Buffering para Otimização de Sistemas Interativos de VoD *

Carlo K. da S. Rodrigues , Luiz J. H. Filho , Rosa M. M. Leão

[1] Universidade Federal do Rio de Janeiro
COPPE/PESC, Rio de Janeiro RJ 21941-972, CxP 68511, Brasil

kleber@land.ufrj.br, ljhfilho@land.ufrj.br, rosam@land.ufrj.br

Abstract. *This work introduces two novel buffer management policies for interactive VoD systems: Unique Buffer (UB) and Precise Buffer (PB). The former uses a buffer which is shared by all clients playing a same single object, and the latter is based on the idea of quantifying the information available in the local buffer before making any stream merge decisions. Through simulations we validate our proposals and carry out a detailed competitive analysis. The final results show for instance that, comparing to more conventional approaches, we may have optimizations in the range of 5%–98% at bandwidth peak values, bandwidth average values and system complexity.*

Resumo. *Este trabalho apresenta duas novas estratégias de gerência de buffer de um sistema de VoD com interatividade: Unique Buffer (UB) e Precise Buffer (PB). A primeira tem por idéia principal o emprego de um único buffer compartilhado por todos os clientes, e a segunda baseia-se na avaliação da quantidade de informação disponível localmente para decidir sobre sua utilização. Por meio de simulações, validamos as estratégias UB e PB e realizamos detalhadas análises competitivas com outras propostas da literatura. Comparativamente com abordagens mais convencionais, podemos obter otimizações de 5%–98% em valores de consumo de banda, pico de banda e complexidade do sistema.*

1. Introdução

O serviço de vídeo sob demanda (VoD) com interatividade tem recebido crescente atenção nos últimos anos. Idealmente, para implementar este serviço, o servidor deve alocar um canal de transmissão de dados exclusivo para cada cliente, permitindo a emulação de ações de interatividade de um aparelho comum de vídeo cassete (VCR). Entretanto, como a banda do servidor é um recurso limitado, esta implementação é inviável quando muitos usuários simultâneos precisam ser atendidos. Para fins de estudo, vemos o aumento da escalabilidade desses sistemas em duas direções: Técnicas de Compartilhamento de Banda [Rodrigues and Leão 2007, Rodrigues and Leão 2006, Rocha et al. 2005, Eager et al. 1999] e Estratégias de Gerência de *Buffer* [Abram-Profeta and Shin 1998, Poon and Lo 1999, Netto 2004].

As técnicas de compartilhamento de banda são algoritmos que definem a forma de atendimento de requisições provenientes de clientes que desejam visualizar objetos multimídia de um servidor. Estes algoritmos respondem a questões como: que fluxo o cliente deve escutar e por quanto tempo? deve ser aberto um novo fluxo e por quanto tempo?

*Este trabalho é parcialmente financiado pelo CNPq e Faperj.

que fluxos devem ser unidos e em que instante de tempo? Já as estratégias de gerência de *buffer* são políticas que definem a utilização do *buffer* devido a essas requisições. Estas políticas respondem a questões como: as informações transmitidas pelo servidor devem ficar armazenadas durante toda a sessão do cliente? quanto de informação deve haver em *buffer* para que o cliente abandone o fluxo sendo transmitido pelo servidor e passe a recuperar dados apenas do *buffer* local? Na prática, as técnicas de compartilhamento de banda são utilizadas com gerência de *buffer*, constituindo uma única solução integrada.

Existe uma significativa diversidade de técnicas de compartilhamento de banda na literatura. Os trabalhos de [Banker and et al. 1994, Dan et al. 1995, Almeroth and Ammar 1996], por exemplo, têm uma implementação bem simples, baseada na localização do canal que transmite a informação (unidade, bloco, etc.) mais próxima àquela de fato solicitada pelo cliente. As técnicas de [Liao and Li 1997, Abram-Profeta and Shin 1998] discutem a união de fluxos em andamento no sistema. A idéia é abrir um fluxo para atender à toda ação de interatividade do cliente, e a união de fluxos é conseguida pela escuta concomitante de dois fluxos. A proposta de [Poon and Lo 1999] utiliza uma taxa de transmissão duas vezes maior que a normal para permitir a união de fluxos. Os trabalhos de [Ma and Shin 2001, Netto 2004, Rocha et al. 2005, Rodrigues and Leão 2005, Rodrigues and Leão 2006] são mais recentes e baseiam-se nos paradigmas de acesso seqüencial Patching e Hierarchical Stream Merging (HSM) [Hua et al. 1998, Eager et al. 1999]. Estas técnicas diferenciam-se entre si na condição de que a estrutura de união de fluxos tem um determinado número de níveis permitidos. Por exemplo, quando são permitidos três níveis, um dado cliente pode escutar um outro cliente que, por sua vez, escuta um outro cliente do sistema, constituindo assim uma árvore de altura três [Bar-Noy et al. 2002]. Em relação ao gerenciamento de *buffer*, temos basicamente duas abordagens formais apresentadas na literatura. A primeira (p.ex., [Viswanathan and Imielinski 1995, Chan and Chang 2001]) traz a idéia de emprego de um *buffer* local para armazenar dados para visualização futura e permitir a sincronização da informação recebida a partir da escuta concomitante de diferentes fluxos de dados. E a segunda abordagem, além de permitir a sincronização da informação vinda de fluxos distintos, também tem a idéia de que, durante uma mesma sessão, o cliente deve manter armazenado em *buffer* local todas as unidades de dados transmitidas pelo servidor (p.ex., [Netto 2004, Rodrigues and Leão 2005, Rocha et al. 2005]).

A principal contribuição deste trabalho é o estudo de duas novas estratégias de gerenciamento de *buffer*: Unique Buffer (UB) e Precise Buffer (PB). A estratégia UB fundamenta-se na idéia de empregar um único *buffer* compartilhado por todos os clientes de uma rede local que desejam assistir a um mesmo objeto, e a estratégia PB é baseada na condição de se verificar a quantidade de informação já disponível em *buffer* local para decidir sobre abertura e extinção de fluxos no sistema. Por meio de simulações, validamos as novas estratégias propostas e realizamos análises comparativas detalhadas com outras propostas da literatura, utilizando diferentes métricas de performance. Dentre outras constatações, os resultados obtidos mostram principalmente que, em comparação com as abordagens mais convencionais, podemos obter otimizações no intervalo de $5\% - 98\%$ em valores médio de banda, pico de banda e complexidade do sistema.

