Técnicas para Sistemas de Vídeo sob Demanda Escaláveis *

Carlo Kleber da S. Rodrigues , Rosa Maria Meri Leão

Universidade Federal do Rio de Janeiro
COPPE/PESC, Rio de Janeiro RJ 21941-972, CxP 68511, Brazil

{kleber,rosam}@land.ufrj.br

Abstract. *This work introduces the novel bandwidth sharing technique denoted as Interactive Merge - MI as well as presents a thorough competitive analysis of the most recent proposals. Through simulations in different scenarios we note for instance that (i) the HSM paradigm is the most efficient one; (ii) the Patching paradigm presents the smallest system complexity and, for a service with small discontinuities, has an attractive competitiveness; (iii) the MI technique is the most efficient and, comparing to the classical Patching scheme, provides bandwidth and peak reductions of 30%–68% and 18%–62%, respectively.*

Resumo. *Este trabalho apresenta a nova técnica de compartilhamento de banda Merge Interativo - MI e realiza uma análise competitiva detalhada das mais recentes propostas. Por meio de simulações em diferentes cenários observamos, por exemplo, que (i) o paradigma de HSM é o mais eficiente; (ii) o paradigma de Patching é o de menor complexidade de sistema e, admitindo pequenas descontinuidades no serviço, pode apresentar uma competitividade bem atrativa; (iii) a nova técnica MI é a de maior eficiência e, comparativamente com a clássica técnica Patching, provê reduções de 30%–68% e de 18%–62% no consumo médio de banda e valor de pico, respectivamente.*

1. INTRODUÇÃO

O serviço de vídeo sob demanda (VoD) com interatividade, voltado principalmente para as áreas de educação e entretenimento, tem recebido crescente atenção nos últimos anos. Para implementar este serviço, de forma ideal, o servidor deve alocar um canal de transmissão de dados exclusivo para cada cliente de tal maneira que ele possa livremente interagir com a informação sendo visualizada. Entretanto, como a banda é um recurso limitado, esta forma de implementação é inviável quando muitos usuários simultâneos precisam ser acomodados no sistema. O emprego de técnicas de compartilhamento de banda torna-se então essencial para obter escalabilidade.

Várias destas técnicas já foram apresentadas na literatura. No trabalho de [4] as requisições são atendidas pela simples escolha de um canal de difusão que está transmitindo a unidade de dados requisitada (ou a mais próxima) pelo cliente. A proposta de [14] tem atenção apenas com ações VCR do tipo *Pause/Resume*. Os autores de [3] apresentam a idéia de se utilizar o *buffer* local para garantir um serviço com interatividade contínuo. Já a proposta de [23] introduz a técnica chamada *Split and Merge* (SAM) e é a primeira

*Este trabalho é parcialmente financiado pelo CNPq e Faperj.

a discutir a união de fluxos em andamento no sistema. Esta união ocorre por meio de um *buffer* compartilhado situado nos nós de acesso da rede. Os autores de [1] melhoram a técnica SAM utilizando um *buffer* local e exclusivo no cliente. No trabalho de [27] ocorre a proposta da técnica *Single-Rate Multicast Double-Rate Unicast* (SRMDRU). Sua idéia base é a utilização de uma taxa de transmissão duas vezes maior que a normal para permitir a união de fluxos. A técnica *Best Effort Patching* (BEP) proposta em [24] segue a concepção básica da técnica original de acesso seqüencial *Patching* [21], com a diferença de que a estrutura de união de fluxos é de até três níveis em vez de dois. Contudo, pesquisas sobre modificações do esquema original de *Patching* para admitir estruturas de união de três ou mais níveis (p. ex., [20, 30]) sugerem que o nível de complexidade para implementação e gerência do sistema não justificam sua implementação em face dos resultados observados de otimização de banda. Os trabalhos de [26, 19] apresentam a técnica *Patching* Interativo (PI), e os autores de [13] apresentam as técnicas *Patching* Interativo Eficiente (PIE) e *Patching* Interativo Completo (PIC). Estas técnicas baseiam-se na clássica técnica *Patching* e diferenciam-se da proposta de [24] basicamente no aspecto de que a estrutura de união de fluxos mantém-se em até dois níveis.

Apesar de originalmente propostas para o acesso seqüencial, i.e., sem interatividade, as técnicas baseadas no paradigma de *Hierarchical Stream Merging* (HSM) como, por exemplo, [16, 17, 9], em geral podem ser modificadas ou servir de base para o desenvolvimento de técnicas para cenários com interatividade. O principal atrativo deste paradigma é a sua eficiência em potencial por admitir estruturas de união de fluxos sem número limitado de níveis [5]. Em especial, a técnica *Closest Target* (CT) [17] constitui-se em uma das mais simples e eficientes técnicas deste paradigma. Os autores de [29, 2] mostram que uma otimização de banda significativa é obtida, em relação a um atendimento *unicast*, através do emprego de esquemas HSM. Já o trabalho de [28] mostra que a introdução de otimizações em esquemas HSM, baseadas principalmente no uso estratégico do *buffer* local do cliente, podem levar a redução da banda do servidor para cenários com interatividade.

Este trabalho apresenta a nova técnica denominada *Merge* Interativo (MI) e realiza uma análise competitiva detalhada das mais recentes propostas supracitadas. Análises e experimentos são realizados com cargas sintéticas de servidores reais e incluem diferentes métricas de comparação como, por exemplo, distribuição da banda, consumo médio de banda e valor de pico. Dentre outras constatações, os resultados obtidos mostram que (i) o paradigma de HSM é o mais eficiente; (ii) o paradigma de *Patching* fornece a menor complexidade de sistema e, admitindo pequenas descontinuidades no serviço, $1\%-5\%$ do tamanho total do objeto multimídia sendo transmitido, pode apresentar uma competitividade bastante atrativa; (iii) a nova técnica MI é a de maior eficiência e comparativamente, por exemplo, com a clássica técnica *Patching*, apresenta reduções de $40\% - 68\%$ no consumo médio de banda, e de $34\% - 63\%$ no valor de pico de banda.

O restante deste trabalho tem a seguinte organização. A Seção 2 introduz conceitos sobre cenários com interatividade. A Seção 3 revisa as mais recentes propostas baseadas nos paradigmas de *Patching* e de HSM. A nova técnica *Merge* Interativo (MI) está na Seção 4. A Seção 5 traz os mais importantes resultados de simulação obtidos. Por último, as conclusões e as propostas para trabalhos futuros constituem a Seção 6.

