COPPE/UFRJ

ALGORITMOS PARA ACESSO INTERATIVO EM APLICAÇÕES DE VÍDEO P2P

Luiz José Hoffmann Filho

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

Orientador: Rosa Maria Meri Leão

Rio de Janeiro

Setembro de 2009

ALGORITMOS PARA ACESSO INTERATIVO EM APLICAÇÕES DE VÍDEO P2P

Luiz José Hoffmann Filho

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

Aprovada por:

Prof.ª Rosa Maria Meri Leão, Dr.

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

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Prof. Artur Ziviani, Dr.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2009

Hoffmann Filho, Luiz José

Algoritmos para acesso interativo em aplicações de vídeo P2P/ Luiz José Hoffmann Filho.– Rio de Janeiro: UFRJ/COPPE, 2009.

XV, 115 p.:il; 29,7 cm

Orientador: Rosa Maria Meri Leão

Dissertação (mestrado) - UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2009

Referência Bibliográficas: p. 107-115

1. Peer-to-Peer. 2. Protocolo BitTorrent. 3. Vídeo sob Demanda. I. Leão, Rosa Maria Meri II.Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A Deus em primeiro lugar.

Aos meus familiares com muito carinho.

Agradecimentos

Primeiramente a Deus, que por sua imensa gratidão e misericórdia nunca nos desampara.

A minha família, pelo amor e apoio incondicionais. Tudo que sou hoje eu devo aos meus pais, Maria Lurdes e Luiz José, que sempre me incentivaram na realização dos meus sonhos, principalmente na minha vida profissional. Aos meus irmãos, Eduardo e Leonardo e a minha irmã Luiza Maria e a meu filho Luiz Guilherme. Não tenho palavras para expressar a minha gratidão.

A minha noiva e futura esposa Dayana pela compreensão e apoio para a realização desse trabalho e principalmente pela paciência em ter de ficar distante durante todo o período da dissertação o que mostra todo o seu amor. TE AMO!!!

Aos meus amigos, sendo que cada um teve uma participação especial nesta caminhada. Primeiramente, aos amigos do LAND. Ao Carlo Kleber pela grande e valiosa ajuda para o desenvolvimento desse trabalho e de alguns artigo, principalmente pelas longas conversas que ajudaram a esclarecer muitas dúvidas. Ao GD pela paciência e ajuda para entender melhor os mistérios do Trangram. Ao Fabrício pela ajuda na administração do LAND e também pela revisão desse texto. Ao Jefferson e Lucas pela ajuda no desenvolvimento do modelo do BitTorrent, ponto de partida para este trabalho. A Carolina Vielmond pela atenção e ajuda a entender o seu modelo. Obrigado também ao Allyson, Bernardo, Guto, Bene, Hugo, Ed, Gaspare, Guilherme Domingues, Flávio, Xandão e Vinícius pela amizade e atenção. Gostaria de agradecer também ao Fred, João Paulo e Daniel pela amizade e atenção.

A Carol, a mãezona de todos do laboratório! Obrigado pelas palavras de carinho, incentivo e apoio em todos os momentos. E não poderia deixar de agradecer pelos inúmeros cafezinhos que me mantiveram acordado e aos pães de queijo!

À Professora Rosa Maria Meri Leão por te sido, além de minha orientadora, uma grande incentivadora. Com sua determinação, conhecimento, paciência e dedicação, me ensinou e iluminou o caminho que tive de percorrer para a realização desse trabalho. Muito cresci intelectualmente e humanamente durante o tempo em que estive sob sua orientação.

A todos os meus professores pelos valiosos ensinamentos e, em especial aos Professores Edmundo A. de Souza e Silva e Daniel Ratton Figueiredo.

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

ALGORITMOS PARA ACESSO INTERATIVO EM APLICAÇÕES DE VÍDEO P2P

Luiz José Hoffmann Filho

Setembro/2009

Orientador: Rosa Maria Meri Leão

Programa: Engenharia de Sistemas e Computação

As aplicações multimídia têm sido empregadas nos mais diversos segmentos, desde *e-learning* até a transmissão de canais de TV pela Internet. Entretanto, para o crescimento dessa indústria é necessário o desenvolvimento de tecnologias e aplicações que proporcionem um serviço com qualidade, eficiência, baixo custo e ainda consigam agregar novas funcionalidades, como interatividade. Uma alternativa é utilizar a arquitetura *Peer-to-Peer* para prover eficiência e baixo custo e aliado a isto, o desenvolvimento de novos algoritmos que consigam prover um serviço com qualidade e com as funcionalidades necessárias para a distribuição de vídeo. Este trabalho propõe um novo algoritmo para distribuição de vídeo sob demanda utilizando o protocolo *BitTorrent* como base, visando tornar a recuperação do conteúdo eficiente quando o acesso dos usuários é interativo. Em sua concepção tem, como inovação principal, a utilização de um modelo que emula o comportamento do usuário. A análise e a validação são feitas por meio de simulações usando cargas obtidas a partir de um servidor multimídia real. Os resultados evidenciam a eficiência do novo algoritmo em relação a outros trabalhos da literatura.

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

ALGORITHMS FOR INTERACTIVE ACCESS IN P2P VIDEO APPLICATIONS

Luiz José Hoffmann Filho

September/2009

Advisor: Rosa Maria Meri Leão

Department: Computer and System Engineering

Multimedia applications have been employed in several segments, from e-learning to the TV broadcast over the Internet. However, in order to this industry to grow, it is mandatory the development of technologies and applications to provide quality of service, efficiency, low cost of deployment and still manage to have functionalities, like interactivity. An alternative is the use of Peer-to-Peer architecture in order to obtain efficiency and lower costs. In parallel, it is necessary the development of new algorithms to improve the quality of service and provide the functionalities needed for video distribution. This work proposes a new algorithm for video distribution based on the BitTorrent protocol aiming at providing efficient recovery content when the users access is interactive. The main feature of the algorithm is the use of a model that emulates the user's behavior. The analysis and validation are done through simulation using logs of a real multimedia system. The results show the efficiency of the new algorithm in comparison to other works of the literature.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nos últimos anos, observou-se uma grande popularização das aplicações multimídia na Internet, principalmente aquelas que têm como objetivo a distribuição de vídeo ou áudio: seja para fins de entretenimento, como um filme ou uma rádio, ou de educação, como uma vídeo aula. Em uma aplicação de *streaming* de vídeo ou áudio, o usuário inicia a sua visualização poucos segundos após iniciar a recepção dos dados do servidor, não existindo a necessidade de receber o arquivo completo para poder visualizá-lo [1]. Um exemplo de aplicação deste tipo é o *YouTube* [2], fundado em 2005, com o objetivo de permitir que o usuário envie e compartilhe facilmente pequenos vídeos na Internet. O *YouTube* tornou-se um fenômeno, chegando a servir mensalmente mais de 100 milhões de vídeos [3].

O *YouTube* utiliza a arquitetura cliente/servidor, auxiliada por CDNs (*Content Distribution Networks*), espalhadas por diversos países, para realizar a distribuição dos vídeos [4]. Nessa arquitetura um servidor é o responsável pelo gerenciamento das requisições dos usuários, enquanto que os demais são pela distribuição dos vídeos/áudios, sendo esses os que formam a CDN. Um aumento no número de usuários requer um maior número de servidores e de banda disponível para servi-los, o que acarreta na elevação dos custos para os provedores de conteúdo. Por isso, diz-se que esse tipo de arquitetura tem escalabilidade limitada e alto custo. Estimativas mostram que, em abril de 2008, a banda consumida pelo *YouTube* foi maior que

a banda total da Internet no ano de 2000 [5] e que são gastos aproximadamente 1 milhão de dólares por dia, somente em banda utilizada [6].

Com o intuito de amenizar os problemas da arquitetura cliente/servidor, foi proposto o *IP multicast*. Essa abordagem mantém um servidor central que realiza a transmissão do fluxo de vídeo/áudio para os usuários, porém por meio de canais *multicast*. A eficiência e a redução nos custos pelo uso de banda são conseguidos porque um mesmo canal pode ser compartilhado por diversos usuários. Vários esquemas desse tipo já foram apresentados na literatura, como por exemplo: os protocolos *Patching* [7] e *Stream Merging* [8]. Conjuntamente a esses protocolos também é possível utilizar-se uma gerência de buffer local [9, 10]. O maior limitante dessa abordagem é o fato de que o serviço *multicast* nem sempre está habilitado ao longo de toda a rede de comunicação a ser utilizada.

Outra abordagem é a arquitetura *Peer-to-Peer* (P2P), muito utilizada para o compartilhamento de arquivos. Os protocolos BitTorrent [11] e Gnutella [12] são exemplos conhecidos desse tipo de arquitetura. Para a distribuição de vídeo/áudio, os usuários se comportam como consumidor, recebendo e visualizando o vídeo/áudio, e também como servidor, distribuindo o vídeo/áudio já recebido. Desta forma, eles estarão colaborando com a distribuição do conteúdo, aumentado a escalabilidade do sistema, reduzindo a banda utilizada do servidor e, por consequência, reduzindo também os custos. Uma limitação na arquitetura P2P decorre do fato de não ter como garantir conexões estáveis e confiáveis entre os nós (*peers*). O ambiente de rede é naturalmente dinâmico: os requisitos de banda podem oscilar significativamente, pois os nós são livres para entrar e sair do sistema tão frequentemente quanto desejarem. Visando resolver estas limitações, algumas propostas tem sido apresentadas, como por exemplo: o *P2Cast* [13], o *BiTos* [14], o *CoolStreaming/Donet* [15], *mTreebone* [16], NetTube [17], Hoffmann *et al* [18], etc.

Os protocolos utilizados para a distribuição de vídeo/áudio têm que levar em consideração algumas características da aplicação. Quando o vídeo/áudio é do tipo sob demanda (*on demand*), um cliente requisita sob demanda um vídeo/áudio já armazenado em um servidor, com é o caso do *YouTube*. Por se tratar de um vídeo

já armazenado, o usuário pode realizar ações interativas, como: saltos para frente e/ou para trás e pausas. Este tipo de aplicação é sensível ao atraso e tolerante a pequenas perdas. Outro tipo de aplicação é aquela dita ao vivo (*live*), muito similar à transmissão tradicional das redes de rádios e/ou televisão, exceto que a transmissão é realizada através da Internet. Como exemplos de aplicações de áudio ao vivo tem-se as diversas rádios que transmitem sua programação pela Internet. Como um vídeo ao vivo não está armazenado em um servidor, não é possível realizar algumas ações interativas, como saltos para frente. Entretanto, se o usuário realizar o armazenamento do que já foi recebido, ele poderá realizar saltos para trás ou pausas. Este tipo de aplicação também é sensível ao atraso, porém é tolerante a pequenas perdas [1].

O nível de interatividade, em ambos os tipos de aplicação, irá depender do seu objetivo. Para uma aplicação de ensino a distância, os usuários não são tão comportados quanto os usuários assistindo a um filme, pois, em uma aula, o aluno deseja executar várias funções como: pausar, parar, retroceder e avançar. Essas funções de interatividade do usuário são essenciais para o bom entendimento da aula por parte do aluno, pois poderá ser feito o retorno a uma determinada parte que ele não entendeu.

Os nós de uma aplicação P2P podem ser organizados em três tipos de estrutura para a distribuição de vídeo/áudio. A primeira, denominada árvore, cria uma relação de pai e filho entre os nós. O pai, ao receber um fluxo, o encaminha para seus filhos sem a necessidade de requisição dos mesmos. Enquanto na segunda, denominada malha (*mesh*), os nós não mantém nenhuma relação de pai e filho. Na estrutura em malha o conceito de fluxo não é mais utilizado e sim o de bloco. Cada bloco contém uma parte do vídeo/áudio. É necessário o envio de pedidos de blocos aos nós vizinhos. A escolha dos blocos a serem pedidos irá depender do algoritmo de seleção utilizado pela aplicação, podendo ser sequencial, aleatório ou mesmo o "bloco mais raro". A terceira estrutura é a união das duas primeiras, denominada híbrida. Nesta estrutura, as relações de pai e filho entre os nós coexistem com outras conexões que não respeitam este tipo de relação. Cada nó recebe um fluxo de vídeo seguindo a

relação pai-filho e ainda realiza pedidos de blocos a seus vizinhos.

Um dos principais protocolos de compartilhamento de arquivos utiliza a estrutura em malha e se mostra muito eficiente, escalável, com baixo *overhead* e de fácil implementação: o protocolo *BitTorrent*. Nele, o arquivo a ser distribuído é dividido em blocos que são transmitidos fora de ordem, podendo ser recuperados em paralelo de diferentes vizinhos. O destaque desse protocolo são os algoritmos de seleção de vizinhos e de seleção de blocos. O algoritmo de seleção de vizinhos é baseado na política *tit-for-tat*, que tenta eliminar os nós egoístas (*free-riders*) e selecionar os nós que mais disponibilizam recursos para o compartilhamento. Esta é uma forma de tornar a distribuição mais justa e eficiente. Já o algoritmo de seleção de blocos seleciona o bloco mais raro (*rarest first*) para ser requisitado. Este algoritmo tenta aumentar a dispersão do objeto no *swarm* - conjunto de nós distribuindo um determinado arquivo - e assim tornar o *swarm* auto-suficiente. No entanto, apesar de sua eficiência para o compartilhamento de arquivos, o protocolo *BitTorrent* não apresenta o mesmo desempenho quando usado para a distribuição de vídeo/áudio. Selecionar sempre o mais raro em vez de tentar recuperar os próximos a serem tocados causa interrupções na visualização. Propostas como *BiToS* [14], Parvez *et al* [19], de *Zhou-Chiu-Lui* [20], de *Shah-Pâris* [21] e Carlsson e Eager [22], utilizam o protocolo *BitTorrent* como base para realizar a distribuição de vídeo/áudio. Isto é possível porque em todas elas são realizadas modificações nos algoritmos, principalmente no algoritmo de seleção de bloco e, às vezes, também, no algoritmo de seleção de vizinhos.

O objetivo do presente trabalho é desenvolver um novo algoritmo para distribuição de vídeo sob demanda com interatividade utilizando a arquitetura *Peer-to-Peer*. Deve-se ressaltar que a interatividade é fundamental para aplicações de ensino à distância [23]. Para o desenvolvimento dessa proposta utilizou-se como base o protocolo *BitTorrent*. Foi modificado o algoritmo de seleção de blocos com a finalidade de atender aos requisitos de tempo e qualidade para a distribuição de vídeo. Já o algoritmo de seleção de vizinhos será o mesmo do protocolo *BitTorrent*. O algoritmo de seleção de blocos é baseado em um modelo de previsão de *Vielmond-Leão-de*

Souza e Silva [24] que emula o comportamento dos usuários. Através do modelo é possível inferir quais ações o usuário poderá realizar no futuro e com isto recuperar previamente os blocos a serem reproduzidos.