O restante deste texto tem a seguinte organização. A Seção 2 revisa os algoritmos de duas recentes técnicas de compartilhamento de banda. Na Seção 3, temos as estratégias

de gerência de *buffer* mais usadas na literatura. As novas estratégias UB e PB estão na Seção 4. A Seção 5 traz os mais importantes resultados de simulação obtidos. Por último, conclusões e propostas para trabalhos futuros constituem a Seção 6.

2. Interatividade e compartilhamento de banda

Considere um grupo de clientes recebendo fluxos de dados relativos a um objeto multimídia armazenado em um servidor através de uma rede de comunicação. O objeto é dividido em unidades de dados u de mesmo tamanho e de duração de uma unidade de tempo. A rede tem *multicast* implementado (IP ou aplicação). Os clientes têm acesso não-seqüencial, i.e., podem realizar ações VCR. O cliente possui *buffer* local capaz de armazenar pelo menos metade do objeto requisitado e sua banda é duas vezes a taxa de exibição desse objeto. Por fim, os fluxos de dados transmitem na mesma taxa de exibição do objeto. Doravante, salvo informado diferentemente, assumimos estas suposições no restante deste texto [Bar-Noy et al. 2002, Almeida et al. 2001, Rocha et al. 2005, Ma and Shin 2001, Bar-Noy et al. 2002].

A seguir revisamos os algoritmos de operação de duas técnicas de compartilhamento de banda utilizadas nos experimentos deste trabalho: Patching Interativo Eficiente (PIE) e Merge Interativo (MI) [Rodrigues and Leão 2005, Rodrigues and Leão 2006, Rodrigues 2006]. Elas foram escolhidas por agregar simultaneamente eficiência e simplicidade [Rodrigues and Leão 2005, Rodrigues and Leão 2006, Rodrigues 2006].

2.1. Patching Interativo Eficiente – PIE

Esta técnica é baseada no paradigma de Patching e tem, como principal premissa, a manutenção da estrutura de união de fluxos em no máximo dois níveis. Além disso, a chegada de uma requisição do cliente é o único evento que provoca uma união de fluxos. A seguir descrevemos brevemente o algoritmo de operação.

EVENTO ÚNICO: Requisição para unidade u_r do objeto.

- Inicialmente busca-se um fluxo *multicast* S_{before}, i.e., um fluxo transmitindo uma unidade de dados igual ou anterior à u_r, dentro de um limiar de tempo δ_{before}. Ou seja, S_{before} pode estar transmitindo desde a unidade u_r até a unidade $u_r - \delta_{before}$. Se S_{before} existe, então a requisição é atendida pelo mesmo.
- Se S_{before} não existe, é verificado se existe um fluxo multicast S_{after}, i.e., um fluxo transmitindo uma unidade de dados posterior à u_r, dentro de um limiar de tempo δ_{after}. Ou seja, S_{after} pode estar transmitindo desde a unidade u_r+1 até a unidade $u_r+\delta_{after}$. Se S_{after} existe, o servidor diz ao cliente para escutá-lo e abre um fluxo *unicast* (i.e., um *patch*) para transmitir as unidades perdidas inicialmente.
- Se S_{after} não existe, um novo fluxo *multicast* S_{new} é aberto para servir a requisição por u_r. É ainda verificado se existe um fluxo *multicast* S_{merge}, i.e., um fluxo transmitindo uma unidade de dados anterior à u_r, dentro de um limiar de tempo δ_{merge}. O fluxo S_{merge} não pode possuir *patches* associados a ele, pois um dos objetivos é manter a estrutura de união de fluxos em no máximo dois níveis.
- Se S_{merge} existe, então o fluxo S_{new} é o seu alvo e os clientes de S_{merge} devem escutar também o fluxo S_{new} para que, eventualmente, S_{merge} se una ao fluxo S_{new}. Caso S_{merge} não exista, então nada mais precisa ser feito.

2.2. Merge Interativo – MI

Esta técnica é baseada no paradigma de HSM e tem sua estrutura de união de fluxos de profundidade (i.e., número de níveis) ilimitada. A seguir tem-se brevemente o algoritmo de operação, onde ressaltam-se os três eventos considerados para a união de fluxos.

- **EVENTO 1: Requisição para unidade u_r do objeto.** Imediatamente abre-se um novo fluxo S_{new} para atender esta requisição. Simultaneamente, o servidor busca por um fluxo em andamento no sistema que esteja transmitindo uma unidade de dados posterior à u_r. Seja S_t este fluxo. Se S_t existe, então o cliente de S_{new} também vai escutá-lo de tal sorte que S_{new} e S_t possam ser unidos mais à frente.
- **EVENTO 2: Término de um fluxo S_j que é o fluxo alvo de um outro fluxo S_i.** Nesta situação, S_j termina antes de ser alcançado por S_i. Os clientes de S_i ficam órfãos (i.e., não possuem mais fluxo a alcançar) e os conteúdos armazenados em seus *buffers*, devido à escuta de S_j, são descartados. O servidor busca um outro fluxo que transmite uma unidade de dados posterior àquela atual do fluxo S_i. Seja S_{subs} este fluxo. Se S_{subs} existe, ele se torna alvo de S_i. Caso S_{subs} termine antes de ser alcançado por S_i, a busca por um outro fluxo S_{subs} é repetida.
- **EVENTO 3: União do fluxo S_i e seu alvo S_j. Seja S_m o fluxo resultante.** O servidor imediatamente busca um fluxo que esteja transmitindo uma unidade de dados posterior àquela do fluxo S_m. Seja S_t este fluxo. Se S_t existe, os clientes de S_m, que originalmente pertenciam ao fluxo S_i, vão também escutá-lo para eventualmente se unir a ele. Já os clientes de S_m, que originalmente pertenciam ao fluxo S_j, não são afetados. Caso S_t não exista, nada precisa ser feito.

3. Estratégias de gerência de buffer

Aqui revisamos duas propostas da literatura que possuem implementações bastante simples e, ainda, são consideravelmente competitivas quanto à otimização do sistema. Estas propostas são avaliadas principalmente em [Netto 2004, Rodrigues and Leão 2005, Rocha et al. 2005, Rodrigues and Leão 2006].