2. CONCEITOS BÁSICOS E TERMINOLOGIA

Considere um servidor de vídeo e um grupo de clientes recebendo um objeto multimídia (aula, *video clip*, etc.) através da Internet. Neste trabalho admitimos que o objeto é dividido em unidades de dados de mesmo tamanho e que existe serviço *multicast* disponível na rede. Embora o serviço *multicast* não esteja atualmente implementado em toda a rede Internet, acreditamos que esta consideração seja válida em determinadas situações: redes corporativas (*enterprise networks*) e redes locais (*local networks*) possuem o serviço *multicast* ou podem implementá-lo sem maiores dificuldades. Alternativamente, temos soluções como *multicast* na camada de aplicação usando *tunelling*. Assumimos também que os clientes têm acesso não-seqüencial, i.e., podem realizar ações VCR (p. ex., *Jump Forwards/Backwards* e *Pause/Resume*). Cada um dos clientes possui *buffer* local capaz de armazenar pelo menos metade do objeto requisitado e sua banda corresponde a duas vezes a taxa de exibição deste objeto. Por fim, os fluxos iniciados (*multicast* ou *unicast*) transmitem individualmente na mesma taxa de exibição do objeto. Estas suposições estão respaldadas conjuntamente ou individualmente em trabalhos anteriores sobre técnicas de compartilhamento de banda como, por exemplo, [2, 28, 25, 24, 5].

O cenário descrito a seguir explica a forma atual mais comum de atendimento de requisições, considerando a interatividade de clientes, para um objeto armazenado no servidor. Assuma que uma requisição ocorre em $t = t_0$ e existem n fluxos *multicast* em andamento. Seja u_r a unidade do objeto solicitada por esta requisição e, sem perda de generalidade, admita que todos os fluxos S_i, $i = 1, \ldots, n$, estão transmitindo unidades de dados posteriores à u_r. Por fim, seja $dist_i$ a distância entre u_r e a unidade sendo atualmente transmitida pelo fluxo *multicast* S_i.

O atendimento da solicitação de u_r pode ser basicamente realizado de duas formas: serviço *descontínuo* ou serviço *contínuo* [25, 1]. No caso de serviço descontínuo, a solução consiste em simplesmente identificar o fluxo que tenha associado o menor valor de $dist_i$ e atender a requisição por meio deste fluxo. Seja então S_{min} o fluxo e $dist_{min}$ o valor de distância associado. Note que se $dist_{min} > 0$ então o cliente que fez a solicitação por u_r receberá na realidade uma unidade posterior àquela solicitada e distante $dist_{min}$, caracterizando o serviço descontínuo por não estar recebendo exatamente aquilo que foi solicitado. Já no caso do serviço contínuo, o valor $dist_{min}$ é determinado e então é verificado se ele está ou não dentro de um limiar de tempo permitido δ_{after}. Caso esteja, o cliente é atendido pelo fluxo correspondente S_{min} e as unidades do objeto que faltam (relativas à distância $dist_{min}$) são enviadas através de um fluxo (canal) *unicast*. Caso $dist_{min} > \delta_{after}$, um novo fluxo *multicast* $S_{n+1} = S_{new}$ é então aberto para atender à solicitação por u_r.

Uma opção para evitar a criação de um fluxo S_{new} é a união do cliente com um outro fluxo que transmite uma unidade de dados anterior, mas *suficientemente próxima* à u_r. Note que nesta situação há descontinuidade do serviço, mas não há perda de informação, pois o cliente recebe unidades de dados à mais do que solicitou. A estimativa de que seja *suficientemente próxima* é bastante subjetiva e está atrelada ao grau de tolerância do cliente. Seja δ_{before} o limiar de tempo utilizado para mensurar esta proximidade, e seja S_{before} o fluxo identificado nesta situação. Estimulando a utilização deste conceito, os autores de [3] inclusive sugerem que clientes de sistemas interativos podem não se importar com pequenas descontinuidades em relação ao serviço oferecido. Mais especificamente, os autores de [26, 19] fazem $\delta_{before} = 60$ s para objetos de tamanho 6000–4200 s, e no trabalho de [13] é usado $\delta_{before} =$10 s para objetos de tamanho 226–4175 s.

3. PROPOSTAS RECENTES

Nesta seção revisamos os algoritmos de operação de quatro das mais recentes técnicas de compartilhamento de banda que consideram o serviço imediato com interatividade: PI, PIE, PIC e CT. Conforme já comentamos, as três primeiras são baseadas no paradigma de *Patching* e a última no paradigma de HSM. Estas técnicas são utilizadas no estudo comparativo que realizamos mais adiante e esta decisão deu-se por serem técnicas recentes e assim conseguirem agregar simultaneamente dois aspectos essenciais: máxima eficiência e simplicidade. As técnicas PI, PIE e PIC, por exemplo, possuem um mecanismo bastante elaborado em termos de decisões que visam o compartilhamento de fluxos e a utilização do *buffer* local do cliente, sem perder com isso a simplicidade da estrutura de união de fluxos em dois níveis, conforme a técnica original *Patching*. Já a técnica CT apresenta resultados de performance comparáveis a escalonamentos ótimos *off-line*, ao mesmo tempo que, apesar de ter uma estrutura de união de fluxos relativamente complexa, seu mecanismo de decisão para compartilhamento de fluxos é dos mais simples já concebidos.

3.1. Técnica Patching Interativo – PI

Nesta técnica a chegada de uma requisição é o único evento que pode provocar uma união de fluxos no sistema como explicamos a seguir. Seja u_W a unidade de dados do objeto correspondente ao tamanho da janela ótima W da técnica *Patching* original [18, 7], (p. ex., se W= 12 s e o objeto é dividido em unidades de 1 s cada então $u_W = 12$), e seja S_W um fluxo *multicast* que transmite, no instante da chegada da requisição, uma unidade de dados anterior ou igual à u_W. Para efeito de análise, assuma que uma requisição ocorre para uma dada unidade u_r do objeto. A tomada de decisões neste algoritmo se dá em função de dois casos: $u_r = 1$ e $u_r \neq 1$, onde 1 representa a primeira (inicial) unidade do objeto.

- Caso 1: $u_r = 1$ – Se S_W existe e está transmitindo uma unidade de dados posterior à u_r, então a requisição é atendida pelo mesmo e as unidades eventualmente perdidas (*patch*) são enviadas através de um fluxo *unicast*. Se S_W não existe então um novo fluxo *multicast* é aberto para atender a requisição.
- Caso 2: $u_r \neq 1$ – Neste caso inicialmente é verificado se existe um fluxo *multicast* S_{before}, i.e., um fluxo transmitindo uma unidade de dados anterior à u_r dentro de um limiar de tempo δ_{before}. Se este fluxo existe então a requisição é atendida pelo mesmo. Caso contrário, 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}. Se este fluxo existe então o servidor informa ao cliente para escutar S_{after} e abre um novo fluxo *unicast* para transmitir as unidades inicialmente perdidas (*patch*). Por outro lado, se S_{after} não existe então um novo fluxo *multicast* S_{new} é aberto para servir a requisição por u_r.

3.2. Técnica Patching Eficiente – PIE

Semelhantemente à técnica PI, aqui a chegada de uma requisição é o único evento que provoca uma união de fluxos no sistema. Também é baseada no clássico esquema *Patching* e tem, como principal premissa, a manutenção da estrutura de união de fluxos em no máximo dois níveis. Como antes, para efeito de análise, admita que ocorre uma requisição para uma dada unidade de dados u_r do objeto.