Outra contribuição desse trabalho foi o desenvolvimento de um modelo detalhado de simulações do protocolo *BitTorrent* utilizando a ferramenta Tangram-II [25]. Esse modelo compreende as principais funções do protocolo e mostrou um comportamento muito acurado e próximo ao real. Por conseguinte, este foi utilizado como base para o desenvolvimento da nova proposta. Outra contribuição foi a realização de uma abrangente pesquisa na literatura sobre distribuição de *streaming* de vídeo/áudio utilizando arquitetura P2P, seja *streaming* ao vivo ou sob demanda.

Esta dissertação está organizada da seguinte forma: no Capítulo 2, são apresentados alguns conceitos relevantes, além de uma revisão bibliográfica abrangendo os principais trabalhos da literatura e classificando-os de acordo com o tipo de vídeo e estrutura utilizado; no Capítulo 3, é realizada uma revisão de três trabalhos relacionados - *BiToS* [14], de *Zhou-Chiu-Lui* [20] e de *Shah-Pâris* [21] - e utilizados para realizar uma avaliação comparativa de desempenho com a nova proposta; no Capítulo 4, são detalhados o modelo do protocolo *BitTorrent* e a nova proposta de distribuição de vídeo sob demanda com interatividade; no Capítulo 5, é validado o modelo do protocolo *BitTorrent* e apresentada uma análise comparativa da proposta em relação às três propostas da literatura; e por fim, no Capítulo 6, são apresentadas as conclusões e os trabalhos futuros.

Capítulo 2

Conceitos teóricos

Este capítulo é dedicado a fazer uma revisão dos conceitos relevantes para o desenvolvimento deste trabalho. Em paralelo à exposição de cada conceito, é feita menção a outros trabalhos na literatura que, por razões diversas, têm relação com o tema desta dissertação.

Mais especificamente, na Seção 2.1, é apresentada uma síntese da arquitetura *Peer-to-Peer* (P2P), tendo como principal foco o protocolo *BitTorrent*. É também apresentada uma descrição detalhada do funcionamento de seus principais algoritmos, seleção de blocos e seleção de vizinhos. Já na Seção 2.2, são apresentados os principais conceitos sobre a distribuição de vídeo/áudio utilizando a arquitetura P2P e também uma revisão sobre os trabalhos na literatura que abordam este assunto.

2.1 Arquitetura *Peer-to-Peer*

A principal motivação para a utilização de uma arquitetura *P2P* é a escalabilidade, haja vista que quanto maior o número de nós conectados, maior será a demanda e mais recursos estarão disponíveis, tornando o sistema robusto e econômico. Porém, esta arquitetura também apresenta problemas, como o alto custo para o seu gerenciamento e manutenção, devido a conexões intermitentes e à volatilidade

dos nós.

Os primeiros sistemas a utilizarem a arquitetura P2P tinham o objetivo de prover a distribuição e/ou compartilhamento de arquivos, como é o caso do Gnutella [12] e do *BitTorrent* [11]. Porém, devido as suas características de escalabilidade, robustez e economia (recursos), sua aplicação foi estendida a sistemas com outros fins, como a distribuição de vídeo (*PPLive* [26] e *PPStream* [27]), no compartilhamento de CPU e armazenamento, (projetos *SETI@Home* [28] e *OceanStore* [29], respectivamente), e ainda na telefonia IP (*Skype* [30]).

Para estes sistemas funcionarem, é necessário que eles construam uma rede sobreposta sobre a Internet, conhecida também como rede *overlay*. Uma rede sobreposta pode ser definida como sendo uma rede abstrata e/ou lógica sobre a Internet, a qual pode ser representada por um grafo nos seguintes termos: se o nó X mantém uma conexão com o nó Y, então existe uma aresta (ligação) entre X e Y. Esse grafo é formado por todos os nós ativos e conectados por arestas formando, assim, uma rede *overlay*. Não existe uma conexão física entre os nós, mas as arestas representam ligações abstratas entre os nós, sobrepostas às ligações físicas da Internet [1].

Os sistemas P2P também podem ser classificados de acordo com a organização da rede sobreposta [31, 32], que são:

- Não-estruturada: a rede sobreposta é criada de forma totalmente aleatória, não respeitando qualquer tipo de roteamento e/ou mapeamento entre conteúdo e localização. Sendo assim, sua principal desvantagem é o tempo para localizar um determinado conteúdo e a não-garantia de encontrá-lo. Esse sistema ainda pode ser classificado em:
  - Centralizado: existe uma entidade central que mantém informações sobre todos os nós da rede, sendo a responsável pelo gerenciamento das chegadas e/ou partidas dos nós, podendo também realizar o mapeamento do conteúdo existente na rede. A principal vantagem do sistema centralizado é a simplicidade na construção e gerenciamento da rede sobreposta,

porém cria-se um ponto crítico de falha: ocorrendo uma falha nessa entidade central, toda a rede será paralisada. O seu principal exemplo é o protocolo *BitTorrent* [11].

- Descentralizado: não existe qualquer entidade central para o gerenciamento ou busca de conteúdo na rede. Todos esses processos são realizados distribuidamente pelos nós. O método mais utilizado por esse tipo de sistema é a *inundação de consultas (flooding controlado)*, ou seja, um nó envia mensagens de busca por conteúdo para todos os seus vizinhos, e estes, por sua vez, respondem a essa mensagem e a retransmitem aos seus vizinhos. Este método é muito custoso e sem garantias de sucesso, mas se mostra muito robusto a falhas. O seu principal exemplo é o *Gnutella* (primeira versão) [12].
- Híbrido: este tipo de sistema se assemelha, inicialmente, aos sistemas descentralizados, haja vista que não existe, oficialmente, uma entidade central de controle. Entretanto, diferentemente dos sistemas descentralizados, nem todos os nós são iguais nesse tipo sistema. Os nós mais poderosos - isto é, os com grande quantidade de recursos disponíveis - são designados líderes (super-nós) de grupo e têm maiores responsabilidades. Se um nó não é líder de grupo, ele fica associado a um líder de grupo. A função de um líder de grupo é gerenciar a chegada e partida de nós e a busca pelo conteúdo na rede. A principal vantagem é a fácil manutenção da rede e um melhor desempenho na busca pelos dados. Entretanto, esta busca não tem garantias de sucesso. Aplicações como *KaZaA* [33], *Skype* [30] e a nova versão do *Gnutella* [12] são exemplos deste tipo de sistema não estruturado híbrido.

- Estruturada: é criada uma rede sobreposta, onde é estabelecido um roteamento prévio para cada um dos nós participantes, provendo um mapeamento entre conteúdo (dados) e localização (nó) na forma de uma tabela de roteamento distribuído. São tipicamente baseados na tecnologia *Distributed Hash Table (DHT)*, que consiste de uma identificação lógica utilizando no espaço da

chave de *hash* a identificação do nó e quais os recursos que ele tem disponível. As suas vantagens são a rápida localização dos dados e a busca garantida. Porém, sua manutenção é complexa e custosa, principalmente em ambientes com altas taxas de chegada e partida. Seus principais exemplos são: *Chord* [34], *PAST* [35].

A compreensão dessa classificação ajuda a entender as características, vantagens/desvantagens e pontos de falha dos muitos sistemas que utilizam a arquitetura *P2P*.

Os primeiros sistemas que utilizaram a arquitetura P2P para a distribuição de conteúdo, como *Napster* [36] e *Gnutella* [12], se mostraram eficientes e robustos. Porém, uma nova geração foi desenvolvida e rapidamente se tornou popular, como é o caso do protocolo *BitTorrent*. Essa aplicação apresentou novas características que aumentaram a eficiência e robustez, como: o desenvolvimento de um simples mecanismo para encorajar os usuários a compartilharem seus recursos, representado pela política *tit-for-tat*, resultando em uma diminuição no número de nós egoístas (*free-riders*), ou seja, nós que não querem compartilhar seus recursos com os demais. Em segundo lugar, a divisão do objeto em *blocos*, que são recuperados paralelamente utilizando a política do bloco mais raro (*Rarest First*). Esse mecanismo utiliza a banda disponível do nó alcançando alta eficiência. Em terceiro lugar, o armazenamento do *hash* de cada bloco, ou seja, uma assinatura que garante a integridade dos dados, evitando ataques por poluição. Na Subseção 2.1.1, será apresentada uma descrição mais detalhada do protocolo *BitTorrent*.

2.1.1 Protocolo *BitTorrent*

O *BitTorrent* [11] foi desenvolvido para a distribuição de conteúdo sobre uma rede P2P. Diferentemente de outras aplicações, o *BitTorrent* não provê nenhuma funcionalidade de pesquisa ou busca pelo conteúdo; seu foco principal está na distribuição.

Para a distribuição de um determinado objeto (conteúdo) é necessária a criação de uma nova rede sobreposta não-estruturada centralizada, denominada de *swarm*, formada pelo servidor, conhecido por *tracker*, e pelos nós (usuários), que podem ser classificados como *seeds* ou como *leechers*. O *tracker* é o responsável pelo gerenciamento e manutenção do *swarm*. Os nós são classificados como *seeds* quando já têm o objeto completo, apresentando-se como distribuidores primários. Por outro lado, os nós que ainda não têm o objeto completo são denominados *leechers*, que além de recuperar os dados, também auxiliam na disseminação do objeto, apresentando-se como distribuidores secundários.

A distribuição de um objeto inicia-se com a geração de um arquivo *.torrent*. O conteúdo desse arquivo contém informações necessárias para a distribuição do objeto, como: número de blocos no qual o objeto foi divido - tipicamente, o objeto é dividido em blocos de $256KB$ e cada bloco é dividido em pedaços de $16KB$ - , *hash* de todos os blocos para validação, endereço IP e porta do *tracker*. A criação desse arquivo é devido ao fato de não existirem outros meios para realizar buscas de conteúdo. Desta forma, publica-se esse arquivo em páginas *web* para que os usuários possam ter acesso.

De posse do arquivo *.torrent*, o usuário tem as informações necessárias para entrar no *swarm*. Primeiramente, o nó estabelece uma conexão com o *tracker* que, por conseguinte, envia uma lista com IPs dos nós (tipicamente são 50 nós) que já fazem parte do *swarm*. Imediatamente após o recebimento da lista, o nó tenta estabelecer conexões TCP com os nós da lista. Aqueles com os quais a conexão for estabelecida com sucesso formam a sua vizinhança. Uma vez que a conexão esteja estabelecida, são trocadas algumas informações entre os nós, como o *bitfield*, que é o mapa dos blocos que o nó tem disponível para compartilhar com seu vizinho.

Com as conexões estabelecidas entre os nós, cada nó presente no *swarm* executa os algoritmos de seleção de blocos e seleção de vizinhos, identificando quais blocos requisitar e quais vizinhos servir, respectivamente. A seguir é apresentado uma descrição detalhada desses dois algoritmos do protocolo *BitTorrent*.

Algoritmo de Seleção de Vizinhos

O algoritmo de seleção de vizinhos encoraja os nós a contribuírem com a maior quantidade possível de seus recursos, como banda de *upload*, para obterem também uma quantidade considerável dos recursos de seus vizinhos, como banda de *download*. Essa política é conhecida como *tit-for-tat* e tem como objetivo, além de melhorar as taxas de compartilhamento, também evitar os *free-riders* que são penalizados por essa política e não conseguem recuperar o objeto com taxas satisfatórias. Quando esse algoritmo é executado os vizinhos de um nó serão classificados em dois conjuntos: os bloqueados (*chocke*), que não podem realizar pedidos e os desbloqueados (*unchocke*), que podem efetuar pedidos. Além disso, os nós classificam os seus vizinhos segundo o seu interesse por ele: o nó pode estar interessado no vizinho, ou seja, o vizinho tem blocos que ele não tem e, que ele deseja recuperar; ou não-interessado, ou seja, o nó já tem todos os blocos que o vizinho tem.

O algoritmo apresenta duas versões, uma para nós *leechers* e outra para nós *seeds*. Basicamente, a diferença está na forma como os vizinhos são avaliados para serem desbloqueados. Para um *leecher*, o vizinho é avaliado considerando sua taxa de *upload*, enquanto que o *seed* avalia seus vizinhos pelo intervalo de tempo em que eles estão desbloqueados.

Primeiramente, será descrito o algoritmo para os nós *leechers*. Seja **P** um nó ativo em um *swarm* e apto a executar o algoritmo de seleção de vizinhos. Ele poderá escolher n vizinhos para serem desbloqueados, sendo n determinado pela banda de *upload* disponível. Por exemplo, se a banda disponível de **P** é de $40KB/s$, o número máximo de nós que poderão ser desbloqueados é de $n = 4$. O algoritmo é executado em intervalos de tempo $t = 10$ segundos ou em todas as ocasiões, nas quais um dos vizinhos deixar o *swarm* ou um vizinho que está desbloqueado e interessado passa a não ter mais blocos de interesse, ou seja, não-interessado. Consequentemente, o intervalo de execução pode ser menor do que os $t = 10$ segundos. Todas as vezes que o algoritmo é executado, uma nova etapa é iniciada e os seguintes passos são executados:

1. **P** ordena todos os seus vizinhos que estejam interessados nele e que disponibilizaram algum recurso para ele. A ordenação é decrescente com a taxa de *download* recebida por **P** de cada um deles. Os vizinhos que não compartilharem seus recursos com **P**, ou seja, não enviaram nenhum bloco nos últimos 30 segundos, serão classificados como *free-riders* e não serão selecionados para receber recursos de **P**, garantindo que somente os vizinhos que contribuíram e que estão interessados serão desbloqueados;

2. Os $n - 1$ primeiros vizinhos são selecionados para serem desbloqueados, ou seja, os que ofereceram a maior taxa de *download*;

3. A cada três etapas, **P** seleciona um vizinho, aleatoriamente, que esteja interessado, para ser desbloqueado via uma procedimento chamado *optimistic unchocke*; se este vizinho não faz parte dos já selecionados a etapa está concluída. Caso contrário, um novo candidato é selecionado aleatoriamente e o procedimento se repete até que um vizinho seja encontrado ou que não existam mais vizinhos para serem escolhidos.

Portanto, **P** poderá ter mais do que n vizinhos no estado desbloqueado, segundo o algoritmo de seleção de vizinhos, mas somente k ($k \leq n$) vizinhos estarão interessados em **P** e poderão ser desbloqueados na mesma etapa.

Existem dois propósitos para a utilização do *optimistic unchocke*: possibilitar a descoberta de novos vizinhos e habilitar novos vizinhos, que acabaram de entrar no *swarm* e que não têm nenhum bloco, a iniciarem o compartilhamento de seus recursos.

