Universidade Federal do Pampa

Autor: Ussamah Jumah Eid Ahmad Laila

PROPOSTA DE PROXY DE VÍDEO ESCALÁVEL COM HIERARQUIA DE MEMÓRIA

Trabalho de Conclusão de Curso II

BAGÉ
2012

USSAMAH JUMAH EID AHMAD LAILA

PROPOSTA DE PROXY DE VÍDEO ESCALÁVEL COM HIERARQUIA DE MEMÓRIA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Leonardo Bidese de Pinho

Bagé
2012

USSAMAH JUMAH EID AHMAD LAILA

PROPOSTA DE PROXY DE VÍDEO ESCALÁVEL COM HIERARQUIA DE MEMÓRIA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 23 de novembro de 2012
Banca examinadora:

_____

Prof. Dr. Leonardo Bidese de Pinho
Orientador
UNIPAMPA

_____

Profª. Drª. Ana Paula Lüdtke Ferreira
UNIPAMPA

_____

Prof. MSc. Bruno Silveira Neves
UNIPAMPA

Dedico este trabalho a minha família pela força e empenho dispensados a mim, não só neste, mas em todos os momentos de minha vida.

AGRADECIMENTO

Primeiramente à Deus, a quem sempre recorri em momentos de incerteza e dúvida e atendeu as minhas preces.

A minha família, que além de me apoiar nos momentos mais difíceis e críticos, confiou em mim e me deu forças para seguir adiante.

Agradeço ao meu orientador professor Leonardo Bidese de Pinho pela dedicação, apoio e por me auxiliar a concluir esta etapa. Também agradeço pela sua paciência durante toda esta jornada. Agradeço também, ao professor Cristian Cechinel pelas palavras de apoio nos momentos difíceis.

E por último, mas não menos importante, agradeço aos membros do LCP/COPPE-UFRJ pela colaboração.

“Descobri como é bom chegar quando se tem paciência. E para se chegar, onde quer que seja, aprendi que não é preciso dominar a força, mas a razão. É preciso, antes de mais nada, querer ”.

Amir Klink

RESUMO

Este trabalho propõe o uso de um modelo hierarquizado de memória para servidores de distribuição de *video on demand* (VoD). Fundamentada em um estudo sobre as características da aplicação e diferentes tecnologias de memória, bem como do estado-da-arte em técnicas de gerenciamento de *proxies* dinâmicos, é detalhada a abordagem de uma nova técnica denominada Memória Cooperativa Colapsada Hierarquizada (MCCH). Esta é baseada em uma técnica denominada Memória Cooperativa Colapsada (MCC), que utiliza somente a memória RAM como *cache* de dados oriundos do servidor, adaptando-a para inclusão de um segundo nível de memória como *cache* de vítimas na hierarquia de memória do *proxy*. Como forma de avaliar detalhadamente o desempenho da MCCH foi desenvolvido um simulador de eventos discretos para descrever o comportamento do sistema em uma escala de tempo. Os resultados de simulação para cenários com apenas um nível de *cache* foram corroborados, por meio de validação cruzada, com os resultados apresentados em trabalhos relacionados à MCC. Com a analise dos resultados obtidos nas simulações pode-se verificar que em cenários com dois níveis de *cache* na hierarquia de memória do *proxy*, a MCCH apresentou desempenho semelhante, entretanto, a um custo menor, quando comparada com a MCC para os mesmos percentuais de recursos no *proxy*. Além desta comparação principal, este estudo também propõe e avalia a otimização no processo de criação de *link slots* da técnica original da MCC de um único nível que leva a melhor desempenho.

Palavras-chave: Vídeo sob Demanda. Tecnologias de Memória. *Proxies* de Vídeo. Memória Cooperativa Colapsada Hierarquizada.

ABSTRACT

This work proposes the usage of a hierarchical memory model in video on demand (VoD) distribution servers. From a study about characteristics of the application and different memory technologies, as well as state-of-the-art techniques for dynamic proxy management, a novel technique called Hierarchical Collapsed Cooperative Memory (MCCH) is presented. This approach is based on a technique called Collapsed Cooperative Memory (MCC), which only uses RAM to cache data received from the system server, adapting it with the inclusion of a second memory level as a victim cache on the proxy memory hierarchy. In order to evaluate performance of MCCH, an analytical model and a discrete events simulator were developed to allow both a quantitative and qualitative analysis. The simulation results for just one cache level were corroborated with cross-validation against data from the model and results presented in previous works about MCC. The results of analyses provide ways to verify that the scenarios consisting of a two-level cache proxy memory hierarchy, the MCCH showed similar performance, though using a lower cost comparing to the MCC for the same proxy resources percentages. Besides such main comparison, this study also proposes and evaluates an optimization on the link slot creation process of the original single-level MCC technique that leads to better performance.

Keywords: Video on Demand. Memory Technologies. Video Proxies. Hierarchical Collapsed Cooperative Memory.

LISTA DE FIGURAS

LISTA DE TABELAS

SIGLAS

BS *Begin Slot*.

DDR – SDRAM *Double Data Rate* – SDRAM.

DDR3 – SDRAM *Double Data Rate Type Three* – SDRAM.

ES *End Slot*.

FS *Free Slot*.

HDD *Hard Disk Drive*.

HTML *HyperText Markup Language*.

LS *Link Slot*.

MCC Memória Cooperativa Colapsada.

MCCH Memória Cooperativa Colapsada Hierarquizada.

MCCH-LS+ MCCH com Otimização no Processo de Criação de Sequências de Link Slots.

MCCH-LS- MCCH sem Otimização no Processo de Criação de Sequências de Link Slots.

MCCL+ Memória Cooperativa Colapsada Local com Otimização.

MCCL- Memória Cooperativa Colapsada Local sem Otimização.

OMNeT++ *Objective Modular Network Testbed in C++*.

PDA *Personal Digital Assistant*.

QoS *quality of service*.

RS *Read Slot*.

SDR – SDRAM *Single Data Rate* – SDRAM.

SDRAM *Synchronous Dynamic Random Access Memory*.

SSD *Solid State Drive*.

VoD *video on demand*.

WS *Write Slot*.

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

As aplicações de vídeo sob demanda (Video on Demand - VoD) têm ganhado cada vez mais espaço na Internet. Isso se deve aos recentes avanços nas tecnologias de compressão, largura de banda e capacidade de armazenamento dos dispositivos de memória, associado ao desenvolvimento de técnicas mais eficientes para a distribuição deste conteúdo. No entanto, requisitos como largura de banda, tempo de duração da sessão, latência e tempo de entrega são ainda os principais fatores limitantes na utilização generalizada através da Internet.

O serviço de VoD abrange muitas aplicações importantes relativas ao entretenimento, educação e informação, tais como filmes, educação a distância e notícias sob demanda, incluindo a possibilidade de ter acesso às operações como pausa, posicionamento randômico durante a reprodução do vídeo e a flexibilidade de assistir o vídeo no momento que desejar sem a necessidade de esperar baixar todo o vídeo para iniciar sua exibição. No entanto, não apenas os usuários de PC tradicionais estão tendo acesso a este serviço, mas também os usuários de TV, telefone móvel e *Personal Digital Assistant* (PDA).

Diferentemente das aplicações tradicionais de texto e imagens estáticas, este tipo de aplicação exige alta largura de banda e capacidade de armazenamento. Além disso, o conteúdo transmitido deve ser reproduzido continuamente ao longo do tempo, respeitando a temporização original do arquivo gravado no servidor. Ou seja, possui requisitos rigorosos de tempo de entrega, o que a caracteriza como uma aplicação de tempo real. Por outro lado, a Internet é um serviço de comutação de pacotes, onde todos os recursos da rede são compartilhados (*buffers*, roteadores, largura de banda dos enlaces) entre os usuários, não oferecendo nenhuma garantia de largura de banda e atraso fim-a-fim.

Devido às características deste tipo de serviço, associado ao aumento na demanda, novas técnicas deverão ser desenvolvidas ou aprimoradas para viabilizar aplicações de VoD para grandes audiências, inclusive por meio da criação de redes de distribuição de conteúdo compostas por servidores intermediários (*proxy*) especializados, com o objetivo de aumentar a capacidade do sistema e reduzir a latência de início de exibição do vídeo. Estes *proxies* armazenam em sua

cache os conteúdos mais acessados, como também são responsáveis pelo gerenciamento do conteúdo que deve ser mantido na *cache* para uso futuro. Contundo, a eficiência do uso de *proxies* está relacionada com a taxa de acertos na *cache proxy*, quanto maior a taxa de acertos, maior será a eficiência do uso de *proxies*.

A MCC (ISHIKAWA, 2003) é uma técnica do estado-da-arte para gerenciamento dinâmico de *cache* para *proxies* de vídeo. Esta técnica utiliza a memória RAM como *cache* de dados recebidos do servidor de vídeo. A dinâmica de gerenciamento de memória da MCC proporciona uma redução de até 80% no tráfego entre o *proxy* e servidor principal quando a capacidade de armazenamento da memória RAM corresponde a 10% do acervo do servidor principal. Entretanto, o uso somente de memória RAM se torna proibitivo para grandes acervos de vídeo.

Uma possível alternativa a esta técnica seria usar níveis intermediários de memória, compostos por disco magnético e memória flash, como *caches* de vítimas, na expectativa de aumentar a capacidade de armazenamento do sistema à um custo menor por gigabyte. Basicamente, a *cache* de vítimas armazenaria os segmentos de vídeo que forem descartados devido a falta de espaço na *caches*. Assim seria possível evitar que o gerenciador da *caches* recorresse ao servidor principal para obter novamente os dados solicitados, aumentando a taxa de acerto no *proxy*, o que consequentemente irá diminuir o tráfego entre o *proxy* e servidor principal. Por um lado, a agregação desta hierarquia de memória aumentaria a taxa de acertos no *proxy*, mas por outro lado poderia degradar a qualidade de serviço do sistema já que estas tecnologias de memória possuem uma vazão menor e um tempo de acesso maior aos dados em comparação com a RAM. Portanto, um sistema com hierarquia precisaria ser bem dimensionado, de acordo com as características particulares da aplicação, para que não ocorra a degradação deste serviço.

Neste trabalho são estudadas as características da aplicação e das tecnologias de armazenamento como base para a proposta de uma nova técnica para *proxy* de vídeo escalável que utiliza uma hierarquia de memória para auxiliar na distribuição de vídeo. Em particular, esta técnica se apresenta como uma extensão da técnica MCC, adaptada para gerenciar a hierarquia de memória.

1.2 Objetivos

Este trabalho apresenta uma nova proposta para o sistema de memória de *proxies* de vídeo, desdobrado em objetivos específicos de acordo com a metodologia adotada para o desenvolvimento do trabalho e para uma avaliação consistente da proposta.

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral propor e implementar um *proxy* de vídeo escalável com hierarquia de memória, baseando-se em uma técnica do estado-da-arte denominada MCC, que usa apenas memória RAM como *cache* de dados.

1.2.2 Objetivos Específicos

- Detalhar e implementar a nova proposta, denominada MCCH;
- Construir o modelo analítico simplificado da MCCH para análise de sensibilidade;
- Validar, por meio de validação cruzada, a equivalência entre a MCC e MCCH para cenários com apenas um nível de *cache* na hierarquia de memória do *proxy*;
- Avaliar através de simulações o desempenho do sistema proposto;

2 DISTRIBUIÇÃO DE CONTEÚDO DE MULTIMÍDIA

Neste capítulo será apresentado brevemente as principais características das aplicações de multimídia, modos e modelos de transmissão através da Internet. A seguir, será descrito uma técnica do estado-da-arte para realizar o gerenciamento do conteúdo de multimídia armazenado em *proxy* de vídeo, denominada MCC. Está técnica será usada como base no desenvolvimento deste trabalho.

2.1 Modos de transmissão

Existem dois modos para a transferência de arquivos de multimídia através da Internet: modo *download* e modo *streaming* (WU et al., 2001).

No modo *download*, o arquivo multimídia somente poderá ser reproduzido pelo cliente após ser totalmente transferido. Esta abordagem tende a produzir uma latência[1] de início de exibição inaceitável devido ao elevado tempo gasto na transferência em função do grande tamanho destes arquivos, tornando este modo pouco atraente. No entanto, neste modo não existe nenhum requisito de tempo de entrega e largura de banda.

Diferentemente do modo *download*, no modo *streaming*, o arquivo de multimídia não precisa ser totalmente transferido para o cliente para que este inicie a sua exibição, podendo ser exibido assim que uma quantidade mínima de segmentos de vídeo estiver armazenada em seu *buffer*. Desta forma, à medida que as demais partes do vídeo chegam ao cliente, as mesmas são decodificadas e reproduzidas, diminuindo assim a latência inicial. Isto implica em restrições de tempo e largura de banda que deverão ser levados em conta no momento do projeto do sistema, de modo que estas sejam consideradas na política de admissão de clientes para que não ocorram interrupções na exibição dos fluxos de vídeo transmitidos pelo sistema.

[1] Período de tempo entre a requisição e o início efetivo de exibição do vídeo

2.2 Classes de aplicações de *streaming*

As aplicações de multimídia no modo *streaming* podem ser classificadas em três classes de aplicações (KUROSE; ROSS, 2006):

Áudio e vídeo de fluxo contínuo armazenado: Nesta classe de aplicação, o conteúdo de multimídia se encontra comprimido e armazenado em um servidor. O conteúdo é enviado aos clientes sob demanda e estes começam a sua reprodução depois de um pequeno intervalo de tempo após a requisição.

Esta classe tem três características fundamentais:

- (i) Conteúdo armazenado: o conteúdo de multimídia está pré-gravado e armazenado em um servidor, permitido que os clientes realizem as operações de pausa, avanço e retrocesso no conteúdo solicitado;
- (ii) Fluxo contínuo: a exibição do conteúdo inicia após alguns segundos de sua solicitação, permitindo que os clientes reproduzam o conteúdo já armazenado em seu *buffer* ao mesmo tempo em que estão recebendo partes do conteúdo que estão mais à frente. Essa técnica é conhecida como fluxo contínuo (*streaming*), evitando que o cliente tenha que esperar a transferência total do conteúdo solicitado antes de sua reprodução, reduzindo-se assim, a latência inicial para a exibição;
- (iii) Reprodução contínua: assim que o cliente começa a exibição do conteúdo de multimídia, ela deve prosseguir de acordo com a temporização original da gravação, ou seja, os dados devem ser recebidos do servidor a tempo de serem reproduzidos no cliente. Caso contrário, pausas e interrupções ocorrerão.

O Youtube (YOUTUBE, 2011), Hulu (HULU, 2011) e Netflix (NETFLIX, 2011) são exemplos deste tipo de classe de *streaming*.

Áudio e vídeo de fluxo contínuo ao vivo: Esta classe de aplicação é semelhante a transmissão de rádio e televisão, exceto que a transmissão é feita através da *Internet*. Devido a sua entrega ser ao vivo, e o conteúdo não ser armazenado, não é possível que os clientes tenham acesso aleatório ao conteúdo. Além disso, se um cliente deseja reproduzir um conteúdo já em andamento, o mesmo se juntará a sessão existente, perdendo assim as partes anteriores do *stream*.

Áudio e vídeo interativos em tempo real: Esta classe de aplicação permite que os clientes se comuniquem entre si interativamente usando áudio/vídeo, em tempo real. Um exemplo é o sistema de VoIP da Skype, adquirido pela Microsoft (MICROSOFT, 2011).

Apesar de menos rigorosa em relação ao *jitter*[2] em comparação com as outras duas classes apresentadas, a transmissão de fluxos contínuos de multimídia armazenados para grandes audiências, demanda gerenciamento otimizado dos recursos disponíveis nos servidores do sistema de distribuição e na rede.

2.3 Características das aplicações de multimídia

Em contraste às aplicações tradicionais de texto e imagens estáticas na *Internet*, as aplicações de multimídia possuem requisitos de largura de banda, capacidade de armazenamento e perda de dados (WU et al., 2001; SILBERSCHATZ; GALVIN; GAGNE, 2008). Por exemplo, considere um vídeo digital com resolução de 800 x 600 *pixels*, e 60 minutos de duração. Se usarmos 24 bits para representar a cor de cada pixel, um único quadro exigirá 11520000 bits. Se os quadros são exibidos a uma taxa de 25 quadros/segundo, taxa necessária para que o vídeo apareça suave ao olho humano, será necessária uma largura de banda de 36 MBps e um espaço de armazenamento de 129,6 GB. Logo, um servidor com centenas ou milhares de vídeos com este padrão de codificação, exigirá vários *terabytes* de armazenamento, além de exigir uma quantidade imensa de largura de banda para manter todos os fluxos de vídeo simultaneamente, mesmo para uma quantidade limitada de clientes concorrentes. Desta forma, fica claro que o armazenamento e a transferência através da *Internet* se tornam impraticáveis sem compressão.

Em relação ao requisito de perda, as aplicações de multimídia no modo *streaming* são até certo ponto tolerantes à perda, incorrendo em pequenas perturbações na exibição do vídeo, diferentemente do modo *download*, que é altamente sensível à perda.

2.4 Métodos de transmissão

A forma mais simples de transmitir o conteúdo multimídia pela *Internet* é por meio do método *unicast*. Neste método, para cada requisição do cliente é criado um fluxo de vídeo independente. Desta forma, o servidor precisa estabelecer uma conexão dedicada para cada requisição, limitando o número de clientes que podem ser atendidos simultaneamente pelo sistema, de acordo com o valor da largura de banda disponível no servidor e com a taxa dos vídeos. No entanto, devido à alta largura de banda exigida, associada com o aumento na demanda, este método se tornou ineficiente para transmitir este tipo de conteúdo em sistemas centralizados, com um único servidor centralizado, sendo necessário adotar novos métodos de transmissão.

Por outro lado, se um vídeo é solicitado, existe a tendência de que outros clientes solicitem

[2] *Jitter* é a variação do atraso entre os pacotes sucessivos de dados enviados através da rede.

sua reprodução em um intervalo de tempo próximo, como por exemplo, no pico do horário nobre da TV, compreendido entre os horários das 20h às 23h, ou em uma aula virtual, onde vários alunos sejam orientados a assistir certo conteúdo durante determinado período. Sendo assim, é possível aproveitar o fato de vários clientes estarem buscando o mesmo conteúdo em um intervalo de tempo curto, permitindo assim o compartilhamento da transmissão.

Uma modo de se compartilhar a transmissão é através do *broadcast*, mais conhecido como difusão. Neste método, o conteúdo é recebido por todos os clientes. Contudo, não são todos os usuários da *Internet* que tem interesse em receber o conteúdo, e sim, um grupo de clientes. Logo, é necessário que seja permitida a seletividade da transmissão. Com base nisso, foi criado o método *multicast*, onde os fluxos de vídeo são recebidos somente pelo grupo de clientes que demonstram interesse em recebê-lo. Desta forma, um único fluxo de vídeo é mantido pelo servidor, replicando-se durante seu percurso pela rede quando necessário. No entanto, para que isso seja possível, é necessário que toda a rede tenha suporte ao *multicast*, o que não ocorre na *Internet*.

2.5 Serviço de distribuição de VoD

Uma forma simples de se explicar o serviço VoD é compará-lo com uma vídeo locadora digital virtual. O cliente, por meio de uma página *Web*, solicita a exibição de um determinado vídeo. Após sua solicitação, em um curto intervalo de tempo, começa a receber o conteúdo solicitado através do modo *streaming*, permitindo que o cliente reproduza o conteúdo à medida que o está recebendo. Note que, este serviço é atraente, pois permite que o cliente possa realizar pausa, avanço, retrocesso no conteúdo de vídeo, de modo que se enquadra na classe de aplicações de áudio e vídeo de fluxo contínuo armazenado.

2.5.1 Componentes do sistema VoD

O serviço VoD é usualmente implementado a partir de um sistema composto basicamente de um servidor de vídeo principal, responsável por armazenar um conjunto de vídeos (acervo) e transmitir fluxos destes quando solicitado; *proxies*, servidores intermediários responsáveis pelo gerenciamento e armazenamento de cópias dos conteúdos mais populares; e por fim, os clientes, que solicitam e reproduzem os vídeos. Em relação ao *proxy*, algumas considerações devem ser destacadas.

Os *proxies* são servidores intermediários que são colocados na borda da rede, mais próximos dos clientes, que tem como principal função armazenar os conteúdos solicitados pelos clientes,

mantendo em sua *cache* os conteúdos mais acessados. Desta forma, é possível reduzir tanto a latência fim-a-fim percebida pelos clientes, em caso de um *hit* na *cache* do *proxy*, como também reduz a carga sobre o servidor principal, uma vez que se o objeto solicitado estiver na *cache* do *proxy*, não será necessário solicitar ao servidor. Estas características dos *proxies* são de suma importância em aplicações de vídeo sob demanda, uma vez que estas possuem requisitos de tempo, largura de banda e espaço de armazenamento.

As aplicações de multimídia são aplicações que exigem alta largura de banda, tanto dos dispositivos de memória quanto da rede, e devem ser mantidas durante todo o período de exibição do vídeo. Portanto, minimizar a largura de banda consumida pelo sistema é prioridade para o gerenciamento da *cache* do *proxy*, pois este recurso é um fator limitante na escalabilidade do sistema, restringindo o número de clientes que podem ser atendidos simultaneamente pelo sistema (TANG; XU; CHANSON, 2005).

O gerenciamento de *proxies* de vídeo se difere de *proxies* Web convencionais, que armazenam páginas *HyperText Markup Language* (HTML) e imagens estáticas. A principal diferença está na forma como os objetos são manipulados. *Proxies* convencionais manipulam objetos, enquanto *proxies* de vídeo manipulam segmentos de objetos de vídeo (ISHIKAWA, 2003). Isso se deve ao fato de que os objetos manipulados em *proxies* convencionais possuem tamanho na ordem de 1KB até 100KB. Sendo assim, uma quantidade grande de objetos poderá ser armazenados na *cache*, mantendo uma taxa de acertos alta no *proxy*. No entanto, os objetos de vídeo exigem grande espaço de armazenamento. Logo, um pequeno número de objetos de vídeo poderá ocupar toda a *cache* e, consequentemente, o *proxy* terá uma baixa taxa de acertos. Devido a isso, os *proxies* de vídeo manipulam segmentos de objetos de vídeo ao invés do objeto por inteiro. Cabe ressaltar que vídeo de curto período de duração (*clips*), devido ao pequeno tamanho do objeto de vídeo, o *proxy* convencional costuma ser eficiente (ISHIKAWA, 2003).

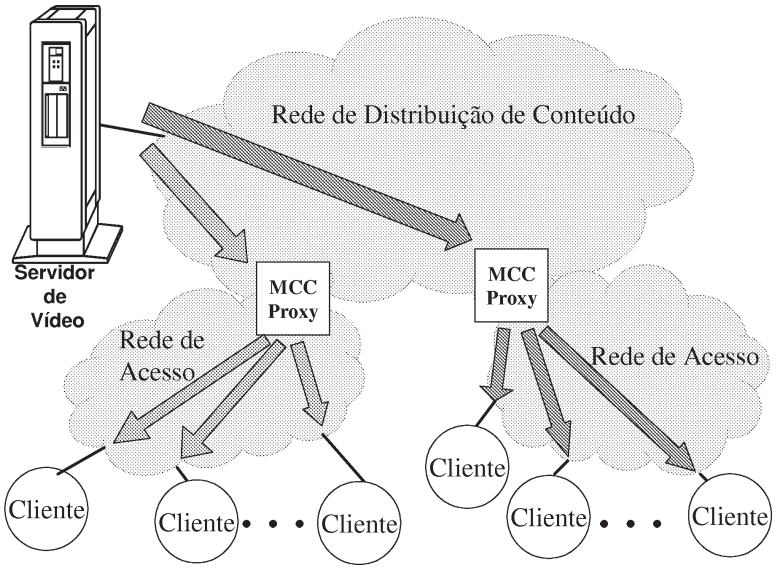
O gerenciamento da *cache* dos *proxies* de vídeo pode ser estático ou dinâmico. No gerenciamento estático, o *proxy* trata cada segmento de vídeo como um objeto distinto dos demais, diferentemente do gerenciamento dinâmico, o qual leva em conta a relação temporal entre os segmentos (objetos) de um mesmo vídeo.

2.5.2 Memória Cooperativa Colapsada (MCC)

A Memória Cooperativa Colapsada - MCC (ISHIKAWA, 2003) é um exemplo representativo do estado-da-arte de técnicas de gerenciamento de *cache* dinâmico para *proxies* de vídeo. Devido aos requisitos das aplicações de VoD, a MCC usa a memória RAM como *cache* de dados recebidos do servidor de vídeo buscando aproveitar a baixa latência de acesso e alta vazão deste

dispositivo de armazenamento. No entanto, devido ao alto custo por GB de armazenamento quando comparado com os discos magnéticos, a capacidade de armazenamento da *cache* é relativamente pequena quando comparada com o tamanho do acervo do servidor. Para contornar esta potencial limitação, a MCC implementa um gerenciador dinâmico de *cache* que leva em conta, além da relação temporal entre os segmentos de um mesmo vídeo, a popularidade dos trechos de vídeo e o custo relativo de substituição de sequências contíguas de segmentos. A Figura 1 apresenta a arquitetura da MCC.

Figura 1 – Arquitetura da MCC.



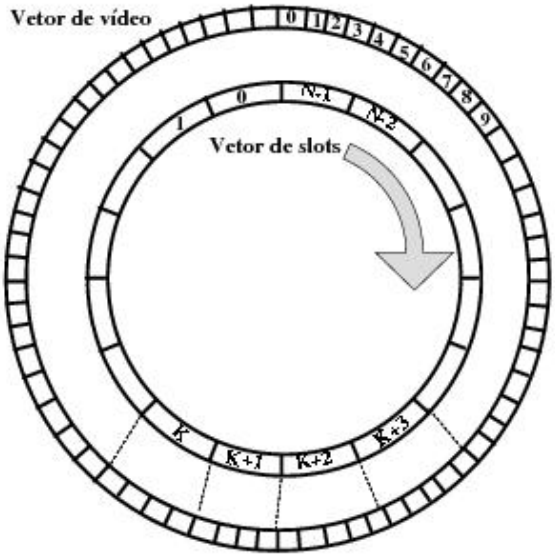
Fonte: (ISHIKAWA, 2003)

2.5.2.1 Estruturas e conceitos da MCC

A MCC possui, para cada vídeo armazenado em sua cache, dois vetores circulares: o vetor de vídeo e o vetor de *slots*, conforme visto na Figura 2. O vetor de vídeo é usado pela MCC para mapear as unidades de vídeo[3], que em conjunto compõem um fluxo de vídeo completo, na *cache* do *proxy*, estando estes trechos presentes ou não. Por meio deste vetor, a MCC acessa os segmentos de vídeo mapeados na *cache*. Para isso, cada posição deste vetor guarda em sua estrutura metadados referentes aos segmentos de vídeo armazenados em sua *cache*, tais como: ponteiro, que é usado para indicar o local de armazenamento da unidade de vídeo na *cache*; validade, usado para indicar se o conteúdo mapeado pelo ponteiro é válido.

[3] A unidade de vídeo é o menor segmento de vídeo que a MCC consegue manipular.

Figura 2 – Estruturas da MCC – Vetor de *slots* e vetor de vídeo.



Fonte: (ISHIKAWA, 2003)

Já o vetor de *slots* é usado pela MCC para realizar o gerenciamento dos *buffers* colapsados e sequências de LS, além de determinar quais unidades de vídeo poderão ser removidas quando a MCC requisitar a liberação de recursos de armazenamento. Sendo assim, um *slot* nada mais é do que uma estrutura de controle usada para gerenciar as unidades de vídeo no vetor de vídeo. Para isso, cada *slot* armazena em sua estrutura metadados de controle, tais como: ponteiros, indicando os limites de cada *slot* no vetor de vídeo; tipo do *slot*, que define o estado (prioridade) das unidades de vídeo mapeadas pelo mesmo em função do tempo. Os estados que cada *slot* poderão assumir são descritos à seguir:

***Read Slot* (RS)** – *Slot* do tipo leitura;

***Write Slot* (WS)** – *Slot* do tipo escrita;

***End Slot* (ES)** – *Slot* do tipo final de *buffer*;

***Begin Slot* (BS)** – *Slot* do tipo início de *buffer*;

***Free Slot* (FS)** – *Slot* do tipo livre;

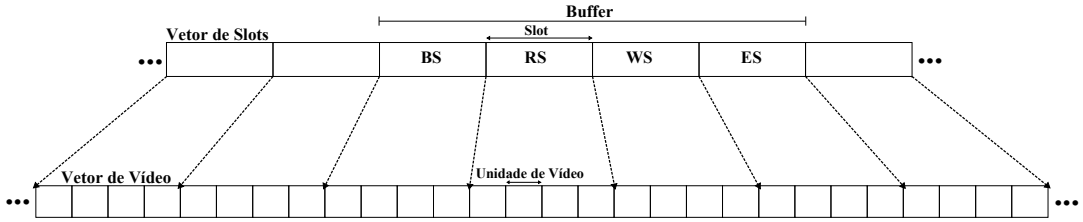
***Link Slot* (LS)** – *Slot* do tipo ligação.