3.1. Estratégia Simple Buffer

A principal função da estratégia SB é permitir a sincronização da escuta de dois fluxos simultâneos que transmitem o mesmo objeto. O tamanho do *buffer* local do cliente é determinado pelo tempo de escuta simultânea permitido para o cliente, e nunca precisa ser maior que a metade da duração total do objeto [Bar-Noy et al. 2002]. O primeiro fluxo é utilizado para transmitir as informações (unidades de dados) que o cliente precisa visualizar instantaneamente. Já o segundo fluxo transmite unidades de dados que são visualizadas a partir de um instante futuro. As unidades de dados do segundo fluxo são armazenadas em *buffer* local. No instante em que a unidade de dados do primeiro fluxo alcançar a primeira unidade de dados armazenada em *buffer* (proveniente do segundo fluxo), o cliente deixa de escutar o primeiro fluxo e passa a ler diretamente de seu *buffer*. O primeiro fluxo pode então ser extinto, redundando em economia de banda do sistema. O segundo fluxo permanece ativo e tendo suas unidades de dados armazenadas em *buffer* local. A Figura 1(a) ilustra esta estratégia. Podemos observar as unidades de dados de dois fluxos $multicast$, S_i e S_j, que estão sendo transmitidos simultaneamente. As unidades provenientes de S_i, desde a unidade u_i até a unidade $u_j - 1$, são exibidas sucessivamente a partir do instante presente t, enquanto que as unidades de S_j, desde a unidade u_j, começam a ser exibidas apenas a partir do instante $t + \delta$.

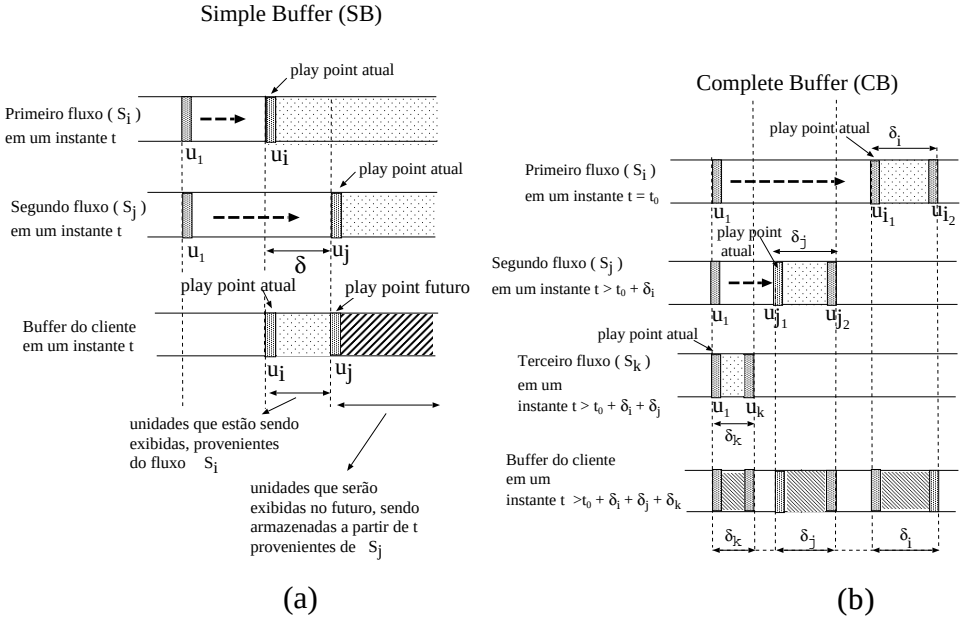

Figura 1. (a) Estratégia Simple Buffer; (b) Estratégia Complete Buffer.

3.2. Estratégia Complete Buffer

A proposta Complete Buffer (CB) fundamenta-se na idéia de que o comportamento interativo do cliente o faz naturalmente realizar acessos a unidades de dados anteriormente já visualizadas. Sendo isso verdade, a otimização de banda pode ser conseguida ao se permitir que as unidades de dados transmitidas pelo servidor sejam permanentemente armazenadas em *buffer* local durante uma mesma sessão, i.e., enquanto o cliente estiver visualizando o objeto em uma mesma sessão, os dados de seu *buffer* local não serão removidos. Quando o cliente realiza uma requisição para uma dada unidade de dados do objeto, primeiramente é verificado se a unidade de dados solicitada já não está em *buffer* local. Se a resposta for afirmativa, o cliente é atendido localmente, sem abertura de um novo fluxo e sem a tentativa de realizar compartilhamento com qualquer outro fluxo. Nesta estratégia o tamanho do *buffer* local é igual ao tamanho do objeto inteiro.

A Figura 1(b) ilustra esta estratégia mostrando o *buffer* de um cliente após este ter realizado a escuta de três fluxos independentes em três períodos distintos de tempo. Primeiramente, no instante de tempo $t = t_0$, o cliente escuta o fluxo S_i e armazena em *buffer*, desde a unidade u_{i_1} até a unidade u_{i_2}. Depois, no instante $t > t_0 + \delta_i$, o cliente escuta o fluxo S_j e armazena em *buffer* desde a unidade u_{j_1} até a unidade u_{j_2}. Finalmente, no instante $t > t_0 + \delta_i + \delta_j$, o cliente escuta o fluxo S_k, armazenando desde a unidade u_1 até a unidade u_k. No instante $t > t_0 + \delta_i + \delta_j + \delta_k$, o cliente tem armazenado em *buffer* um total de $\delta_i + \delta_j + \delta_k$ unidades (agrupadas em três segmentos não-consecutivos), as quais podem então ser utilizadas para atendimento de futuras requisições do próprio cliente.

4. Novas estratégias de gerência de buffer

4.1. Estratégia Unique Buffer

A proposta Unique Buffer (UB) aproveita o efeito conjunto das requisições para um mesmo objeto devido aos vários clientes existentes no sistema e localizados em uma

mesma rede local. O cliente deixa de ter um *buffer* local exclusivo, passando a existir apenas um *buffer* único compartilhado – localizado em um nó (rede) de acesso – para atendimento de todos os clientes que desejam assistir a um mesmo objeto. As solicitações de um cliente no instante t podem então se beneficiar de quaisquer solicitações ocorridas em instantes anteriores a t. Dessa forma, dois efeitos são percebidos: (i) o cliente se beneficia de suas próprias requisições anteriores, assim como na proposta original de CB, e (ii) o cliente se beneficia de requisições anteriores e provenientes de outros clientes. O tamanho do *buffer* único é igual ao tamanho do objeto inteiro.