O algoritmo inicialmente busca por um fluxo *multicast* S_{before}, i.e., um fluxo transmitindo uma unidade de dados anterior à u_r dentro de um limiar de tempo δ_{before}. Se este fluxo existe então a requisição é atendida pelo mesmo. Porém, 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}. Se este fluxo existe então o servidor informa ao cliente para escutar S_{after} e abre um novo fluxo *unicast* para transmitir as unidades inicialmente perdidas (*patch*). Entretanto, se S_{after} não existe então um novo fluxo *multicast* S_{new} é aberto para servir a requisição por u_r. Nesta situação 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}. Note que o algoritmo de PIE é mais elaborado que o de PI por introduzir o conceito do fluxo S_{merge}, o que cria a expectativa de uma maior otimização de banda dado que é uma possibilidade a mais para a união de fluxos.

3.3. Técnica Patching Interativo Completo – PIC

Esta técnica também é baseada no clássico esquema *Patching* e, apesar de bem mais elaborada que as técnicas PI e PIE, tem como premissa básica a manutenção da estrutura de união de fluxos em no máximo dois níveis. Três eventos são considerados para união de fluxos no sistema, conforme explicamos a seguir.

- *evento 1*: admita que ocorra uma requisição para uma dada unidade de dados u_r do objeto. Nesta situação o algoritmo procede de forma semelhante ao de PIE com a diferença básica de que, em vez de buscar por um único fluxo S_{merge}, aqui busca-se um conjunto de fluxos deste tipo. A expectativa natural é que mais otimização de banda possa ser conseguida desde que, potencialmente, mais fluxos são unidos.
- *evento 2*: admita o término precoce de um fluxo *multicast* S_j que era o fluxo alvo de um outro fluxo *multicast* S_i. Em outras palavras, S_j termina antes de ser alcançado por S_i. Os clientes de S_i então tornam-se *órfãos* e os conteúdos respectivamente armazenados em seus *buffers*, devido à escuta de S_j, são descartados. O servidor imediatamente busca por um outro fluxo em andamento no sistema que transmite uma unidade de dados posterior àquela atual do fluxo S_i e dentro do limiar de tempo δ_{merge}. Seja S_{subs} este fluxo. Se este fluxo existe então ele torna-se o novo alvo de S_i. Caso S_{subs} também termine precocemente, i.e., termine antes de ser alcançado por S_i, o processo de busca por um outro fluxo S_{subs} é repetido.
- *evento 3*: admita o término de todos os eventuais fluxos de *patch* associados a um fluxo S_{source}. Nesta situação imediatamente o servidor tenta localizar um fluxo S_{merge} que transmite uma unidade de dados posterior àquela do fluxo S_{source} e dentro de um limiar de tempo δ_{merge}. Se este fluxo existe então ele torna-se alvo de S_{source}. Ou seja, os clientes de S_{source} passam a escutar também o fluxo S_{merge} de tal sorte que S_{source} e S_{merge} possam ser unidos mais à frente.

3.4. Técnica Closest Target – CT

Diferentemente de PI, PIE e PIC, a técnica CT, por ser baseada em HSM, conforme já mencionado, tem um mecanismo de tentativa de compartilhamento de dados bem mais exaustivo. Existe sempre a busca por um fluxo para ser simultaneamente escutado com aquele que está efetivamente servindo a solicitação por uma unidade u_r. Além disso, não existe qualquer limiar de tempo restritivo para a busca por fluxos no sistema. São considerados os três eventos descritos a seguir para união de fluxos no sistema.

- *evento 1*: admita que ocorra uma requisição para uma dada unidade de dados u_r do objeto. O servidor imediatamente abre um novo fluxo multicast S_{new} para atender esta requisição. Simultaneamente o servidor também tenta localizar um outro fluxo em andamento no sistema que esteja transmitindo uma unidade de dados posterior à u_r, sem considerar qualquer limiar de tempo. Seja S' este fluxo. Se S' existe então o cliente de S_{new} também vai escutá-lo de tal sorte que S_{new} e S' possam, eventualmente, ser unidos mais à frente.
- *evento 2*: admita o término precoce de um fluxo S_j que era o fluxo alvo de um outro fluxo S_i. Neste caso o tratamento é análogo ao descrito no *evento 2* de PIC, sendo que aqui não se utiliza limiar de tempo para busca do substituto de S_j.
- *evento 3*: admita que ocorre a união de dois fluxos no sistema: S_i e seu alvo S_j. Seja S_m o fluxo resultante desta união. Os eventuais conteúdos dos *buffers* dos clientes de S_m, que originalmente eram clientes de S_j, são descartados e o servidor imediatamente busca no sistema por um fluxo que esteja transmitindo uma unidade de dados posterior àquela do fluxo S_m, sem considerar qualquer limiar de tempo. Seja S' este fluxo. Se S' existe então os clientes de S_m vão escutá-lo também de tal sorte que S_m e S' possam ser unidos mais à frente.

4. UMA NOVA PROPOSTA: MERGE INTERATIVO

Aqui apresentamos a nova técnica *Merge* Interativo (MI). Assim como CT, esta nova técnica é baseada no paradigma de HSM e, portanto, tem sua estrutura de união de fluxos de profundidade ilimitada (i.e., número de níveis ilimitado). Os seguintes três eventos são considerados para efeito de união de fluxos no sistema.

- *evento 1*: admita que ocorra uma requisição para uma dada unidade de dados u_r do objeto. O servidor imediatamente abre 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 dentro de um limiar de tempo δ_{MI}. Seja S' este fluxo. Se S' existe então o cliente de S_{new} também vai escutá-lo de tal sorte que S_{new} e S' possam ser unidos mais à frente.
- *evento 2*: admita o término precoce de um fluxo S_j que era o fluxo alvo de um outro fluxo S_i. Neste caso o tratamento é análogo ao descrito no *evento 2* de PIC, sendo que aqui utiliza-se o limiar de tempo δ_{MI} para busca do substituto de S_j.
- *evento 3*: admita que ocorra a união de dois fluxos no sistema: S_i e seu alvo S_j. Seja S_m o fluxo resultante desta união. O servidor imediatamente busca no sistema por um fluxo que esteja transmitindo uma unidade de dados posterior àquela do fluxo S_m dentro de um limiar de tempo δ_{MI}. Seja S' este fluxo. Se S' existe então os clientes de S_m, que originalmente pertenciam ao fluxo S_i, vão escutá-lo também para que possam alcançar S' e então se unir a ele. Já os clientes de S_m, que originalmente pertenciam ao fluxo S_j, não são afetados.