A partir do primeiro pedido para um determinado vídeo, o vetor de *slots* começa a girar sincronamente na mesma velocidade de exibição do vídeo no sentido horário em relação ao vetor de vídeo, que permanece estacionário. Como o vetor de *slots* gira, as unidades de vídeo terão sua prioridade alterada de acordo com o tipo de *slot* a qual pertencem. De forma geral, as

unidades de vídeo mapeadas pelos *slots* pertencentes a um *buffer* e *slots* de LS não poderão ser descartadas. Já os *slots* do tipo FS são *slots* que não pertencem a nenhum *buffer* e poderão ter as unidades de vídeo que estão sendo mapeadas por este *slot* liberadas, caso presente, quando a *cache* estiver cheia.

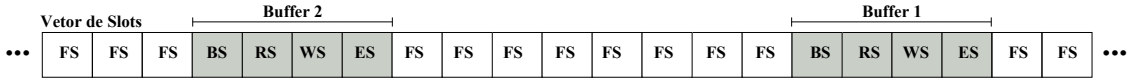
Basicamente, um *buffer* ocupa quatro *slots* do vetor com a seguinte ordem BS,RS,WS,ES, conforme ilustrado na Figura 3. O *slot* RS, representa a porção temporal de vídeo que está sendo enviada a um ou mais clientes contidos neste buffer. O *slot* WS representa a porção temporal de vídeo que está alimentando este *buffer*, ou seja, é o fluxo de vídeo oriundo do servidor. Já os *slots* BS e ES são usados para controle de risco de *underflow* e *overflow*, uma vez que a taxa na qual o fluxo está alimentando este *buffer* pode estar mais lenta ou rápida que a taxa de envio do vídeo. Desta forma, pode ser gerado um sinal de alerta, permitindo que o sistema interaja com o servidor a tempo de reduzir ou interromper o fluxo de vídeo que alimenta este *buffer*.

Figura 3 – Projeção do vetor de *slots* sobre o vetor de vídeo.

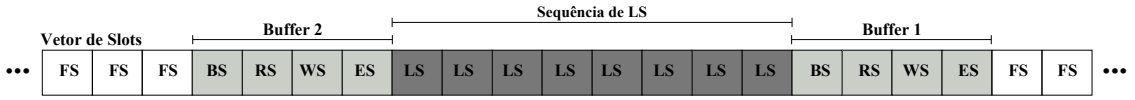


O *slot* LS se diferencia dos demais por ter seu conteúdo reservado na *cache*, independente de pertencer ou não a um *buffer*. Sua função é unir dois *buffers* separados por uma distância temporal, priorizando-se a manutenção do conteúdo de vídeo mapeado pelos *slots* entre os *buffers*, possibilitando que um único fluxo de vídeo oriundo do servidor seja necessário para alimentá-los, reduzindo assim, a carga sobre o servidor, como visto na Figura 4. No entanto, em cenários onde todo o conteúdo da *cache* estiver reservado por *buffers* e *slots* do tipo LS, a prioridade deste tipo de *slot* poderá ser reduzida para FS, permitindo que o conteúdo mapeado pelo mesmo seja liberado da *cache* para que novos *buffers* sejam criados, evitando que os clientes sejam bloqueados, mas com a penalidade no aumento de fluxos entre o servidor e *proxy*. Esta habilidade permite que a MCC seja mais flexível que as soluções existentes na literatura que também implementam *ring buffers* para o gerenciamento da memória dos *proxies* de vídeo.

Figura 4 – Criação da sequência de LS.



(a) Estado do vetor de *slots* antes da criação da sequência de LS



(b) Estado do vetor de *slots* após a criação da sequência de LS

2.5.2.2 Política de substituição na *cache* do *proxy*

Quando a *cache* atingir sua capacidade máxima de armazenamento, a MCC inicia o processo de liberação de recursos, priorizando a manutenção das unidades de vídeo mapeadas pelos *slots* dos vídeos mais populares, em detrimento dos vídeos menos populares[4]. O objetivo é manter na *cache* o conteúdo dos vídeos mais requisitados, permitindo uma maior taxa de acertos (*hits*) na *cache* do *proxy*. No entanto, as unidades de vídeo sob controle dos *buffers* colapsados e *slots* LS são reservados e não poderão ser removidos da *cache*. Portanto, a MCC inicialmente libera as unidades de vídeo que estão presentes na *cache* e sendo mapeadas pelos *slots* do tipo FS, priorizando-se, para o mesmo vídeo, a liberação do conteúdo dos *slots* mais ao final do vídeo. Contudo, dependendo da demanda ao sistema, a quantidade de unidades de vídeos liberadas poderá não ser suficiente para armazenar os segmentos de vídeos oriundos do servidor, logo, resta a MCC reduzir a prioridade dos *slots* do tipo LS para FS, permitindo que o conteúdo mapeado pelos mesmos sejam liberados, em troca da criação de um fluxo entre o servidor e o *proxy*.

O algoritmo de remoção de sequências LS, SR7 – Alocação de Capacidades, demonstrado por (ISHIKAWA, 2003), é NP–Completo, sendo muito custoso para se obter a solução ótima, principalmente considerando-se as restrições de tempo real das aplicações de VoD, mas passível de uma solução próxima da ótima a um custo menor.

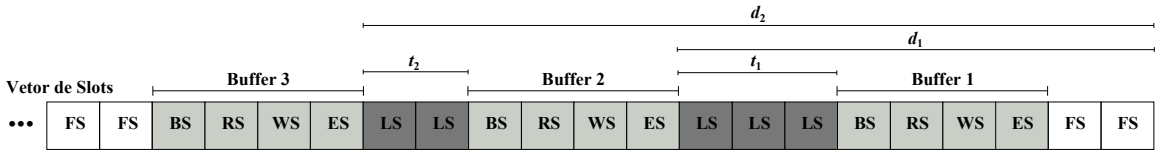
O algoritmo consiste em organizar as sequências de LS levando-se em consideração o custo em removê-las. O custo de remoção de cada sequência é calculado de acordo com a Equação 2.1:

$$F(d_i, t_i, f) = \frac{d_i}{\min(t_i, f)} \tag{2.1}$$

[4] A popularidade de um determinado vídeo no escopo deste estudo é definida em função do número de clientes que o assistem concorrentemente em um período de tempo.

onde, d_i é a distância do primeiro *slot* da i-ésima sequência de LS até o final do vetor de *slots*, t_i é o tamanho da i-ésima sequência de LS, em *slots*, e f é o número de *slots* que devem ser liberados, conforme ilustrado na Figura 5.

Figura 5 – Custo de remoção da sequência de LS.



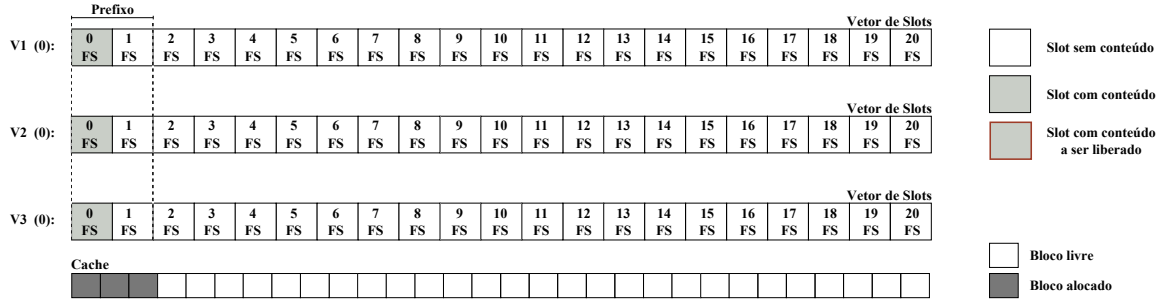
O objetivo deste cálculo é priorizar a remoção de sequências de LS que tenham maior tamanho, cuja tempo de duração do fluxo seja o mínimo possível. Além do custo de remoção, a MCC também utiliza a influência da popularidade de cada vídeo como critério para a escolha das sequências a serem removidas.

2.5.2.3 Funcionamento da MCC

O objetivo desta seção é explicar detalhadamente o funcionamento da MCC. Para isso, considere um servidor com três vídeos disponíveis em seu acervo, representados por V1,V2 e V3, todos com o mesmo tamanho de 21 *slots* (ver Figura 6), onde cada *slot* representa 8 segundos de vídeo. A *cache* está dividida em blocos de tamanho fixo com capacidade total de armazenamento de 30 blocos, no qual cada bloco equivale ao tamanho de um *slot*.

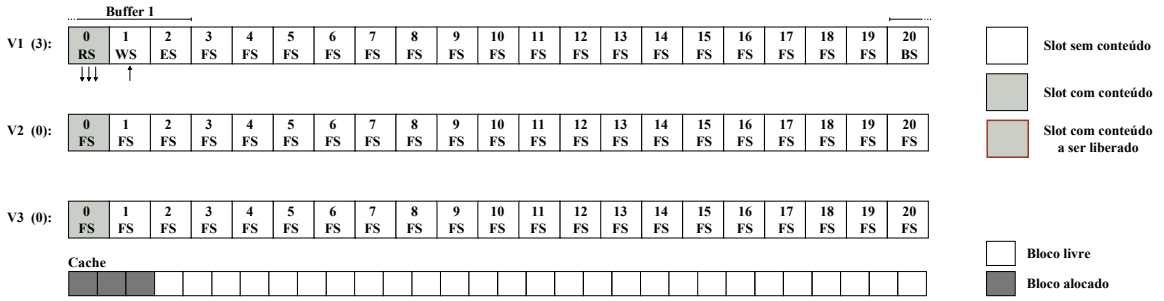
Para simplificar o exemplo, considere que a porção temporal de vídeo associada à k-ésima posição no vetor de *slots* é $[8*k, 8*(k+1))$, para $0 \leq k \leq 20$, além disso, nas ilustrações descritas a seguir, o vetor de vídeo será abstraído.

Figura 6 – Estado inicial dos vetores de *slots* de cada vídeo e da *cache* antes da primeira inclusão dos clientes ao sistema.



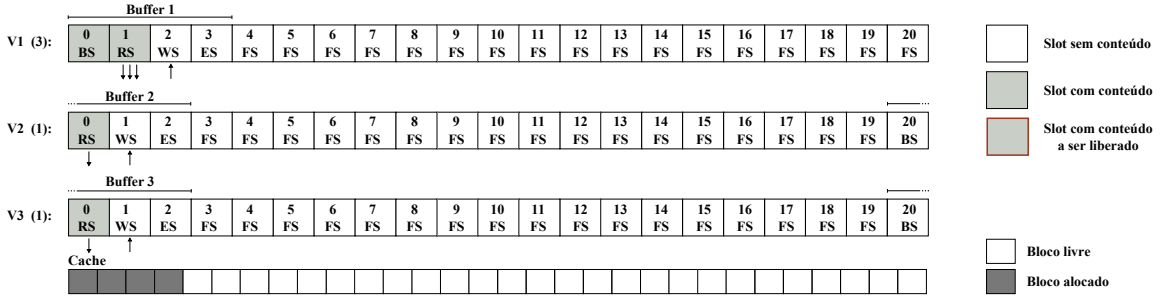
Assumindo-se que, em t_0[5], a MCC inicia o processo de inclusão dos clientes ao sistema. Neste instante, três clientes solicitaram a exibição do vídeo V1 à partir do início. Como não há um *buffer* no ponto solicitado, a MCC deverá criá-lo para atender a estas requisições, como visto na Figura 7. A partir deste momento, para cada cliente incluído no *buffer* 1, um fluxo de vídeo é criado para transmitir[6], de acordo com a taxa de envio do vídeo, o conteúdo mapeado pelo *slot* RS ao destino (seta para baixo), e outro fluxo passa a ser recebido do servidor para alimentar o *slot* WS (seta para cima).

Figura 7 – Primeiro *buffer* criado, em t_0, para atender a requisição de inclusão dos clientes.



No próximo giro, em t_1, a MCC inicialmente processa os pedidos em espera na fila, e verifica que há mais duas requisições de inclusão, uma para o vídeo V2 e outra para o vídeo V3. Novamente, a MCC assegura que há disponibilidade de recursos para a inclusão dos clientes no sistema, permitindo que os *buffer* 2 e *buffer* 3 sejam criados, conforme visto na Figura 8. Já que as unidades de vídeos mapeadas pelo *slot* WS dos *buffers* 1, 2 e 3 não estão presentes[7] em *cache*, a MCC deverá solicitar ao servidor, totalizando, assim, três fluxos de vídeo que serão recebidos do servidor, um por *buffer*.

Figura 8 – Inclusão de novos clientes ao sistema em t_1.



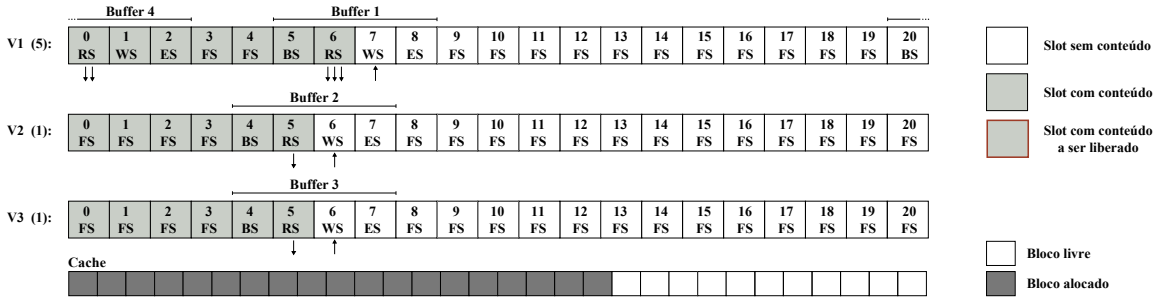
[5] t_i é a variável tempo em unidade de giro, onde i, representa o valor do giro atual do sistema. O período de um giro da MCC é igual ao tempo de envio de um *slot* de vídeo, que neste caso é de 8 segundos.

[6] Considerando-se que o modelo de transmissão de dados por meio da rede é *unicast*.

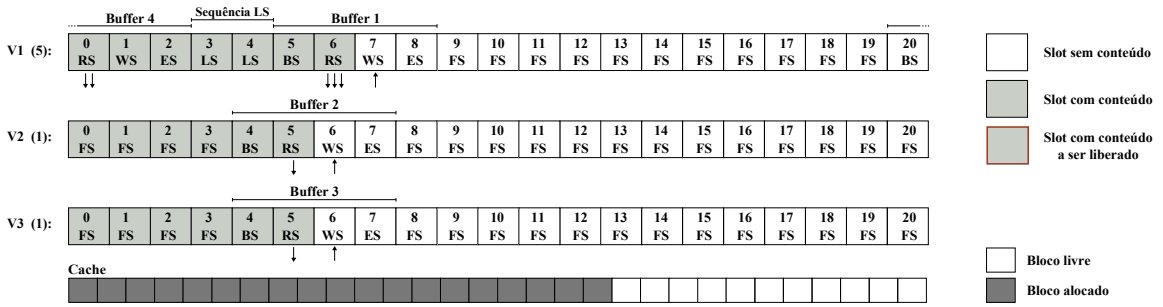
[7] É assumido que os *slots* que não estão presentes em *cache* (*slots* sem conteúdo) não estão alocados.

Após alguns instantes de tempo, mais especificamente, em t_6, ao verificar a disponibilidade de recursos e incluir o *buffer* 4 no vídeo V1 (ver Figura 9a), para atender a requisição de inclusão dos dois novos clientes, a MCC cria a primeira sequência de LS para unir os *buffers* 1 e 4, priorizando-se a manutenção das unidades de vídeo mapeadas por estes *slots*, conforme ilustrado na Figura 9b. Desta forma, o conteúdo de vídeo mapeado por esta sequência de LS não será removido quando a MCC requisitar a liberação de recursos, exceto se a quantidade de *slots* do tipo *FS* liberados não for suficiente para atender a demanda do sistema.

Figura 9 – Criação da primeira sequências de LS em t_6.



(a) Estado do vetor de *slots* antes de criar a sequência de LS para unir os *buffers*.



(b) Estado do vetor de *slots* após criar a sequência de LS para unir os *buffers*.

A medida que o vetor de *slots* gira, as unidade de vídeo mapeadas pelos *slots* do tipo WS que não estão presentes na *cache* são solicitadas ao servidor para que no próximo giro sejam enviadas aos clientes. Note que o *buffer* se comporta como uma janela deslizante sobre o vetor de *slots*, onde, a cada giro, o *buffer* se desloca um *slot* à direita. As Figuras 10 e 11 ilustram o estado da MCC para os instantes de tempo t_7 e t_8, respectivamente.

Figura 10 – Estado da MCC, em t_7, após incluir o *buffer* 5 para atender a requisição de inclusão dos novos clientes ao vídeo V2.

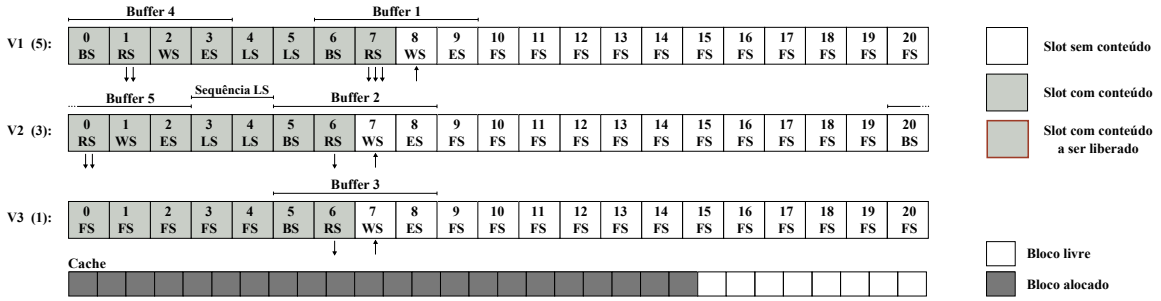
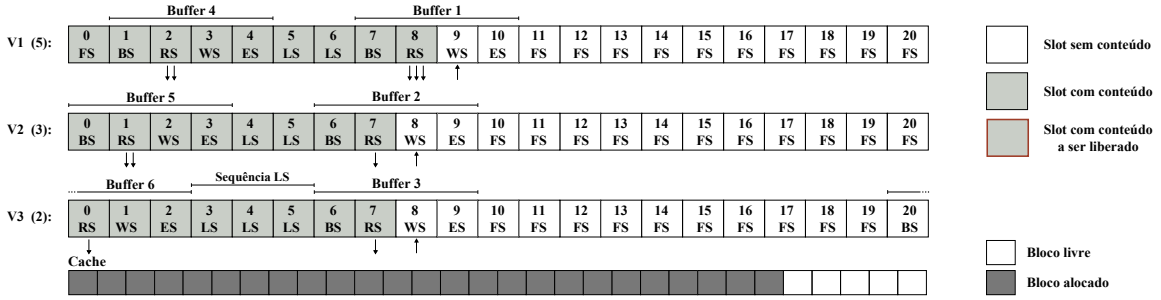


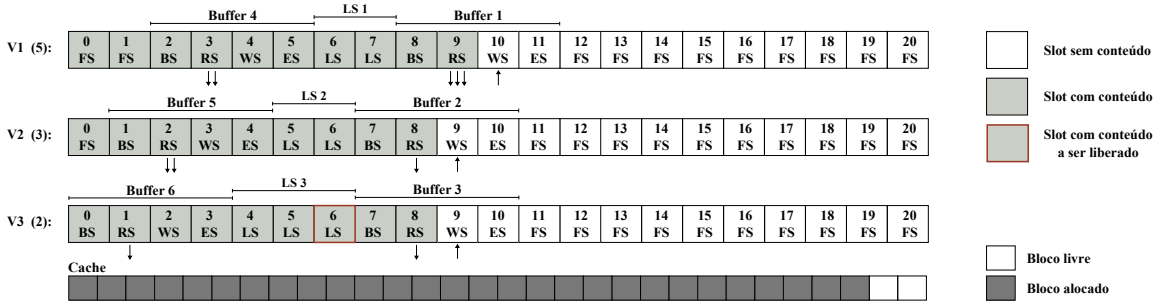
Figura 11 – Estado da MCC, em t_8, após incluir o *buffer* 6 para atender a requisição de inclusão do novo cliente ao vídeo V3.



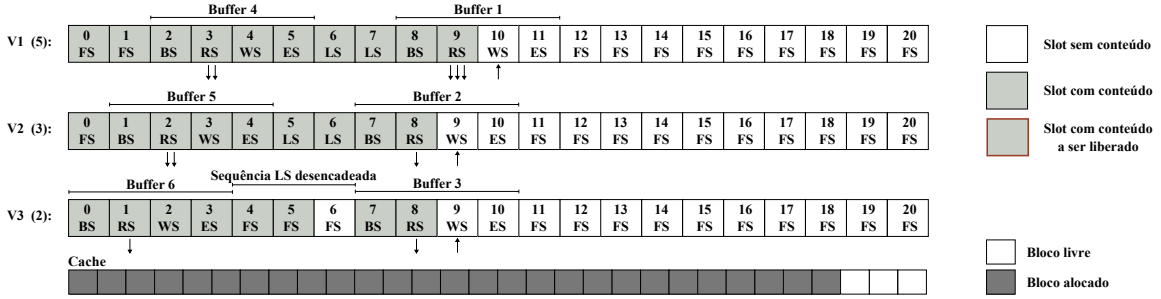
Em t_9, a MCC verifica que o *slot* WS dos *buffers* 1, 2 e 3 (ver Figura 12a) deverão ter seu conteúdo alocado (um bloco por *slot*, totalizando 3 blocos de memória), já que os mesmos não estão presentes em *cache*. Entretanto, ao avaliar a disponibilidade de recursos de armazenamento, dos 30 blocos na *cache*, 28 estão ocupados e apenas dois livres. Todavia, este cenário não é sustentável, pois 31 blocos serão utilizados, sendo superior a capacidade de armazenamento da *cache*. Neste momento, a MCC inicia a liberação de recursos, requisitando a remoção de 1 bloco de memória $((28+3)-30)$. De acordo com o critério de substituição descrito por (ISHIKAWA, 2003), o vídeo V3, que possue menor número de clientes (dois clientes), terá maior prioridade na liberação do conteúdo mapeado pelos *slots* do tipo FS, porém, note que, o vídeo V3 não possuem nenhum *slot* do tipo FS com conteúdo, sendo necessário liberar blocos de memória alocados pelos vídeos de maior popularidade. Tendo em vista que, o vídeo V2, segundo vídeo menos popular (três clientes), tenha um *slot* FS com conteúdo (*slot* na posição 0), o mesmo faz parte do prefixo, e não poderá ser liberado. Sendo assim, a MCC tenta liberar recursos de armazenamento (*slots* FS) alocados para o vídeo V1 (vídeo mais popular) nas posições 1 e 0, conforme a ordem descrita, porém os conteúdos mapeados por estes *slots* fazem parte do prefixo, e não poderão ser liberados. Logo, resta a MCC reduzir a prioridade dos *slots* do tipo

LS para FS, permitindo que os conteúdos mapeados pelos mesmos sejam removidos. Como dito anteriormente, determinar a solução ótima,que passa por, sempre que necessário, encontrar a sequência de LS, dentre as sequências de LS do vídeo menos popular, que represente o menor custo possível, é NP–completo. Portanto, a MCC organiza as sequências de LS de acordo com o custo individual de remoção (conforme descrito na Equação 2.1), priorizando-se a manutenção das sequências de maior custo dos vídeos mais populares, ou seja, as sequências de menor custo pertencentes ao vídeo de menor popularidade terão maior prioridade de remoção. O custo de remoção das sequências LS 1, LS 2 e LS 3 (ver Figura 12a) e a popularidade dos vídeos a que pertencem são apresentados na Tabela 1.

Figura 12 – Primeiro cenário, em t_9, no qual a MCC deverá liberar recursos para armazenar os segmentos oriundos do servidor principal.



(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.



(b) Estado dos vetores de *slots* da MCC após liberar recursos.

Tabela 1 – Custo de remoção das sequências de LS em t_9.

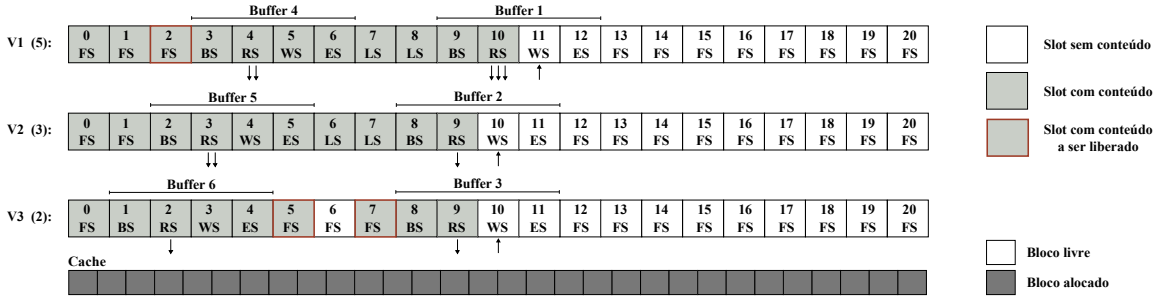
Vídeo	Popularidade	Sequência LS	Custo de Remoção
V1	5	LS 1	15
V2	3	LS 2	16
V3	2	LS 3	17

Baseando-se no critério descrito, os *slots* nas posições 4, 5 e 6 (sequência LS 3), pertencentes ao vídeo V3, terão suas prioridades alteradas para FS, permitindo, assim, que a MCC possa

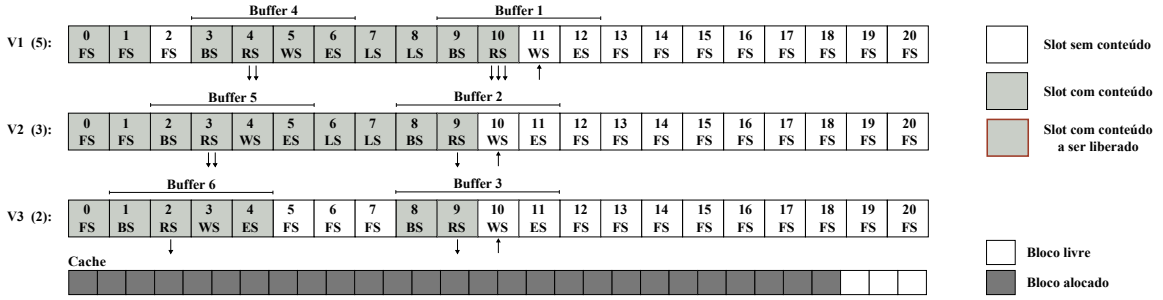
liberar os recursos necessários. Dentre os três *slots* FS, a MCC prioriza a remoção dos *slots* mais ao final do vídeo. Logo, o *slot* na posição 6 terá o conteúdo liberado, conforme ilustrado na Figura 12b.

Em t_{10} mais recursos deverão ser liberados para armazenar os fluxos de vídeo oriundos do servidor. Contabilizando, tem-se onze *slots* com conteúdo no vídeo V1, dez *slots* no vídeo V2 e nove *slots* no vídeo V3, além de três *slots WS*, pertencentes aos *buffers* 1, 2 e 3, que deverão ter seu conteúdo alocados neste giro, totalizando, assim, 33 *slots* a serem utilizados, excedendo a capacidade de armazenamento da *cache* em três blocos, conforme apresentado na Figura 13a. Deste modo, a MCC requisita a liberação de três *slots* de vídeo, no entanto, como há *slots* FS com conteúdo que não fazem parte do prefixo, referentes aos *slots* na posição 7 e 5 do vídeo V3 e o *slot* na posição 2 do vídeo V1, os mesmo serão liberados (de acordo com a ordem descrita), portanto, nenhuma sequência de LS será desencadeada, conforme ilustrado na Figura 13b.

Figura 13 – Estado da MCC em t_{10}.



(a) Estado dos vetores de *slots* e da *cache* antes de liberar recursos.

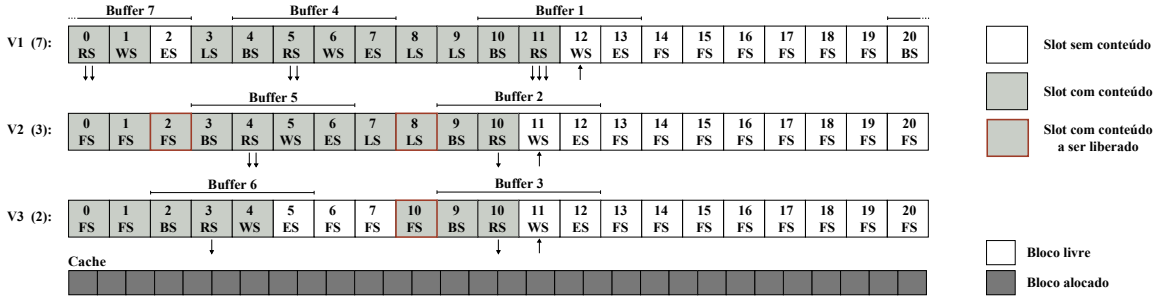


(b) Estado dos vetores de *slots* e da *cache* após liberar recursos.