A Figura 2 ilustra um sistema no qual um servidor armazena n objetos de interesse de um total de j clientes ($C_1, \ldots, C_j$), localizados remotamente em uma mesma rede local. Este clientes estão conectados à Internet por meio de uma rede (nó) de acesso, que possui um conjunto de k *buffers* únicos ($b_1, \ldots, b_k$) e tem implementada a estratégia UB. A expectativa é um alto grau de otimização da banda B do servidor. Em geral, para fins de justificativa de projeto, temos que $j \geq n \geq k$. Os valores de n e j estão atrelados a situações reais. O valor de k pode ser definido como o número de objetos mais freqüentemente requisitados (i.e., populares) e estimado a partir da distribuição *Zipf*, que diz que a probabilidade de selecionar o objeto de categoria β é igual a $(\beta^{1-\alpha} \sum_{x=1}^{n} \frac{1}{x^{1-\alpha}})^{-1}$, onde α é denominado de $skewfactor$ da distribuição [Ma and Shin 2001]. Os k *buffers* são alocados dinamicamente segundo as requisições dos clientes. Quando todos os k *buffers* estiverem alocados, as requisições de outros objetos são atendidas por meio de um *buffer* convencional e local do cliente. A Figura 3(a) ilustra os segmentos de dados armazenados em um dos k *buffers* devido a escuta de três fluxos distintos, realizadas por três clientes que vêem o mesmo objeto simultaneamente. O fluxo S_1 refere-se ao cliente C_1, o fluxo S_2 ao cliente C_2, e o fluxo S_3 deve-se ao cliente C_3. As unidades armazenadas no *buffer* podem ser indistintamente utilizadas por qualquer um dos j clientes do sistema que queiram assistir ao mesmo objeto neste instante de tempo.

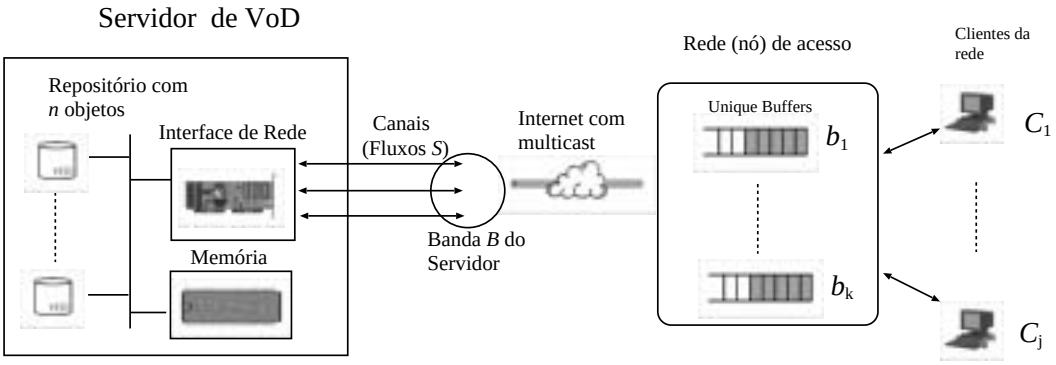

Figura 2. Sistema de vídeo sob demanda.

O compartilhamento eficiente de um único *buffer*, por todos os clientes de uma rede local que desejam assistir a um mesmo objeto, exige o uso de métodos específicos de acesso ao *buffer*. Isto porque é provável termos mais de um cliente realizando acesso simultaneamente. Esta questão pode ser modelada como o *Problema do Produtor/Consumidor* [Yang and Moloney 1988]. A princípio, vemos duas abordagens para sua solução. A primeira é uma proposta denominada de *Software Transactional Memory* [Marathe et al. 2006], a qual baseia-se em conceitos da área de bancos de da-

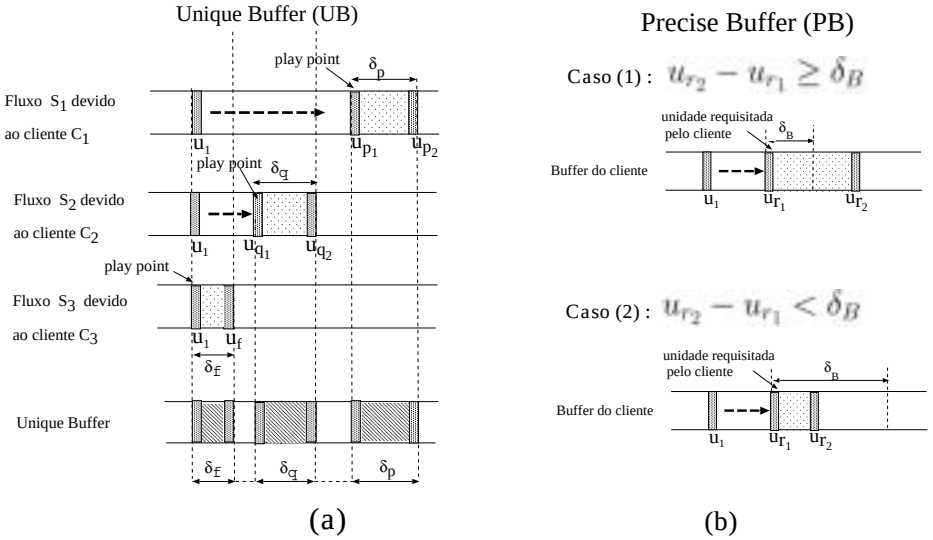

Figura 3. (a) Estratégia Unique Buffer; (b) Estratégia Precise Buffer.

dos para controle de acesso, criando noções de atomicidade, consistência e isolamento de transações. A segunda, mais comum, relaciona-se à utilização de conceitos como semáforos e/ou travas (*locks*) para o controle de acesso.