Note que existem duas principais diferenças em relação à técnica CT explicada anteriormente: a consideração de um limiar de tempo δ_{MI} e a não fusão de grupos de clientes pertencentes a fluxos distintos na ocorrência do *evento 3*. Estas duas diferenças são discutidas em maiores detalhes a seguir. Em relação à primeira diferença, i.e., a utilização de um limiar de tempo denotado por δ_{MI}, argumentamos o seguinte. Embora a técnica CT não faça uso de qualquer limiar de tempo, uma série de outras técnicas baseadas em HSM o fazem na intenção de otimizar as decisões relacionadas a abertura e união de fluxos no sistema (p. ex., [22, 8]). O que visualizamos com esta diferença é a possibilidade de uma máxima otimização da técnica MI.

Já em relação à segunda diferença, i.e., a não fusão de grupos de clientes pertencentes a fluxos distintos na ocorrência do *evento 3*, temos a seguinte análise. Suponha que um fluxo S_i se una com seu alvo S_j. Seja S_m o fluxo resultante desta união. Considere que S_j já tivesse um fluxo alvo S_k antes de ser alcançado por S_i. Na técnica CT, conforme já explicado, um fluxo alvo para S_m é selecionado e toda informação (i.e., unidades de dados do objeto) recebida pelos clientes originalmente pertencentes à S_j (enquanto estavam escutando S_k antes da ocorrência da união) é simplesmente descartada. Para um acesso seqüencial, esta decisão de descarte não aumenta a banda do servidor (embora impacte na banda do cliente), pois esta mesma informação precisa ser retransmitida de qualquer forma para atender os clientes originalmente pertencentes ao fluxo S_i. Por outro lado, para um acesso não-seqüencial, esta decisão pode aumentar a banda do servidor, pois a informação descartada não é mais necessariamente retransmitida conforme explicamos a seguir.

Assuma que, logo após a união dos fluxos S_i e S_j, que resulta no fluxo S_m, todos os clientes originalmente pertencentes à S_i decidam, por exemplo, realizar um salto para trás (ou para frente) do atual ponto em exibição do objeto. Neste caso, se os clientes originalmente pertencentes à S_j descartarem a informação em *buffer* (recebida de S_k), como ocorre na técnica CT, esta mesma informação precisará ser retransmitida novamente, não por causa dos clientes originalmente pertencentes à S_i, mas por causa dos clientes originalmente pertencentes à S_j. A decisão de descarte nesta situação é portanto indevida, pois os clientes de S_j já tinham esta informação em *buffer* antes da união. Para resolver isso, na técnica MI, conforme já explicamos, um fluxo alvo S_t é então selecionado e designado exclusivamente para os clientes de S_m que originalmente pertenciam à S_i, e os clientes que originalmente pertenciam à S_j não são afetados, i.e., não fazem descarte de dados e continuam tendo S_k como fluxo alvo. Note que S_t pode ser um fluxo diferente de S_k, pois os clientes têm um acesso não-seqüencial e assim podem, portanto, abrir fluxos de dados em qualquer instante e em qualquer unidade de dados do objeto sendo visualizado. Assim, temos em MI a possibilidade de um mesmo fluxo com diferentes grupos de clientes associados, cada grupo com um fluxo alvo distinto.

No entanto, é preciso dizer que existem duas situações específicas, advindas desta segunda diferença acima explicada, que podem favorecer a técnica CT em relação à técnica MI. A primeira situação relaciona-se ao fato de que o eventual novo fluxo alvo S_t, designado para os clientes de S_i depois que S_i se une à S_j, pode estar relativamente mais próximo ao fluxo S_j do que aquele originalmente designado para o próprio S_j. Por exemplo, admita que, antes da união de S_i com S_j, o fluxo alvo de S_j, que é o fluxo S_k, esteja à distância de 20 unidades de dados. Admita que imediatamente antes da união de S_i com S_j, um novo fluxo S_t é aberto no sistema e este está à distância de apenas 3 unidades de S_j. Considere que neste instante ocorre a união de S_i com S_j, resultando no fluxo S_m. Considerando este cenário, analisemos a seguir o que ocorre nas técnicas MI e CT. Na técnica CT, todos os clientes de S_m (originalmente pertencentes à S_i ou S_j) são beneficiados pelo fato da identificação do novo alvo indicar o fluxo mais próximo S_t. Na técnica MI, os clientes de S_m, originalmente pertencentes à S_i, são beneficiados pela identificação do fluxo mais próximo S_t; entretanto, os clientes de S_m, originalmente pertencentes à S_j, permanecem tendo S_k como fluxo alvo. Nesta situação, CT pode apresentar uma melhor otimização de banda que MI, pois o fluxo alvo mais próximo atende indistintamente a todos os clientes de S_m. Note que uma maior proximidade do fluxo alvo leva a um menor número de fluxos no sistema, pois o fluxo que tenta alcançar o alvo pode ser extinto mais cedo.

A segunda situação é descrita a seguir. Na técnica MI, se os clientes de S_j não possuem um fluxo alvo, estes clientes ainda permanecerão sem um fluxo alvo mesmo após a união de S_i com S_j. Na técnica CT, isto não ocorre, pois o fluxo alvo é escolhido para todos os clientes de S_m, independentemente de serem originalmente pertencentes à S_i ou S_j. Contudo, este fato, não traz considerável vantagem para CT. A explicação é que, devido à operação intrínseca do paradigma de HSM (abertura e busca por fluxos de forma exaustiva) e ainda à condição de termos interatividade, a probabilidade de um cliente não ter um fluxo alvo associado é praticamente desprezível. Para analisar melhor este aspecto, implementamos neste trabalho uma variante de MI, que denominamos MI-var. A única diferença para MI é que, na técnica MI-var, após ocorrer a união de S_i com S_j, resultando no fluxo S_m, os clientes originalmente pertencentes à S_j, que não possuem um fluxo alvo, são também considerados para a designação de um eventual novo fluxo alvo S_t.

5. Avaliação de Performance

5.1. Métricas e Cargas

Nas análises deste trabalho, as quais consideram as técnicas *Patching*, PI, CT, PIE, PIC, MI e MI-var, utilizamos as seguintes principais métricas: consumo médio da banda, valor de pico da banda, distribuição da banda, razões competitivas e trabalho médio.

Consumo médio e valor de pico da banda são duas métricas comumente utilizadas em experimentos comparativos. Já a distribuição da banda torna-se importante pelo fato de considerarmos em nossos experimentos a interatividade dos clientes. Isto faz com que a banda possa variar significativamente ao longo do tempo. Conseqüentemente, o simples cômputo do valor médio ou do valor de pico não permitem uma análise mais detalhada das técnicas. As razões competitivas [6] aqui utilizadas são assim definidas [13]: $RMax = max(\frac{P[Banda>k]_{tec_1}}{P[Banda>k]_{tec_2}})$ e $RMin = min(\frac{P[Banda>k]_{tec_1}}{P[Banda>k]_{tec_2}})$, para todo k inteiro desde que $P[Banda > k] \geq 10^{-4}$ (por precisão da simulação), onde $Banda$ é medida em número de canais simultâneos em uso no sistema, e tec_1 refere-se à técnica em relação a qual queremos comparar a técnica tec_2. Note que estas razões representam um meio eficiente de quantificar a diferença entre as respectivas distribuições de banda de duas técnicas tec_1 e tec_2 que desejamos comparar. O trabalho médio é utilizado para analisar a complexidade do sistema e é definido conforme mostrado a seguir [13]. Ele é função do número médio de mensagens recebidas pelo servidor e das operações executadas para o tratamento das mesmas. A operação de maior custo 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) sintetiza 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 (detalhes em [13, 12]). Os valores de n, $E[DR]$, $E[MR]$, $E[PR]$, e $E[LR]$ são obtidos a partir do modelo de simulação.