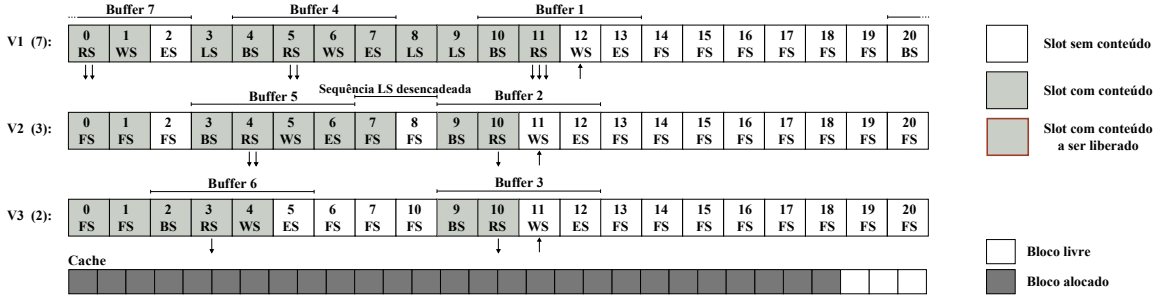
De forma geral, durante o intervalo de tempo entre $[t_i, t_{i+1})$, equivalente ao tempo de um *slot* de vídeo, a MCC deverá: deverá: (i) processar os pedidos em espera na fila de inclusão; (ii) liberar recursos, caso a *cache* atinja sua sua capacidade máxima de armazenamento; (iii) transmitir aos clientes, de acordo com a taxa de envio do vídeo, todo o conteúdo mapeado pelo *slot* RS do *buffer* a que pertencem; e (iv) requisitar ao servidor e armazenar na *cache* as unidades de vídeo mapeadas pelos *slots* WS que não estão presentes na *cache* do *proxy*. Caso os requisitos

de tempo da aplicação não sejam atendidos, a qualidade de serviço (Quality of Service - QoS) poderá ser comprometida, ocasionado interrupções na exibição do vídeo por parte dos clientes. As Figuras 14 à 20 ilustram o estado da MCC a cada giro do sistema.

Figura 14 – Estado da MCC em t_{11}.

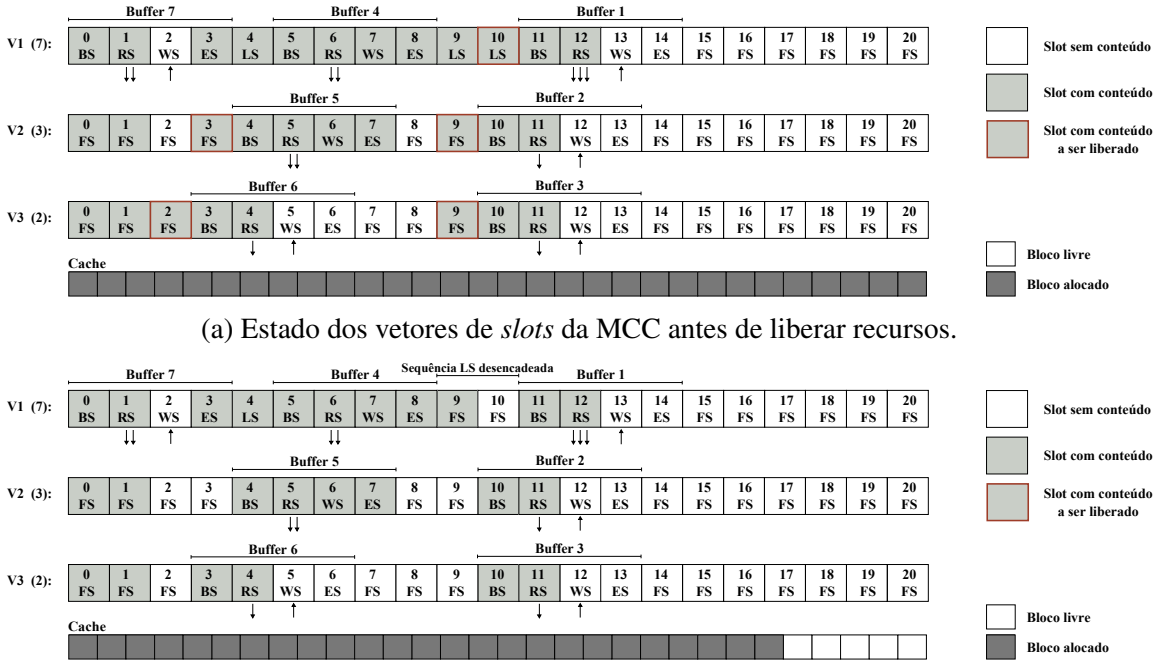


(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.



(b) Estado dos vetores de *slots* da MCC após liberar recursos.

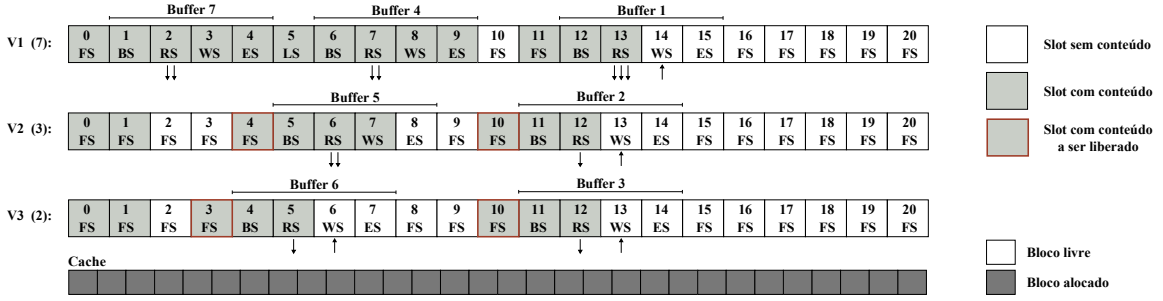
Figura 15 – Estado da MCC em t_{12}.



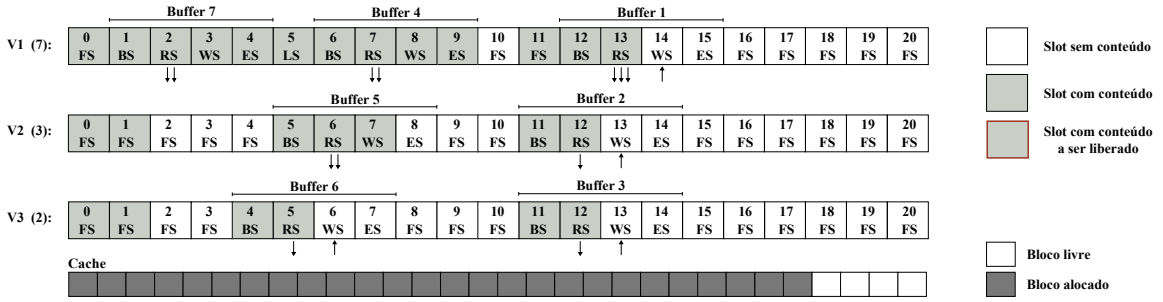
(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.

(b) Estado dos vetores de *slots* da MCC após liberar recursos.

Figura 16 – Estado da MCC em t_{13}.

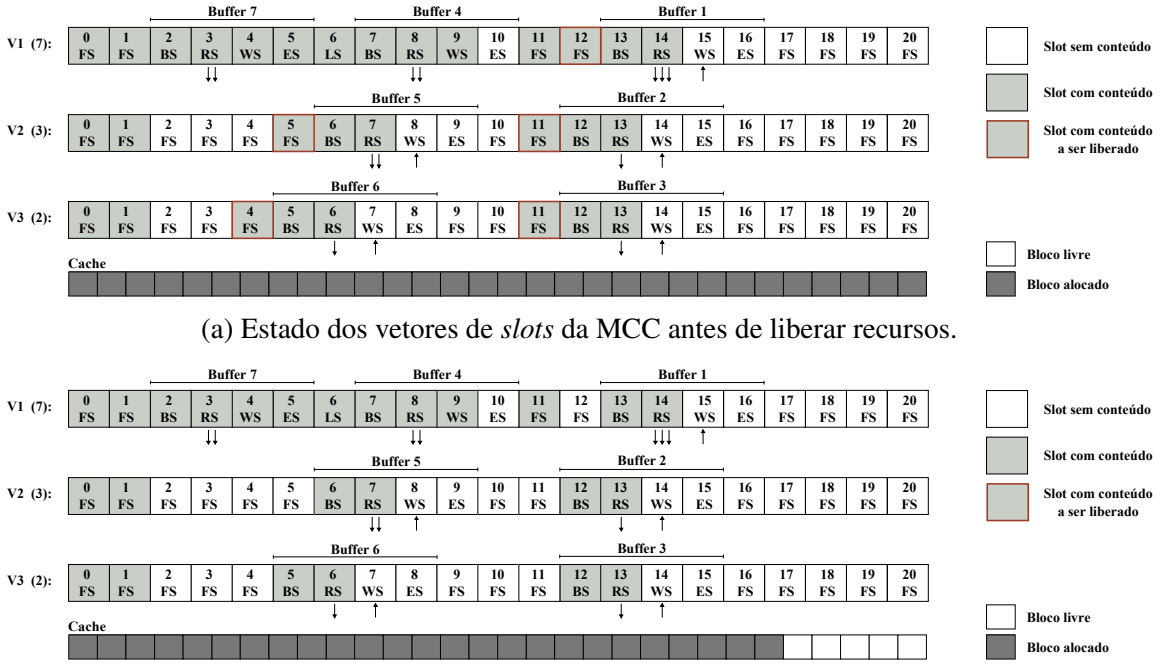


(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.



(b) Estado dos vetores de *slots* da MCC após liberar recursos.

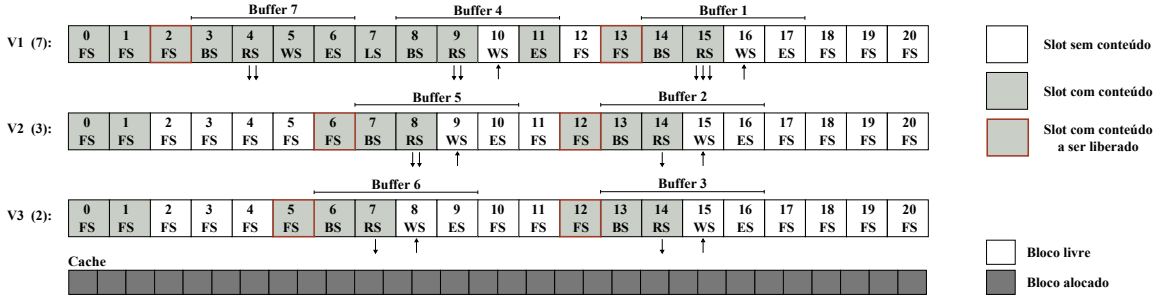
Figura 17 – Estado da MCC em t_{14}.



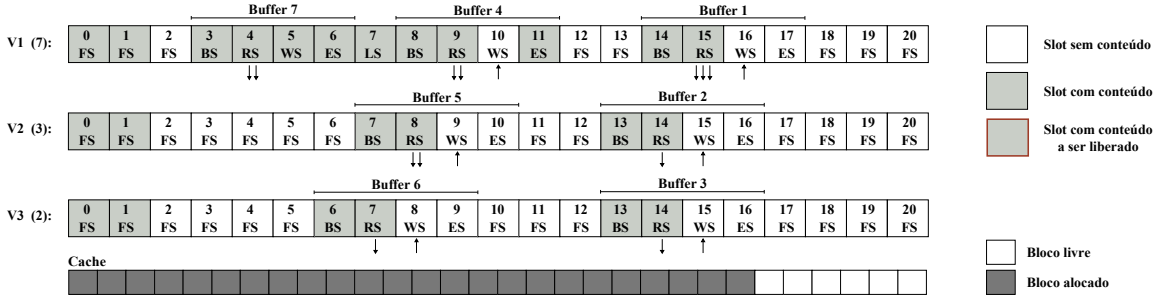
(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.

(b) Estado dos vetores de *slots* da MCC após liberar recursos.

Figura 18 – Estado da MCC em t_{15}.

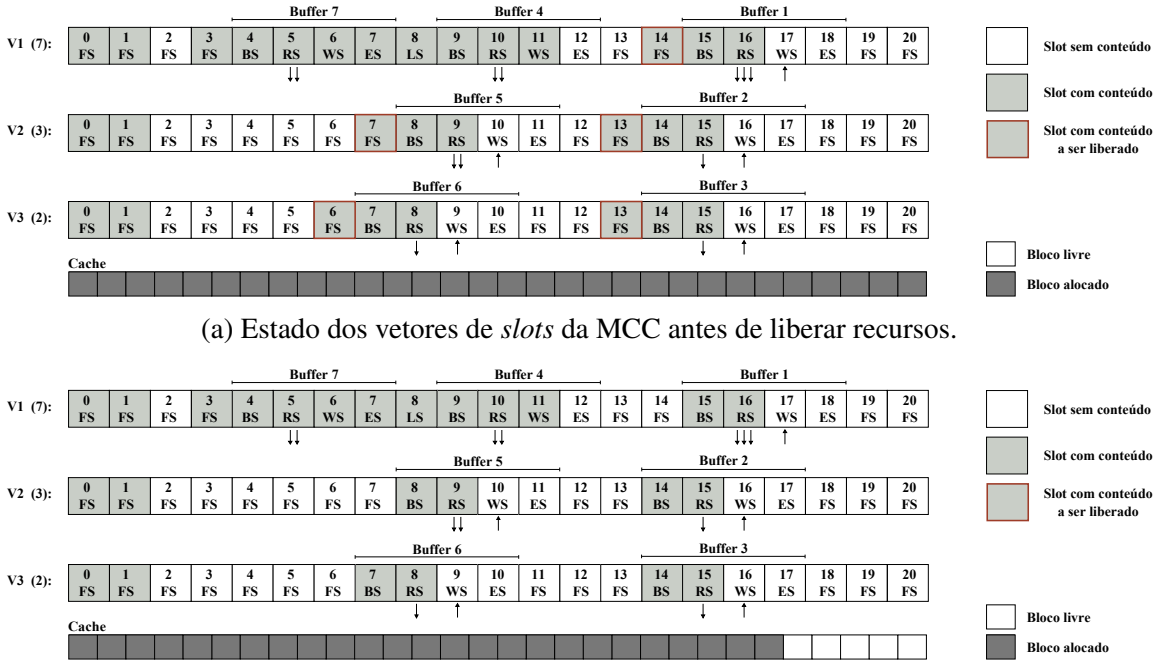


(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.



(b) Estado dos vetores de *slots* da MCC após liberar recursos.

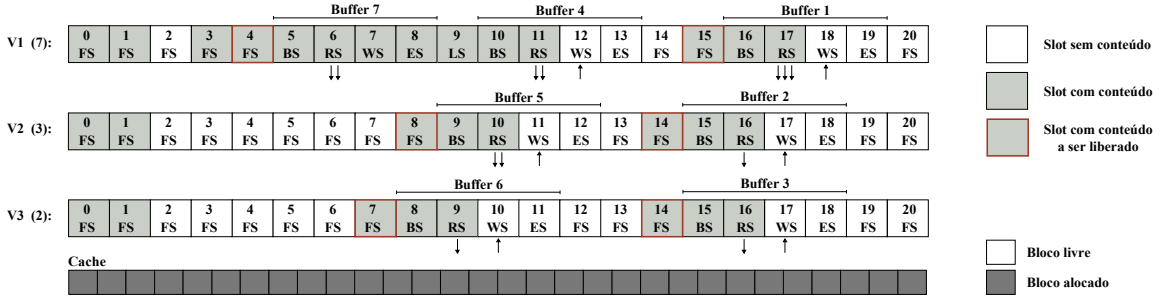
Figura 19 – Estado da MCC em t_{16}.



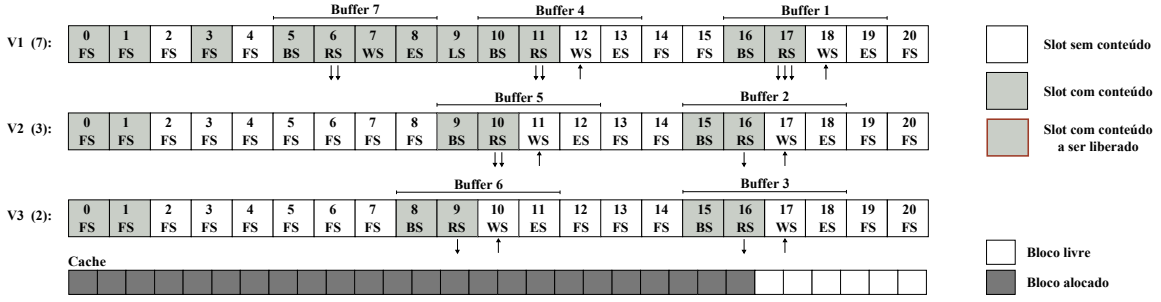
(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.

(b) Estado dos vetores de *slots* da MCC após liberar recursos.

Figura 20 – Estado da MCC em t_{17}.



(a) Estado dos vetores de *slots* da MCC antes de liberar recursos.



(b) Estado dos vetores de *slots* da MCC após liberar recursos.

3 TECNOLOGIAS DE ARMAZENAMENTO

O desempenho dos dispositivos de memória é vital para as aplicações de vídeo sob demanda. Devido às características deste tipo de aplicação é necessário que estes dispositivos de armazenamento sejam capazes de atender a demanda do sistema, caso contrário, se tornará o gargalo, ocasionando o bloqueio do sistema[8].

As principais considerações ao se escolher um dispositivo de memória são: vazão de leitura e escrita, capacidade de armazenamento, latência, custo e consumo energético. Como não há apenas uma memória que se sobressaia em todos os quesitos, torna-se relevante considerar a construção de um sistema hierarquizado, na perspectiva de fornecer um desempenho satisfatório a um custo razoável. Em geral, o nível superior da hierarquia, que está mais próximo da unidade de processamento, possui menor capacidade de armazenamento, menor latência de acesso e maior largura de banda. À medida que o nível se afasta do processador, há uma tendência de se utilizar dispositivos de memórias de menor custo por unidade de armazenamento e menor desempenho de leitura e escrita.

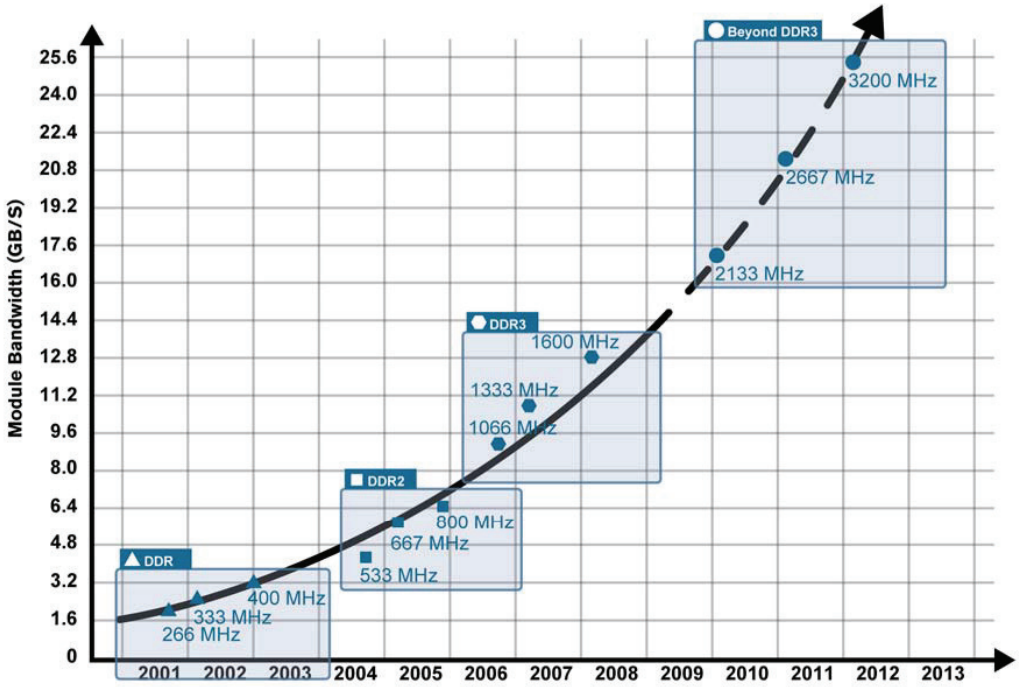
3.1 Memórias SDRAM

O dispositivo de memória *Synchronous Dynamic Random Access Memory* (SDRAM) é um tipo de memória volátil usualmente empregado em sistemas computacionais modernos. Esta tecnologia de memória se caracteriza pela alta vazão de dados, baixa latência de acesso e maior custo por gigabyte (GB) em relação à memória *Flash* e o disco magnético.

As memórias *Double Data Rate* – SDRAM (DDR – SDRAM) permitem transferir os dados tanto na borda de subida do *clock*, quanto na borda de descida, realizando-se duas transferências por ciclos de *clock*, obtendo-se assim, o dobro de desempenho em relação a *Single Data Rate* – SDRAM (SDR – SDRAM) com a mesma frequência de operação das memórias. Atualmente, entre as memórias SDRAM mais utilizadas, as que possuem maior desempenho são do tipo *Double Data Rate Type Three* – SDRAM (DDR3 – SDRAM). A Figura 21 resume a evolução das DDR – SDRAM na última década.

[8]Quando o sistema bloqueia, novas requisições de usuários não são atendidas por falta de recursos

Figura 21 – Tendência na vazão dos dispositivos de memória DDR – SDRAM



Fonte: (RAMBUS, 2009)

3.2 Memórias *Flash*

As unidades de estado sólido – amplamente conhecida como *Solid State Drive* (SSD) – baseadas em memória *Flash* incorporam uma tecnologia de armazenamento relativamente nova, cuja densidade tem dobrado a cada dois anos e deverá continuar nesta tendência até 2012 (RYU; KIM; RAMACHANDRAN, 2011). O fato deste tipo de meio de armazenamento ter ganhado maior popularidade é devido a redução no custo por GB de armazenamento, e que deve continuar a cair como visto na Figura 22.

Atualmente, a principal tecnologia de memória Flash é a NAND, que pode ser dividida em dois tipos: SLC (*Single-Level Cell*) e MLC (*Multi-Level Cell*). A SLC armazena um único bit por célula de memória. Diferentemente da tecnologia SLC, a MCL permite armazenar dois bits por célula, aumentando assim a capacidade de armazenamento do dispositivo em relação a tecnologia SLC. Este tipo de tecnologia é mais interessante quando a capacidade de armazenamento é mais importante que o desempenho.

Figura 22 – Tendência no custo por GB de memória *Flash*.

Fonte: (LEVENTHAL, 2008)

3.3 Discos magnéticos

Os discos magnéticos são responsáveis pela maior parte do armazenamento secundário nos sistemas computadorizados modernos. No entanto, os principais fatores que limitam o tempo de acesso aos dados no disco magnético estão relacionados com a natureza mecânica dos dispositivos, entre eles: tempos de posicionamento e a latência rotacional. O tempo de posicionamento é o tempo gasto até o mecanismo de leitura/gravação se posicionar na trilha onde o bloco se encontra. O tempo de latência rotacional é o tempo de espera até que o setor desejado se posicione sob o mecanismo de leitura/gravação.

Os discos magnéticos são caracterizados pela grande capacidade de armazenamento que são tipicamente de 500 GB a 3 TB, e possuem baixo custo por gigabyte de armazenamento. O tempo de acesso aleatório varia de 5 à 10 ms, que é a soma da latência rotacional com tempo de posicionamento. Diferentemente das memórias *Flash*, os discos podem sobrescrever os dados diretamente. Além disso, não têm um número limitado de gravações (WIKIPEDIA, 2011).

3.4 Comparação das tecnologias

Diferentemente dos discos magnéticos, os dispositivos SSD que utilizam memória Flash tem desempenho de escrita muito diferente do desempenho de leitura. Além do mais, o desempenho destas operações depende do tipo de acesso. Em caso de acesso sequencial, o SSD tem desempenho similar ao *Hard Disk Drive* (HDD). No entanto, em acesso de leitura randômico, os SSD baseados na tecnologia NAND possuem melhor desempenho que os HDD, pois estes dispositivos não possuem o atraso de posicionamento existente nos discos magnéticos.

Em contraste com os discos magnéticos, o SSD é um dispositivo eletrônico no qual nenhuma parte mecânica é necessária. Entretanto, somente um pequeno *overhead* é necessário para localizar o bloco solicitado que é muito inferior a latência de acesso dos discos magnéticos. Porém, estes dispositivos possuem baixo desempenho para acessos de escrita, pois é necessário armazenar o bloco que contem a palavra a ser modificada em um *buffer*, para então apagar o bloco e modificar a palavra desejada, e depois escrever novamente todo o bloco no modulo de memória Flash.

Outra desvantagem dos SSD é que estes possuem uma capacidade limitada de escrita. O número máximo de escritas por bloco para a tecnologia atual varia de 10.000 para MLC NAND até 100.000 no SLC NAND. Atualmente, podem-se encontrar dispositivos SSD NAND *Flash* MLC com capacidades maiores que 256 GB, com vazão de leitura sequencial superiores a 250 MBps e escrita sequencial próxima de 200 MBps.

De acordo com (RYU; KIM; RAMACHANDRAN, 2011), os parâmetros mais importantes em aplicações VoD são a escrita sequencial e a vazão de leitura aleatória. Conforme demonstrado, para leituras maiores que 512 KB, a vazão de leitura aleatória dos dispositivos SSD se aproxima da vazão de leitura sequencial. Esta característica faz com que os dispositivos SSD consigam suportar mais fluxos de vídeo paralelos em relação aos discos magnéticos.

4 MEMÓRIA COOPERATIVA COLAPSADA HIERARQUIZADA (MCCH)

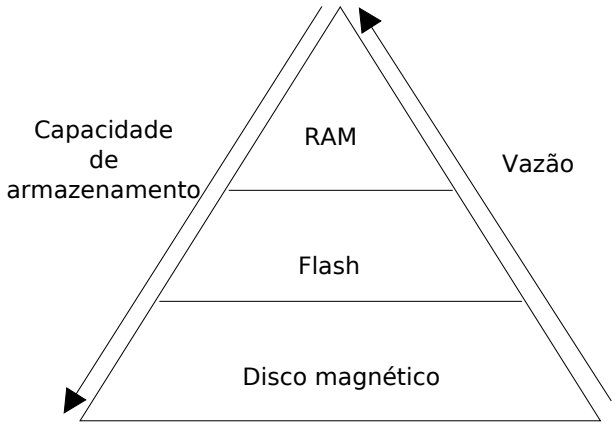
Em Ishikawa (2003), é introduzida a técnica escalável para *proxies* dinâmicos de vídeo baseada unicamente no uso da memória RAM com *cache* de segmentos de vídeo recebidos do servidor principal, devido a sua baixa latência de acesso e alta vazão de dados que são requisitos fundamentais em aplicações de mídia contínua. No entanto, a baixa capacidade de armazenamento desta tecnologia de memória, quando comparada com a tecnologia de discos magnéticos, tende a minimizar a taxa de acertos na *cache*, aumentando, assim, a quantidade de fluxos de vídeo entre *proxy* e servidor. Para minimizar esta desvantagem, a MCC implementa um gerenciador dinâmico de *cache* que leva em conta a relação temporal entre os segmentos de vídeo armazenados em sua *cache*, além da popularidade dos vídeos.

Basicamente, a MCC permite o encadeamento e desencadeamento do *buffer* dos clientes de acordo com a disponibilidade dos recursos de memória no *proxy*. Conforme explicado no capítulo anterior, esta habilidade permite que a *cache* seja mais flexível, pois por um lado a MCC aumenta a taxa de acertos através do encadeamento dos *buffers* dos clientes, que consequentemente irá diminuir o número de fluxos de vídeo entre o servidor e *proxy*, enquanto por outro lado, o encadeamento destes *buffers* irá penalizar o número de *slots* disponíveis para alocação de novos *buffers*. Quando a *cache* atingir sua capacidade máxima de armazenamento, os *slots* de ligação que unem esses *buffers* poderão ser removidos permitindo que novos clientes sejam atendidos pelo sistema com a penalidade no aumento do número de fluxos entre o *proxy* e servidor devido ao desencadeamento do(s) *buffer*(s) colapsado(s). Logo, a MCC obtém uma redução de até 80% no tráfego entre o *proxy* e servidor principal quando a capacidade da *cache* no *proxy* é 10% do acervo do servidor (ISHIKAWA, 2003). Uma forma de reduzir o tráfego entre o *proxy* e o servidor é por meio do aumento na capacidade de armazenamento do *proxy*. Contudo, o uso somente de memória RAM como *cache* de dados, em cenários onde o *proxy* seja capaz de armazenar grandes percentuais do acervo, se tornar inviável, pois o custo por *gigabyte* de memória RAM aumenta significativamente em *proxies* de 16 GB para 32 ou 48 GB, por exemplo.

Uma forma de minimizar a penalidade de desencadeamento dos *buffers* colapsados é utilizar níveis intermediários de memória no *proxy*, como visto na Figura 23, que atuarão como *caches* de

vítimas[9] para armazenar os *slots* de vídeo descartados pelo gerenciador da MCC. Basicamente, a *cache* de vítimas armazenaria os segmentos de vídeo que fossem descartados devido à falta de espaço na *cache* principal (RAM). Assim, é possível evitar que o gerenciador da *cache* principal recorra ao servidor para obter novamente os dados solicitados, aumentando a taxa de acerto no *proxy*, que consequentemente irá diminuir o tráfego entre o *proxy* e servidor, bem como a taxa de bloqueio. Todavia, apesar da agregação desta hierarquia de memória, doravante denominada Memória Cooperativa Colapsada Hierarquizada - MCCH, aumentar potencialmente a taxa de acertos no *proxy*, poderia degradar a *quality of service* (QoS) do sistema, já que estas tecnologias de memória possuem menor vazão e maior latência de acesso aos dados quando comparadas com a memória RAM. Portanto, existe um significativo desafio no que se refere à dinâmica do uso dos dispositivos de memória para que não ocorra a degradação do serviço.