Um nó *seed* executará um algoritmo de seleção diferente, levando em consideração outras métricas para determinar quem terá acesso aos seus recursos. Seja **Q** um nó *seed*. Ele irá escolher n vizinhos para serem desbloqueados. Esse algoritmo é executado no mesmo intervalo de tempo $t = 10$ ou em todas as ocasiões, nas quais um de seus vizinhos deixar o *swarm* ou um vizinho que está no estado desbloqueado e interessado torna-se um *seed*. Consequentemente, o intervalo de execução pode ser

menor do que $t = 10$ segundos. Todas as vezes que o algoritmo é executado uma nova etapa é iniciada. A seguir, será feita uma descrição do algoritmo executado por um nó *seed*:

1. **Q** ordena todos os seus vizinhos, que se encontram no estado desbloqueado, de acordo com o tempo da última mudança do estado bloqueado para desbloqueado, sendo do tempo mais recente para o mais antigo. A ideia é priorizar os vizinhos que estão desbloqueados há menos tempo. Em caso de empate, os vizinhos são ordenados pela taxa de *download* que o *seed* enviou para cada um, onde a maior prioridade é atribuída para maior taxa. **Q** escolhe então os n primeiros vizinhos para serem desbloqueados.

2. **Q** irá escolher a cada 3 etapas um vizinho, aleatoriamente, via *optimistic unchocke*. O vizinho permanece no estado desbloqueado até a próxima etapa.

O passo 1 do algoritmo acima garante que os nós não permanecem por longos períodos realizando pedidos ao *seed* **Q**, somente pelo fato de terem grande disponibilidade de recursos. Pelo Algoritmo, após **Q** ter enviado dados a um de seus vizinhos por um período de X etapas, ele modificará o estado de seu vizinho para bloqueado e selecionará um novo para servir. Assim, **Q** poderá servir tanto os vizinhos que chegaram primeiro ao *swarm*, quanto os que chegaram mais tarde, prevenindo desta forma que alguns poucos vizinhos monopolizem **Q**.

Muito da eficiência do protocolo *BitTorrent* é devido a este algoritmo de seleção de vizinhos, que se mostrou muito eficiente em seu propósito de eliminar os *free-riders* e de prover distribuição uniforme dos blocos entre os nós. Entretanto, existem algumas características do algoritmo que acabam prejudicando os nós. Por exemplo, considere um swarm onde existem vários usuários de banda larga, ou seja, que tem alta capacidade de *upload* e *download* e apenas um usuário com baixa capacidade. Mesmo que este usuário com baixa capacidade disponibilize toda a sua banda disponível para uso do *BitTorrent*, ele acabará sendo punido pelo algoritmo, uma vez que os demais nós têm mais capacidade do que ele e praticam taxas maiores que as

dele. Trabalhos como de Legout *et al* [37] vêm estudando formas de minimizar esta injustiça.

Algoritmo de Seleção de Blocos

O algoritmo de seleção de blocos é dividido em 3 fases: a primeira é a inicialização, com a função de recuperar um número mínimo de blocos iniciais para que o nó possa interagir com os vizinhos mais rapidamente; a segunda é regida pela política *rarest first*; enquanto a terceira é a finalização. A seguir serão explicadas com mais detalhes cada uma dessas fases.

1. A primeira fase, de inicialização, é conhecida como *random first* e seu principal objetivo é permitir que o nó recupere os primeiros pedaços mais rapidamente. Isto é necessário para que o nó obtenha rapidamente alguns blocos para disponibilizar aos seus vizinhos. Por padrão, são recuperados pelo menos 4 blocos nesta fase, sendo pedidos aleatoriamente. Após a recuperação deste número mínimo, o algoritmo passa para a fase seguinte;

2. A segunda fase é a utilização da política *rarest first*, onde serão recuperados os blocos seguindo a política de sempre recuperar o bloco mais raro localmente. Esta estratégia é implementada da seguinte forma: cada nó do *swarm* recebe e armazena o mapa de blocos (*bitfield*) de cada um de seus vizinhos e também recebe mensagens para atualizar estes mapas. Sendo assim, um nó tem o conhecimento de quais blocos estão disponíveis, localmente, entre seus vizinhos. De posse dessa informação, é realizado um cálculo para saber quantas cópias cada bloco tem entre seus vizinhos. Seja $K_i > 0$ o número de cópias do bloco i na vizinhança de um nó. O bloco mais raro é o bloco com menor valor de K_i. Portanto, o nó, sabendo qual é o bloco mais raro, realiza o seu pedido para os respectivos vizinhos. Na ocorrência de mais de um bloco com o mesmo número de cópias, este será escolhido aleatoriamente.

3. A terceira fase, de finalização, é conhecida como modo *end game*. Essa fase

é executada pelo nó quando este já efetuou o pedido de todos os blocos, sem que tenha recebido todos eles. Nesse modo, o nó realiza pedidos dos blocos para todos os seus vizinhos que os tenham disponíveis. Entretanto, quando um bloco é recebido, uma mensagem de cancelamento do pedido é enviada a todos os vizinhos que receberam o pedido desse bloco. A utilização dessa fase tem o objetivo de resolver ou minimizar os efeitos do problema do último bloco. Esse problema surge somente ao final da recuperação do objeto, quando faltam poucos blocos a serem recuperados, pois alguns blocos podem ser disponibilizados por poucos vizinhos, que acabam sendo sobrecarregados pela quantidade de pedidos e consequentemente não conseguem atender a todos.

Durante essas três fases é utilizada a política *strict priority*. O objetivo dessa política é dar celeridade ou maior prioridade para terminar a recuperação de um bloco que já tenha começado a ser recebido. Os pedaços que compõem os blocos são recuperados paralelamente de um ou mais vizinhos e, somente após ter o bloco completo, ele é disponibilizado para que os seus vizinhos possam recuperá-lo.

2.2 Distribuição de Vídeo Utilizando a Arquitetura *Peer-to-Peer*

A arquitetura P2P vem sendo empregada também na distribuição de conteúdo multimídia. Entende-se por conteúdo multimídia um arquivo de vídeo ou áudio, podendo ser **sob demanda** (VoD), ou seja, pré-armazenado e/ou codificado, ou **ao vivo**, ou seja, sendo transmitido durante sua produção e sendo visualizado pelos usuários poucos segundos após iniciada a recepção dos dados. Alguns sistemas em operação, tais como *SopCast* [38], *Joost* [39], *PPLive* [26] e *PPStream* [27], já empregam a arquitetura P2P para a distribuição de vídeo e existem muitos outros trabalhos na literatura, como Carlsson e Eager [22], Dana *et al* [40] e Guo *et al* [13], que propõem novos sistemas para a distribuição de vídeo/áudio. Muitas dessas propostas utilizam protocolos já conhecidos para a distribuição de objetos, sendo

um exemplo o protocolo *BitTorrent*, utilizado nas propostas de *Shah-Pâris* [21], *BiToS* [14], Dana *et al* [40] e Choe *et al* [41]. Porém, a utilização desses protocolos, como é o caso do *BitTorrent*, implica em uma série de ponderações a serem feitas, uma vez que eles não foram projetados para a distribuição de conteúdo com severas restrições de tempo e qualidade, como é o caso da distribuição de vídeo/áudio.

Os sistemas de distribuição de vídeo P2P, basicamente, utilizam uma organização não-estruturada centralizada, e são classificados segundo [42]: a estrutura utilizada para entregar o fluxo de dados e o tipo de vídeo distribuído. Na Subseção 2.2.1, será feita uma discussão sobre os prós e os contras de cada uma das estruturas. Em seguida, na Subseção 2.2.2, serão classificados os principais trabalhos sobre a distribuição de vídeo P2P de acordo com o tipo de vídeo distribuído.

2.2.1 Estruturas

Basicamente existem três estruturas para a distribuição de vídeo P2P: a estrutura em árvore (*tree*), em malha *(mesh)* e a híbrida [43]. As duas primeiras se diferenciam na forma como os nós se organizam para encaminhar o fluxo de vídeo, como pode ser observado na Figura 2.1. Na estrutura em árvore (Figura 2.1(a)) os nós se organizam em uma única ou em múltiplas árvores, nas quais a fonte de vídeo é a raiz da árvore. São formadas relações de pai e filho entre os nós, nas quais somente o pai encaminha o fluxo de vídeo a seus filhos. Sendo assim, cada nó que desejar participar da distribuição, deverá entrar na árvore e, sem que este faça requisições, começará a receber o fluxo de vídeo, podendo ser do nó fonte ou de outro nó. Por outro lado, na estrutura em malha (Figura 2.1(b)) os nós formam uma malha, ou seja, uma rede sobreposta totalmente distribuída e realizam a disseminação do vídeo em blocos - pequenos fragmentos do vídeo de tamanho fixo. Os nós não possuem funções específicas, podem tanto receber quanto enviar os blocos a quaisquer outros nós. Em ambas as estruturas existem nós fontes, que somente encaminham os blocos, contudo, na estrutura em malha, um nó pode receber de vários nós fontes simultaneamente. Vale ressaltar que os nós pertencentes à malha de distribuição têm

que saber quais blocos cada um de seus vizinhos possui e explicitamente requisitar o bloco que deseja ao vizinho indicado. Já a estrutura híbrida (Figura 2.1(c)) é a união das estruturas em árvore e em malha, proposta com o intuito de construir uma estrutura que maximize as boas características e minimize os problemas de ambas, tornando-se eficiente e robusta.

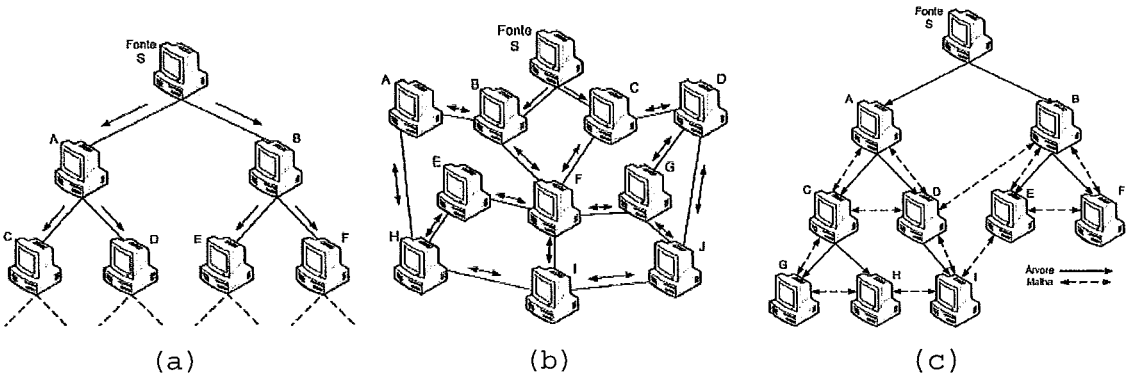

Figura 2.1: Estruturas de distribuição de vídeo em redes Peer-to-Peer: (a) Estrutura em árvore; (b) Estrutura em malha; (c) Estrutura híbrida [1].

Estrutura em Árvore

Similar à árvore formada pelo *IP Multicast* em nível de rede, os usuários participantes de um fluxo de vídeo podem formar uma árvore de distribuição em nível de aplicação, tendo como raiz a fonte do fluxo. Cada usuário que chega à árvore entra em um determinado nível. Os usuários pertencentes à árvore possuem relações de pai (nível superior) e filho (nível inferior), ou seja, quando um nó recebe um dado de seu pai, ele encaminha uma cópia para cada um de seus filhos, sem que exista uma requisição explícita por parte deles. Desta forma, o fluxo é distribuído para os filhos. Os trabalhos Chu *et al* [44] e Jannotti *et al* [45] são exemplos de distribuição de vídeo utilizando uma estrutura em árvore.

Dado um conjunto de nós, existem várias maneiras de construir uma árvore para a distribuição de vídeo. As principais características para a construção de uma árvore são a altura e o grau do nó. Os nós dos níveis inferiores (filhos) recebem o fluxo após os nós dos níveis superiores (pais) receberem. Para reduzir o atraso

imposto aos nós filhos, é preferível que a árvore tenha o menor número de níveis possível, pois quanto maior a altura da árvore, maior será o atraso para os nós mais distantes da raiz. Uma forma de reduzir a altura da árvore é aumentar o grau dos nós, ou seja, aumentar o número de filhos que cada nó pode servir. Entretanto, o grau máximo de um nó é limitado pela sua capacidade de *upload*. Uma proposta para a construção da árvore que leva em conta o compromisso entre a altura da árvore e o grau dos nós pode ser encontrada em Miranda e Figueiredo [46].

Outro fator importante é a manutenção da árvore, uma vez que as chegadas e partidas de usuários na árvore podem ser frequentes, seja por problemas na rede, na máquina do usuário ou por seu desejo. Após a partida de um nó, todos os seus descendentes na árvore ficam desconectados da fonte do fluxo e não receberão mais o vídeo. Uma forma de minimizar este problema é utilizar algoritmos que recuperem a estrutura da árvore o mais rápido possível.

A construção e manutenção da árvore podem ser realizadas de forma centralizada ou distribuída. Na solução centralizada, um nó central é a fonte do fluxo e também é responsável pela construção e a manutenção da árvore. Na ocorrência da chegada de um novo nó à árvore, primeiro ele conecta-se ao nó central e, baseado na topologia da árvore existente e nas características do novo nó, como localização e capacidade, o nó central decide a posição do novo nó na árvore e notifica o nó-pai a que ele irá se conectar. O nó central, identificando uma partida da árvore, seja por notificação do próprio nó ou por *timeout*, reorganiza toda a árvore. Entretanto, para grandes sistemas de distribuição de vídeo em árvore, a utilização de um nó central pode ser um ponto único de falha ou mesmo um gargalo do sistema. Portanto, alguns algoritmos de construção e manutenção distribuídos têm sido propostos, como por exemplo Tran *et al* [47] que mostrou melhor desempenho e robustez.

Quando um nó deixa a árvore, todos os seus descendentes deixam de receber o fluxo de vídeo, o que pode acarretar em descontinuidades na reprodução. Sendo assim, o tempo de reparo da árvore deve ser pequeno a fim de reduzir a perda de pacotes causada pela saída do nó-pai. Outra desvantagem é que todos os nós-folhas (nós sem filhos, pertencentes ao último nível) não contribuem com a distribuição do

vídeo, uma vez que eles não têm a quem enviar o fluxo, o que se torna ainda mais grave considerando que grande parte dos nós de uma árvore são folhas.

Uma solução para resolver o problema da não-contribuição pelos nós-folhas e minimizar os efeitos das falhas foi proposta em Magharei e Rejaie [48], denominada de estrutura de múltiplas árvores. O nó-fonte divide o fluxo em n sub-fluxos, formando assim n sub-árvores, uma para cada sub-fluxo. Cada novo usuário fará parte de todas as sub-árvores e receberá todos os sub-fluxos. Dentro de cada sub-árvore, o funcionamento é idêntico ao de uma árvore simples, pois o fluxo passa de pai para filho. Um nó tem diferentes posições em cada uma das sub-árvores, ou seja, em uma sub-árvore ele é um nó interno, enquanto que em outra ele poderá ser um nó-folha. Desta forma, todos os nós contribuirão para a distribuição do vídeo, uma vez que ele será um nó interno em pelo menos uma sub-árvore. O número máximo de sub-árvores em que um nó pode ser nó interno é limitado pela sua capacidade.