Tabela 1. Complexidade de tempo e trabalho médio do servidor

Técnica	DR	MR	PR	LR
Patching	$O(n)$	—	$O(C)$	$O(C)$
PI	$O(n)$	—	$O(C)$	$O(C)$
PIE	$O(n)$	$O(C)$	$O(C)$	$O(C)$
PIC	$O(2n)$	$O(n)$	$O(n)$	$O(n)$
MI, MI-var	$O(2n)$	$O(n)$	—	$O(n)$
CT	$O(2n)$	$O(n)$	—	$O(n)$

(a) Complexidade de tempo

Técnica	Fórmulas
Patching	$E[DR]*n+E[PR]+E[LR]$
PI	$E[DR]*n+E[PR]+E[LR]$
PIE	$E[DR]*n+E[MR]+E[PR]+E[LR]$
PIC	$E[DR]*2n+(E[MR]+E[PR]+E[LR])*n$
MI, MI-var	$E[DR]*2n+(E[MR]+E[LR])*n$
CT	$E[DR]*2n+(E[MR]+E[LR])*n$

(b) Trabalho médio

Nossas simulações usam quatro cargas sintéticas obtidas a partir de servidores reais nos trabalhos de [10, 28]. Dois destes servidores são de ensino a distância (eTeach e MANIC). O outro é o *Universo Online* (UOL), um dos maiores provedores de conteúdo da américa latina. Os objetos são divididos em unidades de dados de 1 s e o cliente pode executar as seguintes ações VCR: *Play, Stop, Pause/Resume, Jump Forwards* e *Jump Backwards*. As principais características das cargas que utilizamos são apresentadas na Tabela 2. Note que elas são estatisticamente diferentes. As cargas têm, por exemplo, diferentes níveis de interatividade: o número médio de requisições por sessão varia de 10.29 (eTeach) a 1.35 (MANIC-1). O tamanho médio L do segmento do objeto requisitado tem uma grande variação associada – de 118 s (eTeach) até 1190 s (MANIC-1) – e seu desvio padrão varia dentro de uma ordem de magnitude. A popularidade N do objeto denota o número médio de requisições que chegam durante o período de tempo T (i.e., tamanho do objeto). A partir desta característica, é possível ter-se um sentimento de quanto um objeto é em média solicitado pelos clientes do sistema [16, 21]. Uma outra característica importante que consideramos é a distribuição da variável aleatória X, assim definida: X é a variável que representa a primeira unidade do segmento de tamanho médio L do objeto requisitado. Podemos notar que, para carga eTeach, X tem distribuição praticamente uniforme. Para as outras cargas, existe uma distribuição mais concentrada em torno de algumas unidades.

5.2. Resultados e Análise

Os resultados de simulação são obtidos usando a ferramenta Tangram-II [15]. Esta ferramenta constitui-se em um ambiente de modelagem e experimentação de sistemas computacionais/comunicações, desenvolvido na Universidade Federal do Rio de Janeiro (UFRJ), com participação de UCLA/USA, com propósitos de pesquisa e educação. Este ambiente combina uma interface de usuário sofisticada em um paradigma de orientação a objeto e novas técnicas de solução para análise de performance e disponibilidade. O

Tabela 2. Cargas sintéticas

Estatística	eTeach	UOL	MANIC-1	MANIC-2
tamanho do objeto T (s)	2199	226	4175	4126
popularidade do objeto N	98	97	99	100
número total de requisições	5146	1114	677	2366
número médio de req/sessão	10.29	2.23	1.35	4.73
tamanho médio do segmento L (s)	118	134	1190	496
desvio padrão de L (s)	143	91	1184	473
coef. de variação de L	1.21	0.68	1.00	0.95
número de diferentes valores assumidos pela v.a. X	≈ 2199	≈ 24	≈ 24	≈ 98

usuário especifica um modelo em termos de objetos que interagem por meio de um mecanismo de troca de mensagens. Um vez que o modelo seja compilado, pode ser resolvido analiticamente, se for Markoviano ou pertencer a uma classe de modelos não Markovianos, ou resolvido via simulação. É possível obter soluções tanto para estado estacionário como para estado transiente.

Salvo informado diferentemente, aqui consideramos $\delta_{after} = \delta_{merge}$, e fazemos $\delta_{before} = 10\,\text{s}$. Estas suposições são respaldadas nos resultados obtidos em [13]. A banda, como já comentado, é medida em número de canais simultâneos em uso no sistema. Devido a restrições de espaço, não ilustramos todos os experimentos realizados (resultados completos estão em [12, 11]). Inicialmente analisamos as técnicas baseadas em *Patching* (i.e., PI, PIE e PIC) no intuito de obtermos aquela de melhor performance. Em seguida, de forma análoga, avaliamos as técnicas baseadas em HSM (i.e., CT, MI e MI-var). Finalmente, temos uma análise competitiva considerando a mais eficiente técnica de cada paradigma. Também em nossas análises consideramos dois tipos de uso de *buffer* local do cliente: *buffer* simples (SB) e *buffer* completo (CB). O primeiro tipo refere-se ao uso convencional do *buffer* para fins de sincronização de dois fluxos que o cliente escuta simultaneamente, objetivando a união. Já o segundo refere-se a um uso estratégico de *buffer* em que, além da convencional sincronização, todas as unidades de dados transmitidas pelo servidor são armazenadas pelo cliente. Isto faz com que só exista requisição para uma unidade que o cliente ainda não possui armazenada em *buffer* local.

Em todas as cargas consideradas, PI, PIE e PIC apresentam melhor performance que a clássica técnica *Patching*. Este resultado já era esperado, pois *Patching* é um esquema para acesso seqüencial e, portanto, não é capaz de lidar eficientemente com a interatividade dos clientes. Esta superioridade em performance é quantificada por meio das razões competitivas computadas: $RMin = 1.0$ e $RMax \in [120.2, 3634.9]$. Ilustramos ainda esta diferença através das Figuras 1(a) e 1(b). A primeira mostra a distribuição (CCDF - *Complementary Cumulative Distribution Function*) da banda para PIE-CB e *Patching*-CB na Carga eTeach, e a segunda mostra para PIE-SB e *Patching*-SB na Carga MANIC-2. As técnicas PIE e PIC têm performance semelhante em todas as cargas e são ambas, de forma geral, mais eficientes que PI. Esta superioridade em performance é quantificada por meio das razões competitivas computadas: $RMin \in [0.5, 1.0]$ e $RMax \in [1.6, 25.8]$. Também para ilustrar este fato apresentamos a distribuição (CCDF) da banda para PIE-SB, PIC-SB e PI-SB na Figura 1(c). Isto favorece enormemente

PIE por ter uma implementação bem mais simples que PIC. Nestas avaliações usamos $\delta_{after} = 50\%$ da janela ótima de *Patching* W [7, 18], entretanto, enfatizamos que as observações então feitas são válidas independentemente do valor absoluto de δ_{after} no intervalo de 0–100% de W ou L. Valores fora deste intervalo se mostram ineficientes.