Figura 23 – Hierarquia de memória de três níveis. O sentido da seta indica melhoria na característica do dispositivo.



4.1 Dinâmica de um sistema baseado em MCCH

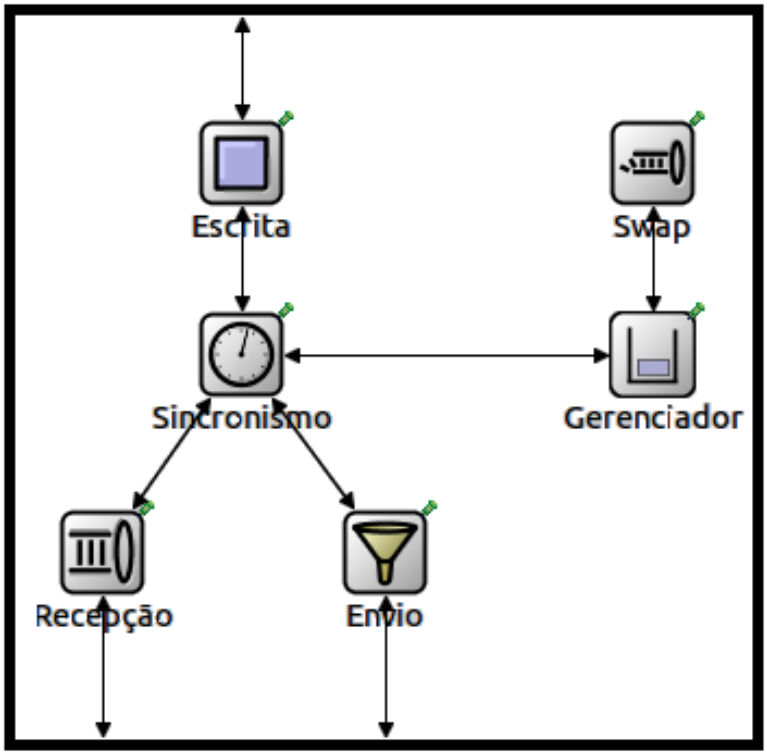
Uma forma de implementar a MCCH consiste na adaptação da MCC para a inclusão de um segundo nível de *cache* na hierarquia de memória do *proxy*. Sendo assim, à medida que os segmentos de vídeo são descartados da *cache* principal (memória RAM) serão armazenados no segundo nível da hierarquia, para que, posteriormente, quando requisitados, sejam novamente copiados para a *cache* principal. Neste sentido, a inclusão deste nível na hierarquia de memória do *proxy* atuará como *cache* de vítima da *cache* principal.

[9] *Cache* de vítimas foi proposta por (JOUPPI, 1990) para melhorar a taxa de falhas em *cache* diretamente mapeadas. Esta *cache* armazena os blocos descartados pela *cache* principal como resultado de uma substituição.

4.1.1 Descrição dos módulos da MCCH

O desenvolvimento da MCCH foi baseado no modelo descrito por Bragato (2006), porém, adaptado para a inclusão de um segundo nível de *cache* na hierarquia de memória do *proxy*. O sistema é constituído por seis módulos principais. Com exceção ao módulo de sincronização, os demais módulos possuem sua própria *thread* de execução. A Figura 24 apresenta os módulos do sistema.

Figura 24 – Descrição da arquitetura da MCCH.



Modulo de envio: é responsável pela leitura das unidades de vídeo mapeadas pelo *slot* RS e transmissão das mesmas aos clientes de acordo com a taxa de envio do vídeo.

Modulo de escrita: é responsável por requisitar ao servidor o conteúdo mapeado pelos *slots* WS que não estão presentes na *cache* do *proxy*, e armazenar os mesmos na *cache* principal. Para isso, é mantida uma lista de todos os *buffers* no sistema, assim, é possível obter os *slots* WS que terão que ter seu conteúdo preenchido para que no próximo giro sejam enviados aos clientes. Entretanto, antes de requisitar ao servidor, uma verificação é realizada a fim de verificar se o conteúdo mapeado pelo *slot* WS está presente no nível inferior da hierarquia, caso esteja, e haja disponibilidade de recursos de vazão de leitura no dispositivo de *cache* de vitima, o conteúdo é copiado para a *cache* de principal, caso contrário, será solicitado ao servidor.

Modulo de gerenciamento: é responsável por processar os pedidos em espera na fila de inclusão, girar os vetores de *slot*, liberar recursos da *cache* principal, criar uma lista dos *slots* que possivelmente terão o conteúdo liberados pelo algoritmo de substituição da MCC no próximo giro do sistema, de acordo com a disponibilidade de recursos de vazão de escrita da *cache* de vítimas, e por fim, liberar recursos da *cache* de vítimas para a cópia desta lista, caso necessário.

Módulo de recepção: é responsável por recepcionar os pacotes de controle enviados pelos clientes e incluí-los na fila de inclusão, para que posteriormente possam ser processados pela módulo de gerenciamento.

Módulo de *swap*: é responsável pela cópia dos *slots* previamente selecionados pelo módulo de gerenciamento para a *cache* de vítimas.

Módulo de sincronismo: é responsável pela sincronização das *threads* do sistema. A máquina de estados é formada por dois estados. No estado 1, as *threads* dos módulos de envio, escrita, recepção e *swap* executam paralelamente, enquanto a *thread* do módulo de gerenciamento fica parada. Neste estado, o sistema está enviando aos clientes os blocos de vídeo, recebendo os fluxos de vídeo do servidor, processando os pacotes de controle enviados pelos clientes e copiando a lista de *slots* que possivelmente serão vitimados da *cache* principal no próximo giro do sistema. No estado 2, somente as *thread* dos módulos de gerenciamento e recepção executam, enquanto as demais aguardam a finalização da execução da *thread* de gerenciamento. Neste estado a *thread* de gerenciamento deverá atualizar todas as estruturas de controle do sistema, processar as requisições dos clientes, criar a lista de *slots* a serem copiados pela *thread* do módulo de *swap*, e, por fim, liberar recursos da *cache* principal e *cache* de vítimas caso seja necessário. Portanto, a execução desta *thread* deverá ser o mais eficiente possível de modo a não influenciar na temporização do sistema, caso contrário, poderá causar a degradação da QoS do sistema.

4.1.2 Adaptação para MCCH

Como a execução do módulo de gerenciamento é síncrona, ocorrendo periodicamente a cada T segundos (onde T é o tamanho do slots de vídeo em segundos), a execução da *thread* deste módulo deverá ocorrer o mais rápido possível, de modo a não influenciar nos requisitos de tempo da aplicação. Portanto, a cópia dos slots a serem liberados não poderá ser realizada durante este processo. Logo, uma lista de *slots* que possívelmente serão liberados na próxima execução (após T segundos) é criada, para que, posteriormente, durante o estado 1 do sistema, sejam copiadas pelo módulo de *swap*. Para isso, a MCCH deve calcular os recursos de armazenamento que serão necessários para a próxima liberação, baseando-se na disponibilidade de recursos atual do sistema.

A quantidade de recursos a serem liberados pela MCCH dependerá da quantidade de recursos de armazenamento disponível na *cache* principal, bem como da quantidade de *slots* WS que terão que alocar recursos para armazenar os fluxos de vídeo oriundos do servidor. De posse desses valores, uma lista de *slots* é criada a partir dos *slots* que possivelmente terão seus recursos desalocados no próximo giro. Entretanto, talvez nem todos estes *slots* possam ser copiados devido a limitação na vazão de escrita do dispositivo de memória usado como *cache* de vítimas. Nestes casos, a MCCH prioriza a cópia das sequências de LS de maior custo, pertencentes aos vídeos de maior popularidade, em detrimento aos menos populares, e, posteriormente, a cópia dos *slots* FS. Neste caso, além de usar a popularidade dos vídeos, considera, como critério adicional, a posição do *slot* no vetor de *slots*, de modo a priorizar aqueles que estão mais no início do vídeo.

Porém, os *slots* copiados para a *cache* de vítimas poderão não ser exatamente os mesmo a serem liberados pelo algoritmo de substituição da *cache* principal utilizado pela MCC.Isto decorre do processo de inclusão e exclusão de *buffers* no sistema para vídeos que foram previamente copiados, o qual altera a prioridade dos mesmos. Para contornar este efeito, o algoritmo de substituição da *cache* principal foi modificado para inicialmente priorizar os slots que possuem o conteúdo duplicado na hierarquia de memória do *proxy*, baseando-se nos mesmos critérios do algoritmo originalmente descrito por Ishikawa (2003).Contudo, a prioridade dos *slots* LS não é alterada para FS, quando liberados seus recursos de armazenamento.

4.1.3 Política de substituição da *cache* de vítimas

A política usada pela MCCH para substituição do conteúdo na *cache* de vítimas é a mesma utilizada pela MCC, conforme descrito no Capítulo 2.

4.2 Modelo analítico

A avaliação de desempenho de um sistema é um processo complexo que envolve análises e decisões. Para realizar estas análises é necessário definir quais são as métricas de desempenho adequadas a serem avaliadas. Uma vez definidas estas métricas, é necessário determinar as principais variáveis que influenciam em seu comportamento e a relação entre elas para então criar o sistema de equações.

As principais métricas de desempenho a serem quantificadas são:

- Vazão do sistema: é a capacidade máxima de envio de fluxos de vídeo pelo *proxy*, deter-

minada pelo mínimo entre a vazão do enlace de saída do *proxy* e a vazão sustentada de leitura do sistema de memória no estado estacionário;

- Taxa de bloqueio: é a relação entre a quantidade de requisições não atendidas e o número total de requisições solicitadas pelos clientes ao sistema.

Seguem na Tabela 2 as principais variáveis do sistema.

Tabela 2 – Principais variáveis do sistema

Variável	Descrição
λ	Taxa média de requisições de vídeo por segundo
V_{rate}	Taxa de consumo do vídeo em Mbps
V_{length}	Tempo de reprodução do vídeo em segundos
S_{length}	Tamanho do *slot* de vídeo em segundos
$CA_{dispositivo}$	Capacidade máxima de armazenamento do dispositivo de memória em Mb
$CV_{dispositivo}$	Capacidade máxima de vazão de leitura do dispositivo de memória em Mbps

De acordo com os cenários focados a serem avaliados neste trabalho, as principais variáveis que influenciarão o desempenho do sistema serão $CA_{dispositivo}$ e $CV_{dispositivo}$, sendo um fator limitante na escalabilidade do sistema, considerando a hipótese de que a vazão da rede não seja um gargalo.

4.2.1 Considerações preliminares

Para facilitar a implementação das equações do modelo analítico simplificado será necessário primeiramente descrever algumas restrições no projeto quanto aos clientes, MCC e sistema.

- Em relação aos clientes: Embora os clientes possam solicitar a exibição do vídeo a partir de um ponto aleatório e realizar operações de pausa, avanço, retrocesso, ou até mesmo a desistência de assistir todo o vídeo, assume-se que os clientes solicitam a reprodução do vídeo do início e permanecem no sistema até o final da reprodução, e não realizam nenhuma operação de pausa, avanço ou retrocesso. Impondo estas restrições, o período de tempo que os recursos deverão permanecer alocados para um cliente é igual ao tempo de envio do vídeo. Além disso, também se assume que os clientes possuem largura de banda de descida suficiente para receber o fluxo de vídeo de acordo com a taxa de envio do sistema, que é igual a taxa de exibição do vídeo, sendo estes codificados no modo CBR (*Constant Bit Rate*);

- Em relação à MCC: Considera-se que o *buffer* do cliente no vetor de *slot* é composto apenas pelos *slot*s RS e WS, ao invés de quatro *slots*, como descrito por Ishikawa (2003). Isto é válido, pois é suficiente para que a MCC possa interagir com o servidor a tempo de controlar o preenchimento do *slot* de escrita WS à medida que os dados contidos no *slot* de envio RS são enviados aos clientes, sem a necessidade do uso dos *slots* BS e ES. Por motivos de simplificação, a política adotada para substituição dos segmentos de vídeo é a LFU (*Least Frequently Used*) ao invés da política adotada pela MCC.

- Em relação ao sistema: Assume-se que todos os clientes solicitam a reprodução do mesmo vídeo (único vídeo armazenado no servidor principal) e que a capacidade de armazenamento do *proxy* é igual ao tamanho deste vídeo. Além disso, assume-se que o sentido do fluxo de dados entre os dispositivos de memória é unidirecional, ou seja, quando ocorrer o descarte dos segmentos do nível superior da hierarquia, o mesmo não poderá retornar. Como também, será adotado o modelo *unicast* para a transmissão dos fluxos de vídeo entre servido/*proxy* e entre o *proxy*/clientes , uma vez que o modelo *multicast* não é suportado atualmente pela Internet.

4.2.2 Desenvolvimento do modelo analítico

Nesta seção são descritas analiticamente as equações necessárias para o desenvolvimento do modelo de vazão do sistema para acessos de leitura no *proxy* e taxa de bloqueio do sistema. Cabe salientar que as equações do modelo descrito foram modeladas somente para dois níveis de armazenamento, que incluem a memória RAM, como *cache* principal, e a memória *Flash*, como *cache* de vítimas. No entanto, a inclusão do disco magnético como terceiro nível de armazenamento não será realizado neste momento, pelos motivos que serão discutidos a frente.

Para o desenvolvimento do modelo foca-se a análise em um sistema contendo apenas um arquivo de vídeo de duração V_{length} segundos, que deve ser consumido a uma taxa de V_{rate} Mbps. Além disso, assume-se que os clientes solicitam a exibição do vídeo a uma taxa média de λ requisições por segundo, seguindo uma distribuição de Poisson. Assim, o intervalo de tempo entre chegadas é exponencialmente distribuído com média de $\delta = \frac{1}{\lambda}$ segundos.

4.2.2.1 Modelo de inclusão no sistema

Nesta seção é descrito o modelo que representa o tempo no qual o controle de admissão deverá incluir clientes no sistema. Para isso deve-se saber que todas as requisições solicitadas ao *proxy* são inicialmente colocadas em um fila de inclusão. Sua função é armazenar as requisições

para que o sistema possa, posteriormente, em tempos específicos, acessá-las, processá-las e então incluí-las ou não no sistema de acordo com a disponibilidade de recursos.

Para determinar os instantes de tempo em que o sistema irá acessar esta fila, se faz necessário determinar as principais varáveis que influenciam em seu comportamento. Estas variáveis estão relacionadas à velocidade de giro do vetor de *slots*, instante da primeira inclusão no sistema e do tamanho de cada *slot* de vídeo do vetor de *slots*.

Seja δ, o intervalo de tempo entre as chegadas dos clientes ao sistema, em segundos. Definido este intervalo, o total de clientes requisitando a exibição do vídeo ao sistema é calculado de acordo com a seguinte equação:

$$R_{total}(\tau) = \left\lfloor \frac{1}{\delta} * \tau \right\rfloor, \quad \tau \geq 0 \tag{4.1}$$

onde τ representa a variável tempo em segundos.

De acordo com a Equação 4.1, o primeiro cliente a solicitar o vídeo entra no sistema no tempo $\tau = \delta$ segundos. Como este é o primeiro cliente, seria possível incluí-lo diretamente no sistema, mas ao invés disso, é aguardado um período de tempo T para absorver a possível chegada de um número maior de clientes que desejem receber do primeiro *slot* de vídeo. Em função disso, a primeira inclusão ocorrerá em $\tau_0 = \delta + T$. Isso é vantajoso, pois, se um vídeo foi solicitado pela primeira vez, existe uma tendência de que novos clientes solicitem a exibição do mesmo vídeo em um curto intervalo de tempo, permitindo, assim, que uma quantidade maior de clientes compartilhe o mesmo *buffer*. No entanto, a desvantagem é que o primeiro cliente terá que esperar um tempo maior para começar a receber o fluxo de vídeo. Porém, este tempo de espera está na ordem de 1 a 10 segundos, um tempo relativamente pequeno quando comparado com a duração do vídeo.

Note que, o período de inclusão T dos clientes no sistema deve ser proporcional ao tempo de envio de cada *slot* de vídeo. Como o vetor de *slots* gira sincronamente na mesma velocidade de exibição do vídeo, a taxa de envio do vídeo aos clientes é igual a taxa de exibição do vídeo. Logo, o tempo de envio de cada *slot* é igual a porção temporal de vídeo associada a este *slot*. Já que o vetor de *slots* consiste em *slots* de tamanho fixo de S_{length} segundos, o tempo de envio para qualquer *slot* deste vetor é o mesmo. Desta forma, a retirada de clientes da fila e inclusão no sistema, caso haja a disponibilidade de recursos, ocorrerá periodicamente a cada $T = S_{length}$ segundos a partir da primeira inclusão.

Definido o instante da primeira inclusão (τ_0) e o período de inclusão T, pode-se então determinar os instantes de tempo no qual o sistema irá retirar os clientes da fila e processar, caso

existam clientes, de acordo com a seguinte equação:

$$\delta + T * t \quad \text{, se} \quad t \geq 1 \tag{4.2}$$

onde $t \in \mathbb{Z}$ é a quantidade de giro de período T no vetor de *slot*. Cabe observar que a primeira inclusão ocorrerá em t=1, a segunda em t=2, e assim por diante.

Para facilitar o desenvolvimento das equações, a partir deste momento, são definidas todas as equações em função da variável t ao invés de τ. Para determinar a quantidade de giros de período T, após a chegada do primeiro cliente no sistema em função do tempo, é possível usar a seguinte equação:

$$NG(\tau) = \left\lfloor \frac{\tau - \delta}{T} \right\rfloor, \tau \geq \delta \tag{4.3}$$

Com base nas Equações 4.1 e 4.2 é possível então determinar o número médio total de requisições dos clientes que foram retiradas da fila de inclusão e processados pelo sistema a partir da primeira inclusão, ou seja, a partir de t=1. Esta equação inclui todas as requisições aceitas e bloqueadas pelo sistema, como descrito na equação abaixo:

$$NC_{total}(t) = \begin{cases} 0 & \text{, se } t < 1 \\ R_{total}(\delta + T * t) & \text{, se } t \geq 1 \end{cases} \tag{4.4}$$

A Equação 4.4 fornece o acúmulo de requisições solicitadas ao sistema. No entanto, o principal fator a determinar é o número médio de requisições que serão retiradas da fila a cada giro. Para isso, subtrai-se a quantidade total de requisições no tempo de giro anterior de inclusão $NC_{total}(t-1)$ da quantidade total de clientes no tempo de giro atual $NC_{total}(t)$ como visto na equação abaixo:

$$NC_{fila}(t) = NC_{total}(t) - NC_{total}(t-1), \ t \geq 0 \tag{4.5}$$

Com o uso da Equação 4.5 pode-se identificar a quantidade de clientes que serão incluídos no sistema a cada giro do vetor de *slot*. Contudo, para determinar o total de requisições aceitas, é necessário especificar a equação que modele a disponibilidade de recursos no sistema, conforme descrito na seção 4.2.2.2.

4.2.2.2 Modelo de controle de admissão de clientes

Nesta seção é descrito o modelo matemático que representa o comportamento do controle de admissão de clientes. Toda vez que o sistema for incluir um novo cliente, é necessário que o sistema assegure que os outros pedidos em atendimento não sejam prejudicados. Desta forma, no momento de incluir um cliente, é necessário garantir que haverá recursos disponíveis

durante o período de permanência do cliente em atendimento no sistema. Caso esta garantia seja satisfeita, o novo cliente (ou conjunto de clientes que esteja na fila de inclusão) será admitido, caso contrário, será bloqueado e removido da fila de inclusão.

Para este modelo, os recursos a serem considerados são somente recursos relacionados a capacidade de armazenamento (CA) e a capacidade de vazão de leitura (CV)[10] nos dispositivos de memória RAM e *Flash*.

Algumas considerações devem ser enfatizadas em relação a restrições no desenvolvimento do modelo. Estas restrições ou limitações do modelo serão descritas a medida que as equações forem modeladas. Como o sistema, modelado inicialmente, é composto por dois níveis de memória, a capacidade de armazenamento do *proxy* é definida pela soma das capacidades de armazenamento dos dispositivos de memória RAM e *Flash*:

$$CA_{proxy} = CA_{ram} + CA_{flash} \tag{4.6}$$

onde CA_{ram} e CA_{flash} representam a capacidade de armazenamento da memória RAM e *Flash*, respectivamente.

No entanto, no modelo descrito, as equações são válidas somente para cenários onde a capacidade de armazenamento da memória RAM e *Flash* são inferiores ao tamanho do vídeo, mas a soma das capacidades de armazenamento dos dispositivos de memória (CA_{proxy}) é igual ao tamanho do vídeo[11]. Considerando esta restrição, a capacidade de armazenamento de cada nível da hierarquia é calculada da seguinte forma:

$$CA_{ram} = \alpha * V_{length} * V_{rate} \tag{4.7}$$

$$CA_{flash} = \beta * V_{length} * V_{rate} \tag{4.8}$$

tal que,

$$\alpha + \beta = 1$$

onde α e β representam percentagens do tamanho do vídeo.

Como a capacidade de armazenamento do sistema é igual ao acervo do servidor, que contém apenas um vídeo, o *proxy* sempre será capaz de cachear todo o vídeo. Devido a isso, a largura de banda consumida pelo sistema para acessos de escrita no dispositivo de memória RAM para fluxos de vídeo oriundos do servidor para *proxy*, não será um fator limitante para a escalabilidade do sistema, uma vez que existirá apenas um fluxo de vídeo, que permanecerá durante o tempo de

[10] Vazão de escrita será modelada posteriormente

[11] Cenário que foi adotado como premissa para a análise preliminar

envio do vídeo, ou seja, por um período igual a V_{length}. Além disso, o sistema não será bloqueado devido à indisponibilidade de recursos de armazenamento no *proxy*, pois no pior cenário, onde todo o conteúdo da *cache* estiver reservado por *buffers*, não será possível o descarte de nenhuma unidade de vídeo, mas sempre haverá a disponibilidade de adicionar uma nova requisição a um *buffer* já existente, visto que todo o conteúdo do vídeo poderá ser cacheado no *proxy*.

O mesmo não é válido quanto a disponibilidade de recursos de largura de banda para acessos de leitura no *proxy*, uma vez que a capacidade de vazão do sistema estará limitada ao somatório das vazões de leitura nos dispositivos de memória usados. Sendo assim, para cada requisição solicitada ao sistema, é necessário garantir que este recurso esteja disponível durante toda a permanência do cliente em atendimento. Esta garantia deve ser satisfeita uma vez que na falta deste recurso, o sistema deverá remover ou interromper o envio do fluxo de vídeo ao cliente.

Portanto, o único recurso que deverá ser levado em conta pelo controle de admissão de cliente é a capacidade de vazão de leitura dos dispositivos de memória. Este recurso restringirá o número de fluxos de vídeo permitidos por cada dispositivo de memória, e pode ser calculado da seguinte maneira:

$$MF_{flash} = \left\lfloor \frac{CV_{flash,leitura}}{V_{rate}} \right\rfloor \tag{4.9}$$

$$MF_{ram} = \left\lfloor \frac{CV_{ram,leitura}}{V_{rate}} \right\rfloor \tag{4.10}$$

tal que,

$$CV_{flash} < CV_{ram}{}^{12}$$

onde MF_{flash} e MF_{ram} representam o número máximo de fluxos de vídeo permitidos para acessos de leitura nos dispositivos de memória *Flash* e RAM, respectivamente.

Definidas as Equações 4.9 e 4.10, o número máximo de fluxos de vídeo aceitos pelo sistema para acessos simultâneos de leitura no *proxy* é calculado de acordo com a seguinte equação:

$$MF_{sistema} = MF_{ram} + MF_{flash} \tag{4.11}$$

No entanto, dependendo de como o sistema for projetado, talvez não seja possível somar os fluxos, como descrito na Equação 4.11. Desta forma, o valor efetivo $\left(MF_{eft}\right)$ poderá variar de $MF_{flash} \leq MF_{eft} \leq MF_{sistema}$, sendo que, no pior caso, estará limitado ao número de fluxos para acessos de leitura na memória *Flash*.

Cabe ressaltar que a capacidade de vazão de leitura da memória RAM não é um gargalo para atender a demanda requisitada ao sistema. Assim sendo, o único recurso que será levado em

[12] Restrição tecnológica, conforme o levantamento realizado e sintetizado no capítulo anterior

conta pelo controle de admissão dos clientes é a capacidade de vazão de leitura do dispositivo de memória *Flash*.

Quando a *cache* principal atingir sua capacidade máxima de armazenamento, em $t = \left\lceil \frac{\alpha * V_{length}}{S_{length}} \right\rceil$, será necessário descartar unidades de vídeo para que o fluxo de vídeo oriundo do servidor possa ser armazenado na *cache* principal. A medida que as unidades de vídeo são descartadas, as mesmas são armazenadas na *cache* de vítimas, para que, futuramente, quando referenciadas, o gerenciador da *cache* possa acessá-las sem a necessidade de solicitar ao servidor, economizando-se largura de banda no enlace entre servidor/*proxy*.

Contudo, a política de substituição irá determinar quais unidades de vídeo estão presentes na *cache* principal e quais estão na *cache* de vítimas. Para este modelo foi adotada a política LFU, que irá descartar as unidades de vídeo menos frequentemente acessadas. Esta política não é a mais adequada para este tipo de aplicação, mas sua escolha se deve a simplicidade de modelar o sistema utilizando-a, associado ao fato de que este tipo de aplicação possui alta localidade espacial, ou seja, se o *k-ésimo slot* de vídeo no vetor de *slots* é referenciado, então o *k+1-ésimo slot* será referenciado após o período de um giro, além de possuir alta localidade temporal, pois se um vídeo é solicitado, existe uma tendência que novas requisições para a exibição deste mesmo vídeo sejam feitas em um intervalo de tempo curto.

Como este tipo de aplicação permite que os clientes possam ter acesso aleatório ao *stream* e inicializar a exibição do conteúdo a partir de qualquer ponto, se torna difícil determinar a frequência de acessos aos dados contidos na *cache*, logo, parte-se da premissa de que todos os clientes solicitam a exibição sempre a partir do início e permanecem até o final da exibição, implicando em acesso uniforme aos *slots* de vídeo. Impondo esta restrição, quando a *cache* atingir sua capacidade máxima de armazenamento, a tendência é descartar as unidades de vídeo pertencentes ao final do vídeo, presentes na *cache* principal e que não estejam reservadas, mantendo assim, o prefixo do vídeo.

Embora no modelo atual o descarte seja feito a partir dos *slots* mais ao final do vídeo, esta característica não influenciará no desempenho do sistema, pois as análises são feitas considerando um acervo com apenas um vídeo, não se beneficiando da popularidade dos vídeos. Contudo, em cenários onde existam vários vídeos sendo requisitados, com o uso da política LFU, a tendência é manter na *cache* principal os vídeos mais populares (com maior frequência de acesso) e descartar as unidades de vídeo dos vídeos menos populares, com menor frequência de acesso, para a *cache* de vítimas. Desta forma, a tendência é manter os vídeos com maior demanda no dispositivo de memória de maior vazão e os vídeos com menor demanda no dispositivo de menor vazão.

No entanto, devido à restrição imposta no projeto do sistema, referente ao sentido do fluxo

de dados entre os dispositivos de memória ser unidirecional, permitindo fluxos somente no sentido da RAM para a *Flash*, quando as unidades de vídeo descartadas forem referenciadas por um *buffer*, o gerenciador da *cache* será forçado a atender estas requisições pela *cache* de vítimas. Desta forma, o número máximo de clientes em atendimento simultâneo pela *cache* de vítimas não poderá ultrapassar o valor MF_{flash} pois, caso contrário, interrupções no envio do vídeo ocorrerão.

Considerando-se os parágrafos anteriores, quando a *cache* principal atingir sua capacidade máxima de armazenamento, o sistema colocará em prática a política de descarte, vitimando os *slots* de vídeo menos frequentemente acessados. Estes *slots* são aqueles que estão entre os dois primeiros *buffers* incluídos no sistema. Desta forma, todas as requisições que entrarem no sistema a partir de $TE \geq 2$ serão forçados a degradar ao nível inferior da hierarquia, para receber o fluxo de vídeo. Desta maneira, se *k*-ésimo *slot* for descartado para armazenar o fluxo de vídeo que alimenta o primeiro *buffer* incluído no sistema, no próximo giro, o *slot* a ser descartado será $(k+1)$-ésimo. Seguindo esta analogia, após todo o vídeo ser armazenado no *proxy*, as unidades de vídeo indexadas pelo vetor de *slots* entre o *k*-ésimo *slot* e $(k+\left\lfloor \frac{\beta * V_{length}}{S_{length}} \right\rfloor - 1)$-ésimo *slot*, estarão presentes na *cache* de vítimas, enquanto os demais *slots* estarão na *cache* principal. Desta forma, o tempo de permanência dos clientes na *cache* de vítimas, a partir do momento de requisitar o *slot* descartado até consumirem o último *slot* de vídeo, será $TA_{flash} = \left\lfloor \frac{\beta * V_{length}}{S_{length}} \right\rfloor$.

Entretanto, em alguns casos, não será possível descartar nenhum *slot* de vídeo da *cache* principal. Isto ocorre em cenários onde a distância temporal entre as requisições dos clientes é menor que duas vezes o período de inclusão[13], $\delta \leq 2 * T$. Ou seja, todo o conteúdo da *cache* principal estará reservado por *buffers* quando a *cache* principal atingir sua capacidade máxima de armazenamento, causando o bloqueio do sistema devido a indisponibilidade de recursos de armazenamento. No entanto, é possível usar o segundo nível da hierarquia para atender as requisições, para isso, duas abordagens podem ser adotadas.