O armazenamento de unidades de dados no *buffer* único é dinâmico, i.e., é determinado exclusivamente pelo acesso dos clientes às unidades do objeto que está sendo visualizado. Diferentemente de técnicas baseadas em *proxy*, em UB não há necessidade de cópias de partes ou de todo o objeto do servidor multimídia para o *proxy* sem que o cliente tenha requisitado, também não há premissa de uso de quaisquer tipos de codificação em camadas (*layer-encoded streaming*) visando a melhor adaptação de qualidade dos fluxos. Uma análise competitiva mais detalhada com técnicas de *proxy* é considerada em trabalhos futuros.

4.2. Estratégia Precise Buffer

A principal motivação da estratégia PB está na tentativa de evitar que a fragmentação de informação no *buffer* local do cliente comprometa a otimização do sistema. A fragmentação se dá quando a informação armazenada apresenta-se em pequenos segmentos espaçados entre si, fazendo com que na maioria das vezes o cliente não seja atendido por apenas um único segmento e, assim, necessite entrar em um ciclo de leitura: ora a partir de *buffer* local, ora a partir de fluxos do sistema. Isto cria uma maior dificuldade para o compartilhamento de dados, pois os fluxos passam a ter uma menor duração [Rodrigues 2006, Rocha et al. 2005]. Evitando os malefícios da fragmentação, podemos ainda esperar que haja uma minimização da variabilidade dos requisitos de banda (*traffic smoothing* [de Souza e Silva et al. 2002]) e do *overhead* [Bar-Noy et al. 2002, Rodrigues and Leão 2006] do servidor para o tratamento das mensagens entre clientes e servidor, que se traduz em complexidade do sistema e é avaliada nos experimentos adiante.

Na proposta PB, a requisição somente é atendida na condição de que o tamanho da informação contígua em *buffer* (i.e., número de unidades consecutivas), a partir da unidade de dados requisitada, seja suficiente para evitar que o cliente entre no ciclo de

leitura descrito no último parágrafo. Seja então δ_B a variável que designa esse tamanho. A Figura 3(b) ilustra esta estratégia para um cliente que requisita a unidade u_{r_1}, sendo que ele já tem armazenado desde a unidade u_{r_1} até a unidade u_{r_2}. Quando $u_{r_2} - u_{r_1} \geq \delta_B$, a informação é utilizada para servir o cliente. Por outro lado, quando $u_{r_2} - u_{r_1} < \delta_B$, a informação é ignorada e o cliente deve escutar fluxos de dados do servidor.

Para efeito de análise, resolvemos neste trabalho realizar dois conjuntos distintos de experimentos para PB. No primeiro conjunto, assumimos $\delta_B = L$, onde L é o tamanho médio do segmento do objeto requisitado pelo cliente quando este faz uma requisição ao servidor. O valor de L pode ser estimado dinamicamente por medição durante a operação e/ou a partir de *logs* de clientes [Rodrigues and Leão 2006, Rodrigues 2006, Rocha et al. 2005, Almeida et al. 2001, Costa et al. 2004]. Para os experimentos seguintes, variamos δ_B em função do tamanho de L. O objetivo é avaliar a possibilidade da identificação de um valor ideal para δ_B. Discussões sobre a sensibilidade de PB com respeito à precisão das estimativas de L em um ambiente real são consideradas em trabalhos futuros.

5. Avaliação de performance

5.1. Métricas

Nesta seção avaliamos o desempenho das estratégias de gerência de *buffer*, utilizando conjuntamente técnicas de compartilhamento de banda. Nos experimentos consideramos as seguintes estratégias: Simple Buffer (**SB**), Complete Buffer (**CB**), Unique Buffer (**UB**) e Precise Buffer (**PB**); e as seguintes técnicas de compartilhamento de banda: Patching Interativo Eficiente (PIE) e Merge Interativo (MI). Consumo médio da banda, valor de pico da banda, distribuição da banda e trabalho médio do servidor são as principais métricas aqui utilizadas para avaliação das otimizações alcançadas.

Valor médio e valor de pico da banda são medidos em número de canais (fluxos) simultaneamente em uso pelo servidor para transmitir os objetos. Estas métricas são comumente utilizadas em trabalhos da literatura que visam a comparar o desempenho de técnicas de compartilhamento de banda. O comportamento interativo do cliente faz com que o consumo de banda possa variar significativamente ao longo do tempo. Portanto, é necessário avaliarmos a variabilidade da banda ao longo do tempo, o que é feito através do cálculo da distribuição da banda do servidor. Por último, o trabalho médio é aqui tomado como uma grandeza adimensional e utilizado para analisar a complexidade do sistema. O trabalho é função do número médio de mensagens recebidas pelo servidor e das operações executadas para o tratamento das mesmas. A seguir, explicamos o método de quantificação desta métrica conforme definido em [Rodrigues and Leão 2005]. A operação de maior custo no sistema é a busca por um fluxo *multicast*. Faz-se então a análise de pior caso e admite-se que a operação de busca seja executada em tempo $O(n)$, onde n é o número de fluxos *multicast* em andamento no sistema. O servidor pode receber quatro tipos de mensagens: requisição para uma unidade de dados (DR), requisição para o término de uma união de fluxos (MR), requisição para o término de *patch* (PR), e requisição para fim de exibição (LR). A Tabela 1(a) traz as complexidades de tempo relativas às operações devido a cada tipo de mensagem, onde $O(C)$ denota uma complexidade de tempo constante; a Tabela 1(b) apresenta as equações para calcular o trabalho médio. Os valores de n e dos números médios de mensagens (por minuto) do sistema –

a saber, $E[DR]$, $E[MR]$, $E[PR]$, $E[LR]$ – são obtidos a partir do modelo de simulação (detalhes em [Rodrigues 2006]).

Tabela 1. COMPLEXIDADE DE TEMPO E TRABALHO MÉDIO

Técnica	DR	MR	PR	LR
PIE	$O(n)$	$O(C)$	$O(C)$	$O(C)$
MI	$O(2n)$	$O(n)$	—	$O(n)$

(a) Complexidade de tempo

Técnica	Fórmulas
PIE	$E[DR] * n + E[MR] + E[PR] + E[LR]$
MI	$E[DR] * 2n + (E[MR] + E[LR]) * n$

(b) Trabalho médio

5.2. Cargas

Consideramos quatro cargas sintéticas. Cada uma refere-se a um cenário de avaliação distinto no qual um objeto multimídia está sendo transmitido. Estas cargas são obtidas por meio de um gerador proposto no trabalho de [Rocha et al. 2005] e referem-se aos servidores multimídia eTeach, MANIC e Universo *Online* UOL [Almeida et al. 2001, Costa et al. 2004]. Os dois primeiros servidores são de ensino a distância e o último é um servidor de conteúdo.

