMORIM, C. L. "GloVE : A Distributed Environment for Scalable Video-On-Demand Systems", *International Journal of High Performance Computing Applications (IJHPCA)*, v. 17, n. 2, May 2003. ISSN: 1094-3420, Sage Publications.

[32] PINHO, L. B., AMORIM, C. L. "A Practical Performance Analysis of Stream Reuse Techniques in Peer-to-Peer Video on Demand Systems".

In: Danelutto, M., Vanneschi, M., Laforenza, D. (Eds.), *Euro-Par 2004 Parallel Processing, 10th International Euro-Par Conference, Pisa, Italy, August 31-September 3, 2004, Proceedings*, v. 3149, *Lecture Notes in Computer Science*, Springer, pp. 784–791, August 2004. ISBN: 3-540-22924-8, ISSN: 0302-9743.

[33] PINHO, L. B. *Estratégias Escaláveis para Distribuição de Mídias Contínuas sob Demanda em Redes sem Fio*. Tese de Doutorado, COPPE/Sistemas, UFRJ, December 2007. (in Portuguese).

[34] ABRAM-PROFETA, E. L., SHIN, K. G. "Scheduling video programs in near video-on-demand systems". In: *MULTIMEDIA '97: Proceedings of the fifth ACM international conference on Multimedia*, pp. 359–369, New York, NY, USA, 1997. ACM. ISBN: 0-89791-991-2. doi: http://doi.acm.org/10.1145/266180.266387.

[35] "Laboratório de Computação Paralela (LCP) - COPPE/UFRJ". . www.lcp.coppe.ufrj.br.

[36] "Iperf". . iperf.sourceforge.net/.

[37] ROSS, S. M. *A course in simulation*. Macmillan Publishing Company, 1990.

[38] "LightTPD". , 2007. www.lighttpd.net.