Na primeira, quando a *cache* principal atingir sua capacidade máxima de armazenamento, o fluxo de vídeo que alimenta o *buffer* incluído no sistema, em $t = 1$, passaria a ser armazenado na *cache* de vítimas. Desta forma, o sistema seria forçado a atender as requisições dos clientes por um dispositivo de menor vazão, causando a degradação do desempenho do sistema.

A segunda abordagem tenderia a se beneficiar da capacidade de vazão de leitura da *cache* principal para atender as requisições dos clientes que foram incluídas no sistema a partir da primeira inclusão, em $t = 1$, até o momento da *cache* principal atingir sua capacidade máxima

[13]Cada *buffer* é formado por um *slot* de leitura e *slot* de escrita

de armazenamento, em $t = \left\lceil \frac{\alpha * V_{length}}{S_{length}} \right\rceil - 1$. Estas requisições seriam atendidas somente pela *cache* principal durante todo tempo de envio do vídeo, equivalente ao tempo de $\left\lceil \frac{V_{length}}{S_{length}} \right\rceil$, mesmo impondo a restrição no sentido dos fluxos entre os dispositivos de memória.

Para isso assume-se que, quando o sistema atingir sua capacidade máxima de armazenamento, existirá um *slot* de folga reservado ao primeiro *buffer* incluído no sistema. Isso é necessário para que o vetor de *slots* possa girar sem causar *deadlock* do sistema, permitindo que a porção temporal de vídeo indexada pelo primeiro *slot* no vetor de *slots* não esteja reservada no momento de sobrescrevê-la.

No entanto, como esta abordagem tem o mesmo comportamento de um *ring buffer*, a medida que o vetor de *slots* gira, as unidades de vídeo indexadas pelo primeiro *slot* de vídeo no vetor de *slots* serão, em sequência, copiadas para a *cache* de vítimas, para que as requisições a partir do tempo de entrada $TE = \left\lceil \frac{\alpha * V_{length}}{S_{length}} \right\rceil$, tempo do giro posterior em que a *cache* principal atingiu sua capacidade máxima de armazenamento, possa ser atendida pelo sistema. Sendo assim, quando todo o fluxo de vídeo for transferido do servidor ao *proxy*, o prefixo do vídeo estará cacheado na *cache* de vítimas e o sufixo na *cache* principal. Logo, o tempo que os recursos deverão permanecer alocados para cada cliente é igual ao tempo de atendimento na *cache* de vítimas, $TA_{flash} = \left\lfloor \frac{\beta * V_{length}}{S_{length}} \right\rfloor$. Portanto, para todos os clientes que forem incluídos no sistema após TE, o controle de admissão deverá garantir que o número de clientes em atendimento simultâneo na *cache* de vítimas não exceda o valor de MF_{flash}.

Desta maneira, se $\delta > 2 * T$, são constituídos cenários onde ocorrerá o descarte, sendo que o tempo no qual o sistema estará restrito a vazão da *cache* de vítimas será a partir do tempo de inclusão $TE = 2$. Nos casos em que $\delta \leq 2 * T$, são gerados cenários onde todo o conteúdo da *cache* principal estará reservado, sendo $TE = \left\lceil \frac{\alpha * V_{length}}{S_{length}} \right\rceil$. Cabe ressaltar que o desempenho desta abordagem dara-se-á baseando-se na idéia de que a demanda pelo sistema é exigida ao máximo somente em determinados horários do dia, onde há um pico de audiência. Logo, é possível que a *cache* principal supra uma quantidade maior de clientes até o momento de atingir seu limite de armazenamento, reduzindo a taxa de bloqueio do sistema, mesmo com a imposição da restrição no sentido do fluxo de dados entre os dispositivos de memória.

Definidos o valores de TE e TA_{flash}, além das Equações 4.5 e 4.9, é possível determinar o número médio total de clientes aceitos pelo sistema que serão atendidos pela *cache* principal e

cache de vítimas para o envio do fluxo de vídeo, $NC_{aceitos}(t)$, como visto na equação abaixo:

$$NC_{aceitos}(t) = \begin{cases} 0 & , \text{se } 0 \leq t < TE \\ min(NC_{aceitos}(t-1) + NC_{fila}(t), MF_{flash} + NC_{flash}(t)) & , \text{se } t \geq TE \end{cases} \tag{4.12}$$

onde $NC_{flash}(t)$ é o total de clientes que saem da *cache* de vítimas, liberando recursos de vazão de leitura no dispositivo podendo ser definido da seguinte forma:

$$NC_{flash}(t) = \begin{cases} 0 & , \text{se } 0 \leq t < TE + TA_{flash} \\ NC_{aceitos}(t)(t - TA_{flash}) & , \text{se } t \geq TE + TA_{flash} \end{cases} \tag{4.13}$$

Desta forma, se no instante de inclusão t, o valor da expressão $NC_{aceitos}(t-1) + NC_{Fila}(t) \leq MF_{flash} + NC_{flash}(t)$ for satisfeita, o sistema poderá admitir os clientes, já que o número médio total de clientes aceitos no instante anterior de inclusão, $NC_{aceitos}(t-1)$, somado ao número médio de clientes retirados da fila, $NC_{Fila}(t)$, é inferior ou igual ao número de fluxos, MF_{flash}, somado ao número médio total de clientes que saíram de atendimento da *Flash* , $NC_{flash}(t)$, liberando recursos de vazão da memória *Flash*. Logo, é possível garantir que o número médio de clientes em atendimento simultaneamente na *cache* de vítima não exceda o valor máximo.

No entanto, se $NC_{aceitos}(t-1) + NC_{Fila}(t) > MF_{flash} + NC_{flash}(t)$, o sistema não poderá atender à todas as requisições retiradas da fila de inclusão, ocorrendo o bloqueio do sistema até que novos recursos sejam liberados pela *cache* de vítimas. Cabe salientar que a Equação 4.12 representa o acúmulo de clientes aceitos no sistema, que será usado posteriormente para calcular a taxa de requisições bloqueadas pelo sistema.

As Equações 4.12 e 4.13 são válidas para os casos onde os clientes que entram no sistema, estarão restritos a vazão da *Flash*. No entanto, os clientes que entram antes do tempo TE serão atendidos somente pela *cache* principal, como descrito anteriormente. Portanto, pode-se equacionar o total de clientes aceitos pelo sistema para serem atendidos somente pela *cache* principal, o qual pode ser calculado da seguinte forma:

$$NC_{aceitos,ram}(t) = \begin{cases} NC_{total}(t) & , \text{se } 0 \leq t < TE \\ NC_{total}(TE-1) & , \text{se } t \geq TE \end{cases} \tag{4.14}$$

Modeladas as Equações 4.12 e 4.14, pode-se então determinar as métricas de vazão do sistema para acessos de leitura, $BC(t)$, e taxa de bloqueio, $TB(t)$, que serão descritos nas seções seguintes.

4.2.2.3 Modelo de largura de banda

Nesta seção será modelada a largura de banda consumida pelo sistema para acessos de leitura no *proxy*. A largura de banda consumida pelo sistema dependerá do número médio de cliente em atendimento no sistema $AT(t)$ e da taxa de consumo do vídeo (V_{rate}) podendo ser calculada da seguinte forma:

$$BC(t) = AT(t) * V_{rate} \tag{4.15}$$

O valor de $AT(t)$, pode ser calculado da seguinte forma:

$$AT(t) = Entra(t) - Sai(t) \tag{4.16}$$

Onde, $Entra(t)$ representa o total de clientes aceitos pelo sistema e $Sai(t)$ representa o total de clientes aceitos pelo sistema que receberam todo o conteúdo do vídeo.

O total de clientes que entram no sistema $(Entra(t))$ é dado pela soma das Equações 4.12 e 4.14, como visto na equação abaixo:

$$Entra(t) = NC_{aceitos,ram}(t) + NC_{aceitos}(t) \tag{4.17}$$

Como os clientes permanecem no sistema durante o período de $\left\lceil \frac{V_{length}}{S_{length}} \right\rceil$, o total de clientes que saem do sistema pode ser calculado da seguinte forma:

$$Sai(t) = Sai_{ram+flash}(t) + Sai_{ram}(t) \tag{4.18}$$

onde $Sai_{ram+flash}(t)$ é o total de clientes que foram atendidos por ambos os níveis da hierarquia, e pode ser calculado da seguinte forma:

$$Sai_{ram+flash}(t) = \begin{cases} 0 & \text{, se } 0 \leq t < TE + \left\lceil \frac{V_{length}}{S_{length}} \right\rceil \\ NC_{aceitos}(t - \left\lceil \frac{V_{length}}{S_{length}} \right\rceil) & \text{, se } t \geq TE + \left\lceil \frac{V_{length}}{S_{length}} \right\rceil \end{cases} \tag{4.19}$$

e $Sai_{ram}(t)$ é o total de clientes que foram atendidos somente pela cache principal, e pode ser calculado da seguinte maneira:

$$Sai_{ram}(t) = \begin{cases} 0 & \text{, se } 0 \leq t < 1 + \left\lceil \frac{V_{length}}{S_{length}} \right\rceil \\ NC_{aceitos,ram}(t - \left\lceil \frac{V_{length}}{S_{length}} \right\rceil) & \text{, se } t \geq 1 + \left\lceil \frac{V_{length}}{S_{length}} \right\rceil \end{cases} \tag{4.20}$$

4.2.2.4 Modelo da taxa de bloqueio

Para avaliar o desempenho do sistema proposto, nesta seção é descrita analiticamente a taxa de bloqueio do sistema, $TB(t)$, em função do tempo. Esta métrica serve como base para determinar a escalabilidade do sistema, e é calculada da seguinte forma:

$$TB(t) = \frac{NC_{denied}(t)}{NC_{total}(t)} \tag{4.21}$$

onde $NC_{denied}(t)$ representa o número de clientes bloqueados por indisponibilidade de recursos e $NC_{total}(t)$ é o total requisições de vídeo solicitadas pelos clientes. Logo, quanto menor é $TB(t)$, maior é a escalabilidade do sistema.

O total de clientes bloqueados devido a indisponibilidade de recursos de vazão para acessos de leitura no *proxy* pode ser determinado subtraindo-se o total de clientes aceitos pelo sistema do total de clientes na fila de inclusão no momento da retirada, como visto na equação abaixo:

$$NC_{denied}(t) = NC_{total}(t) - (NC_{ram}(t) + NC_{aceitos}(t)) \tag{4.22}$$

Portanto, para obter a taxa de bloqueio do sistema, basta substituir as Equações 4.22 e 4.4 na Equação 4.21.

5 ANÁLISE DE DESEMPENHO

Neste capítulo é apresentada a análise de sensibilidade realizada no modelo do sistema proposto, referenciado como MCCH. Para isso, são descritas inicialmente a metodologia e carga de trabalho utilizadas para a validação e avaliação do sistema proposto, além de definir as principais métricas usadas para medir e analisar quantitativamente o desempenho do sistema. A seguir, é realizada a validação dos resultados obtidos nas simulações da MCCH, por meio de validação cruzada, a partir de resultados apresentados em Ishikawa (2003) e Granado (2010) para cenários com 1 e 100 vídeos disponíveis no acervo do servidor para um sistema com apenas um nível de *cache* na hierarquia de memória do *proxy*. O objetivo desta validação é demonstrar a equivalência de desempenho da MCC e MCCH quando colocadas em cenários equivalentes, para, posteriormente, comparar o real ganho de desempenho da MCCH quando agregado um segundo nível de memória, de menor velocidade e mais baratas por bit, como *cache* de vítimas. Além desta comparação, é avaliado o comportamento da MCCH para diferentes cenários, representados por diferentes combinações dos parâmetros de entrada.

5.1 Metodologia de avaliação

Para avaliar o desempenho da MCCH utilizou-se o ambiente de simulação do *Objective Modular Network Testbed in C++* (OMNeT++), em conjunto com o *INET framework*, que é um pacote desenvolvido em C++ que utiliza a API do OMNeT++ para simular a comunicação de redes. Todos os módulos do sistema foram desenvolvidos na linguagem de programação C++.

Para todos os cenários de simulação da MCCH assumiu-se que sempre haverá disponibilidade de recursos de largura de banda para o envio e recebimento dos fluxos de vídeo, visto que o objetivo deste estudo é atacar a escalabilidade do sistema quando agregado um segundo nível de *cache* na hierarquia de memória do *proxy*, de modo que a vazão dos enlaces não se tornem o fator limitante na escalabilidade do sistema, ou seja, o gargalo. Portanto, o total de fluxos simultâneos que o sistema conseguirá sustentar é igual ao pico de clientes em atendimento atingidos durante o período de simulação. Além disso, assumiu-se que os clientes solicitam a exibição do vídeo a partir do início, e permanecem no sistema até o final de exibição da mídia.

A corroboração dos resultados obtidos nas simulações da MCCH, em cenários onde há apenas um nível de *cache*, é feita por meio de validação cruzada a partir de resultados obtidos em simulações e experimentos de protótipos da MCC. Além do mais, para cenários com apenas um vídeo no acervo do servidor, os resultados também são comparados com os resultados do modelo analítico descrito anteriormente. Contudo, um dos quesitos de suma importância que deverá ser levado em conta para reproduzir os experimentos de trabalhos relacionados à MCC é o modo de como a carga do sistema foi gerada, ou seja, as distribuições usadas para gerar os instantes de tempo entre as chegadas das requisições de inclusão dos clientes ao sistema e seleção dos vídeos disponíveis no acervo, descrita na seção seguinte.

Na Tabela 3 é apresentada a descrição dos principais parâmetros utilizados nas simulações da MCCH.

Tabela 3 – Descrição dos parâmetros de simulação.

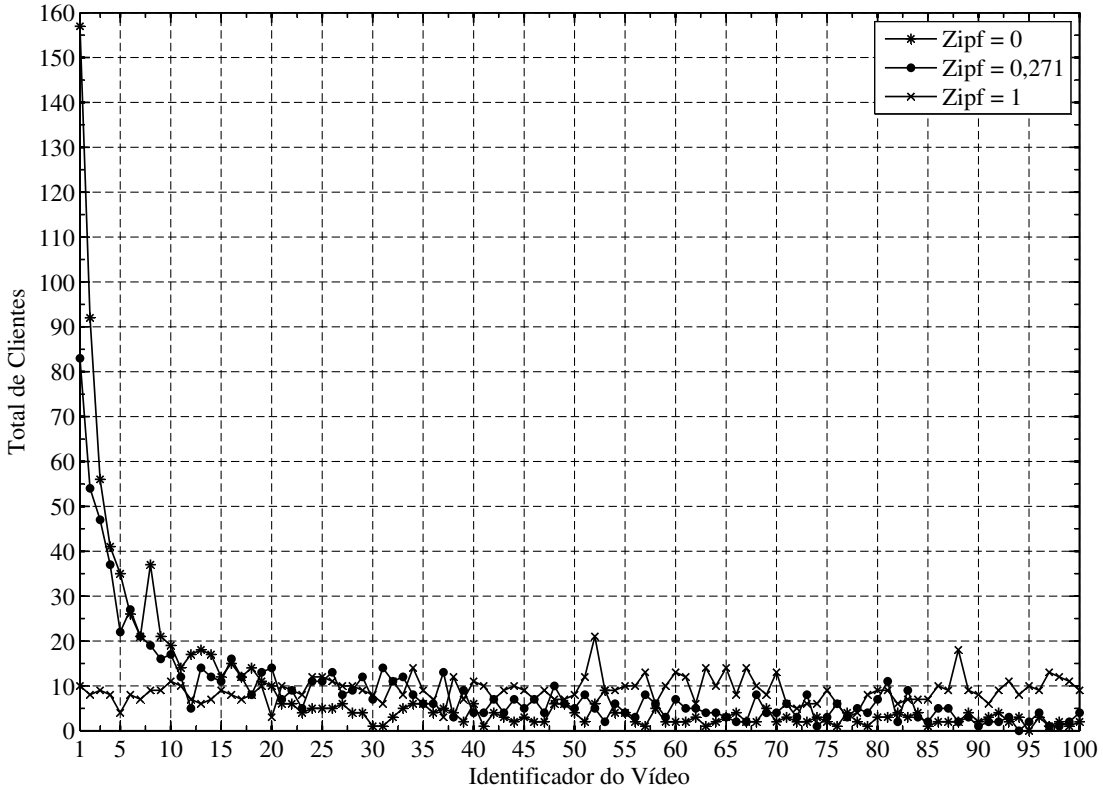
Parâmetro	**Descrição**
Número de vídeos	Indica o número de vídeos disponíveis no acervo do servidor.
Tamanho do *slot*	Indica o tamanho da porção temporal de vídeo, em segundos, que cada *slot* mapea no vetor de vídeo.
Tamanho da *cache* principal	Indica o percentual do tamanho da *cache* principal em relação ao tamanho do acervo do servidor.
Tamanho da *cache* de vítimas	Indica o percentual do tamanho da *cache* de vítimas em relação ao tamanho do acervo do servidor.
Intervalo de tempo entre chegadas	Indica o intervalo médio de tempo, em segundos, entre a chegada de requisições de inclusão dos clientes ao sistema.
Taxa do vídeo	Indica a taxa de reprodução do vídeo em Mbps.
Tamanho do prefixo	Indica o tamanho do prefixo do vídeo, em *slots*, que deverá ter seu conteúdo priorizado.
Número máximo de clientes	Indica o número máximo de clientes que poderão ser criados durante o período de simulação.
Capacidade de vazão de leitura	Indica a capacidade de vazão de leitura nominal do dispositivo de memória da *cache* de vítimas em Mbps.
Capacidade de vazão de escrita	Indica a capacidade de vazão de escrita nominal do dispositivo de memória da *cache* de vítimas em Mbps.
Duração do vídeo	Indica o tempo de reprodução do vídeo em minutos.
Tamanho da sequência de LS	Indica o tamanho máximo da sequência de LS, em *slots*, que poderá ser criada para unir dois *buffers*.
Tempo de simulação	Indica o tempo de simulação dos experimentos em minutos.
Zipf	Indica o valor do expoente da distribuição de Zipf usado para modelar a popularidade de acesso aos vídeos disponíveis no acervo.

5.2 Carga de trabalho

Para gerar a carga de trabalho semelhantes à observada em servidores reais, a chegada das requisições de inclusão dos clientes ao sistema foi modelada como um Processo de Poisson, com tempos entre chegadas exponencialmente distribuídos com média de $\frac{1}{\lambda}$, onde λ representa a taxa de chegada. A escolha dos vídeos foi baseada na Distribuição Zipf, utilizada para modelar a diferença de popularidade existente entre os diferentes elementos do acervo.

A Figura 25 apresenta a distribuição de 900 clientes para um acervo com 100 vídeo variando-se o parâmetro Zipf. As Figuras 25a e 25b[14] demonstram o efeito de diferentes valores de Zipf na distribuição de acesso dos clientes aos vídeos. Note que, a curva com valor de $Zipf = 1$, a distribuição está mais próxima da uniforme, sendo que a probabilidade de cada vídeo ser requisitado é de aproximadamente 1%, diferentemente da curva $Zipf = 0$, onde há uma concentração maior de clientes em um subconjunto dos vídeos contidos no acervo, no qual aproximadamente 70% dos clientes requisitam 20% do conteúdo do acervo.

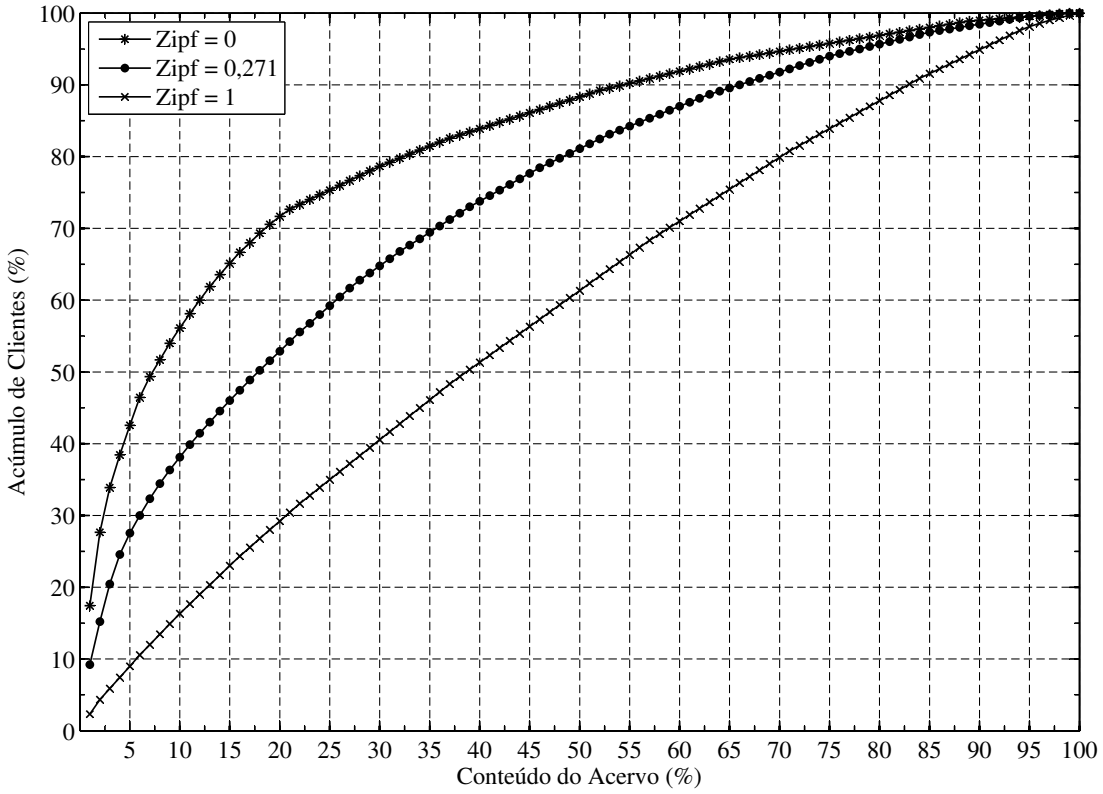
Figura 25 – Influência do parâmetro Zipf na distribuição de acesso dos clientes aos vídeos do acervo.



(a) Popularidade dos vídeos

[14] O eixo horizontal representa o percentual do conteúdo do acervo do servidor classificados do mais popular ao menos popular.

Figura 25 – Influência do parâmetro Zipf na distribuição de acesso dos clientes aos vídeos do acervo.



(b) Teste do Princípio de Pareto[15]

5.3 Métricas de desempenho

Para validar e analisar quantitativamente o desempenho do sistema proposto foram utilizadas as seguintes métricas:

Pico de Utilização: é a maior taxa de ocupação atingida no enlace entre o *proxy* e o servidor durante o período de simulação.

Taxa de Utilização: é a taxa média de utilização do enlace entre o *proxy* e servidor. Quanto menor[16] este valor, mais eficiente é o algoritmo de substituição da *cache* no *proxy*.

Número de Fluxos: é o número de fluxos oriundos do servidor principal para o *proxy* de vídeo computados a cada giro[17]. Quanto menor este valor, mais eficiente é o gerenciamento da *cache*

[15] O Princípio de Pareto, também conhecido como regra $80-20$, afirma que, para muitos eventos, cerca de 80% dos efeitos vêm de 20% das causas.

[16] Note que, embora um sistema possa ter uma taxa média de utilização baixa, se em pequenos intervalos de tempo, a taxa instantânea de utilização for muito alta, poderá ocasionar atraso e perda de pacotes, comprometendo a QoS do sistema.

[17] Um giro do sistema equivale ao tempo de envio de um *slot* de vídeo.

do *proxy*.

Taxa de Bloqueio – TB: é a percentagem de requisições não atendidas por falta de recursos no sistema.

$$TB = \frac{R_{denied}}{R_{total}}$$

onde, R_{denied} é o total de requisições bloqueadas devido a indisponibilidade de recursos, e R_{total} representa o total de requisições que solicitaram o serviço. Esta métrica indica a escalabilidade de um sistema de VoD. Quanto menor este valor, mais escalável é o sistema.

5.4 Validação da MCCH

O objetivo desta seção é comprovar, por meio de validação cruzada, a equivalência entre o simulador da MCCH desenvolvido e as implementações de (GRANADO, 2010) e (ISHIKAWA, 2003) da MCC quando colocadas em cenários equivalentes, onde possuem o mesmo percentual de recursos. De forma secundária mas de igual modo relevante, a análise dos resultados apresentados nesta seção permite um melhor entendimento da aplicação da MCC.

Na Tabela 4, tem-se os principais parâmetros utilizados nos experimentos com o protótipo (GRANADO, 2010) e simulações de (ISHIKAWA, 2003) da MCC, complementados com parâmetros adicionais necessários para a validação do simulador desenvolvido. Cabe observar que o tamanho da *cache* de vítimas é 0% do tamanho do acervo, portanto, apenas um nível de *cache* (*cache* principal) será utilizado pela MCCH para armazenar os segmentos de vídeos no *proxy*, caracterizando uma arquitetura equivalente à MCC.

Tabela 4 – Parâmetros de simulação da MCCH para uma arquitetura com apenas 1 nível de memória.

Parâmetro	**Padrão**	**Variação**
Número de vídeos	100	1
Duração do vídeo (minutos)	60	N/A
Taxa do vídeo (Mbps)	1	N/A
Intervalo de tempo entre chegadas (segundos)	3,1	3,1 – 150
Tamanho do *slot* (segundos)	8	N/A
Tamanho do prefixo (*slots*)	0	N/A
Tamanho da sequência de LS	∞	N/A
Tamanho da *cache* principal (% do tamanho do acervo)	10	10 – 100
Tamanho da *cache* de vítimas (% do tamanho do acervo)	0	N/A
Tempo de simulação (minutos)	60	N/A
Número máximo de clientes	900	1000
Zipf	0,271	0; 1

Cabe ressaltar que nas versões dos protótipos da MCC desenvolvidos por Bragato (2006) e Granado (2010), o *buffer* é constituído por três *slots* na seguinte ordem: BS, RS e WS,

diferentemente da versão originalmente descrita por Ishikawa (2003), no qual a MCC utiliza quatro *slots* por *buffer*. A remoção do *slot* ES deu-se pelo fato de que é possível interagir com o servidor à tempo para controlar o fluxo de vídeo que alimenta o *slot* WS sem comprometer a QoS do sistema. Na versão da MCCH, o *slot* BS foi removido, deste modo, o *buffer* é formado apenas pelos *slots* RS e WS. Note que, esta alteração na estrutura do *buffer* influência diretamente na quantidade de recursos de armazenamento disponível para alocação, e, consequentemente, na taxa de bloqueio do sistema, conforme será mostrado posteriormente.

5.4.1 Validação com 1 vídeo

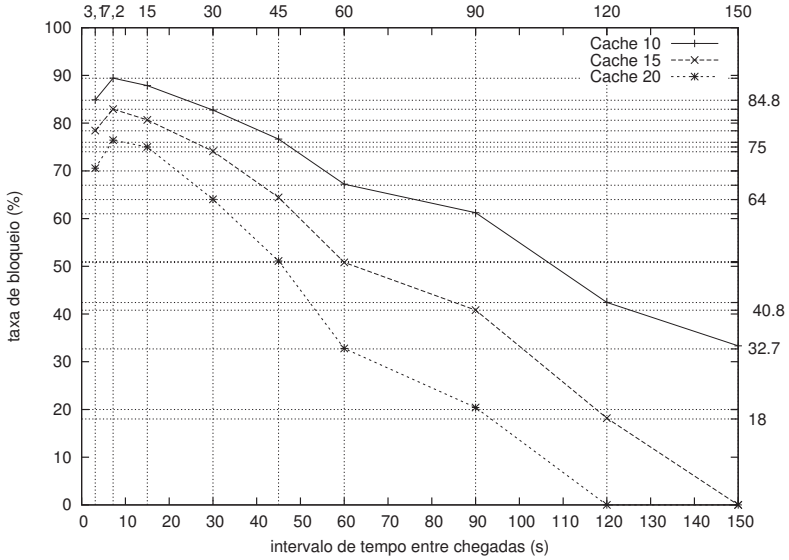
Nesta subseção serão apresentados os principais resultados obtidos nas simulações da MCCH para cenários com apenas um vídeo no acervo do servidor. Estes resultados serão confrontados com os resultados dos experimentos da MCC (GRANADO, 2010) e do modelo matemático. Todas as simulações para o cenário atual foram realizadas limitando a 900 o número máximo de clientes.

A Figura 26 apresenta os resultados da taxa de bloqueio da MCC e MCCH em função do intervalo de tempo entre as chegadas das requisições de inclusão dos clientes ao sistema para os percentuais de tamanho da *cache* principal de 10%, 15% e 20% do tamanho do acervo do servidor. Comparando-se os resultados dos experimentos de Granado (2010) (ver Figura 26a), com os resultados obtidos nas simulações da MCCH, apresentados na Figura 26b, é possível observar que ambas possuem o mesmo comportamento, entretanto, existem diferenças nos resultados. Tais diferenças se devem ao fato de que o número de *slots* por *buffer* nas duas implementações são diferentes, influenciando diretamente na quantidade de *slots* disponíveis para alocação. Para investigar a influência do tamanho do *buffer* na taxa de bloqueio do sistema, será analisado o comportamento da MCC e MCCH.