O gerador de carga sintética tem como entrada um *trace* real de sessões para um dado objeto com uma certa taxa de requisições. Sua saída é um *trace* sintético com estatísticas semelhantes àquelas do *trace* original. Sucintamente, descrevemos sua idéia a seguir. O gerador inicialmente constrói um modelo de transição de estado, onde cada estado corresponde a um segmento de tamanho fixo do objeto. As probabilidades de início de uma sessão em cada segmento, assim como as probabilidades de transição entre os estados, são calculadas a partir do *trace* real. Um *trace* de sessões é então produzido assumindo o início de sessão sendo dado por um processo de Poisson [Almeida et al. 2001, Costa et al. 2004], e o comportamento do cliente dentro de cada sessão é extraído a partir das características do cenário (*trace*) real. O cliente pode executar as seguintes ações VCR: *Play*, *Stop*, *Pause/Resume*, *Jump Forwards* e *Jump Backwards*.

Tabela 2. CARGAS SINTÉTICAS

Estatística	eTeach	MANIC-1	MANIC-2	UOL
Tamanho do objeto T (s)	2199	4175	4126	226
Tamanho da unidade de dados (s)	1	1	1	1
Nr. médio de requisições em T	98	99	100	97
Nr. total de requisições	5146	677	2366	1114
Nr. médio de requisições por sessão	10.29	1.35	4.73	2.23
Tamanho médio do Segmento L (s)	118	1190	496	134
Desvio padrão de L (s)	143	1184	473	91
Coef. de variação de L	1.21	1.00	0.95	0.68
Nr. de valores diferentes da v.a. X	2199	24	98	24

As principais estatísticas das cargas sintéticas estão na Tabela 2. Para garantir um significativo espectro de análise, escolhemos cargas estatisticamente diferentes. Na Carga eTeach (Workload 1), o número de ações Jump Backwards é bem razoável, resultando em

localidade de acesso, i.e., ocorre um grande número de retornos a unidades de dados anteriormente já visualizadas pelo cliente, o que deve favorecer às estratégias de gerência de *buffer* onde o cliente armazena todos os dados transmitidos pelo servidor. Outro aspecto interessante é que a maioria das unidades de dados são acessadas como primeira unidade do segmento. Seja X a variável aleatória que representa a primeira unidade acessada pelo cliente quando este faz um movimento, ou seja, quando requisita um novo segmento. Ter uma unidade i como a primeira unidade de um segmento significa que a variável aleatória X assume o valor i. Por último, a distribuição de acesso a unidades é aproximadamente uniforme. Em relação à Carga MANIC-1 (Workload 2), o nível de interatividade (i.e., nr. de requisições por sessão) é relativamente baixo. Também a distribuição de acesso é não-uniforme e apenas 24 unidades são acessadas como a primeira do segmento. Já na Carga MANIC-2 (Workload 3), poucas unidades são efetivamente acessadas como primeiras unidades do segmento. Existe ainda alguma uniformidade na distribuição de acesso. Finalmente, para a Carga UOL (Workload 4) há significativa localidade de acesso e a primeira unidade do objeto é bastante requisitada (popular).

5.3. Experimentos

Os resultados de simulação são obtidos usando a ferramenta Tangram-II [de Souza e Silva et al. 2006]. Esta constitui-se em um ambiente de modelagem e experimentação de sistemas computacionais e de comunicações, desenvolvido na Universidade Federal do Rio de Janeiro (UFRJ). Em relação à técnica PIE, utilizamos valores ideais para seus respectivos parâmetros δ. Esses valores são obtidos em função das características de cada carga sintética: $\delta_{before} = 10.0\,\text{s}$ e $\delta_{after} = \delta_{merge} = 117.0\,\text{s}$, $386.9\,\text{s}$, $271.9\,\text{s}$ e $29.3\,\text{s}$ para as Cargas eTeach, MANIC-1, MANIC-2 e UOL, respectivamente. Análises para derivação desses valores estão em [Rodrigues and Leão 2005, Rodrigues and Leão 2006]. A seguir, comentamos os dois conjuntos de experimentos realizados.

CONJUNTO 1: Análise Competitiva entre SB, CB, PB ($\delta_B = L$) e UB

Valor Médio e Valor de Pico. As Figuras 4 e 5 resumem os resultados obtidos considerando estas métricas. A estratégia **UB** é notadamente a de melhor desempenho, pois reduz significativamente os requisitos de banda média e valor de pico de banda, comparativamente às outras estratégias. Os requisitos de banda média são inferiores a 1 canal para todas as cargas consideradas. As reduções chegam até 98% em relação a **SB**. A estratégia **SB** é a menos eficiente. Ainda notamos alguma equivalência nos resultados provenientes do uso de **CB** e **PB**. No que se refere aos valores médios de banda, a diferença entre **CB** e **PB**, mantém-se inferior a 1 canal; em relação aos valores de pico, a maior diferença entre **CB** e **PB** ocorre na Carga MANIC-2, onde registramos, com a técnica MI, uma diferença aproximada de oito canais em favor de **PB**. Já as reduções de **CB** e/ou **PB** em relação à **SB** chegam a até 43,0% (valor médio) e a apenas 10,9% (valor de pico), onde evidencia-se uma menor influência sobre esta última métrica.