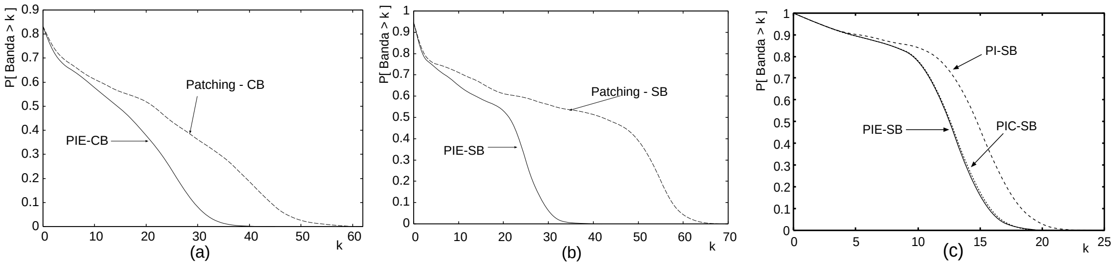

Figura 1. (a) CCDF para PIE-CB e Patching-CB em eTeach; (b) CCDF para PIE-SB e Patching-SB em MANIC-2; (c) CCDF para PIE-SB, PIC-SB e PI-SB em MANIC-1.

Ao compararmos MI e CT, notamos que, na maioria dos cenários, MI é mais eficiente que CT quando usamos a concepção de *buffer* completo (CB) e, de forma contrária, quando usamos *buffer* simples (SB), CT torna-se mais eficiente que MI. Este resultado está embasado na segunda diferença de MI com relação à CT, conforme detalhado na Seção 4. Em linhas gerais, quando usamos *buffer* completo, existe uma tendência de termos um menor número de fluxos ativos no sistema e daí uma menor probabilidade de selecionar novos fluxos alvos, em substituição a fluxos alvos originais, que de fato venham a prover economia de banda. Para ilustrar este ponto temos as Figuras 2(a), 2(b) e 2(c). Nas primeiras duas figuras temos a distribuição (CCDF) da banda quando usamos *buffer* completo (CB) nas Cargas UOL e eTeach, respectivamente, onde vemos a vantagem de MI sobre CT. Na última figura temos o caso de emprego de *buffer* simples (SB) na Carga eTeach. A performance é justamente contrária: CT é mais eficiente que MI. Agora, desde que a implementação de *buffer* completo é relativamente simples e vantajosa, devido à eventual economia de banda que pode ser atingida, decorre como conclusão natural que MI é uma melhor escolha para a maioria dos cenários examinados.

Comparando MI com MI-var, notamos performances semelhantes, ou seja, a otimização de MI-var em relação à MI é insignificante. Isto mostra que a probabilidade de que um cliente não tenha um fluxo alvo é praticamente nula (segunda situação explicada na Seção 4). Para ilustrar, temos as Figuras 3(a), 3(b) e 3(c). As duas primeiras apresentam a distribuição (CCDF) da banda para MI-SB e MI-var-SB nas Cargas MANIC-1 e eTeach, respectivamente. A terceira mostra a distribuição (CCDF) da banda para MI-CB e MI-var-CB na Carga UOL. As curvas praticamente se superpõem. Isto favorece MI por ter uma implementação mais simples.

Em relação ao valor do parâmetro δ_{MI}, os experimentos mostram que, ao simplesmente não limitarmos este valor, já obtemos a performance otimizada ou bastante próxima da mesma. Isto equivale a dizer que, diferentemente do observado no paradigma de *Patching*, o paradigma de HSM, em sua forma mais simples, dispensa a determinação de limiares de tempo para decisões de união e abertura de fluxos no sistema quando usado em cenários com interatividade. Isto naturalmente favorece à implementação, pois exclui a necessidade de formulações matemáticas para cálculo de limiares de tempo, conforme ocorre no paradigma de *Patching*. Podemos ver este fato, por exemplo, representado nas

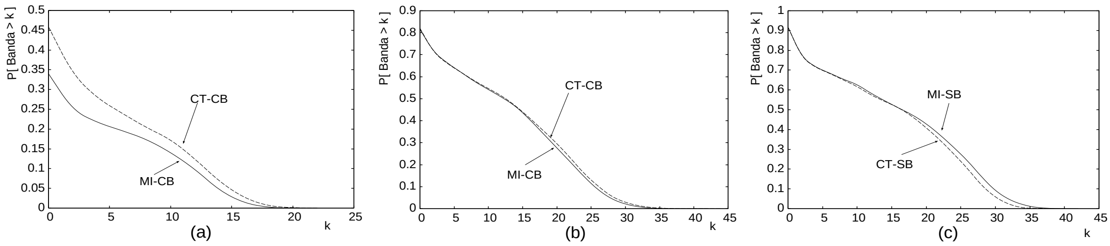

Figura 2. (a) CCDF para MI-CB e CT-CB em UOL; (b) CCDF para MI-CB e CT-CB em eTeach; (c) CCDF para MI-SB e CT-SB em eTeach.

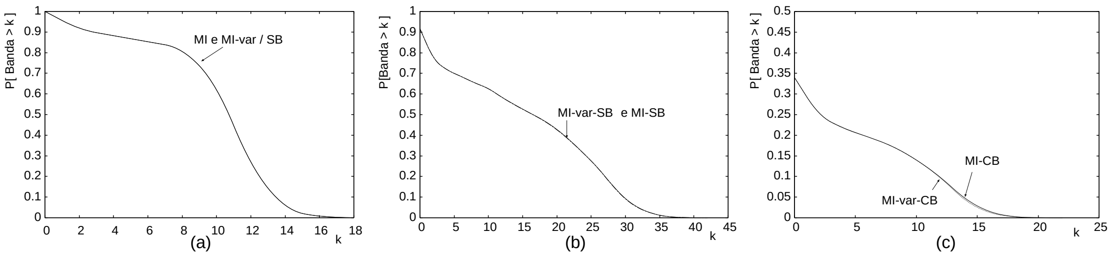

Figura 3. (a) CCDF para MI-SB e MI-var-SB em MANIC-1; (b) CCDF para MI-SB e MI-var-SB em eTeach; (c) CCDF para MI-CB e CT-CB em UOL.