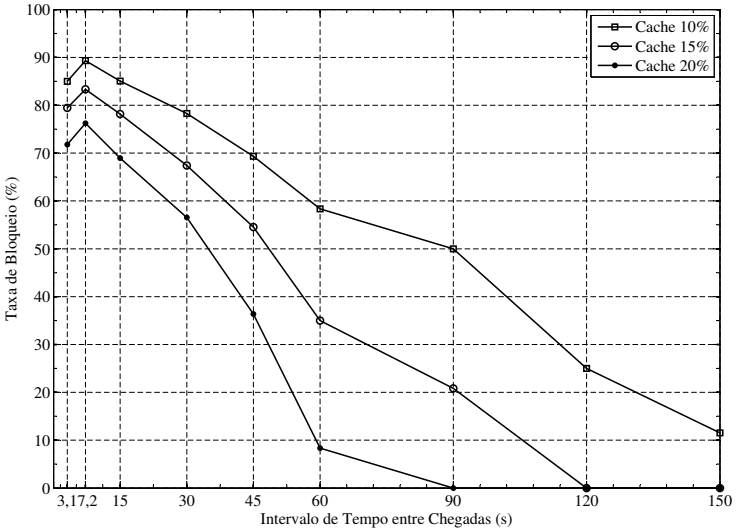
Analisando-se os gráficos apresentados na Figura 26, note que existe um pico na taxa de bloqueio quando o intervalo de tempo entre chegadas de requisições é $7,2$ segundos, embora para o intervalo $3,1$ segundos a demanda ao sistema seja maior. Isso ocorre para intervalos de tempo entre chegadas menores que o tamanho do *slot*, que neste caso é de 8 segundos, pois uma quantidade maior de clientes são agrupados e incluídos no mesmo *buffer*, estratégia conhecida como *batching*, permitindo que o mesmo *buffer* seja compartilhado entre os clientes. Deste modo, a quantidade de recursos de armazenamento alocados para a criação dos *buffers* para suprir a demanda média de $19,4$ clientes/minuto $\left(\frac{1}{3,1} * 60\right)$ é a mesma quantidade usada para atender a demanda de $8,3$ clientes/minuto $\left(\frac{1}{7,2} * 60\right)$. Além do mais, nesta faixa de intervalo ocorre a sobreposição dos *buffers*, permitindo que dois *buffers* distintos compartilhem a mesma

área de memória, reduzindo-se, assim, o total de *slots* que deverão ser reservados para a criação do *buffer*, ou seja, com exceção ao primeiro *buffer* criado pela MCCH, que reservará dois *slots* (RS e WS), os demais *buffers* reservarão somente um *slot* e compartilharão o outro. Sendo assim, para cenários onde o intervalo de tempo entre chegadas de requisições dos clientes é menor que o tamanho do *slot*, a taxa de bloqueio da MCC e MCCH não será influenciada pela diferença no número de *slots* por *buffer*, conforme apresentado no gráfico das Figuras 26a e 26b.

Figura 26 – Validação dos resultados da MCCH para cenários com apenas 1 vídeo disponível no acervo do servidor – Taxa de Bloqueio vs Intervalo de Tempo entre Chegadas para percentuais de *cache* principal de 10%, 15% e 20% do tamanho do acervo.



(a) Experimento da MCC



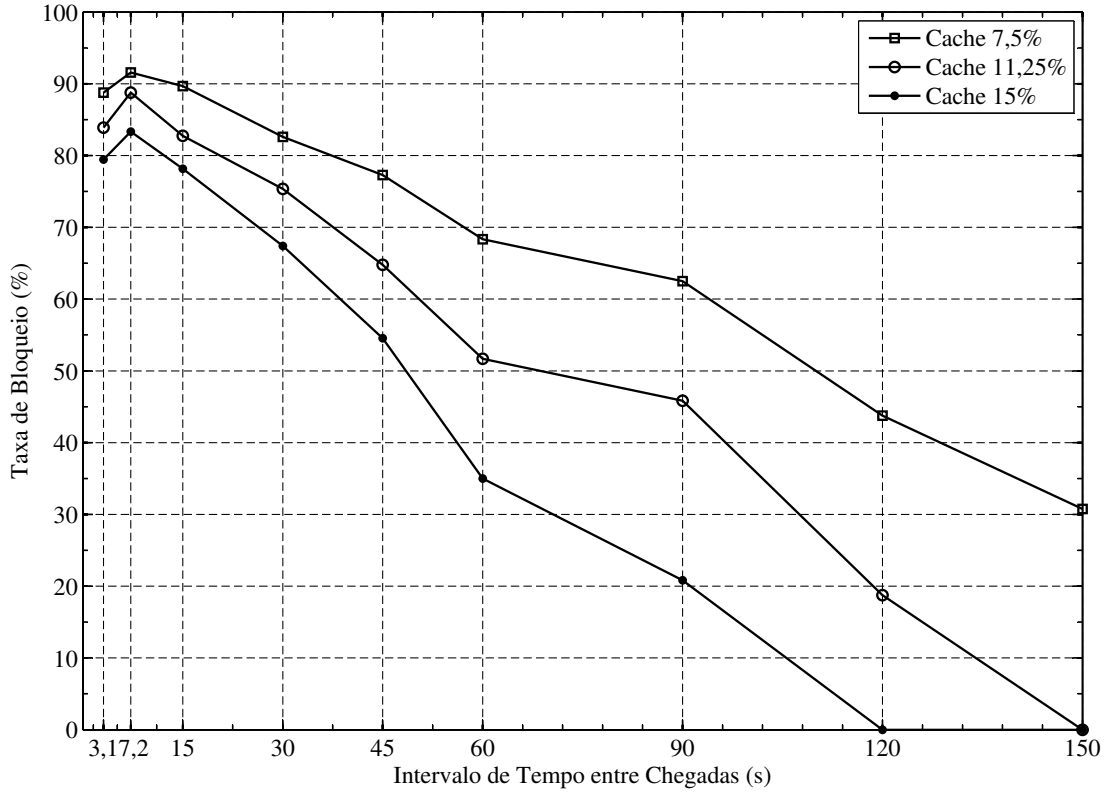
(b) Simulação da MCCH

Para intervalos de tempo entre chegadas acima de 30 segundos, cenários onde não há

sobreposição dos *buffers* tanto na MCC quanto na MCCH, a taxa de bloqueio da MCC para o percentual de tamanho de *cache* de 20% (ver Figura 26a) é igual a taxa de bloqueio da MCCH para o percentual de 15% (ver Figura 26b), ou seja, existe uma relação entre o número de *slot* por *buffer* e a taxa de bloqueio, sendo que, a taxa de bloqueio da MCCH é equivalente a taxa de bloqueio da MCC quando o percentual do tamanho da *cache* da MCCH é igual a $0,75$ do percentual do tamanho da *cache* da MCC.

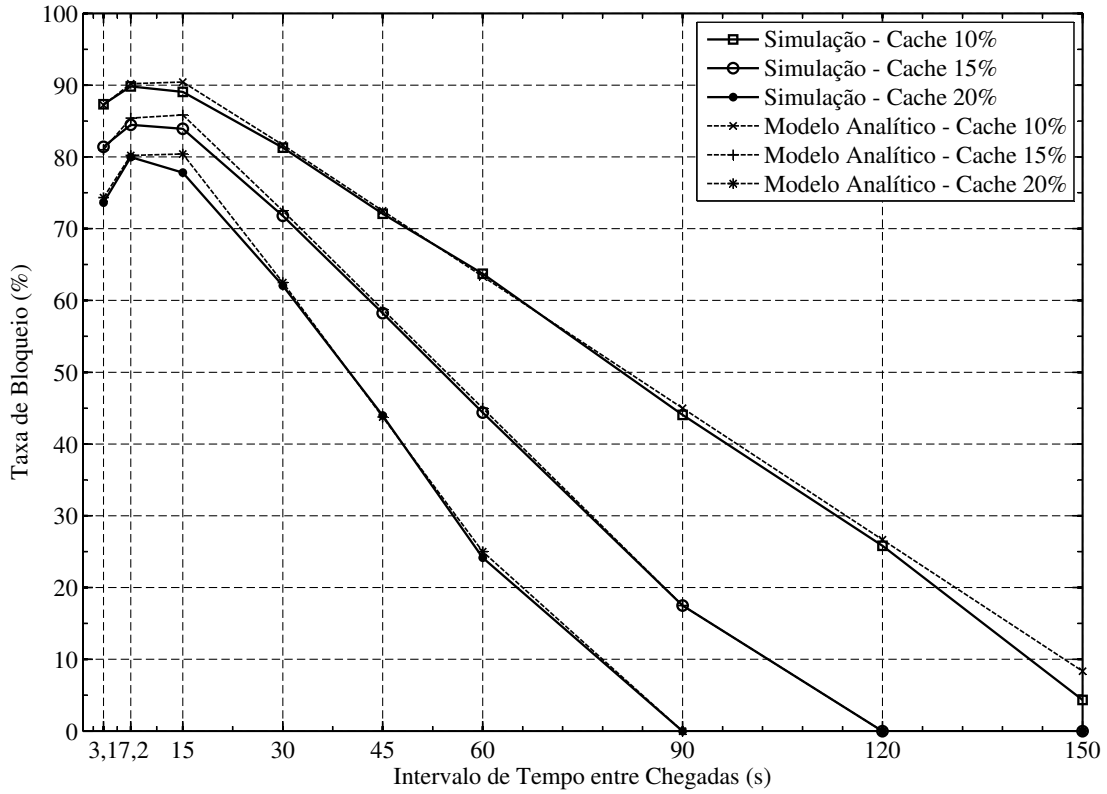
A Figura 27 apresenta os resultados de simulação da MCCH para percentuais de *cache* de $7,5\%$, $11,25\%$ e 15% do tamanho do acervo, cenário equivalente a $0,75$ do percentual do tamanho de *cache* de 10%, 15% e 20% da MCC, respectivamente.

Figura 27 – Simulação da MCCH para cenários com apenas 1 vídeo no acervo do servidor para percentuais de *cache* principal de $7,5\%$, $11,25\%$ e 15% do tamanho do acervo.



Embora a análise dos resultados apresentados nas Figuras 26 e 27 indiquem uma relação entre o número de *slots* por *buffer* e a taxa de bloqueio, foi desenvolvido um modelo matemático simplificado, descrito no capítulo anterior, que modela a taxa de bloqueio da MCCH. A Figura 28 demonstra a comparação entre os resultados do modelo analítico e simulações da MCCH.

Figura 28 – Comparação entre a taxa de bloqueio do modelo analítico e simulação da MCCH para cenários com apenas 1 vídeo no acervo do servidor para percentuais de *cache* principal de 10%, 15% e 20% do tamanho do acervo.



5.4.2 Validação com 100 vídeos

Nesta subseção são apresentados os resultados de simulação da MCCH para cenários com 100 vídeos no acervo do servidor. Resultados estes que serão comparados com os resultados obtidos em experimentos de protótipo da MCC (GRANADO, 2010) e simulações de Ishikawa (2003).

Análise do Número de Fluxos

A Figura 29 mostra os resultados do número de fluxo obtidos nos experimentos do protótipo da MCC (GRANADO, 2010) e simulações da MCCH para o percentual de tamanho de *cache* principal de 10% do tamanho do acervo do servidor, segundo os intervalos de tempo entre as chegadas de requisições de inclusão dos clientes ao sistema de $3,1$, 15 e 30 segundos, limitando a 900 o número máximo de clientes.

Na Figura 29a, existem dois tipos de técnicas de gerenciamento da MCC indicadas como Memória Cooperativa Colapsada Local sem Otimização (MCCL-) e Memória Cooperativa

Colapsada Local com Otimização (MCCL+). Na MCCL-, desenvolvido por Bragato (2006), o compartilhamento da *cache* não é global, ou seja, cada vídeo reservará um percentual do tamanho da *cache*. Além do mais, o sistema de admissão de clientes não considera os *slots* do tipo LS como *slot* que podem ser removidos, desta forma, uma vez criada a sequência de LS, a mesma não poderá ser removida, embora este tipo de *slot* possa ter sua prioridade de descarte aumentada, permitindo a liberação de recursos para que novos *buffers* possam ser criados. A MCCL+ foi desenvolvida por Granado (2010) com o objetivo de tornar global o compartilhamento da *cache* entre os vídeo, além de otimizar o sistema de admissão de clientes da MCCL-, permitindo que o conteúdo mapeado pelas sequências de LS sejam liberados para que novos *buffers* possam ser criados, equivalente a técnica de gerenciamento de *cache* de *proxy* de vídeo originalmente descrita por Ishikawa (2003). Neste trabalho, somente a técnica MCCL+ será relevante para a validação dos resultados.

Cabe ressaltar que, a taxa de bloqueio para todos os cenários apresentados nesta seção foi zero, tanto na MCCL+ quanto na MCCH. Note que, esta métrica influencia diretamente na utilização dos recursos de enlace entre o servidor e o *proxy*, visto que, o bloqueio de novas requisições de inclusão diminui o número de *buffers* ativos no sistema, e consequentemente, reduz a carga sobre o servidor. Este efeito pode ser visto na Figura 29a, onde a MCCL- possui um desempenho superior a MCCL+ quando comparado o número de fluxos para intervalos de tempo entre chegadas de $3,1$, porém, a taxa de bloqueio da MCCL- é aproximadamente 35%, diferentemente da MCCL+, onde a taxa é zero (GRANADO, 2010).

Para analisar o comportamento dos resultados obtidos nas simulações da MCCH, a curva representada pelo intervalo de tempo entre chegadas de $3,1$ segundos, conforme ilustrado na Figura 29b, foi dividida em 4 faixas. A primeira faixa é representada pelo intervalo de tempo entre 0 e aproximadamente 11 minutos, onde para cada primeira requisição de inclusão para um determinado vídeo contido no acervo um fluxo de vídeo é gerado para alimentar o *slot* WS do *buffer*, sendo que os demais *buffers* a serem futuramente criados serão encadeados através de sequências de LS com os *buffers* já existentes.Deste modo, apenas um fluxo de vídeo será necessário para alimentar todos os *buffers* colapsados para um mesmo vídeo. Sendo assim, espera-se que o comportamento da curva neste intervalo tenda a se estabilizar em 100 fluxos (número de vídeos disponíveis no acervo) à medida que o tempo de simulação progrida. Entretanto, vale ressaltar que o número máximo de fluxo atingido foi de 79 fluxos, isso ocorre visto que a carga gerada pela distribuição de Zipf tende a concentrar as requisições de inclusão dos clientes em um subconjunto dos vídeos contidos no acervo (veja a curva Zipf = $0,271$ apresentada nos gráficos da Figura 25), assim, nem todos os vídeos do acervo foram requisitados nesta faixa de tempo.

Todavia, em aproximadamente 11 minutos, a *cache* principal da MCCH atinge sua capacidade máxima de armazenamento iniciando a liberação de recursos de armazenamento para que os fluxos oriundos do servidor possam ser cacheados no *proxy*. Sendo assim, a partir deste instante, o aumento no número de fluxos não se deu somente pelo fato de que novos *buffers* serão criados para atender a novas requisições de inclusão para os vídeos que não possuem clientes em atendimento, mas também pelo fato de que a liberação de recursos da *cache* implica na criação de fluxos de *path* entre o servidor e o *proxy* quando o *slot* WS dos *buffers* requisitarem ao servidor o conteúdo de vídeo mapeado pelos *slots* liberados. Como neste instante de tempo todos os *buffers* estão encadeados através de sequências de LS e o tamanho do prefixo do vídeo a ser priorizado é zero, os *slots* do tipo FS que mapeiam o início[18] do vídeo serão liberados, partindo-se dos vídeos de menor popularidade (menor número de clientes), de modo que os *slots* FS que estão mais ao final do vídeo terão maior prioridade de descarte. Sendo assim, na segunda faixa representada pelo intervalo aproximado de tempo entre 11 e 17 minutos, indicado respectivamente pela seta azul e seta vermelha da curva de $3,1$ segundos apresentada na Figura 29b, a MCCH irá liberar somente o conteúdo de vídeo alocado pelos *slots* do tipo FS.

Contudo, observe que a partir de 15 minutos há uma tendência no aumento do número de fluxos, embora nenhuma sequência de LS tenha sido desencadeada. Esse efeito é decorrente da liberação dos *slots* que mapeiam o início do vídeo, resultando na criação de fluxos de *path* entre o servidor e *proxy* quando novos *buffers* forem incluídos em vídeos que tiveram seu prefixo removido, além do mais, a medida que o tempo de simulação se aproxima de 17 minutos, instante no qual a MCCH iniciará o processo de desencadeamento das sequências de LS, a tendência é que o conteúdo dos *slots* FS dos vídeos de maior popularidade sejam rapidamente liberados, visto que o processo de inclusão de *buffers* reduz a prioridade de descarte dos *slots* FS para LS, diminuído-se, assim, a quantidade de *slots* FS disponíveis para desalocação, logo, para atender a demanda do sistema, uma quantidade maior de vídeos terão seu prefixo removido quando o sistema solicitar a liberação de recursos.

Em aproximadamente 17 minutos, a quantidade de unidades de vídeo mapeadas pelos *slots* do tipo FS não é suficiente para atender a demanda do sistema, logo, a MCCH aumenta a prioridade de descarte dos *slots* do tipo LS, iniciando o processo de liberação de sequências de LS para que novos *buffers* possam ser criados. Como as sequências de LS substituem canais regulares do servidor, o desencadeamento destas sequências e liberação do conteúdo da mesma, aumentam o número de fluxos.

Na quarta faixa, representada pelo intervalo aproximado de tempo entre 44 e 60 minutos, o

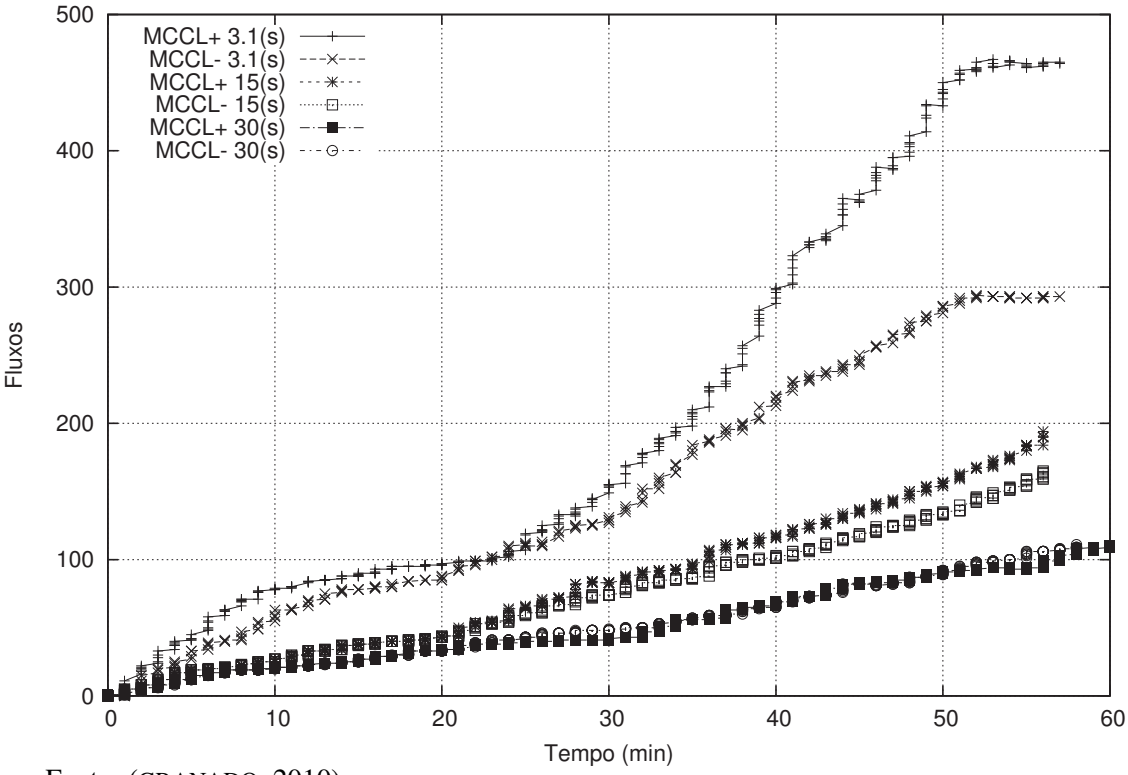
[18] *Slots* que indexam a porção temporal de vídeo associada à primeira unidade de vídeo no vetor de vídeo até a última unidade anterior ao *slot* RS mais próximo do início do vídeo (último *buffer* criado).

número de fluxos tende a estacionar em torno de 525 fluxos com pico de 527 (aproximadamente $58,6\%$ na utilização do enlace). Esse comportamento é resultante do limiar imposto no processo de criação da carga que limita o número máximo de clientes (900 clientes) que poderão ser gerados durante o período de simulação, sendo assim, a partir deste instante, nenhum *buffer* será incluído no sistema, portanto, a quantidade de *slots* FS com conteúdo (alocados) é igual ou superior à quantidade de recursos de armazenamento a serem alocados a cada giro do sistema pelo *slot* WS dos *buffers*. Note que, caso este limiar não fosse imposto, o sistema somente tenderia a estabilizar após o tempo de simulação de 60 minutos, pois, a partir deste instante, os primeiros clientes incluídos passariam a sair do sistema.

De modo análogo, a explicação do comportamento descrito para a curva de $3,1$ segundos é válida para as curvas representadas pelos intervalos de tempo entre chegadas de 15 e 30 segundos, porém, a quarta faixa, que demonstra o estado estacionário do sistema, não está presente para as curvas de 15 e 30 segundos, visto que, para tais demandas, o pico de clientes que requisitaram o serviço durante o período de simulação foi em média de 240 e 120, respectivamente, que está abaixo do limiar de 900 clientes.

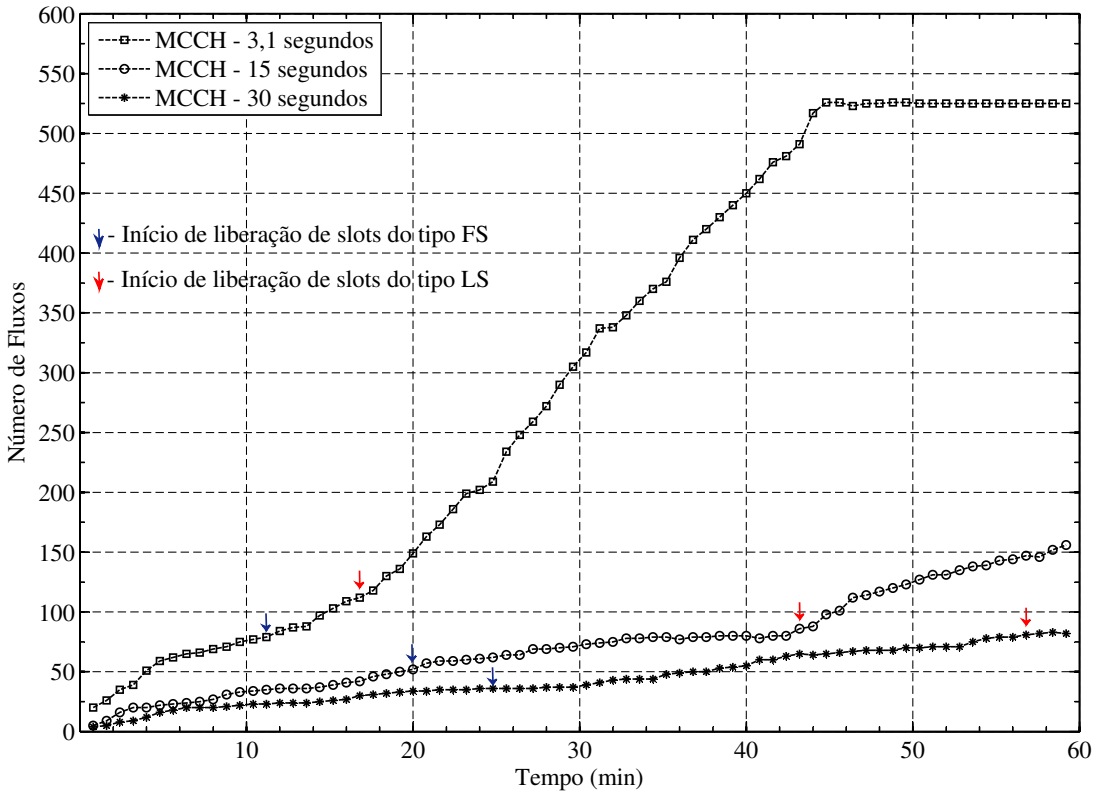
De maneira geral, o comportamento obtido pela MCCH é bastante similar ao apresentado pela MCCL+, com exceção a diferença apresentada durante o processo de liberação dos *slots* do tipo FS (representada pela segunda faixa), de modo que a MCCL+ consegue sustentar por um período de tempo maior (até aproximadamente 23 minutos) o número de fluxos abaixo de 100, diferentemente da MCCH, que mantém até aproximadamente 15 minutos. Esta melhoria no desempenho da MCCL+ para a curva de $3,1$ segundos está possivelmente relacionada ao critério utilizado para liberação do conteúdo mapeado pelos *slots* do tipo FS, sugerindo que a MCCL+ não baseia-se apenas na popularidade dos vídeos, mas também utiliza a posição do *slot* no vetor de *slots* como critério adicional, de modo que, o processo de liberação de recursos prioriza, para os vídeos de mesma popularidade (ou popularidade próxima), a remoção do conteúdo dos *slots* FS que estão mais ao final, balanceando-se, assim, a remoção do prefixo, e consequentemente, adiando-se o processo de criação de fluxos de *path* entre o servidor e o *proxy*.

Figura 29 – Validação da MCCH para cenários com 100 vídeos disponíveis no acervo do servidor – Número de Fluxos vs Tempo para o tamanho de *cache* principal de 10%, segundo os diferentes intervalos de tempo entre chegadas de 3, 1, 15 e 30 segundos.



Fonte: (GRANADO, 2010)

(a) Número de fluxos da MCC



(b) Número de fluxos da MCCH

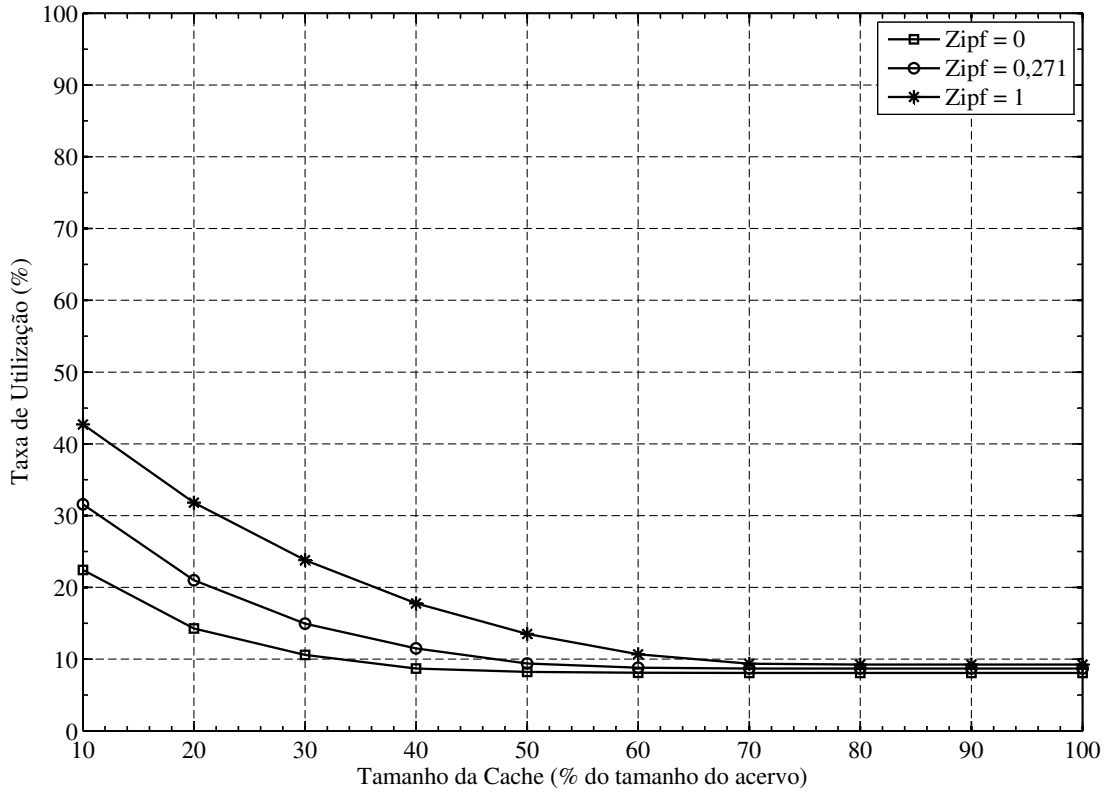
Análise da Taxa e Pico de Utilização

Os gráficos ilustrados na Figura 30 apresentam os resultados da taxa e pico de utilização do enlace entre o servidor e o *proxy* obtidos nas simulações da MCCH em função do tamanho da *cache* principal, segundo os diferentes valores do parâmetro Zipf. Resultados estes que serão analisados e validados com os resultados de simulação da MCC (ISHIKAWA, 2003) apresentados na Figura 30. Conforme utilizado nas simulações de Ishikawa (2003), o número máximo de clientes foi limitado a 1000 e o intervalo de tempo entre chegadas de requisições de inclusão dos clientes ao sistema foi de $3,1$ segundos.