Trabalho. Conforme podemos observar na Figura 6, a estratégia **UB** é aqui também a mais eficiente. As reduções chegam a até duas ordens de grandeza em relação à **SB**. Considerando **CB** e **PB**, podemos observar que, apesar da proximidade dos valores em ordem de grandeza, a estratégia **PB** mostra-se geralmente mais eficiente que **CB**, pois apenas na Carga MANIC-1 não ocorre redução no valor desta métrica. Por exemplo, para

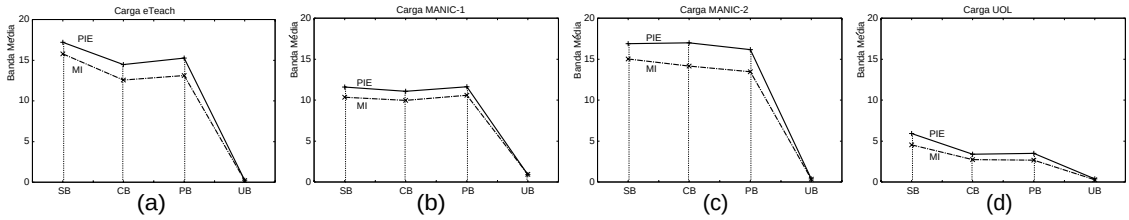

Figura 4. Valor da Banda Média:(a) eTeach; (b) MANIC-1; (c) MANIC-2; (d) UOL.

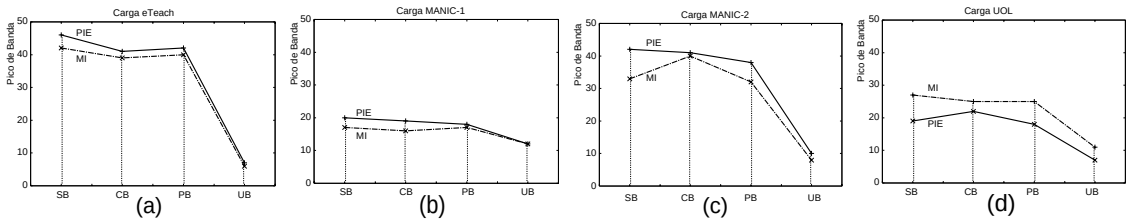

Figura 5. Valor de Pico da Banda:(a) eTeach; (b) MANIC-1; (c) MANIC-2; (d) UOL.

MANIC-2 e técnica MI, o valor obtido por **PB** é aproximadamente três vezes menor que aquele de **CB**. As reduções de **CB** com relação à **SB** não são expressivas. Há inclusive casos desfavoráveis, onde observamos aumentos. Estes casos são aqueles em que possivelmente ocorre a fragmentação da informação em *buffer*. Por fim, as reduções de **PB** em relação à **SB** são um pouco mais atrativas. Por exemplo, para UOL e técnica MI, o valor relativo à **PB** é aproximadamente duas vezes menor que aquele registrado para **SB**.

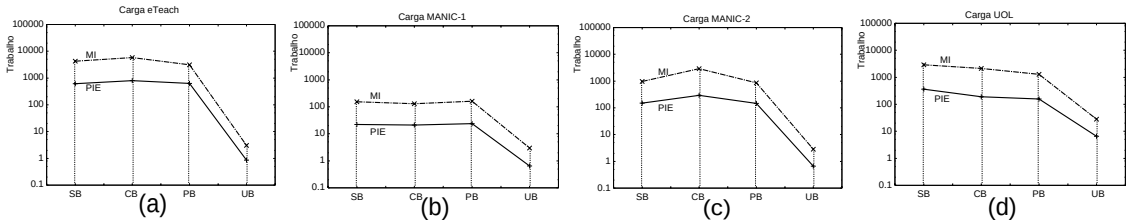

Figura 6. Trabalho:(a) eTeach; (b) MANIC-1; (c) MANIC-2; (d) UOL.

Distribuição. Os resultados desta métrica também confirmam que **UB** é a mais eficiente. A distribuição complementar (CCDF) para a estratégia **UB** aparece sempre bem destacada das demais. A probabilidade do números de canais usados pelo servidor ser maior do que 2 é menor do que 0.1 para a grande maioria das cargas, enquanto que, para as outras estratégias de gerência de *buffer*, esta probabilidade é maior que 0.7 na maioria dos cenários. Também, a partir das curvas da distribuição, observamos que **SB** é a estratégia menos eficiente. Para as estratégias **CB** e **PB**, os resultados aparecem divididos e não há significativa diferença entre elas. Por exemplo, para a técnica MI, observamos que na Carga eTeach, **CB** mostra-se mais eficiente; por outro lado, na Carga MANIC-2, **PB** é mais eficiente. Alguns desses experimentos são ilustrados na Figura 7.

CONJUNTO 2: Análise do Parâmetro δ_B de PB

Aqui variamos o valor de δ_B em função de L e observamos as distribuições complementares da banda do servidor. Também consideramos o valor de δ_B em função do tamanho do objeto inteiro T. Observe que, quando $\delta_B \rightarrow 0$, **PB** torna-se equivalente à

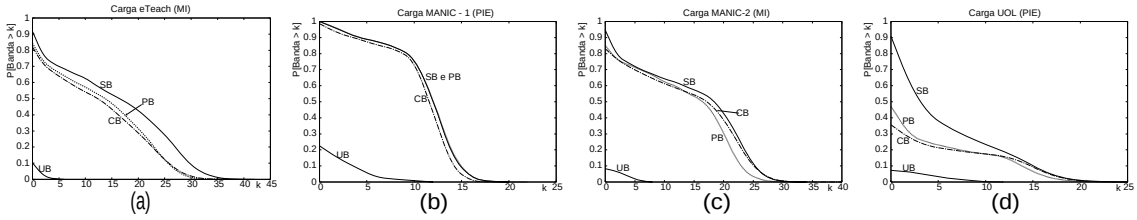

Figura 7. Distribuição:(a) eTeach; (b) MANIC-1; (c) MANIC-2; (d) UOL.

CB; por outro lado, quando $\delta_B \rightarrow T$, **PB** torna-se equivalente à **SB**. O objetivo é então verificar a existência de um valor no intervalo $[0;\ \mathrm{T}]$ que otimize a performance de **PB** e ainda estimar esse valor como função de L.

Para efeito de maior clareza e organização da análise, assumimos então dois casos: $\delta_B \geq L$ e $\delta_B < L$. No primeiro, não verificamos quaisquer otimizações. Isso revela que o tamanho médio do segmento L é possivelmente um valor *upper bound* para as cargas examinadas. No segundo caso e considerando a maioria das cargas, alguma otimização foi produzida. Inclusive, houve situações em que **PB** passou a ser mais eficiente que **CB**. O valor ideal de $\delta_B \geq L$ estabeleceu-se no intervalo de 25%–50% de L. Isso evidencia a importância da determinação do valor ideal de δ_B para o emprego da técnica **PB**. Alguns resultados destes experimentos são ilustrados na Figura 8.