Figuras 4(a), 4(b) e 4(c). Nestas figuras são mostradas as distribuições (CCDF) da banda de MI-SB, para as Cargas eTeach, UOL e MANIC-2, respectivamente, considerando o valor de δ_{MI} como um percentual do tamanho médio de L.

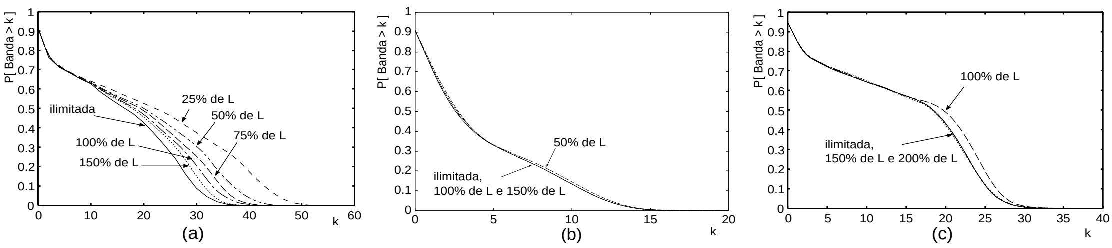

Figura 4. (a) Estudo de δ_{MI} para a Carga eTeach; (b) Estudo de δ_{MI} para a Carga UOL; (c) Estudo de δ_{MI} para a Carga MANIC-2.

Por fim, comparamos a técnica mais eficiente no paradigma de *Patching* com a mais eficiente no paradigma de HSM: técnicas PIE e MI, respectivamente. Em todas as cargas, MI tem melhor performance que PIE. Este fato é ilustrado nas Figuras 5(a), 5(b) e 5(c). Para efeito de justiça, nesta análise utilizamos os valores ideais (obtidos experimentalmente) do parâmetro δ_{after}, explicitados nas próprias figuras em função do valor da janela ótima de *Patching* W. A título de ilustração, as Figuras 8(a) e 8(b) trazem as reduções (%) em relação ao consumo médio de banda e aos valores de pico de banda, respectivamente, registrados por PIE e MI em relação à técnica *Patching*-SB. As reduções são de fato, como já esperado, bem expressivas, tanto para o consumo médio de banda ($\approx 30\% - 68\%$), como para o valor de pico ($\approx 18\% - 62\%$).

A maior eficiência de MI sobre PIE é, contudo, acompanhada por uma maior complexidade de sistema, avaliada por meio da métrica trabalho médio realizado pelo servidor.

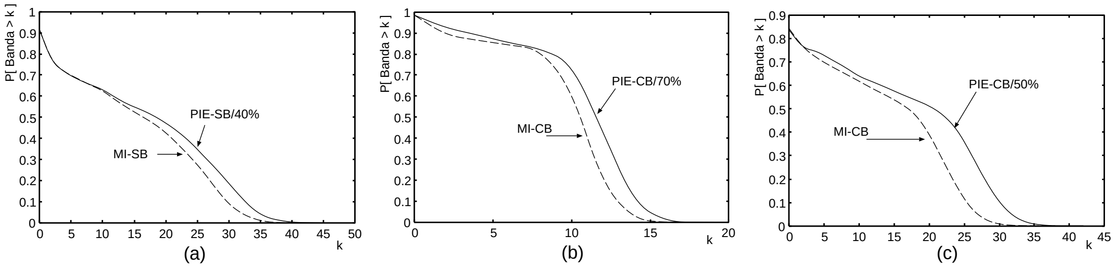

Figura 5. (a) CCDF para PIE-SB e MI-SB em eTeach; (b) CCDF para PIE-CB e MI-CB em MANIC-1; (c) CCDF para PIE-CB e MI-CB em MANIC-2.

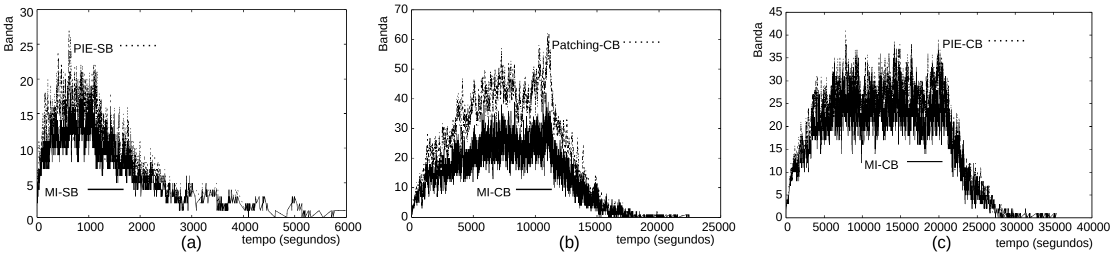

Figura 6. (a) Banda para PIE-SB e MI-SB em UOL; (b) Banda para MI-CB e Patching-CB em eTeach; (c) Banda para MI-CB e PIE-CB em MANIC-2.

Nos experimentos realizados, esta métrica chega a ser mais que uma ordem de grandeza maior para MI. Resultados semelhantes foram apresentados em [13] para a técnica CT em comparação com a técnica PIE. No entanto, se utilizarmos $\delta_{before} \geq$ 1–3% (3–5%) de T, a técnica PIE-SB (PIE-CB) torna-se bastante competitiva com relação a MI-SB (MI-CB) em todas as cargas, exceto para UOL. Esta observação é graficamente ilustrada nas Figuras 7(a), 7(b) e 7(c). Notamos ainda que, para tornar PIE mais eficiente que MI, o valor experimental obtido para δ_{before} é sempre maior no caso de *buffer* completo (CB) do que no caso de *buffer* simples (SB). Isto é explicado pelo fato de que, quando usamos o *buffer* completo, existe uma tendência de termos um menor número de fluxos no sistema e, portanto, uma maior dificuldade para efetuar o compartilhamento de dados. Esta condição favorece à técnica MI por ter um mecanismo de busca mais exaustivo que PIE.

O fato de δ_{before} não melhorar a performance de PIE na carga UOL é explicada a seguir. Avaliamos a probabilidade de $X = 1$, i.e., a probabilidade de que a primeira unidade (do segmento de tamanho médio L do objeto) solicitada pelo cliente seja igual à primeira unidade do objeto. O valor desta probabilidade é ≈ 0.78. Isto significa que a maioria dos clientes requisita a primeira unidade do objeto e, assim sendo, não há fluxos possíveis no limiar de δ_{before} para atender a requisição ocorrida. Portanto, a variação deste parâmetro termina não afetando a distribuição da banda. Por fim, a Figura 8(c) faz uma síntese gráfica comparativa entre MI-SB e PIE-SB considerando dois resultados: (i) o trabalho médio em função do nível de interatividade (número médio de requisições por sessão) e (ii) o valor de δ_{before} para o qual temos praticamente a mesma distribuição de banda em ambas as técnicas. Note que, conforme o nível de interatividade aumenta, o trabalho médio de MI também aumenta enquanto que a descontinuidade introduzida por δ_{before} diminui. Comportamento semelhante é obtido também para o caso com *buffer* completo (CB). Este resultado geral mostra portanto que, apesar de MI ser de fato a técnica de melhor performance, PIE pode ser vista como uma técnica competitiva para cargas em que o nível de interatividade é relativamente alto.