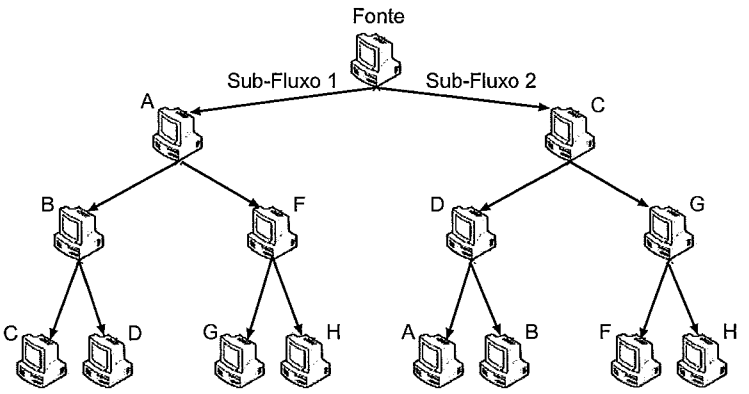

Figura 2.2: Estrutura em múltiplas árvores, com dois sub-fluxos e sete nós.

Na Figura 2.2 é apresentado um exemplo da formação com múltiplas árvores. Existem sete nós e são criados dois sub-fluxos. A fonte divide o fluxo em dois sub-fluxos e envia um para a sub-árvore esquerda (sub-fluxo 1) e o outro para a direita (sub-fluxo 2). Os nós **A**, **B**, e **F** são nós internos na sub-árvore esquerda e folhas na direita. Similarmente, os nós **C**, **D** e **G** são nós internos na sub-árvore direita e folhas na esquerda.

Estrutura em Malha

Os sistemas P2P para distribuição de vídeo em árvore mostram-se frágeis, em cenários com muitas chegadas e partidas de nós devido à manutenção da estrutura da árvore. Outra estrutura que pode ser utilizada para este fim, totalmente distribuída, não existindo nenhuma topologia estática pré-determinada é conhecida como malha (*mesh*). Os trabalhos de Megharei e Rejaie [49] e de Zhang *et al* [18] são alguns dos exemplos que utilizam essa estrutura.

Na estrutura em malha, todos os nós estabelecem relações dinamicamente e, em um curto intervalo de tempo um nó já terá diversos vizinhos, com os quais poderá realizar a troca de dados, de acordo com a sua necessidade e/ou disponibilidade. Ocorrendo a partida de um de seus vizinhos, o nó não ficará desconectado da rede e continuará recebendo o conteúdo dos demais vizinhos, ao mesmo tempo em que poderá buscar por novos vizinhos. Essas características tornam a estrutura em malha extremamente robusta para ambientes dinâmicos.

Para construir e manter um sistema para distribuição de vídeo P2P em malha é necessária a utilização de um nó central (servidor) responsável pela monitoração da distribuição e gerenciamento de todas as ações, como chegadas e partidas de nós da malha.

Na estrutura em árvore, o fluxo é orientado da fonte para todos os nós ao longo da árvore. Na estrutura em malha, devido à sua topologia, o conceito de fluxo não se aplica. Para essa estrutura, a unidade básica de dados é o bloco de vídeo, ou seja, o nó fonte (servidor) divide o vídeo em pequenos blocos, cada um contendo um pequeno fragmento do vídeo, como por exemplo, um intervalo de 1 segundo. Cada bloco contém um número de sequência e por conseguinte, os blocos com os menores números de sequência representam a parte inicial do vídeo. Como cada bloco pode fazer uma rota diferente, acabam ocorrendo chegadas fora de ordem. Portanto, o nó precisa armazenar os blocos e colocá-los na ordem correta para então começar a sua reprodução.

Existem duas políticas principais para recuperação de vídeo: *push* (empurrar) ou *pull* (puxar). Na estrutura em árvore é utilizada a política de empurrar, uma vez que a estrutura propicia a utilização dessa política, na qual o fluxo do vídeo é enviado para os nós-filhos assim que o nó-pai o recebe, sem a necessidade de um pedido do nó-filho. Entretanto, utilizar essa política para a estrutura em malha pode acarretar alguns problemas, uma vez que não existe uma estrutura definida, nem mesmo uma relação de pai e filho entre os nós. Um problema que pode ocorrer é o recebimento de blocos duplicados ou não mais necessários, o que acarretará desperdício de recursos do nó. A política de empurrar pode ser utilizada na estrutura em malha, desde que seja estabelecida uma relação de pai-filho, e esta relação deve ser refeita em todas as ocasiões de partida e/ou chegada de novos nós à malha. Consequentemente, isto causa um aumento do *overhead* total do sistema.

A política puxar é a mais indicada para ser utilizada na estrutura em malha, uma vez que o nó realiza pedidos dos blocos que deseja para seus vizinhos, evitando assim os problemas que acontecem com a política empurrar. O funcionamento da política puxar é baseado na troca do mapa de blocos de cada nó periodicamente, isto é, o envio de mensagem contendo o número de cada bloco disponível para troca. Após receber o mapa de blocos de seus vizinhos, o nó pode decidir quais blocos de seus vizinhos deseja requisitar, e enviar uma mensagem de pedido ao seu vizinho.

Estrutura Híbrida

Nem a estrutura em árvore nem em malha resolvem completamente os desafios introduzidos pela dinâmica dos sistemas P2P na distribuição de vídeo. A estrutura em árvore sofre com a instabilidade provocada pela chegada e partida de nós, com a sobrecarga de controle para manter a árvore, com a sub-utilização de banda dos nós-folha e com a escolha de nós-pai ineficientes. Por outro lado, a estrutura em malha apresenta-se mais simples, porém com um compromisso entre latência e sobrecarga de controle. Se os usuários enviarem notificações a cada bloco de vídeo recebido, a sobrecarga de controle aumenta. Enviar os mapas de blocos periodicamente reduz

a sobrecarga de controle, porém aumenta a latência.

Portanto, uma questão que surge é: como combinar as duas estruturas para construir uma estrutura de distribuição híbrida, eficiente e robusta, unindo o que há de melhor em cada uma das estruturas? Os trabalhos *GridMedia* [50], *CoolStreaming+* [51], *mTreebone* [19] e *AnySee* [52] já tentam responder esta pergunta. No trabalho [19] é proposta a utilização de uma estrutura híbrida, denominada *backbone* em árvore. Segundo [19] existem evidências de que mesmo utilizando uma estrutura em malha, a distribuição de um objeto tende a formar uma árvore específica ou um pequeno conjunto de árvores. Essas árvores são formadas por um subconjunto de nós que demonstram ser mais estáveis, ou seja, permanecem por longos períodos no sistema. Portanto, o trabalho propõe que seja mantida uma hierarquia no núcleo com esses nós estáveis, enquanto que os demais nós se organizem ao redor do núcleo em uma estrutura em malha. Entretanto, o desafio é identificar e posicionar os nós estáveis para formar o núcleo apropriado. Além de que algumas questões devem ser resolvidas antes, nem sempre os nós estáveis são os que apresentam a maior capacidade, e se forem colocados nós de baixa capacidade no núcleo, eles podem prejudicar o desempenho de seus filhos de alta capacidade. Além disso, a estabilidade desse sistema depende da vontade humana, ou seja, quando o usuário irá desejar entrar ou sair do sistema.

2.2.2 Tipos de Vídeos

A distribuição de vídeo ou áudio pode ser classificada em dois tipos: ao vivo (Live) ou sob demanda (VoD). No vídeo/áudio ao vivo, o conteúdo é disseminado a todos os usuários em tempo real, como a programação de uma rede de televisão ou de uma estação de rádio pela Internet. Desta forma, todos os usuários estão sincronizados, ou seja, assistindo ou ouvindo o mesmo conteúdo. Já no caso do vídeo sob demanda, o usuário tem a flexibilidade de assistir a qualquer parte do vídeo no momento em que desejar, uma vez que o vídeo completo já está armazenado.

Distribuição de vídeo ao vivo

A distribuição de vídeo ao vivo é similar às tradicionais transmissões de rádio e televisão, exceto que o meio utilizado para a transmissão é a Internet. Elas permitem ao usuário receber a programação ao vivo de uma rede de televisão ou de rádio, estando ele em qualquer lugar do mundo. Neste tipo de aplicação não existe um armazenamento prévio das informações, os vídeos são transmitidos logo após a sua produção. Nos dias de hoje, existem milhares de estações de rádio transmitindo sua programação na Internet e uma grande quantidade de sistemas que também distribuem a programação de emissoras de televisão, como por exemplo *Sopcast* [39], *PPLive* [27] e *PPStream* [28].

Como o fluxo ao vivo não pode ser previamente armazenado, o usuário não pode realizar uma ação interativa de salto para frente, ou seja, assistir a uma parte futura do vídeo. Entretanto, com o armazenamento local dos dados já recebidos, em alguns sistemas, é possível realizar ações de pausa ou salto pra trás, ou seja, assistir novamente a um determinado trecho do vídeo.

Existem várias propostas na literatura de sistemas para a distribuição de vídeo ao vivo utilizando arquitetura P2P. Em alguns trabalhos, esses sistemas são denominados como protocolo *IPTV*. Eles se diferenciam, principalmente, pela estrutura utilizada, seja ela árvore, malha ou híbrida. As principais propostas utilizando a estrutura em árvore são: o *NICE* [53], *ZIGZAG* [47, 54], *Overcast* [45] e o *ESM* [44]. Todas essas propostas se caracterizam por utilizar uma árvore simples, enquanto que a proposta *SplitStream* [48] é caracterizada pela utilização de múltiplas árvores. Para estrutura em malha, podem ser citados os trabalhos *Donet/CoolStreaming* [18], *PRIME* [49] e *ChainSaw* [55], todos desenvolvidos pela comunidade acadêmica. Existem também sistemas desenvolvidos pela indústria, como *PPLive* [27, 56], SopCast [39, 56] e *PPStream* [28], mas como são proprietários, não existe uma especificação ou código fonte aberto e, por isto, muitas das características relacionadas a estrutura desses sistemas não são conhecidas. Existem trabalhos que tentam estimar características desses sistemas, como de Sentinelli *et*

al [56]. Além disso, existem alguns trabalhos que utilizam a estrutura híbrida, como o *GridMedia* [50], *CoolStreaming+* [51], *mTreebone* [19] e *AnySee* [52].

Muitos desses sistemas já estão em uso ou em fase de protótipo (em avaliação), se mostrando bastante eficientes. Entretanto, existem alguns desafios e/ou vulnerabilidades que ainda são objeto de estudos, como por exemplo, a segurança e o incentivo à cooperação entre os nós. Alguns desses sistemas desenvolvidos inicialmente para a distribuição de vídeo ao vivo estão sendo adaptados à distribuição de vídeo sob demanda, como no trabalho de Li *et al* [51].

Distribuição de vídeo sob demanda (VoD)

Um serviço de vídeo sob demanda (VoD - *Video on Demand*), idealmente, deve permitir que clientes remotamente localizados possam assistir, no momento desejado, a qualquer um dos objetos multimídia armazenados em um servidor central ou em diferentes servidores. Para tanto, os objetos são distribuídos para os clientes através de redes de comunicação. Este tipo de serviço tem uma grande diversidade de aplicações como, por exemplo, ensino a distância, filmes sob demanda e bibliotecas virtuais. Usualmente, a caracterização deste serviço é feita com base nos aspectos mencionados a seguir [57].

1. Duração da sessão: um sistema de VoD deve ser capaz de atender igualmente sessões de longa e curta duração. Por exemplo, um filme típico tem duração de 90-120 min., enquanto que uma visita a uma biblioteca virtual tem uma duração imprevisível, podendo ser bem mais longa ou bem mais curta;

2. Requisitos de banda: os requisitos para as taxas de transmissão dos objetos estão usualmente entre 1.5 Mbps (MPEG-1) e 3-10 Mbps (MPEG-2) por fluxo iniciado. Assim, a banda total para transmissão dos vários objetos simultaneamente tende a ser significativa;

3. Interatividade: idealmente, o serviço de VoD deveria ter condições de emular todas as ações disponíveis de um aparelho de DVD como, por exemplo: *Play* ,

Stop, *Fast Forward*, *Rewind* e *Pause/Resume*. Entretanto, as propostas atuais têm mostrado que o custo/benefício, em termos da banda a ser utilizada, nem sempre se apresenta perfeitamente satisfatório para o emprego comercial. Daí, a solução adotada é, em sua maioria, a emulação parcial dessas ações como, salto para frente e para trás, *Pause/Resume* e *Stop*;

4. Qualidade de serviço (QoS): a avaliação da qualidade de serviço reside principalmente nos aspectos comentados a seguir: primeiro, a latência do serviço, ou seja, a demora até o início da recepção do objeto; segundo, a taxa de desistência, ou seja, a taxa com que os usuários abandonam o sistema por impaciência devido à demora para o início da recepção do objeto; terceiro, a qualidade percebida pelo cliente durante a visualização do objeto; e, por fim, a abrangência da emulação de interatividade.

Tradicionalmente, para prover o serviço VoD é empregado um modelo Cliente-Servidor, onde cada cliente cria conexões com o servidor sobre um canal *unicast*. Com o aumento da popularidade desse serviço, o servidor tornou-se um "gargalo", uma vez que seus recursos, como banda e capacidade de processamento, são finitos. Sendo assim, o problema de como prover um serviço de VoD com qualidade e escalabilidade é bastante estudado.

Várias propostas têm sido exploradas, principalmente na tentativa de resolver o problema da escalabilidade. Outra opção é o protocolo *IP multicast* sobre a camada de rede. Essa abordagem mantém um servidor central que realiza a transmissão do fluxo de vídeo/áudio para os usuários, porém por meio de canais *multicast*. A eficiência e a redução nos custos pelo uso de banda são conseguidos porque um mesmo canal pode ser compartilhado por diversos usuários. Entretanto, como esta tecnologia não abrangem toda a rede, o que inviabiliza a sua utilização. Porém uma série de propostas baseadas no conceito *multicast*, mas sobre a camada de aplicação, como a difusão (*broadcast*) periódica [58], o *Patching* [11, 59] e o *Stream Merging* [12], foram desenvolvidas com o objetivo de diminuir os requisitos de banda do servidor. Conjuntamente a esses protocolos também é possível utilizar-se uma

gerência de buffer local [13, 14]. Outras propostas, como *proxy caching* ou *content distribution networks (CDN)*, também têm sido estudadas para resolver essa questão. Nessas propostas, o conteúdo (vídeo) é disponibilizado em servidores *proxies* ou servidores CDN próximos aos clientes. Assim, como o conteúdo fica replicado em vários servidores, é selecionado aquele que esteja menos congestionado para servir o usuário. Entretanto, esta estratégia ainda apresenta problemas de escalabilidade.