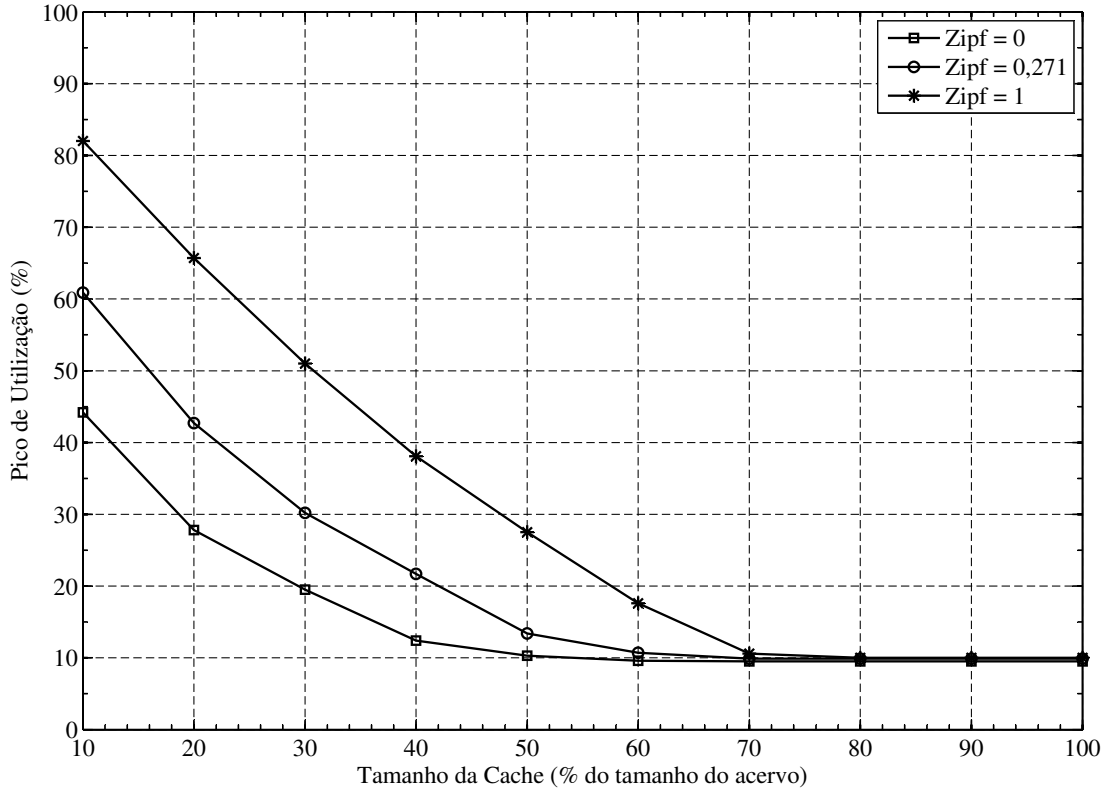
Analisando-se os resultados da taxa e pico de utilização apresentados respectivamente nas Figuras 30a e 30b, a medida que o valor do parâmetro Zipf tende a zero, menor será a utilização dos recursos de enlace entre o servidor e *proxy*. Para compreender este comportamento, as curvas apresentadas nos gráficos da Figura 25 serão de suma importância para esta análise.

Observe na Figura 25a que a variação do parâmetro Zipf altera a distribuição de acesso dos clientes aos vídeos, acentuando a concentração das requisições de inclusão para determinados vídeos contidos no acervo quando o valor do parâmetro Zipf se aproxima de 0, ao contrário do valor 1, no qual a distribuição Zipf tende a distribuir as requisições dos clientes uniformemente entre os 100 vídeos disponíveis no acervo. Como a MCC utiliza a popularidade dos vídeos como critério para escolha dos trechos de vídeos que terão seus recursos liberados, os mais populares terão maior prioridade na manutenção das suas unidades de vídeos armazenadas em *cache*, em detrimento daqueles de menor popularidade, que terão seus recursos liberados quando solicitado pelo sistema. Portanto, os vídeos mais requisitados poderão alocar um percentual maior de espaço de armazenamento, permitindo, assim, que uma fração maior do conteúdo destes vídeos sejam mantidos em *cache*, de modo que os *buffers* pertencentes aos vídeos de maior populares se beneficiem do princípio da localidade temporal e espacial, aumentando, consequentemente, a taxa de acertos. Entretanto, para cenários no qual há pequenas diferenças na popularidade dos vídeos, a distribuição de acesso dos clientes ao conteúdo do acervo está mais próxima da uniforme, de modo que o critério de liberação de recursos da *cache* não se beneficiará da popularidade dos vídeos. Esta explicação justifica na melhoria de desempenho obtida pelo sistema quando o valor do parâmetro de Zipf tende a 0.

Figura 30 – Comparação entre os resultados de simulação da MCCH com os resultados obtidos em simulações da MCC (ISHIKAWA, 2003).

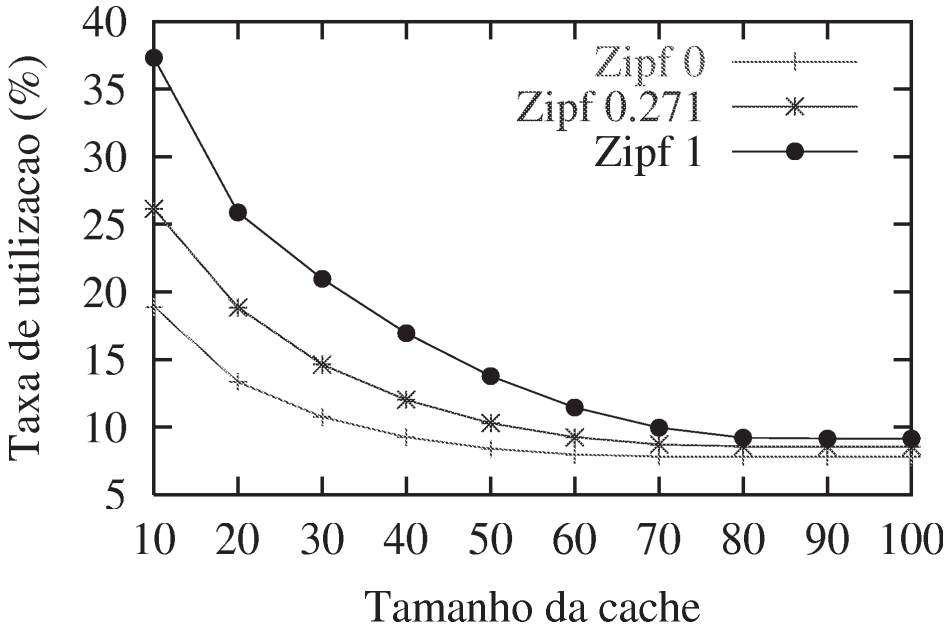


(a) Taxa média de utilização – MCCH

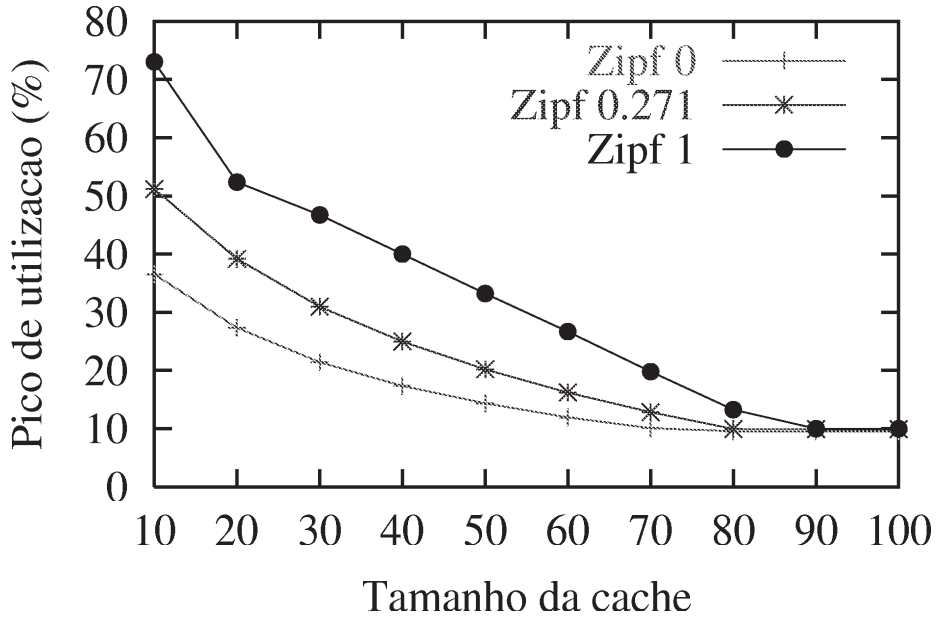


(b) Pico de utilização – MCCH

Figura 30 – Comparação entre os resultados de simulação da MCCH com os resultados obtidos em simulações da MCC (ISHIKAWA, 2003).



(c) Taxa média de utilização – MCC



Fonte: (ISHIKAWA, 2003)

(d) Pico de utilização – MCC

5.4.3 Discussão

Na seção atual apresentamos os resultados de simulação da MCCH para um sistema com um nível de *cache* na hierarquia de memória do *proxy*, objetivando validar os resultados obtidos

nas simulações, por meio de validação cruzada, com os resultados apresentados em trabalhos relacionados a MCC.

Para cenários com apenas 1 vídeo disponível no acervo, os resultados obtidos nas simulações demonstraram que o comportamento da taxa de bloqueio da MCCH condiz com o apresentado pela MCC (GRANADO, 2010), porém, a MCCH obteve um desempenho superior, visto que, o *slot* BS existente na implementação da MCC (GRANADO, 2010) foi removido na implementação atual. Entretanto, para cenários com 100 vídeos, observou-se um comportamento similar entre as curvas do número de fluxos, taxa e pico de utilização nas duas implementações, contudo, para determinados cenários, a MCCH mostrou desempenho inferior.

De forma geral, os resultados de simulação da MCCH apresentados na seção atual demonstram que a implementação do simulador desenvolvido para cenários com apenas um nível de *cache* na hierarquia de memória do *proxy* é equivalente a MCC.

5.5 Otimização de desempenho da MCC

Uma das características não abordadas em trabalhos de Ishikawa (2003), Bragato (2006) e Granado (2010) sobre a MCC foi a influência do processo de criação de sequências de LS na QoS do sistema. Portanto, nesta seção, investigamos, de forma preliminar, o efeito do tamanho da sequência de LS na utilização dos recursos de enlace entre o servidor e *proxy*. A Tabela 5 resume os principais parâmetros usados nas simulações.

Tabela 5 – Parâmetros de simulação da MCCH.

Parâmetro	**Padrão**	**Variação**
Número de vídeos	100	N/A
Duração do vídeo (minutos)	60	N/A
Taxa do vídeo (Mbps)	1	N/A
Intervalo de tempo entre chegadas (segundos)	3,1	N/A
Tamanho do *slot* (segundos)	8	N/A
Tamanho do prefixo (*slots*)	0	N/A
Tamanho da sequência de LS	450	0 – 450
Tamanho da *cache* principal (% do tamanho do acervo)	10; 15; 20	N/A
Tamanho da *cache* de vítimas (% do tamanho do acervo)	0	N/A
Tempo de simulação (minutos)	60	N/A
Número máximo de clientes	900	N/A
Zipf	0,271	N/A

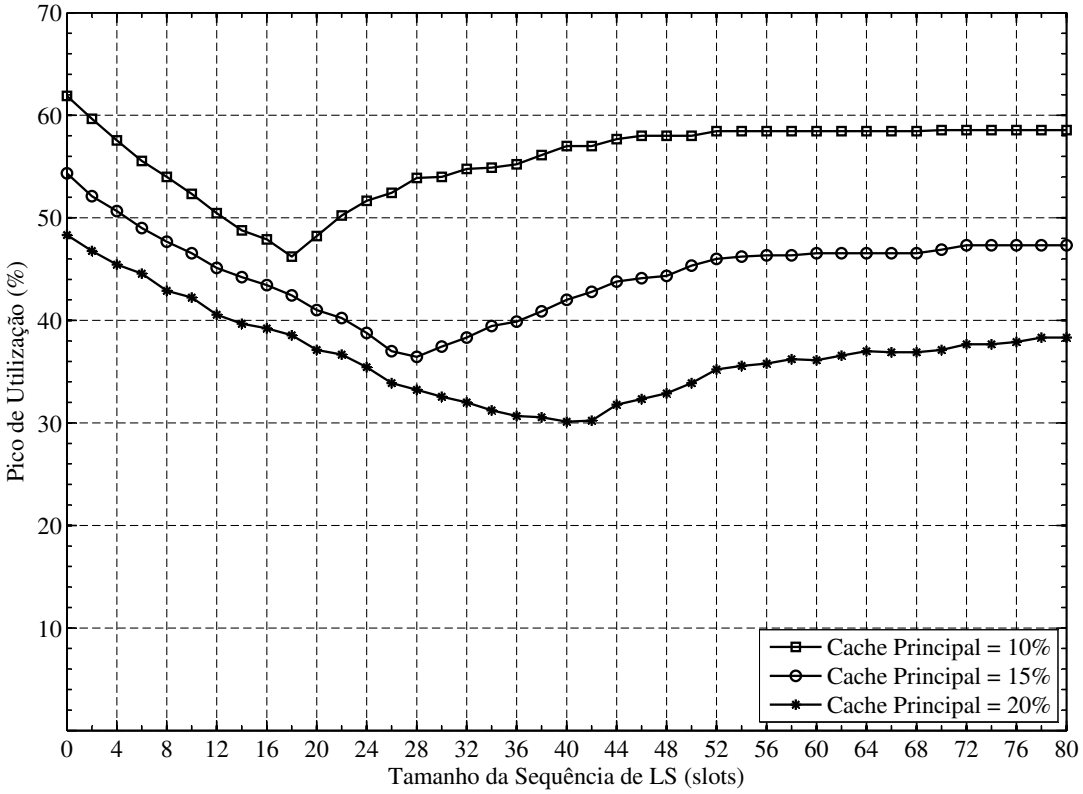
Os gráficos da Figura 31 demonstram o efeito do tamanho da sequência de LS no pico e taxa de utilização na QoS do sistema, segundo os percentuais de tamanho de *cache* principal de 10%, 15% e 20%.

Embora não apresentadas no gráfico, foram realizadas simulações com tamanhos de sequências de até 450 *slots* (equivalente ao tamanho do vetor de *slots*), porém não foram mostradas, pois as curvas tendem a se estabilizar para sequências de LS muito grande, visto que, para cenários com alta demanda ao sistema e grandes concentrações de clientes em um subconjunto dos vídeos do acervo (popularidade), o percentual de sequências muito longas é quase nulo, de maneira a não influenciar no comportamento da curva. Cabe ressaltar, que para o tamanho de sequências de LS igual a zero, a MCCH equivale-se a abordagem tradicional de *ring buffer*, e o extremo (450 slots), a técnica MCCL+.

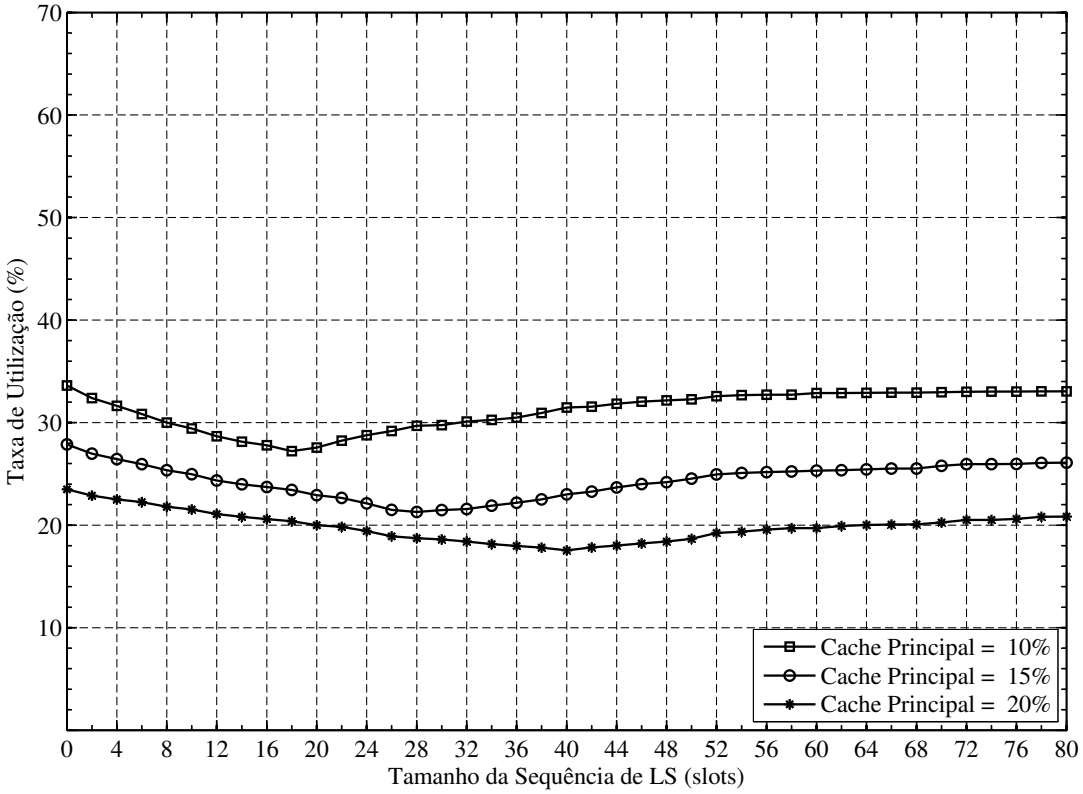
Analisando a curva de tamanho de *cache* principal de 10% apresentada no gráfico da Figura 31a, note que, a medida que o tamanho da sequência de LS aumenta, o pico de utilização diminui, atingindo um valor mínimo de $46,22\%$ quando o tamanho da sequências de LS é de 18 *slots*, porém, para sequências acima deste valor, o pico de fluxos tende a aumentar.

Esta redução ocorre, porque o aumento no tamanho da sequência de LS permite que *buffers* separados por pequenas distancias temporais (vídeos de alta popularidade) tenham seu conteúdo priorizado, logo, apenas um fluxo de vídeo será necessário para alimentá-los, beneficiando-se, assim, da localidade temporal e espacial existente. Entretanto, como o tamanho da *cache* é um pequeno percentual do tamanho do acervo do servidor, a criação de sequências de LS muito longas (maiores que 18 *slots*), permitem que os vídeos de baixa popularidade tenham o conteúdo entre os *buffers* priorizado, reduzindo-se, assim, a quantidade de recursos de armazenamento disponíveis para liberação, e, consequentemente, forçando a MCCH a liberar *slots* FS dos vídeos de maior popularidade que fazem parte do prefixo do vídeo. Como dito anteriormente, a remoção do prefixo implica na criação de fluxos de *path* entre o servidor e o *proxy* quando novos *buffers* forem incluídos no sistema.

Figura 31 – Influência do tamanho da sequência de LS na utilização dos recursos de enlace entre o servidor e o *proxy*.



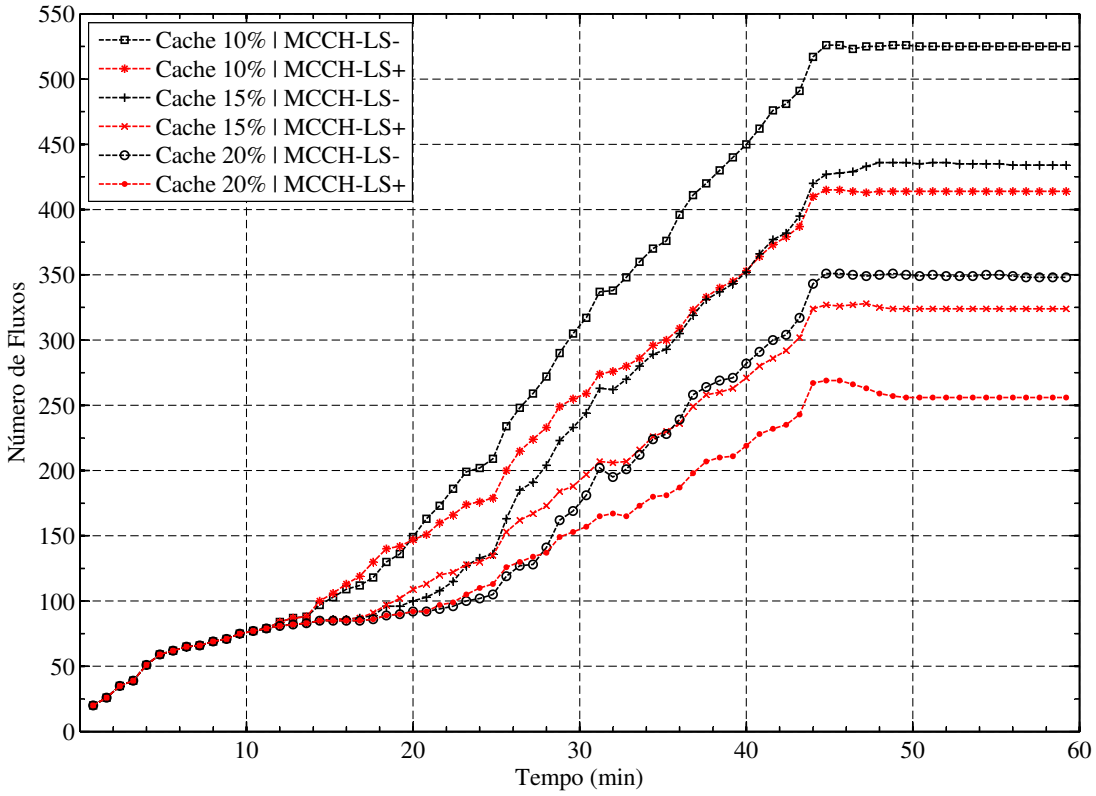
(a) Pico de utilização.



(b) Taxa média de utilização.

A Figura 32 apresenta a comparação desempenho entre MCCH com Otimização no Processo de Criação de Sequências de Link Slots (MCCH-LS+) no melhor ponto de operação apresentado na Figura 31a, e a MCCH sem Otimização no Processo de Criação de Sequências de Link Slots (MCCH-LS-) para diferentes percentuais de tamanho de *cache* principal.

Figura 32 – Gráfico comparativo de desempenho entre a MCCH-LS- e MCCH-LS+.



5.6 Análise dos resultados da MCCH

Nesta seção será avaliado e discutido os resultados de desempenho obtidos pela MCCH quando agregado um segundo nível de *cache* na hierarquia de memória do *proxy*.

5.6.1 Simulação base

Inicialmente, como simulação base, buscou-se avaliar o efeito da política de substituição da *cache* de vítimas no desempenho do sistema, quando comparada com a MCC, que utiliza apenas um nível de *cache* de dados, de modo que as restrições de vazão de leitura e escrita da *cache* de vítimas não influenciassem no desempenho do sistema. Para isso, simulou-se cenários onde ambas possuíssem o mesmo percentual de recursos de armazenamento na *cache* do proxy. A

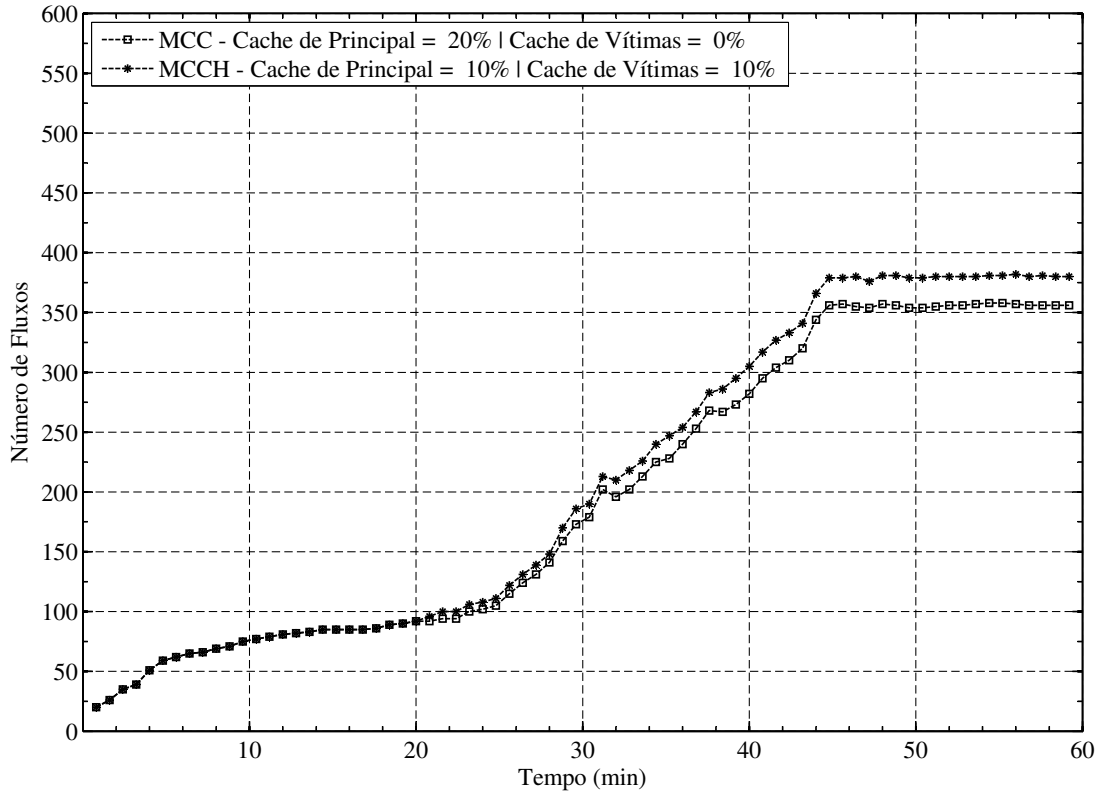
Tabela 6 apresenta os parâmetros usados nas simulações base.

Tabela 6 – Parâmetros de simulação base.

Parâmetro	**Padrão**	**Variação**
Número de vídeos	100	N/A
Duração do vídeo (minutos)	60	N/A
Taxa do vídeo (Mbps)	1	N/A
Intervalo de tempo entre chegadas (segundos)	3,1	N/A
Tamanho do *slot* (segundos)	8	N/A
Tamanho do prefixo (*slots*)	0	N/A
Tamanho da sequência de LS	∞	N/A
Tamanho da *cache* principal (% do tamanho do acervo)	10; 20	N/A
Tamanho da *cache* de vítimas (% do tamanho do acervo)	10	0 – 100
Tempo de simulação (minutos)	60	N/A
Número máximo de clientes	900	N/A
Zipf	0,271	N/A

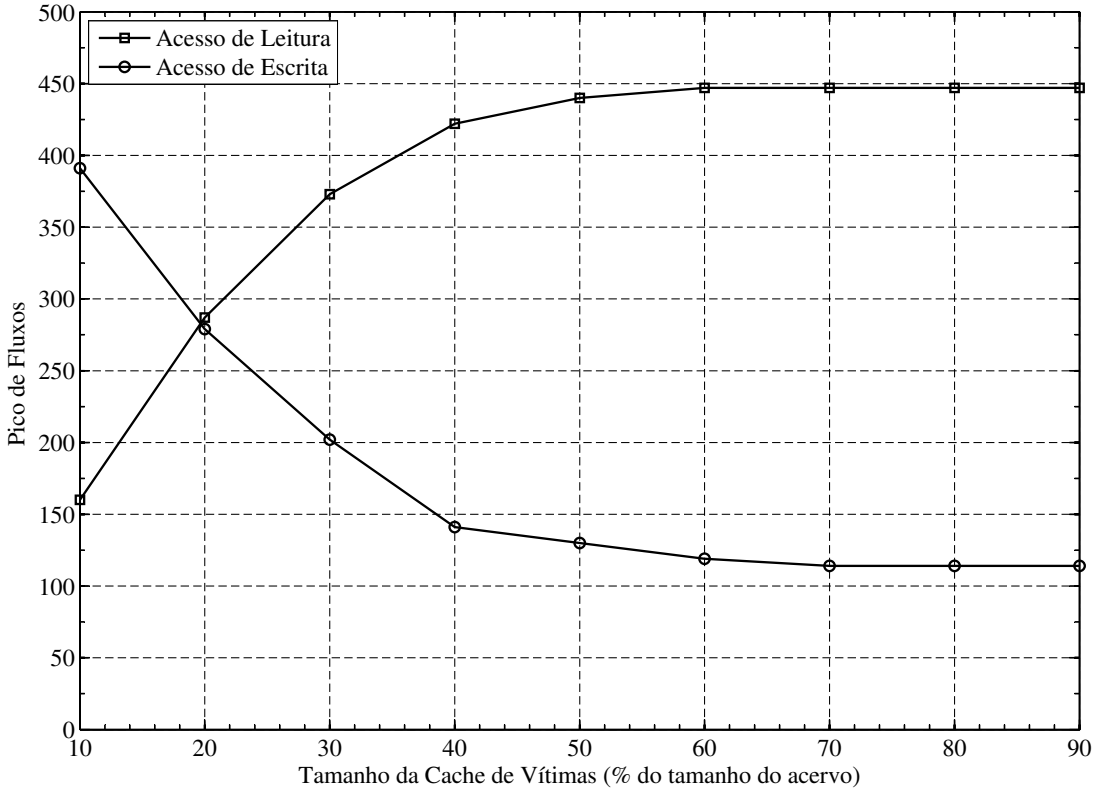
Conforme apresentado no gráfico da Figura 33, a diferença entre o número de fluxos oriundos do servidor nas duas técnicas é pequena , portanto, ambas possuem taxas de *miss* similares, visto que, um fluxo de vídeo é gerado em detrimento de um *miss* na *cache* do *proxy*. A Figura

Figura 33 – Gráfico comparativo de desempenho entre a MCC e a MCCH.