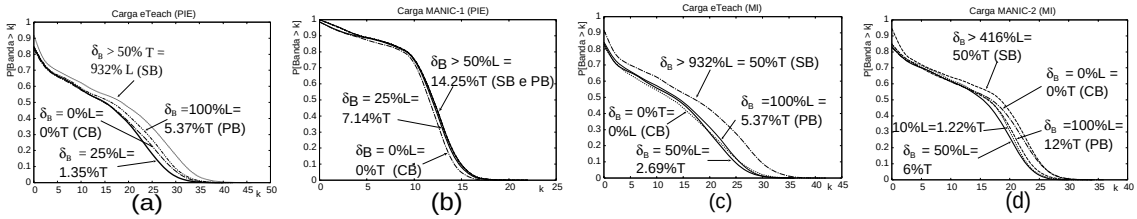

Figura 8. Distribuição:(a) eTeach; (b) MANIC-1; (c) eTeach; (d) MANIC-2.

Em síntese, considerando ambos conjuntos de experimentos, constatamos que: (i) a estratégia **UB** é a mais eficiente. As otimizações produzidas são bem expressivas, revelando um padrão de comportamento semelhante entre clientes independentes que assistem a um mesmo objeto; (ii) as estratégias **CB** e **PB** apresentam certa semelhança nos resultados obtidos, exceto para a métrica trabalho. Isto significa que **PB** tem uma menor complexidade de sistema que **CB**. A complexidade é estimada em função do número de mensagens trocadas entre clientes e servidor e das operações para tratamento das mesmas pelo servidor; (iii) a estratégia **PB** é possível de ser otimizada determinando-se um valor ideal para o parâmetro δ_B, o qual na maioria das cargas utilizadas estabeleceu-se no intervalo de 25%–50% de L; (iv) a estratégia **SB** é a mais simples, mas também é, em geral, a de performance mais sofrível. Esta desvantagem tende a tornar-se inclusive mais acentuada em cenários com alto nível de interatividade por parte dos clientes; (v) os valores de otimização que registramos enaltecem a importância do uso das estratégias de gerência de *buffer*.

6. Conclusões e trabalhos futuros

Aqui foram avaliadas duas novas estratégias de gerência de *buffer* para sistemas de VoD interativos: Unique Buffer (UB) e Precise Buffer (PB), operando em conjunto

com técnicas de compartilhamento de banda. Por meio de simulações, utilizando cargas sintéticas obtidas de servidores reais, validamos nossas propostas. Os resultados finais, os quais englobaram diferentes métricas de comparação de performance, mostraram principalmente que a estratégia UB é a mais eficiente, e também que a estratégia PB é competitiva. Considerando abordagens mais convencionais, alcançamos otimizações de 5%–98% em valores médio de banda, pico de banda e complexidade de mensagens do sistema. Como trabalhos futuros, podemos citar uma análise detalhada do desempenho da estratégia UB quando comparada ao uso de um *proxy*, a avaliação das estratégias propostas usando outras cargas sintéticas, e o desenvolvimento de modelos analíticos para a análise de técnicas de compartilhamento de banda em ambientes com interatividade.

Referências

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

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

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

Banker, R. O. and et al. (1994). Method of providing video-on-demand with VCR-like functions. U. S. Patent 5357276.

Bar-Noy, A., Goshi, G., Ladner, R. E., and Tam, K. (2002). Comparison of stream merging algorithms for media-on-demand. In *Proc. Multimedia Computing and Networking (MMCN'02)*, pages 18–25, San Jose, CA.

Chan, S.-H. G. and Chang, E. (2001). Providing scalable on-demand interactive video service by means of multicast and client buffering. In *Proc. IEEE ICC*, pages 11–15, Helsinki.

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

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

de Souza e Silva, E., Leão, R. M. M., da Silva, A. P. C., de A. Rocha, A. A., Duarte, F. P., Filho, F. J. S., Jaime, G. D. G., and Muntz, R. R. (2006). Modeling, Analysis, Measurement and Experimentation with the Tangram-II Integrated. In *International Conference on Performance Evaluation Methodologies and Tools*.

de Souza e Silva, E., Leão, R. M. M., Ribeiro-Neto, B. A., and Campos, S. V. A. (2002). Performance issues of multimedia applications. In *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, pages 374–404, London, UK. Springer-Verlag.

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

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

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

Ma, H. and Shin, K. G. (2001). A new scheduling scheme for multicast true VoD service. *Lecture Notes in Computer Science*, 2195:708–715.

Marathe, V. J., Spear, M. F., Heriot, C., Acharya, A., Eisenstat, D., III, W. N. S., and Scott, M. L. (2006). Lowering the overhead of nonblocking software transactional memory. In *Workshop on Languages, Compilers, and Hardware Support for Transactional Computing (TRANSACT)*.

Netto, B. C. M. (2004). Interactive Patching: A new resource sharing technique for high-interactivity continuous media delivery. Master's Thesis, UFRJ, COPPE/PESC. In Portuguese.

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

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

Rodrigues, C. K. S. (2006). Mecanismos de Compartilhamento de Recursos para Aplicações de Mídia Contínua na Internet. Tese de Doutorado, UFRJ, COPPE/PESC. http://www.land.ufrj.br/publications/publications.php?

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

Rodrigues, C. K. S. and Leão, R. M. M. (2006). Técnicas para sistemas de vídeo sob demanda escaláveis. In *XXIV Simpósio Brasileiro de Redes de Computadores (SBRC2006)*, pages 247–262, Curitiba, PR, Brasil.

Rodrigues, C. K. S. and Leão, R. M. M. (2007). Bandwidth usage distribution of multimedia servers using patching. *Computer Networks*, 51(3):569–587.

Viswanathan, S. and Imielinski, T. (1995). Pyramid broadcasting for video-on-demand service. In *Proc. SPIE Multimedia Computing and Networking Conference*, pages 66–77.

Yang, I. and Moloney, W. (1988). Concurrent reading and writing with replicated data objects. In *CSC '88: Proceedings of the 1988 ACM sixteenth annual conference on Computer science*, pages 414–417, New York, NY, USA. ACM Press.