Uma terceira proposta é a utilização da arquitetura P2P na distribuição de vídeo. Serão analisados a seguir os trabalhos mais relevantes baseados nesta arquitetura.

Os trabalhos de Guo *et al* [17], Do *et al* [60] utilizam a estrutura em árvore, não prevêem nenhum tipo de ação interativa. A proposta P2Cast [17] é inspirada no esquema de *Pacthing* [11, 59] e usa uma estrutura em árvore para distribuir um único fluxo de vídeo. Quando ocorre a chegada de um novo nó, ele é incluído em um grupo (ramo da árvore) escolhido de acordo com um *threshold* T, que é a diferença máxima tolerável entre o ponto que o usuário deseja (T_u) e o ponto em que está o fluxo (T_f). A diferença entre $T_f - T_u$ é o *patch*, que será recuperado por uma conexão *unicast*. A outra proposta P2VoD [60] introduz um novo conceito de cache. Cada cliente terá uma cache que armazena os dados mais recentemente recebidos, aqueles usuários que possuírem os mesmo dados na cache, mesmo que chegarem ao sistema em intervalos de tempo diferentes, formarão uma geração. Quando um nó pertencente a uma geração parte do sistema, os demais nós desta geração continuam provendo o fluxo do vídeo, sem nenhum atraso, para os descendentes do nó que partiu.

Outros trabalhos utilizam a estrutura em malha, uma vez que são baseados no protocolo *BitTorrent*. Nenhum deles permite realizar ações interativas. A proposta BASS [41] é um sistema híbrido Cliente-Servidor/*BitTorrent*. Naquela proposta, o protocolo *BitTorrent* é utilizado para a recuperação de partes futuras do vídeo, como forma de economizar a banda do servidor. Para as partes do vídeo que estão próximas de serem reproduzidas, a recuperação é realizada diretamente do servidor através de fluxos *unicast*. Não foi feita nenhuma modificação no protocolo *BitTorrent* para prover distribuição do vídeo. Uma segunda proposta, denominada *Toast* [42],

também utiliza um sistema híbrido Cliente-Servidor/*BitTorrent*. Foram realizadas modificações na política de seleção de blocos do protocolo *BitTorrent*. Três novas formas de seleção de blocos foram acrescentadas além da política do mais raro, a saber: (1) sequencial, (2) beta - utiliza uma distribuição beta para selecionar os blocos mais próximos do ponto de reprodução primeiro; e (3) híbrida - onde os blocos estão divididos em 2 conjuntos, os mais próximos do ponto de reprodução são recuperados sequencialmente e os mais distantes usando a distribuição beta. O servidor VoD é utilizado como *backup*: se um bloco que está muito próximo de ser reproduzido ainda não foi recuperado, o cliente realiza um pedido *unicast* para o servidor.

Os trabalhos [24, 2, 23, 3] propõem modificações na política de seleção de blocos e/ou de seleção de vizinhos do protocolo *BitTorrent*.

No trabalho de Carlsson e Eager [24] são apresentas duas novas propostas para seleção de bloco. A primeira denominada *Portion(p)* seleciona com probabilidade p um novo bloco utilizando o algoritmo sequencial e com probabilidade $(1-p)$ seleciona o bloco mais raro. Na segunda estratégia denominada *Zipf(Θ)*, a probabilidade da escolha de um bloco é proporcional a $\frac{1}{(k+1-k_0)^{\Theta}}$, onde k é o índice do bloco a ser selecionado e k_0 e o índice do primeiro bloco que falta. Os trabalhos *BiToS* [2], de *Zhou-Chiu-Lui* [23] e de *Shah-Pâris* [3], que também se baseiam no *BitTorrent*, e implementam algumas modificações, serão discutidos com mais detalhes no Capítulo 3 pois, serão usados na comparação com o algoritmo proposto neste trabalho.

Todos os trabalhos citados até o momento não prevêem a realização de ações interativas. Entretanto, outros trabalhos abordam essa característica. Muitos dos trabalhos se propuseram a utilizar novos algoritmos, seja para seleção de blocos ou seleção de vizinhos, ou seja, não utilizam o protocolo *BitTorrent* como base. A seguir serão citadas as propostas mais relevantes da literatura para a distribuição de vídeo sob demanda com interatividade.

A proposta de Guo *et al* [61] utiliza uma estrutura em árvore para a distribuição do fluxo de vídeo. A raiz da árvore, além de enviar o fluxo de vídeo, também é o

gerenciador de toda a árvore. Todos os nós têm função cliente e servidor implementadas. Em cada nó existe um *buffer* para armazenar os dados recebidos pelo fluxo. Assim que um nó cliente tenha recebido todo o vídeo, ele poderá se desligar da árvore e formar uma nova árvore independente. Quando um determinado nó deseja realizar um salto, seja para frente ou para trás, é executado o algoritmo de chegada de novo nó, ou seja, primeiro ele é desligado da árvore, como se tivesse partido e em seguida é reintegrado com um novo nó da árvore em sua nova posição.

Já a proposta *PONDER* [62] é utilizada uma estrutura em malha, em conjunto com um servidor de VoD, que realiza a mesma função que na arquitetura Cliente/Servidor. Este servidor é utilizado como *backup*, assim como no trabalho de Choe *et al* [42]. Para cada bloco é calculado um *deadline*, ou seja, um tempo máximo para sua chegada. Os blocos que estão próximos ao *deadline* são buscados no servidor e os demais são recuperados usando a rede P2P. Na ocorrência de uma ação interativa, o ponto de visualização é atualizado e o *deadline* é recalculado para todos os blocos. O algoritmo de seleção de blocos, utilizado na parte P2P, segue o *deadline* de cada bloco; o algoritmo de seleção de vizinhos leva em consideração algumas características do vizinho, como taxa de *upload* e contribuição para a rede.

Os trabalhos de Guo *et al* [61] e PONDER [62] não foram utilizados nas comparações com o algoritmo proposto nesse trabalho, principalmente, por não utilizarem o protocolo *BitTorrent* e também por apresentarem características que fogem ao escopo deste trabalho. No trabalho de Guo *et al* [61] é utilizado uma estrutura em árvore para distribuir o vídeo, porém a proposta desse trabalho utiliza uma estrutura em malha, o que torna a comparação entre as duas inviável. Principalmente, por que o objetivo não é comparar o desempenho da estrutura utilizada pelas propostas, e sim o desempenho global do algoritmo. Na proposta PONDER [62] apesar de utilizar a estrutura em malha, também utiliza um servidor, como na arquitetura cliente/servidor, para servir os usuários independente da rede P2P, portanto a sua utilização nesse estudo torna-se inviável. Haja vista, que o objetivo desse trabalho é utilizar uma arquitetura totalmente P2P, sem o auxilio de nenhuma arquitetura que lembre a cliente/servidor.

Capítulo 3

Trabalhos Relacionados

Neste capítulo é apresentado um resumo de três trabalhos encontrados na literatura sobre distribuição de vídeo sob demanda que serviram de base para esta proposta, a ser apresentada no Capítulo 4. Todos estes trabalhos têm em comum a finalidade de distribuição de Vídeo sob Demanda sem interatividade baseado no protocolo *BitTorrent*, com algumas modificações nos algoritmos de seleção de blocos e de vizinhos originais do *BitTorrent*.

A escolha dessas três propostas é pelo fato de apresentarem um algoritmo simples baseado no *BitTorrent* e o bom desempenho para acesso sequencial. Porém, como foram desenvolvidas para vídeo sob demanda sem interatividade foram feitas algumas adaptações nas propostas de forma a adequá-las para a distribuição de vídeo sob demanda com interatividade. O objetivo dessas adaptações foi tornar a comparação entre elas e a nova proposta justa. No decorrer da apresentação de cada uma das propostas serão explicadas as adaptações.

3.1 Proposta *BitTorrent Streaming (BiToS)*

A proposta *BitTorrent Streaming (BiToS)* [2] difere do protocolo *BitTorrent* por apresentar algumas modificações no algoritmo de seleção de blocos. Pelas caracterís-

ticas do algoritmo de seleção de blocos do *BitTorrent*, aplicações sensíveis ao tempo, como a distribuição de vídeo sob demanda, apresentam um desempenho ruim uma vez que não existe a preocupação com o tempo da recuperação ou a ordem de chegada dos blocos, mas sim com a dispersão dos blocos no sistema a fim de evitar a extinção dos mesmos. Consequentemente, foi proposta uma modificação deste algoritmo para que ele seja mais adequado à distribuição de vídeo sob demanda. O algoritmo de seleção de vizinhos, segundo a proposta, não necessita de nenhuma modificação.

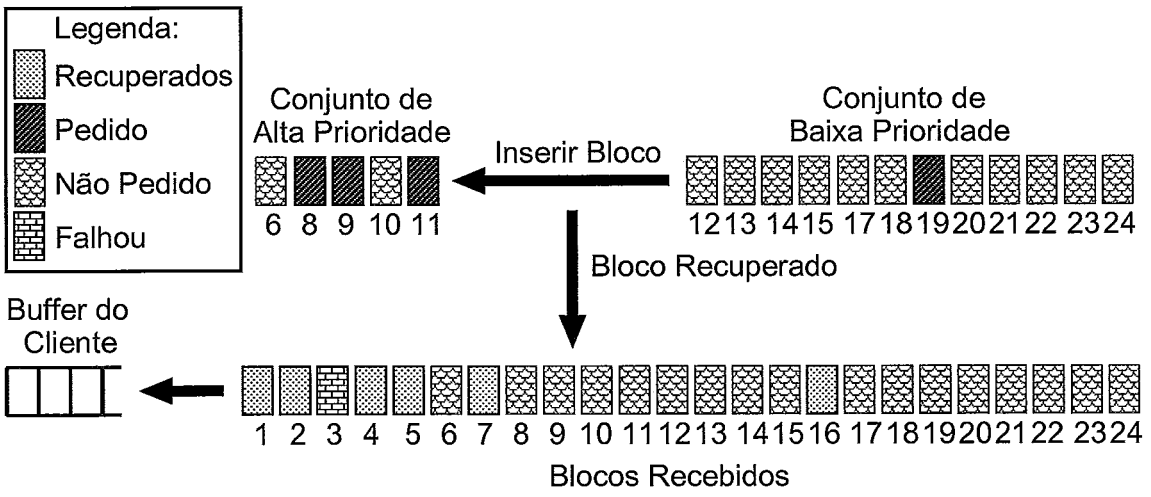

Figura 3.1: Esquema da proposta *BiToS* [2].

O *BiToS* [2] divide os blocos em dois conjuntos principais destacados na Figura 3.1. A seguir, é descrito cada um desses conjuntos:

- Conjunto de Alta Prioridade: contém todos os blocos que ainda não foram recuperados cuja distância em relação ao ponto de reprodução é menor ou igual a t, onde t é o tamanho do conjunto de alta prioridade. Desta forma, os blocos desse conjunto têm uma prioridade maior para serem recuperados do que os blocos do conjunto de baixa prioridade. Este conjunto tem um tamanho fixo de t blocos. Os blocos podem estar no estado **não pedido** e **pedido**;

- Conjunto de Baixa Prioridade: contém todos os blocos que ainda não foram recuperados e cuja distância em relação ao ponto de reprodução é maior que t.

Esses blocos serão necessários em um futuro próximo e podem aguardar mais tempo pela sua recuperação. Os blocos podem estar nos estados **não pedido** e/ou **pedido**.

No processo de seleção, é utilizada uma probabilidade p para escolher de qual conjunto o próximo bloco será recuperado. Com probabilidade p, é escolhido um bloco do conjunto de alta prioridade, enquanto que com probabilidade $(1-p)$, é escolhido um bloco do conjunto de baixa prioridade. Essa probabilidade p representa um equilíbrio entre os blocos imediatamente necessários e a aquisição de blocos futuros. A probabilidade pode ser ajustada dinamicamente para se adaptar às diferentes condições do sistema.

A política utilizada para escolher quais blocos recuperar do conjunto de alta prioridade ou do conjunto de baixa prioridade é a do bloco mais raro (*Rarest First*). A única modificação nesse algoritmo é que se dois ou mais blocos têm a mesma raridade, ou seja, o mesmo número de cópias, o bloco mais próximo a ser reproduzido será o escolhido.

Após a recuperação de um bloco, ele é removido do seu conjunto corrente e acrescentado ao conjunto dos blocos recebidos (função Bloco Recuperado na Figura 3.1). Se este bloco pertencer ao conjunto de alta prioridade, ele será substituído pelo primeiro bloco do conjunto de baixa prioridade (função Inserir Bloco na Figura 3.1).

Uma importante função do sistema é a função *deadline* dos blocos. Para cada um dos blocos que estão nos estados **pedido** e/ou **não pedido**, essa função determina se o bloco será recuperado a tempo de poder ser reproduzido ou não. Caso o bloco não possa ser recuperado dentro do tempo determinado, ele não será pedido; mas se já foi pedido, passará para o estado **falhou**, como por exemplo o bloco três, apresentado na Figura 3.1. Para tomar esta decisão, é estimado o tempo necessário para a recuperação do bloco T_D e o tempo até a sua reprodução T_R. Se $T_R > T_D$, então o bloco chegará a tempo de ser reproduzido. Entretanto, se $T_R < T_D$, o bloco não é mais necessário.

A probabilidade p tem grande impacto no desempenho do sistema. Grandes valores de p garantem que os blocos, que serão reproduzidos em breve, possam ser recuperados antes que os demais. Entretanto, esta situação pode produzir um efeito indesejado, ou seja, o de um nó pedir um bloco muito popular entre os seus vizinhos. Desta forma, o nó não terá nenhum bloco raro para oferecer aos seus vizinhos e, consequentemente, tornar-se-ia bloqueado para muitos deles, devido ao algoritmo de seleção de vizinhos do protocolo *BitTorrent*. Além disso, blocos raros que estão disponíveis agora podem não estar mais no futuro, visto que os nós que têm esses blocos podem deixar o sistema. Por outro lado, pequenos valores para p farão com que os blocos do conjunto de baixa prioridade sejam recuperados antes que os do conjunto de alta prioridade, acarretando em uma perda de qualidade na visualização do vídeo.

As inovações desta proposta [2] são: a classificação dos blocos, seguindo o critério de tempo para a reprodução do bloco; e a utilização de uma probabilidade para a escolha do conjunto a ser recuperado.