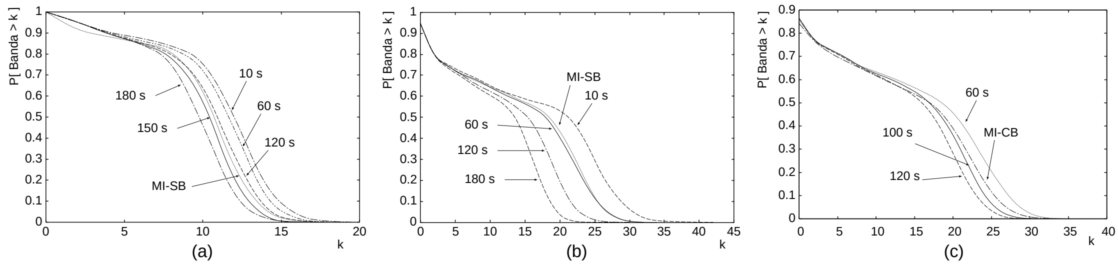

Figura 7. CCDF de PIE com diferentes valores de δ_{before}: (a) Carga MANIC-1 (SB); (b) Carga MANIC-2 (SB); (c) Carga MANIC-2 (CB).

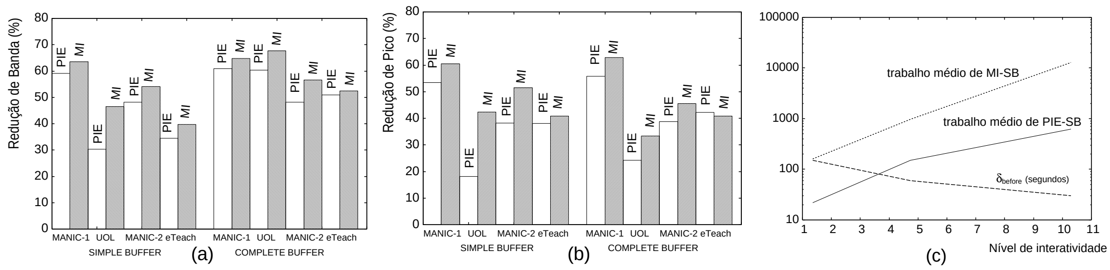

Figura 8. (a) Reduções de banda de PIE e MI em relação à Patching-SB; (b) Reduções de pico de banda de PIE e MI em relação à Patching-SB; (c) Comparação entre MI-SB e PIE-SB.

6. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho apresentamos a nova técnica de compartilhamento de banda *Merge* Interativo (MI), cuja concepção baseia-se no paradigma de HSM. Também realizamos uma análise competitiva minuciosa das mais recentes propostas para cenários com interatividade. A partir dos resultados, vimos que a nova técnica MI é de fato a de maior eficiência. Também observamos, por exemplo, que o paradigma mais simplificado de *Patching* pode tornar-se bastante competitivo com a introdução de pequenas descontinuidades do serviço. Como trabalhos futuros, apontamos avaliações do impacto de descontinuidades do serviço na percepção do cliente, uso inteligente de *buffer* local do cliente conjuntamente com técnicas de compartilhamento de banda e, por fim, análise da possibilidade de aproveitamento de recursos ociosos dos clientes para transmissão de fluxos, criando um ambiente de compartilhamento distribuído.

Referências

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

[2] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon. 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, June 2001.

[3] K. C. Almeroth and M. H. Ammar. 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, August 1996.

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

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

[6] A. Borodin and R. El-Yaniv. *On-line computation and competitive analysis*. Cambridge University Press, The Pitt Building, Trumpington Street, Cambridge, United Kingdom, 1998.

[7] Y. Cai, K. Hua, and K. Vu. Optimizing patching performance. In *Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, pages 204–215, January 1999.

[8] Wun-Tat Chan, Tak-Wah Lam, Hing-Fung Ting, and Wai-Ha Wong. On-line stream merging in a general setting. *Theoretical Computer Science*, 296(1):27–46, 2003.

[9] Wun-Tat Chan, Tak-Wah Lam, Hing-Fung Ting, and Wai-Ha Wong. On-line stream merging, max span, and min coverage. In *Proc. of the 5th Conference on Algorithms and Complexity (CIAC)*, pages 337–346, 2003.

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

[11] C. K. da S. Rodrigues. Resource sharing mechanisms for continuous media delivery in Internet. Qualification Examination, UFRJ, COPPE/PESC, July 2004. In Portuguese.

[12] C. K. da S. Rodrigues and R. M. M. Leão. Novel bandwidth sharing techniques for VoD servers with interactivity. Technical Report, UFRJ-COPPE/PESC, December 2004. In Portuguese.

[13] C. K. da S. Rodrigues and R. M. M. Leão. 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, Maio 2005.

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

[15] E. de Souza e Silva and Rosa M. M. Leão. The Tangram-II Environment. In *Proc.11th Int'l Conference TOOLS2000*, pages 366–369, 2000.

[16] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. In *Proc. 5th Int'l Workshop on Multimedia Information Systems (MIS'99)*, pages 80–87, October 1999.

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

[18] L. Gao and D. Towsley. Supplying instantaneous video-on-demand services using controlled multicast. In *Proc. IEEE Multimedia Computing Systems*, pages 117–121, June 1999.

[19] M. L. Gorza. A resource sharing technique for high-interactivity continuous media delivery and experiments. Master's Thesis, UFF, Computer Science Department, December 2003. In Portuguese.

[20] D. Guan and S. Yu. A two-level patching scheme for video-on-demand delivery. *IEEE Transactions on Broadcasting*, 50(1):11–15, March 2004.

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

[22] E. G. Coffman Jr., P. Jelenkovic, and P. Momcilovic. Provably efficient stream merging. In *Proc. 6th Int'l Workshop on Web caching and Content Distribution*, Boston, MA, June 2001.

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

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

[25] Huadong Ma and K. G. Shin. Multicast video-on-demand services. *ACM SIGCOMM Computer Communication Review*, 32(1):31–43, 2002.

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

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

[28] Marcus Rocha, Marcelo Maia, Ítalo Cunha, Jussara Almeida, and Sérgio Campos. Scalable Media Streaming to Interactive Users. In *MULTIMEDIA'05: Proceedings of the 13th annual ACM international conference on Multimedia*, Singapore, November 2005.

[29] H. Tan, D. Eager, and M. Vernon. Delimiting the range of effectiveness of scalable on-demand streaming. *Performance Evaluation*, 49:387–410, 2002.

[30] Y. W. Wong and Jack Y. B. Lee. Recursive Patching - An efficient technique for multicast video streaming. In *Proc. 5th International Conference on Enterprise Information Systems (ICEIS) 2003*, pages 23–26, Angers, France, April 2003.