Posteriormente, buscou-se determinar a capacidade máxima de vazão de leitura e escrita do dispositivo de memória utilizado como *cache* de vítimas necessárias para atender cenários com alta demanda ao sistema, objetivando determinar os requisitos mínimo de vazão para que o dispositivo de memória não se tornasse o fator limitante na escalabilidade do sistema. Para isso, simulou-se a MCCH para percentuais de tamanho de *cache* de vítimas entre 10% e 90%, fixando o percentual do tamanho da *cache* principal em 10%, conforme ilustrado na Figura 34.

Figura 34 – Requisitos de vazão da *cache* de vítimas.



A Figura 34 apresenta a influência do tamanho da *cache* de vítimas na utilização dos recursos de vazão do dispositivo, demonstrando os picos de fluxos[19] para acessos de leitura e escrita atingidos durante o período de simulação. A partir dos valores apresentados no gráfico é possível determinar os requisitos de vazão necessários para que a *cache* de vítimas não se torne o gargalo do sistema, ocasionando a degradação da QoS. Note que para o percentual de 10%, o pico de fluxos de escrita é relativamente superior ao de leitura. Isso ocorre porque a MCCH não possui um mecânismo de selação dos vídeos, de modo a restringir a cópia do conteúdo dos vídeos de baixa popularidade do nível superior da hierarquia para o nível inferior, logo, a *cache*

[19] O pico de fluxos representa o número máximo de fluxos de leitura ou escrita na *cache* de vítimas atingidos durante o período de simulação, onde cada fluxo é igual a taxa do vídeo.

principal não irá se benificar do principio de localidade espacial e temporal dos *slots* copiados pertencentes aos vídeos de baixa popularidade, pois, estes vídeos terão seus recursos liberados antes de terem sido acessados pelo nível superior da hierarquia , visto que, a distância temporal entre os *buffers* é relativamente grande quando comparados com os vídeos de maior popularidade. Note que, a utilização de dispositivos de memória que possuem limitações no número de escrita, por exemplo, tecnologias de memória *Flash* SSD, degradará o tempo de vida do dispositivo, e, consequentemente, a QoS do sistema.

Embora não implementado um mecanismo para seleção dos vídeos à serem copiados para a *cache* de vítimas, é possível reduzir o pico de acessos de escrita restringido a quantidade de recursos de vazão que poderá ser utilizado pelo sistema para cópia do conteúdo para a *cache* de vítimas, de modo que não influencie na degradação do pico de acessos de leitura.

As Figuras 35 e 36 demonstram de forma geral, o comportamento da MCCH quando utilizado um segundo nível de *cache* na hierarquia de memória do *proxy*.

Figura 35 – Pico de utilização da MCCH para diferentes percentuais de tamanho de *cache* de vítimas.

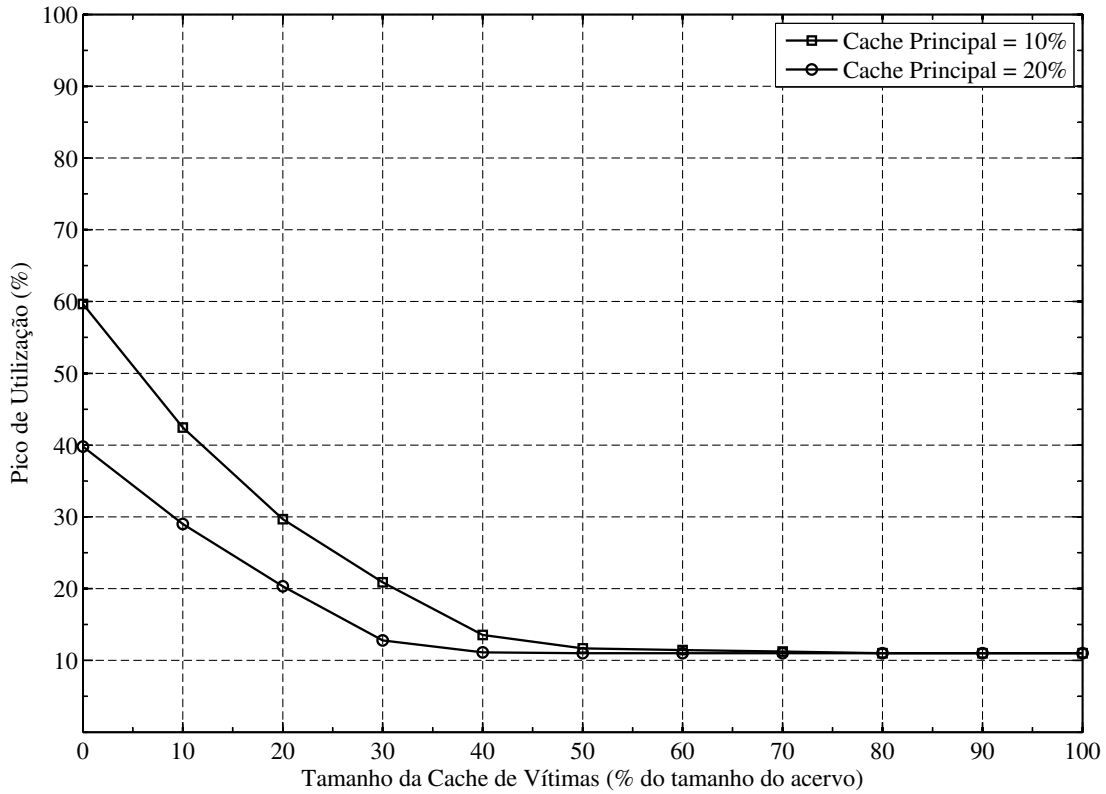
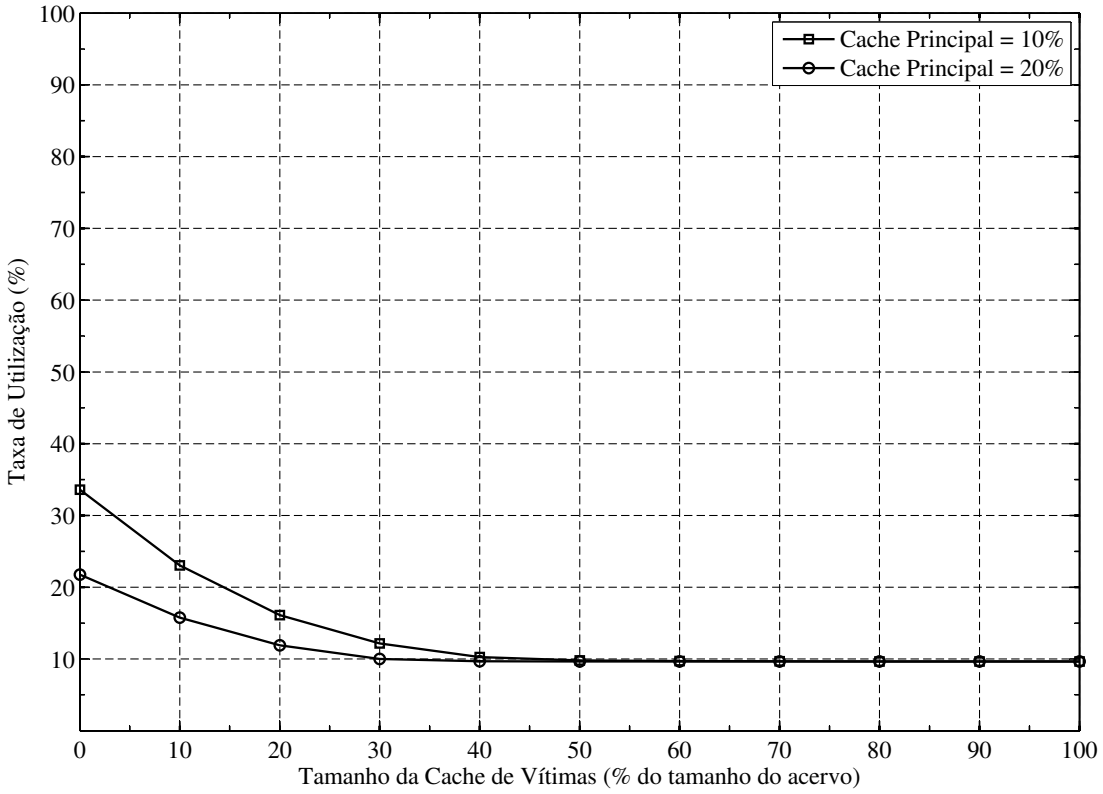


Figura 36 – Taxa de utilização da MCCH para diferentes percentuais de tamanho de *cache* de vítimas.



5.6.2 Análise da vazão

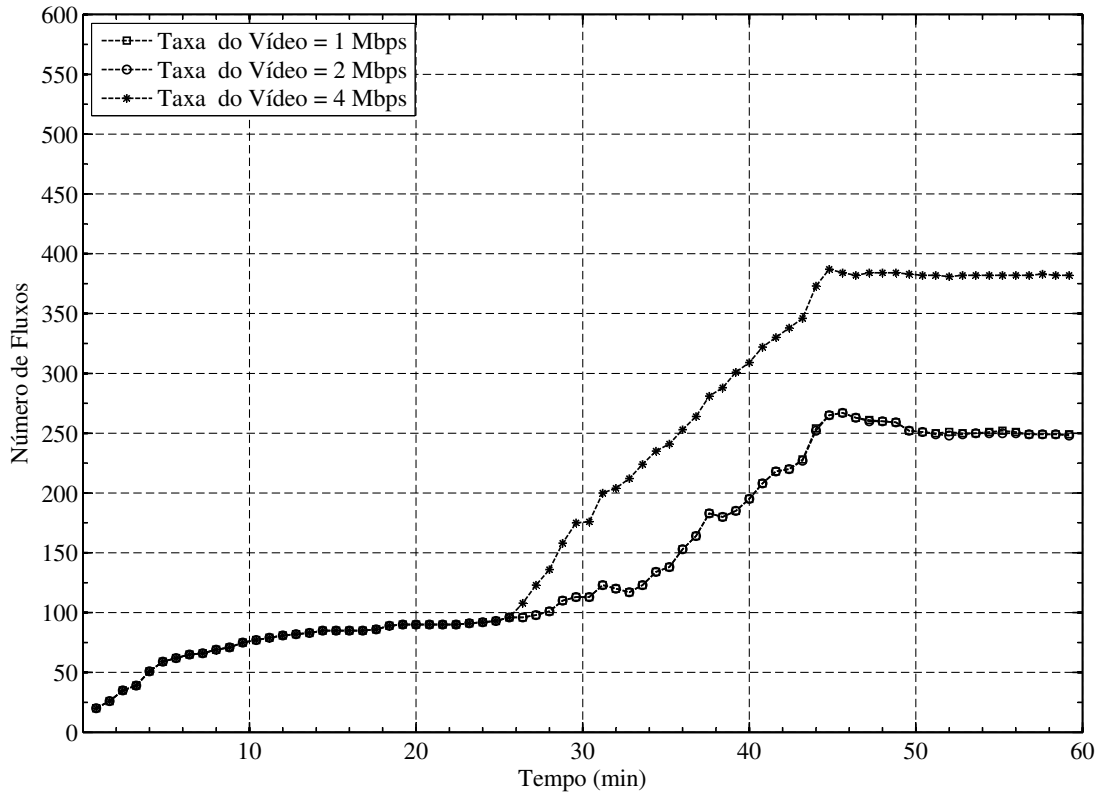
Nesta subseção são analisados os efeitos das limitações na capacidade de vazão de leitura e escrita do dispositivo de memória usado como *cache* de vítimas na QoS do sistema. Os parâmetros da *cache* de vítima para as simulações efetuadas foram baseados em dados referenciados em estudos de Ryu, Kim e Ramachandran (2011), além de trabalhos relacionados a MCC (GRANADO, 2010), conforme apresentado na Tabela 7.

O gráfico da Figura 37 demonstra a influência da limitação na vazão da *cache* de vítima na QoS do sistema. Note que, a partir de aproximadamente 26 minutos, a curva representada pela taxa de 4 Mbps tende a aumentar o número de fluxos oriundos do servidor, em relação as curvas de 1 e 2 Mbps. Para investigar esta degradação na QoS do sistema, foi analisada a vazão de leitura e escrita da *cache* de vítimas para os vídeos de taxa de 1, 2 e 4 Mbps.

Tabela 7 – Parâmetros de simulação da MCCH.

Parâmetro	**Valor**
Número de vídeos	100
Duração do vídeo (minutos)	60
Taxa do vídeo (Mbps)	1; 2; 4
Intervalo de tempo entre chegadas (segundos)	3,1
Tamanho do *slot* (segundos)	8
Tamanho do prefixo (*slots*)	2
Tamanho da sequência de LS	∞
Tamanho da *cache* principal (% do tamanho do acervo)	10
Tamanho da *cache* de vítimas (% do tamanho do acervo)	20
Capacidade de Vazão de Leitura (Mbps)	640
Capacidade de Vazão de Escrita (Mbps)	480
Tempo de simulação (minutos)	60
Número máximo de clientes	900
Zipf	0,271

Figura 37 – Desempenho da MCCH para cenários de diferentes taxas de vídeo.

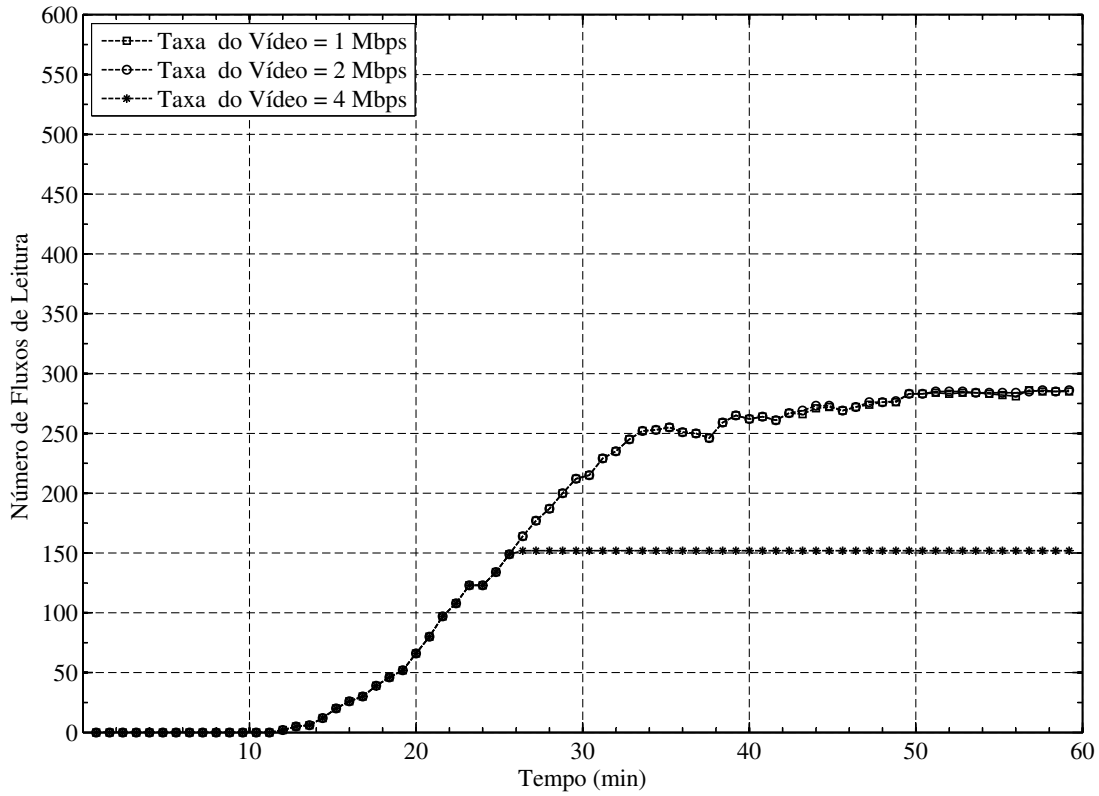


Análise da Vazão de Leitura

Para realizar a análise da vazão de leitura, a curva apresentada na Figura 38 que demonstra a influência da taxa do vídeo no número de fluxos para acessos de leitura na *cache* de vítimas em função do tempo, foi dividida em três faixas de tempo.

Na primeira faixa, representada pelo intervalo entre 0 e 11 minutos, o número de fluxos é zero, pois a *cache* do *proxy* está inicialmente vazia, logo, a MCCH deverá recorrer ao servidor para obtenção dos blocos de vídeo requisitados pela aplicação.

Figura 38 – Influência da vazão de leitura da *cache* de vítimas no desempenho do sistema.



A segunda faixa representada pelo intervalo entre 11 e 25 minutos, o número fluxos passa a aumentar gradativamente, visto que, a partir deste instante, os slots de vídeo a serem vitimados do nível superior da hierarquia são copiados para o nível inferior. Portanto, o *slot* WS dos *buffers*[20] teve suas requisições providas pela *cache* de vítimas (nível inferior), reduzindo-se, assim, o número de fluxos oriundos do servidor, e, consequentemente, aumentando a QoS do sistema.

[20]Com exceção ao primeiro *buffer* criado para cada vídeo, visto que, o *slot* WS destes *buffers* terão que solicitar ao servidor o conteúdo de vídeo.

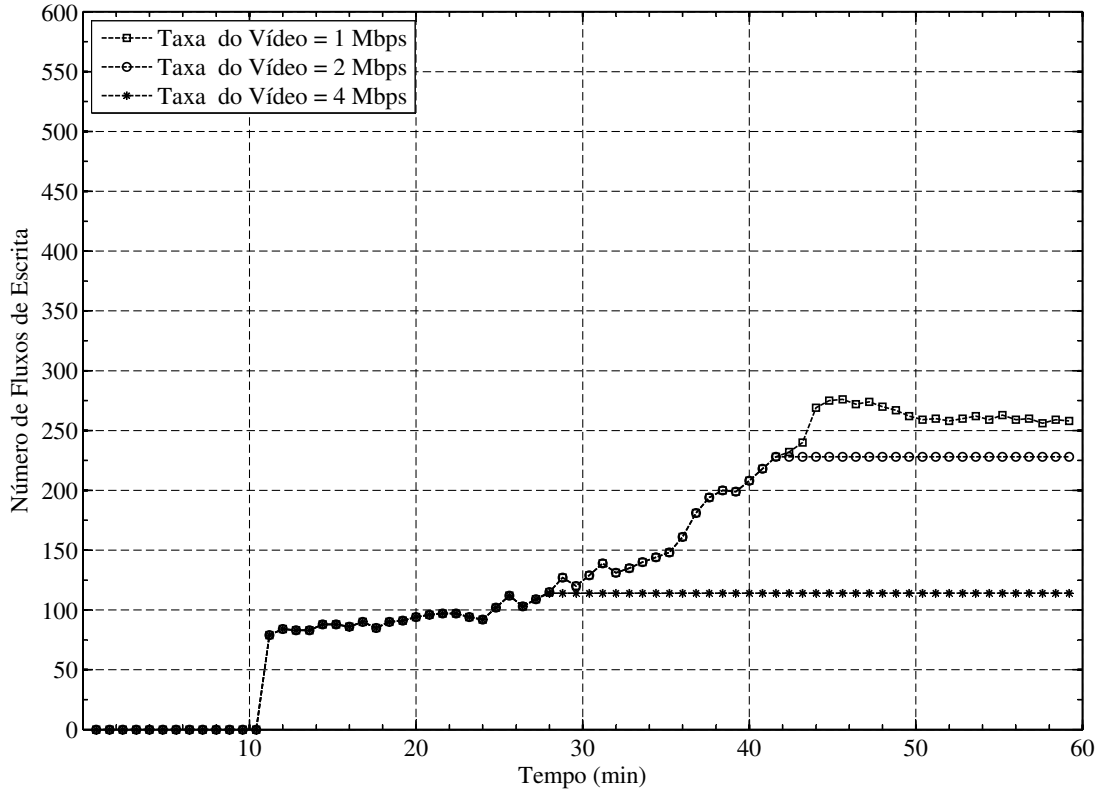
Entretanto, em aproximadamente 25 minutos, a *cache* de vítimas atinge sua capacidade máxima de armazenamento iniciando o processo de liberação de recursos. Sendo assim, na terceira faixa de tempo, indicada pelo intervalo entre 25 e 60 minutos, o número de fluxos tende a se estabilizar. Isto ocorre porque os vídeos de baixa popularidade, que possuem longa distância temporal entre os *buffer*, não terão seus recursos mantidos na *cache* de vítimas, sendo assim, o nível superior da hierarquia não se beneficiou da localidade temporal e espacial dos *slots* de vídeos de baixa popularidade previamente copiados.

Contudo, a partir de 26 minutos, diferentemente do comportamento apresentado pelas curvas de 1 e 2 Mbps, a curva de 4 Mbps, se estabilizou em 152 fluxos. Isso ocorreu, porque a vazão de leitura do dispositivo atingiu a vazão máxima, portanto, a partir deste instante, a MCCH tece que recorrer ao servidor para obtenção dos blocos requisitados, embora os mesmo estejam presentes na *cache* de vítimas. Esta limitação na vazão de leitura ocasiona a degradação da QoS do sistema, conforme apresentado na Figura 37.

Análise da Vazão de Escrita

A curva ilustrada na Figura 39 demonstra a influência da taxa do vídeo no número de fluxos para acessos de escrita na *cache* de vítimas em função do tempo para diferentes taxas de vídeo.

Figura 39 – Influência da vazão de escrita da *cache* de vítimas no desempenho do sistema.



Até aproximadamente 11 minutos, a *cache* principal não atingiu sua capacidade máxima de armazenamento, logo, o conteúdo de nenhum *slot* de vídeo será descartado pela MCCH, portanto, o número de fluxos de escrita é zero. Entretanto, a partir de 11 minutos até aproximadamente 25, o número de fluxos de escrita se estabiliza, embora a quantidade de *slots* liberados pelo nível superior da hierarquia aumente. Este comportamento ocorre porque o algoritmo de substituição da MCC foi adaptado para inicialmente liberar o conteúdo dos *slots* de vídeo da *cache* principal que estão presentes em ambos os níveis da hierarquia, desta forma, somente os *slots* que não possuem o conteúdo na *cache* de vítimas foram copiados.

No instante 25, a *cache* de vítimas atinge sua capacidade máxima de armazenamento, iniciando o processo de liberação de recursos. Portanto, a partir deste momento, o número de *slots* com conteúdo duplicado nos dois níveis da hierarquia, a serem liberados pelo algoritmo de substituição da *cache* principal, está limitado, portanto, o aumento no número de *buffers* criados para atender a requisição de inclusão de novos clientes, ocasionou o aumento na quantidade de recursos a serem liberados pela *cache* de principal, e, consequentemente, aumentará o número de *slots* copiados para a *cache* de vítimas. Todavia, diferentemente do comportamento apresentado pela curva de 1 Mbps, o número de fluxos de escrita para as curvas de 2 e 4 Mbps estabilizaram-se em 114 e 228, respectivamente. Isso ocorre, porque a partir destes instantes, a vazão de escrita da *cache* de vítimas atingiu sua capacidade máxima, restringido, assim, o número de *slots* copiados.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi proposta e avaliada uma nova técnica para aumentar a eficiência de *proxies* dinâmicos de VoD com o uso de uma *cache* hierarquizada que utiliza a memória RAM como *cache* principal de dados recebidos do servidor de vídeo, e um segundo nível de memória como *cache* de vítimas para os dados descartados pela *cache* principal, sendo esta proposta fundamentada em um estudo sobre a evolução dos dispositivos de memória RAM, *Flash* e disco magnético.

A partir dos resultados obtidos nas simulações da MCCH, quando comparados com a técnica da MCC para os mesmos percentuais de recursos de armazenamento na *cache* do *proxy*, pode-se observar que, a hierarquização do sistema de memória do *proxy* obteve desempenho semelhante ao apresentado pela MCC para vídeos de até 2 Mbps. Entretanto, para vídeos de 4 Mibps, verificou-se a degradação na qualidade de serviço do sistema, pois a capacidade de vazão de leitura e escrita destes dispositivos se tornaram o gargalo do sistema. Contudo, as características de vazão do dispositivo de *cache* de vítimas usadas para a análise de sensibilidade do sistema eram relativamente limitantes. Para resolver este impasse, podem ser utilizadas tecnologias de memórias mais modernas que atualmente possuem capacidades máximas de vazão de leitura e escrita acima da vazão de 1 Gbps adotada como referência para a rede.

Além do mais, para percentuais de *cache* principal e de vítimas de 10% e 10%, respectivamente, do tamanho do acervo, a MCCH apresentou um alto pico de escrita na *cache* de vítimas e baixo pico de leitura, indicando que o algoritmo usado deverá utilizar critérios mais seletivos para a cópia do conteúdo a ser descartado pela *cache* principal, de modo a limitar a cópia de vídeos de baixa popularidade, pois para pequenos percentuais de *cache* de vítimas, os conteúdos destes *slots* serão liberados antes de serem acessados pelo nível superior da hierarquia.

Este trabalho também contribuiu na otimização do desempenho da técnica da MCC, usada como base para este estudo, investigando-se o efeito do tamanho da sequência de LS na utilização dos recursos de enlace entre o servidor e o *proxy*, obtendo-se reduções significativas de 21% no pico de utilização para o percentual de tamanho de *cache* principal de 10% do tamanho do acervo, equivalendo-se ao mesmo desempenho de uma *cache* de 15% que não possui mecanismo de limitação no processo de criação de sequências de LS. Entretanto, as análises realizadas, apesar

de superficiais, servirão como marco inicial para futuras análises objetivando-se determinar os principais parâmetros que influenciam neste processo, a fim de criar um mecanismo que determine em tempo de execução o tamanho máximo da sequência de LS que poderá ser criada para unir os *buffers* de modo a maximizar o desempenho da MCCH. Além do mais, está em fase pré-operacional o desenvolvimento do protótipo da MCCH, construída para permitir contratar, futuramente, resultados práticos com os resultados obtidos nas simulações.

E por fim, merece ser avaliado em trabalhos futuros a análise da influência da taxa de substituição na *cache* de vítimas no tempo de vida do dispositivo, visto que, determinadas tecnologias, possuem limitações no número de escrita.

REFERÊNCIAS

BRAGATO, L. L. S. *Implementação e Avaliação de um Sistema de Vídeo Sob Demanda Baseado em Cache Cooperativa Colapsada de Vídeo*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2006.

GRANADO, A. C. *Avaliação Experimental da Memória Cooperativa Colapsada para Sistemas de Vídeo sob Demanda*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2010.

HULU. *Hulu*. 2011. [Online; accessado em 17-Julho-2011]. Disponível em: <http://www.hulu.com>.

ISHIKAWA, E. *Memória Cooperativa para Distribuição de Vídeo Sob Demanda*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2003.

JOUPPI, N. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In: *Computer Architecture, 1990. Proceedings., 17th Annual International Symposium on*. [S.l.: s.n.], 1990. p. 364 –373.

KUROSE, J. F.; ROSS, K. W. *Redes de computadores e a Internet: uma abordagem top-down*. 3th. ed. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2006. ISBN 978-85-88639-18-8.

LEVENTHAL, A. Flash storage memory. *Commun. ACM*, ACM, New York, NY, USA, v. 51, p. 47–51, July 2008. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1364782.1364796>.

MICROSOFT. *Microsoft to Acquire Skype*. 2011. [Online; accessado em 03-Julho-2011]. Disponível em: <http://www.microsoft.com/presspass/press/2011/may11/05-10corpnewspr.mspx>.

NETFLIX. *Netflix*. 2011. [Online; accessado em 17-Julho-2011]. Disponível em: <http://www.netflix.com>.

RAMBUS. *Challenges and Solutions for Future Main Memory*. 2009. [Online; accessado em 03-Julho-2011]. Disponível em: <http://www.rambus.com/assets/documents/products/future_main_memory_whitepaper.pdf>.

RYU, M.; KIM, H.; RAMACHANDRAN, U. Impact of flash memory on video-on-demand storage: analysis of tradeoffs. In: *Proceedings of the second annual ACM conference on Multimedia systems*. New York, NY, USA: ACM, 2011. (MMSys '11), p. 175–186. ISBN 978-1-4503-0518-1. Disponível em: <http://doi.acm.org/10.1145/1943552.1943577>.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Sistemas Operacionais com Java*. 7th. ed. [S.l.]: John Wiley & Sons, 2008. ISBN 978-85-352-2406-1.

TANG, X.; XU, J.; CHANSON, S. T. *Web Content Delivery (Web Information Systems Engineering and Internet Technologies Book Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN 0387243569.

WIKIPEDIA. *Solid-state drive*. 2011. [Online; accessado em 03-Julho-2011]. Disponível em: <http://en.wikipedia.org/wiki/Solid-state_drive>.

WU, D. et al. Streaming video over the internet: approaches and directions. *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 11, n. 3, p. 282 –300, mar 2001. ISSN 1051-8215.

YOUTUBE. *Youtube*. 2011. [Online; accessado em 17-Julho-2011]. Disponível em: <http://www.youtube.com>.