A adaptação que foi implementada nessa proposta é: a atualização do conjunto de alta prioridade sempre que ocorrer um salto, seja para frente ou para trás, e o ponto do salto não pertencer ao conjunto de alta prioridade, ou seja este salto é para fora do conjunto de alta prioridade. Desta forma, toda vez que o usuário realizar um salto o conjunto de alta prioridade é atualizado caso seja para fora dele.

3.2 Proposta de *Zhou-Chiu-Lui*

No trabalho de *Zhou-Chiu-Lui* [23] é apresentado um modelo analítico para a distribuição de vídeo sob demanda sem interatividade. O propósito deste modelo é entender melhor as características e o funcionamento de um sistema de distribuição de vídeo, que utiliza como base o protocolo *BitTorrent*, principalmente no que se refere ao impacto da política de seleção de blocos. Além disso, é proposta uma nova política de seleção de blocos, denominada mista, que tenta unir o melhor de duas

outras políticas: a do mais raro e a sequencial. O trabalho não menciona que política de seleção de vizinhos é utilizada, entretanto, como utiliza o protocolo *BitTorrent*, assume-se que também utilize o algoritmo de seleção de vizinhos baseado na política *tit-for-tat*.

A proposta de *Zhou-Chiu-Lui* [23] é muito semelhante à proposta *BiToS* [2], pois também classifica os blocos de acordo com a sua proximidade para reprodução. Os blocos são classificados em dois conjuntos: alta prioridade, contém os n blocos mais próximos de serem reproduzidos; e baixa prioridade, contém os demais blocos que estão mais distantes da sua reprodução. Para a escolha de qual conjunto deve ser recuperado o próximo bloco, é utilizado um parâmetro p que é a probabilidade com a qual o bloco será recuperado do conjunto de alta prioridade e obviamente com $(1 - p)$ será recuperado do conjunto de baixa prioridade. Porém, diferentemente do *BiToS* [2] que utiliza a política do mais raro para ambos os conjuntos, nessa proposta são utilizadas duas políticas diferentes, uma para cada conjunto. Para o conjunto de alta prioridade é utilizada a política sequencial, enquanto que para o conjunto de baixa prioridade é usada a política do mais raro. Sempre que um bloco for recuperado, este é retirado do seu conjunto correspondente e torna-se parte do conjunto dos blocos recebidos e disponíveis para reprodução. Quando o bloco recuperado faz parte do conjunto de alta prioridade, o primeiro bloco não recuperado do conjunto de baixa prioridade passará a compor o conjunto de alta prioridade, deixando o conjunto de baixa prioridade.

Uma das contribuições desse trabalho foi a demonstração de que quando utilizada somente a política sequencial para a recuperação de todos os blocos o desempenho é melhor do que o da política do mais raro em relação à continuidade da reprodução, entretanto, a política do mais raro possui desempenho bastante superior à sequencial com relação a disseminação rápida de um objeto entre os nós do *swarm*. Foi mostrado que a adoção de uma política mista e a classificação dos blocos em dois conjuntos com prioridades distintas, apresenta um bom compromisso entre a continuidade e a disseminação eficiente do objeto dentro do *swarm*.

A principal adaptação aplicada a essa proposta é a atualização do conjunto de

alta prioridade quando ocorre um salto, como foi feito para a proposta *BiToS* [2].

3.3 Proposta de *Shah-Pâris*

No trabalho de *Shah-Pâris* [3] é apresentada uma nova proposta para a distribuição de vídeo sob demanda sem interatividade, utilizando o protocolo *BitTorrent* como base, como nas demais propostas [2, 23]. Porém, diferentemente das anteriores, que modificavam apenas o algoritmo de seleção de blocos, essa proposta também apresenta modificações no algoritmo de seleção de vizinhos.

Em *Shah-Pâris* [3] é usado o conceito de janela deslizante, ilustrado na Figura 3.2. A janela contém os próximos w blocos a serem reproduzidos pelo usuário. Os blocos que falharem, ou seja, não forem recuperados antes da sua reprodução, não são mais necessários e também não serão mais recuperados. Recuperar blocos que falharam ou que estejam fora da janela não é estrategicamente pertinente, pois consome tempo que poderia ser utilizado para recuperar blocos contidos dentro da janela e que no momento são mais importantes que os demais. Esta restrição na seleção dos blocos notadamente pode diminuir a velocidade de distribuição do vídeo, mas melhora a qualidade de sua distribuição, evitando que ocorram interrupções e/ou descontinuidades por falta de blocos.

Adotar uma política sequencial para a seleção de blocos faria com que os blocos do início da janela recebessem sempre a maior prioridade. Esta política teria a desvantagem de não levar a raridade do bloco em consideração o que prejudicaria a disseminação do arquivo. Desse modo, a política de seleção de blocos dentro da janela é a do bloco mais raro.

A janela irá se movimentar (deslizar) para a direita em duas situações distintas. Estas situações podem ser observadas na Figura 3.2:

1. Chegada do Primeiro Bloco da janela: quando o primeiro bloco da janela é recuperado. Ela é deslocada para a posição seguinte que ainda não foi recu-

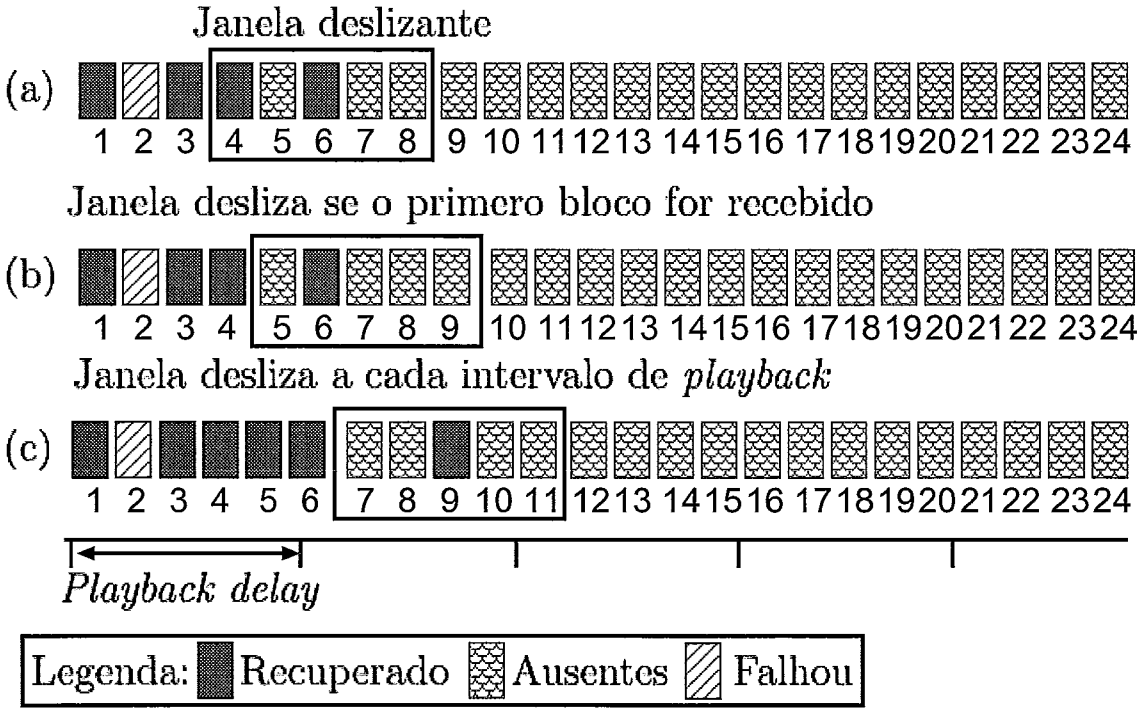

Figura 3.2: Esquema da proposta Shah-Pâris [3].

perada. Parte (b) da Figura 3.2;

2. Ao término do *playback delay*: o *playback delay* é um tempo estimado para que o nó recupere todos os blocos da uma janela. Ocorrendo o seu esgotamento, desloca-se toda a janela para as w posições seguintes à última posição da janela anterior. Parte (c) da Figura 3.2.

O tamanho da janela, segundo [3], denotado por w, deve ser calculado em função do *playback delay* (d), ou seja, tempo para começar a reproduzir o vídeo. Este tempo é estimado para recuperar todos os blocos da janela. A reprodução do vídeo só é iniciada após a recuperação de todos os blocos da janela. O tamanho da janela, medido em número de blocos é dado por $\mathrm{w} = \frac{\mathrm{db}}{\mathrm{c}}$, onde d é o *playback delay*; c é o tamanho do bloco e b é a taxa de reprodução do vídeo.

O algoritmo de seleção de vizinhos no protocolo *BitTorrent* pode ser modelado pela política *tit-for-tat*. Mais precisamente, cada nó seleciona seus vizinhos baseando-se em suas taxas de *download*, sendo a prioridade de escolha dada para os

vizinhos que oferecem a maior taxa, acarretando em um aumento global da velocidade de distribuição dos blocos no sistema.

Ao longo do tempo, entretanto, pode acontecer que um subconjunto de nós que possuam grande disponibilidade de recursos, como banda de *upload* e *download*, consumam boa parte da banda disponível da fonte (*seed*) e de seus vizinhos. O resultado disso é que nós com baixa capacidade de *download* e *upload* podem ter um tempo de *download* demasiadamente longo para a recuperação dos blocos. Muito provavelmente, apenas através do processo de *optimistic unchocke*, que ocorre a cada 30 segundos, é que estes nós menos privilegiados poderão ser servidos. Portanto, um nó menos privilegiado no sistema tem que esperar até que seja eventualmente selecionado por um nó privilegiado ou por outro igual a ele, para só então poder receber os blocos.

Sendo assim, foi proposto em *Shah-Pâris* [3] uma modificação no algoritmo de seleção de vizinhos. A cada janela de w blocos recuperada, o nó selecionará n vizinhos aleatoriamente da lista recebida do *tracker*, para que possa realizar pedidos de blocos a ele. Independente desse processo, durante a recuperação dos blocos de uma janela e entre intervalos de tempo de 10 segundos, os nós se comportam segundo a política *tit-for-tat*, ou seja, empregam o algoritmo de seleção de vizinhos original do *BitTorrent* para a escolha de quais vizinhos serão desbloqueados. Este procedimento é realizado para evitar os nós *free-riding*.

A adaptação realizada nessa proposta foi o deslocamento da janela sempre que ocorrer um salto, seja para frente ou para trás, e o novo ponto de reprodução não pertencer a janela de deslizante atual. Após o salto, a nova janela compreenderá os w blocos a partir do ponto inicial do salto.

Capítulo 4

Protocolo *BitTorrent Interactive Video on Demand* - BIVoD

Neste Capítulo, é apresentada uma nova proposta, denominada *BitTorrent Interactive Video on Demand* - BiVoD, para distribuição de vídeo sob demanda com interatividade, utilizando o protocolo BitTorrent como base. Esse Capítulo está estruturado da seguinte forma: na Seção 4.1 será descrito, brevemente, o modelo de simulação desenvolvido do protocolo *BitTorrent* pois, este é usado como base para o desenvolvimento dos modelos das propostas analisadas. Na Seção 4.2, será apresentada uma descrição completa da proposta BIVoD.

4.1 Modelo do Protocolo *BitTorrent*

Inicialmente, desenvolveu-se um modelo detalhado de simulação do protocolo *BitTorrent* utilizando a ferramenta Tangram-II [26]. O objetivo desse modelo é entender melhor o comportamento do protocolo *BitTorrent*, como também utilizá-lo como base para o desenvolvimento dos modelos da nova proposta BIVoD. Entender o comportamento do *BitTorrent* é o primeiro passo para o desenvolvimento de qualquer tipo de aplicação que o tenha como base, uma vez que auxiliará no entendimento de quais características são úteis ou não e, consequentemente focar no

melhoramento ou na adaptação dos pontos do protocolo que não são favoráveis à nova aplicação.

O modelo do protocolo *BitTorrent* foi desenvolvido com todas as suas características seguindo a implementação do protocolo no cliente principal 4.0.2 [63]. Porém, no algoritmo de seleção de blocos foi realizada uma simplificação: o modo *end game* não foi implementado, pois segundo o trabalho [64], este modo não apresenta significativa melhora no desempenho do protocolo e ainda aumenta a sobrecarga do sistema com o envio de várias mensagens de pedido e de cancelamento de pedido desnecessárias.

No modelo são modeladas separadamente cada uma das entidades que compõem o *swarm*, o *tracker* e os nós *seeds* e *leechers*. Em cada um dos nós, existe um modelo de rede que emula a banda de *upload* disponível, ou seja, acrescenta um atraso para a entrega de cada bloco, que é variável de acordo com o número de pedidos na fila de cada nó. Porém, para a banda de *download* não foi desenvolvido nenhuma forma de emulação como foi para a banda de *upload*, ou seja, no modelo a banda de *download* é ilimitada. Entretanto, segundo o trabalho [65], a banda de *download* é limitada pela banda de *upload*, devido à ação do algoritmo de seleção de vizinhos. Por simplicidade, não foi implementado nenhum tipo de atraso de propagação, acreditando que não apresentasse significativo impacto no desempenho do modelo. Não foi implementado também nenhum tipo de dinâmica das conexões TCP (por exemplo, *timeouts*, etc.), pois tentou-se eliminar variáveis desnecessárias.

4.2 Protocolo *BitTorrent Interactive Video on demand - BIVoD*

As propostas *BiToS* [2], de *Zhou-Chiu-Lui* [23] e de *Shah-Pâris* [3] foram desenvolvidas para a distribuição de vídeo sob demanda sem interatividade e utilizam o protocolo *BitTorrent* como base. Sendo assim, as três propostas apresentam algumas características em comum, como por exemplo, todas classificam os blocos mais

próximos a serem reproduzidos como sendo de alta prioridade para a recuperação e utilizam a política de seleção de blocos mais raro com algumas modificações ou combinada com a política sequencial. Entretanto, da mesma forma que apresentam semelhanças, existem diferenças, uma vez que as propostas *BiToS* e de *Zhou-Chiu-Lui* recuperam os blocos tanto do conjunto de alta prioridade como do de baixa prioridade, enquanto que a proposta de *Shah-Pâris* recupera somente os blocos do conjunto (janela) de alta prioridade. Outra diferença é que nas duas primeiras propostas não são realizadas modificações no algoritmo de seleção de vizinhos, porém na proposta de *Shah-Pâris* este algoritmo foi modificado. As propostas *BiToS* e de *Shah-Pâris* recuperam os blocos do conjunto de alta prioridade seguindo o algoritmo do mais raro, enquanto que a de *Zhou-Chiu-Lui* recupera sequencialmente. A escolha dessas três propostas é devido à utilização do protocolo *BitTorrent* como base para a sua implementação.

A proposta BIVoD foi desenvolvida tendo como base o protocolo *BitTorrent* e utilizando as melhores características de cada uma das propostas citadas no parágrafo anterior. Como diferencial, a nova proposta tem a finalidade de distribuir vídeo sob demanda com interatividade, ou seja, o usuário poderá executar ações como pausa, saltos para frente ou para trás, etc. As modificações implementadas no *BitTorrent* abrangem somente o algoritmo de seleção de blocos, pois é esse algoritmo que no *BitTorrent* prejudica a distribuição de vídeo sob demanda. O principal problema do algoritmo de seleção do bloco mais raro é a recuperação dos blocos fora da ordem em que eles devem ser tocados. A ideia por trás de se utilizar a raridade do bloco é o que torna essa política muito eficiente para a disseminação do objeto. Porém, quando se trata de aplicações que apresentam fortes restrições de tempo, o desempenho e a qualidade são fortemente afetados. Uma aplicação de vídeo sob demanda, utilizando esse algoritmo, pode apresentar muitas interrupções, seguidas de longos períodos de inatividade, ou seja, o retorno de uma interrupção é muito longo e a qualidade acaba sendo degradada. Por outro lado, o algoritmo de seleção de vizinhos, que é baseado na política *tit-for-tat*, se mostrou eficiente mesmo para aplicações de distribuição de vídeo sob demanda sem interatividade [2, 23]. Entretanto,

na proposta de *Shah-Pâris* [3] são realizadas modificações também no algoritmo de seleção de vizinhos, com a justificativa de diminuir o tempo que um nó espera para iniciar suas trocas, principalmente em um ambiente heterogêneo, ou seja, em um ambiente onde exista grande diferença entre as bandas disponíveis dos nós. Na nova proposta não foram implementadas modificações no algoritmo de seleção de vizinhos devido a sua comprovada eficiência, já demonstrada na distribuição de vídeo sob demanda em *BiToS* [2] e *Zhou-Chiu-Lui* [23], como também pelas características do ambiente no qual será desenvolvido: um ambiente homogêneo, onde todos os nós terão a mesma largura de banda disponível. Entretanto, não é descartada a possibilidade de também adotar-se um ambiente heterogêneo e se necessário aplicar modificações no algoritmo de seleção de vizinhos com o objetivo de melhorar o seu desempenho em experimentos futuros.

A principal semelhança entre as três propostas estudadas *BiToS* [2], *Zhou-Chiu-Lui* [23] e *Shah-Pâris* [3], está na preocupação de recuperar os blocos mais próximos ao ponto de reprodução primeiro, ou seja, impor um critério de recuperação que beneficie os blocos imediatamente necessários. Sendo assim, a nova proposta utiliza uma janela, denominada **janela de *playback***, formada pelos m blocos imediatamente seguintes ao ponto da reprodução no tempo t. O seu comportamento será o de uma janela deslizante, ou seja, sempre que o primeiro bloco da **janela de *playback*** for recuperado, essa será deslocada até a posição seguinte que ainda não teve seu bloco recuperado. A recuperação dos blocos da **janela de *playback*** é feita segundo a política do mais raro.

A Figura 4.1 ilustra o comportamento da **janela de *playback*** quando ocorre um salto, seja para frente ou para trás. Seja, P_{salto} a posição do salto, $P_{inicial}$ e P_{final} a primeira e a última posição da **janela de *playback*** respectivamente, supondo que P_{salto} não faça parte da **janela de *playback*** atual, ou seja, $P_{salto} > P_{final}$ ou $P_{salto} < P_{inicial}$, caso (1) e (2) da Figura 4.1 respectivamente, ela será deslocada até P_{salto}, onde esse se tornará $P_{inicial}$ e os próximos m blocos subsequentes farão parte da nova **janela de *playback***. Caso contrário se $P_{inicial} \leq P_{salto} \leq P_{final}$, caso (3) da Figura 4.1, a janela permanecerá na mesma posição, a ocorrência de um

salto interno à janela de *playback* elimina a necessidade de sua atualização, uma vez que a sincronização entre janela de *playback* e ponto de reprodução é mantida. Por outro lado, o deslocamento da **janela de *playback*** é necessário para evitar que ocorra uma falta de sincronização entre o novo ponto de reprodução, após o salto, e a **janela de *playback*** atual.

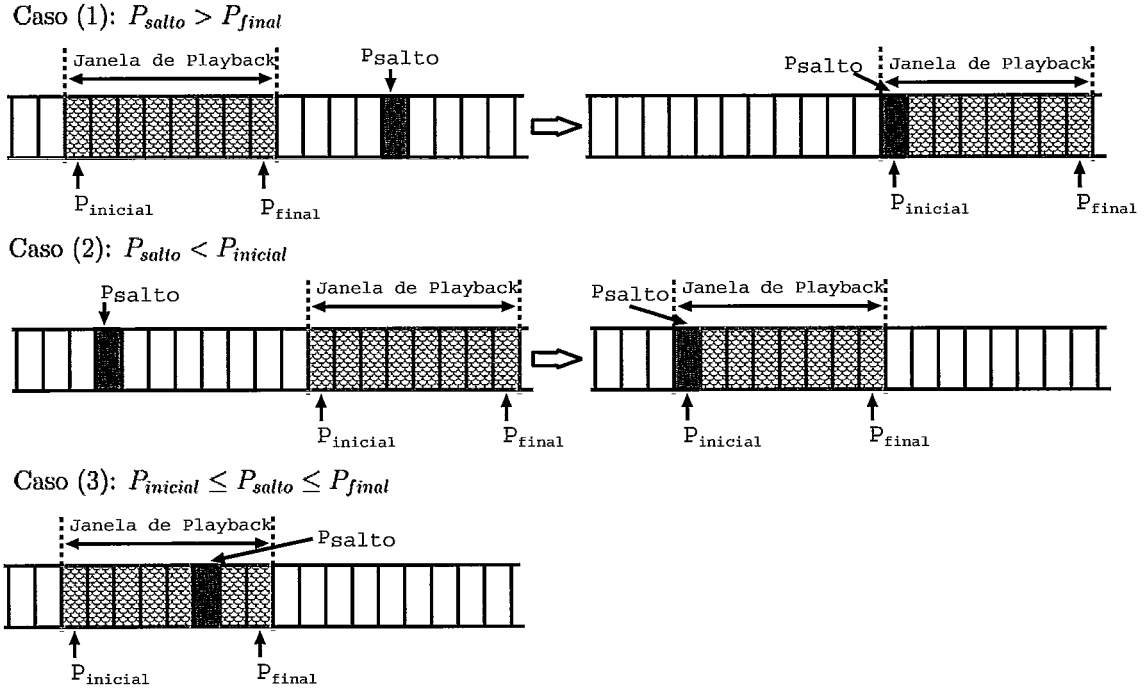

Figura 4.1: Comportamento da **janela de *playback*** quando ocorre um salto, seja para trás ou para frente.

Uma segunda janela, denominada **janela de previsão**, é utilizada para recuperar os n blocos, a partir de uma determinada posição $P_{previsto}$. Essa posição $P_{previsto}$ é determinada pelo modelo de *Vielmond-Leão-de Souza e Silva* [4, 66], utilizado para representar o comportamento do usuário e prever seus futuros saltos. O modelo de comportamento do usuário é baseado em uma cadeia de *markov* oculta e utiliza *logs* de usuários que já passaram pelo sistema na sua parametrização. A saída do modelo é uma sequência de ações que o usuário executa durante uma sessão. A recuperação dos blocos da **janela de previsão** promove o armazenamento de blocos os quais o usuário poderá precisar no futuro. A janela de previsão é atualizada sempre que ocorre um salto do usuário e o algoritmo do mais raro é o utilizado. Na

Subseção 4.2.1 serão apresentados mais detalhes sobre o funcionamento do modelo de *Vielmond-Leão-de Souza e Silva* [4, 66].

A seleção da janela da qual será recuperado um bloco ocorre de forma alternada, ou seja, as janelas de ***playback*** e de **previsão** têm a mesma prioridade. Por exemplo, na Figura 4.2 no tempo t_0, um bloco da **janela de *playback*** é pedido; com o recebimento desse bloco no tempo t_1 é realizado um segundo pedido, agora da **janela de previsão** e se procede desta forma até que todos os blocos do vídeo tenham sido recuperados. Se os blocos da janela a ser realizado o pedido não estiverem disponíveis no momento, serão requisitados os blocos da outra janela, por exemplo, se no tempo t_{100} é a vez de selecionar os blocos da **janela de *playback***, mas os blocos que compõem a **janela de *playback*** não estão disponível na vizinhança para serem recuperados, então são selecionados os blocos da **janela de previsão**. No tempo seguinte, em t_{101} serão selecionados os blocos da **janela de *playback***, se agora este estiverem disponíveis, caso contrário novamente são selecionados os blocos da **janela de previsão**.

Na Figura 4.2 são demonstrados 4 instantes de tempo diferentes de um usuário que executa a nova proposta. No tempo t_0, o usuário acabou de chegar ao *swarm* e iniciou a recuperação para, assim que possível, começar a reproduzir o vídeo. A **janela de *playback***, com tamanho igual a 4 blocos, contém inicialmente os blocos 1, 2, 3 e 4, sendo que o bloco 1 já foi pedido, enquanto que a **janela de previsão**, com tamanho igual a 3 blocos, contém os blocos 17, 18 e 19. O ponto de *play* se encontra no bloco 1, o primeiro a ser reproduzido. No instante de tempo t_1, o bloco 1, já recuperado, é reproduzido e o play é deslocado para o bloco seguinte. A **janela de *playback*** é deslocada para a direita porque o bloco 1 é o primeiro bloco da janela. Em seguida, o bloco 18 da **janela de previsão** é pedido, por ser o mais raro dentro dela. No tempo t_{10}, o play está sobre o bloco 4, esperando para ser recuperado e, em seguida, reproduzido. Novamente a **janela de previsão** é selecionada, sendo realizado o pedido do bloco 21. No tempo t_{12}, o usuário executa um salto para frente, mais especificamente para o bloco 16. Portanto, o *play* é atualizado para este ponto e a **janela de *playback*** também, fazendo agora parte dessa janela os blocos 16,

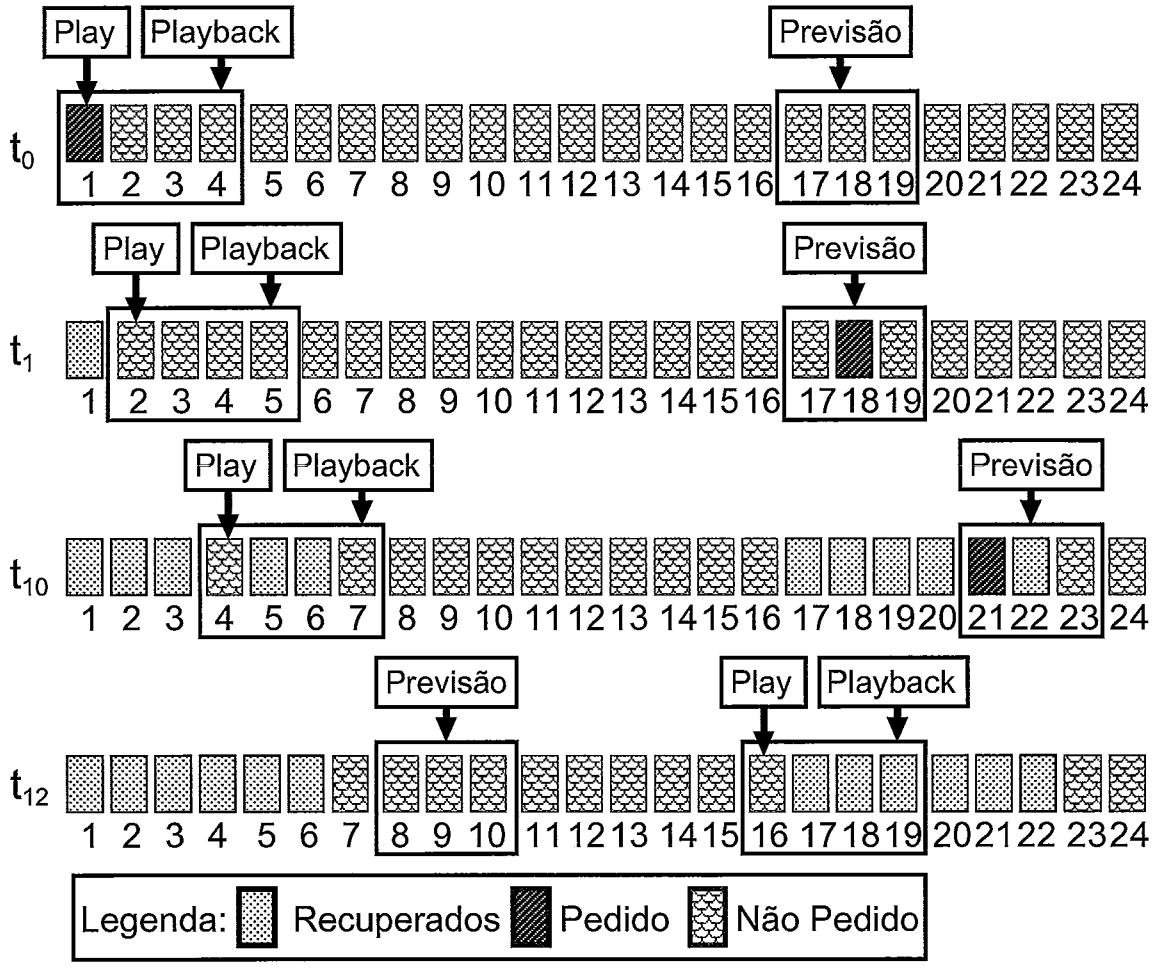

Figura 4.2: Esquema da nova proposta.

17, 18 e 19. Porém, observe que somente o bloco 16 ainda não foi recuperado e, assim que este for recuperado, a janela se deslocará completamente para a direita, até o próximo bloco ausente. Ao mesmo tempo, a **janela de previsão** também é atualizada para os blocos 8, 9 e 10, sendo esses possivelmente os próximos que o usuário poderá precisar, devido a um salto, por exemplo.

A Figura 4.3 demonstra a chegada de uma das janelas, playback ou *previsão*, ao último bloco do objeto no caso em que todos os seus blocos já foram recuperados. Neste cenário podem ocorrer duas ações: Caso (1) na Figura 4.3, o objeto ainda não foi totalmente recuperado, a janela retornará ao início do objeto e deslocará até o primeiro bloco que ainda não foi recuperado, os blocos seguintes a este farão parte da janela; Caso (2) na Figura 4.3, o objeto já foi completamente recuperado,

as janelas retornam ao primeiro bloco do objeto e permanecem paradas.

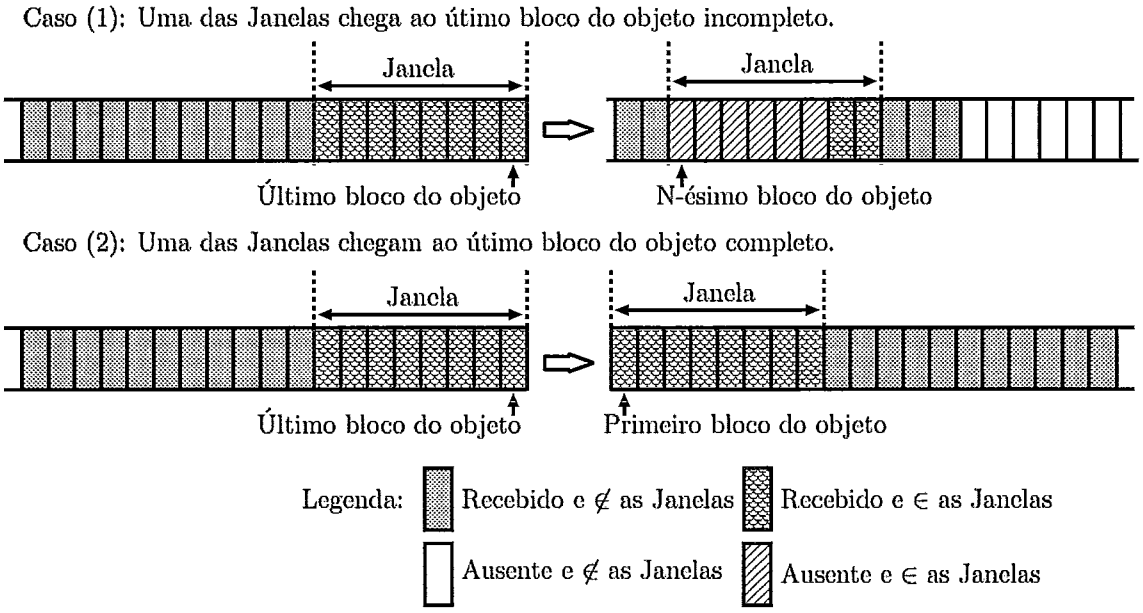

Figura 4.3: Chegada de uma das janela ao último bloco do objeto.

Antevendo possíveis problemas de continuidade na reprodução do vídeo, foi desenvolvida uma variação da proposta BIVoD, denominada BIVoD-*Buffer*. Nesta variação é acrescentado um *buffer* para o armazenamento de k blocos no cliente. A finalidade do *buffer* é de evitar que ocorra um excessivo número de interrupções durante a reprodução, causadas pela ausência de blocos consecutivos. De forma que a reprodução somente será iniciada quando o *buffer* estiver completo. Na ocorrência de uma interrupção por falta de bloco, a reprodução é paralisada e retomada somente após o *buffer* estar novamente completo. A única alteração na proposta BIVoD é a adoção desse *buffer*.

4.2.1 Modelo de *Vielmond-Leão-de Souza e Silva*

O modelo de *Vielmond-Leão-de Souza e Silva* [4, 66] é um modelo de Markov Oculto (*Hidden Markov Model* - HMM) hierárquico para emular o comportamento de usuários acessando um servidor de ensino à distância. Este modelo é baseado em uma aplicação real de ensino à distância, onde alunos do curso de graduação de Tecnologia da Computação do CEDERJ (Centro de Educação Superior à Distância

do Rio de Janeiro) [25] assistem a vídeos-aula previamente gravadas e sincronizadas com transparências, por meio do servidor RIO (*Randomized I/O Multimedia Storage Server*) [67].

Este modelo foi desenvolvido para ser um modelo genérico, ou seja, podendo se adequar a outros sistemas multimídia, não ficando preso às características da aplicação real utilizada inicialmente como base. Em relação a outros modelos da literatura que apresentam esta mesma finalidade, este modelo tem duas vantagens principais: número reduzido de estados, o que acarreta em uma diminuição da complexidade; e uma estrutura que não permite que sequências de ações que notadamente não ocorrem na aplicação real sejam geradas. Um exemplo de sequência de ações inválida é a ocorrência de duas pausas sem que ocorra um *play* entre elas. O trabalho de *Vielmond-Leão-de Souza e Silva* [66] mostra a acurácia desse modelo em comparação a outros da literatura.

A estrutura hierárquica possui propriedades interessantes: a complexidade da fase de treinamento é menor que o HMM convencional, as dependências de curto prazo são capturadas pela cadeia da hierarquia inferior e a dinâmica de longo prazo é governada pela cadeia de *Markov* oculta (a hierarquia superior). A cadeia de *Markov* oculta governa a dinâmica de uma sessão de usuário e os estados ocultos capturam a dependência das ações do usuário dentro do contexto de uma transparência. Sendo assim, dentro de um estado oculto se tem a dinâmica das ações do usuário.

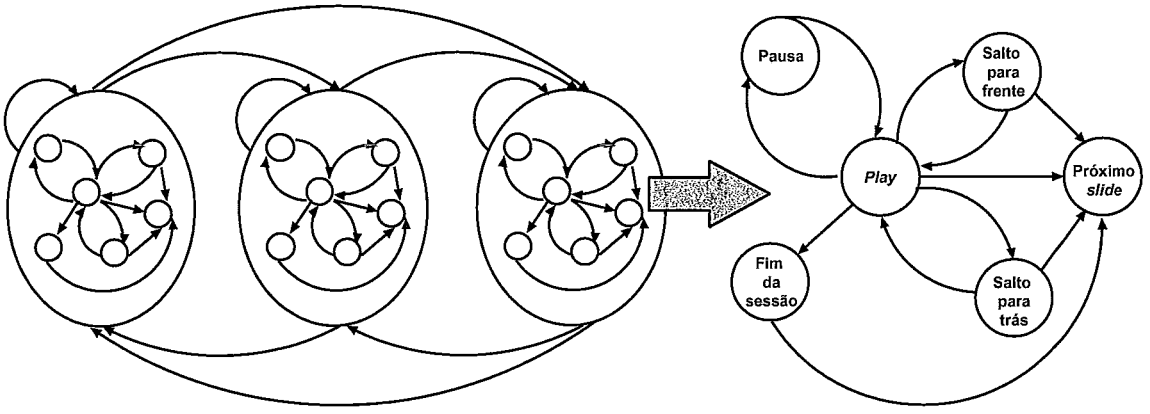

Figura 4.4: Modelo HMM hierárquico [4].

A Figura 4.4 ilustra o modelo HMM hierárquico desenvolvido por *Vielmond-Leão-de Souza e Silva* [4, 66], e apresenta os símbolos que são emitidos em cada um dos estados ocultos. Cada um destes símbolos representa um estado da cadeia de Markov discreta que governa os estados ocultos. Os símbolos utilizados no modelo são: *play*, pausa, salto para frente, salto para trás, próximo *slide* e saída da sessão. Este conjunto foi escolhido baseado na caracterização do comportamento dos usuários do CEDERJ. A cadeia de *Markov* discreta, que governa o estado oculto, sempre é iniciada no estado *play*. O símbolo próximo *slide* causa uma transição entre estados ocultos, já que representa o fim de um *slide*. A partir dos estados salto para frente e salto para trás é possível voltar para o estado *play*, caso o salto não gere uma mudança de *slide*, ou transacionar para o estado próximo *slide*, caso contrário.

Um ponto a ressaltar é sobre a quantidade de estados ocultos que podem ser utilizados, pois utilizar um número elevado de estados representa um ganho em precisão que nem sempre compensa a complexidade do modelo. Este valor dependerá de cada tipo de aplicação, do modelo e da carga utilizada para parametrizá-lo. Um exemplo é a proposta apresentada neste trabalho que apresentou resultados muito satisfatórios para 4 estados. Quando foram utilizados 20 estados, não foram obtidos ganhos significativos se comparados ao resultado com 4 estados. Portanto, o tempo gasto para treinar o modelo de 20 estados não justifica o seu uso.

O resultado da etapa de treinamento é uma sequência de ações interativas que podem ser realizadas pelo usuário, sendo essa sequência de ações formada pelos símbolos vistos anteriormente. Porém, não é especificada a posição no vídeo associada a cada uma dessas ações. Portanto, é necessário analisar características dos *logs*, tais como: a distribuição de probabilidade do tamanho dos saltos, do tempo em *play* e do tempo em pausa, para inserir esses dados no comportamento gerado pelo modelo. Para obter as distribuições de probabilidade que caracterizam as métricas desejadas, primeiramente são calculados os parâmetros para diversas famílias de distribuições. Depois disso, é usado um método para escolher a distribuição mais adequada. Um exemplo de *log* de ações gerado pelo modelo é mostrado na Figura 4.5. Na Figura 4.5

pode-se observar as ações e os respectivos tempos de ocorrência obtidos a partir das amostras das distribuições de probabilidade.

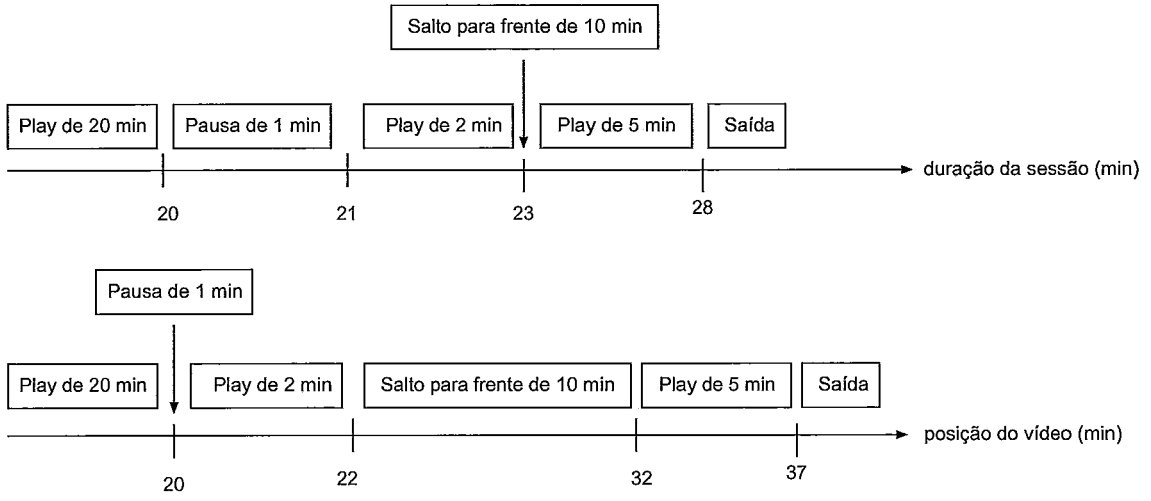

Figura 4.5: Exemplo de uma sessão de usuário gerada pelo modelo após a obtenção das medidas de tempo através das distribuições de probabilidade [4].

4.2.2 Caracterização do nível de interatividade do usuário

Um ponto importante no funcionamento das propostas BIVoD e BIVoD-*Buffer* é a integração com o modelo de comportamento do usuário (*Vielmond-Leão-de Souza e Silva* [4, 66]), ou seja, como o modelo de previsão é utilizado por essas duas novas propostas.

Por tratar-se de distribuição de vídeo sob demanda com interatividade, o usuário pode realizar ações interativas como, saltos para frente, saltos para trás, pausas, etc. Cada usuário pode apresentar um nível de interatividade diferente, isto é, um usuário pode realizar poucas ações enquanto que outros podem realizar mais ou menos ações interativas do que ele. Portanto, utilizar *logs* de usuários muito heterogêneos para a parametrização da cadeia de *markov* oculta (HMM) utilizada pelo modelo de comportamento do usuário poderá afetar negativamente na sua acurácia. Então, com o objetivo de tornar o conjunto de *logs* para a parametrização mais homogêneo foi realizada uma classificações dos *logs* dos usuários de acordo com o

seu nível de interatividade. Para realizar esta classificação foram definidos 3 nível de interatividade, de acordo com o número de ações realizadas pelos usuários que são: alta interatividade, número de ações interativas entre 16 e 40; média interatividade, número de ações interativas entre 6 e 15; e baixa interatividade, número de ações interativas entre 0 e 5.

Por conseguinte, tendo realizado a classificação dos *logs* dos usuários, um modelo de comportamento de usuário para cada nível de interatividade será treinado, ou seja, teremos um modelo treinado para os usuários de alta, um segundo para os de média e um terceiro para os de baixa interatividade. Cada um desses modelos irá simular uma sequência de ações interativas, como por exemplo a da Figura 4.5, com as características dos *logs* utilizados em sua parametrização, ou seja, o modelo parametrizado com *logs* de alta interatividade irá gerar uma sequência de ações correspondendo ao comportamento de um usuário com alta interatividade. Cada uma dessas sequência geradas pelos modelos será utilizada pelas propostas BIVoD e BIVoD-*Buffer* como sendo a previsão das ações futuras do usuário do nível de interatividade correspondente. Todo este processo é realizado *off-line*, ou seja, em um determinado momento o sistema refaz a parametrização dos três modelos, sempre acrescentando novos *logs* de usuários que já passaram pelo sistema, de acordo com a sua classificação, e simula nova sequência de ações interativa para ser utilizada pelos próximos usuários a chegarem ao sistema. Inicialmente, optou-se por realizar este procedimento *off-line* com o objetivo de tornar o sistema mais simples e para manter um custo computacional baixo, porém futuramente este procedimento poderá ser realizado *on-line*.

Devido à utilização de diferentes modelos para representar o comportamento do usuário é preciso determinar o nível de interatividade de um usuário que acaba de chegar ao sistema para a escolha do modelo mais adequado. As únicas informações que se tem deste novo usuário são: qual o vídeo que ele deseja assistir e após o início da reprodução as ações realizadas por ele, que são armazenadas no seu arquivo de *log*. A classificação do usuário é importante para a escolha do modelo de previsão correspondente. Para definir o nível de interatividade do novo usuário

é utilizada uma amostra do *log* de ações gerado por ele. Essa amostra tem um tamanho determinado X e ela é utilizada como entrada em cada um dos três modelos de comportamento do usuário (alta, média e baixa interatividade) para o cálculo do logaritmo da máxima verossimilhança, $logP(X|\lambda)$ (*maximum likelihood*), ou seja, a probabilidade de que essa amostra de ações tenha sido gerada por um desses modelos. Para cada um dos modelos é calculada esta probabilidade e aquele que apresentar o maior valor será o que melhor caracteriza o nível de interatividade daquele usuário, portanto esse modelo será o escolhido para simular a sequência de ações futuras desse usuário.

Outra questão que surge é em relação ao *trade-off* que existe entre o tamanho da amostra X e a acurácia da classificação, ou seja, uma amostra grande aumenta as chances de uma classificação correta, enquanto que uma amostra pequena diminui as chances de uma classificação correta. Portanto, para avaliar este *trade-off* foi realizado um experimento que consiste na variação do tamanho da amostra (X) versus a fração de usuários classificados corretamente. Para realizar esse experimento foram utilizados *logs* já classificados, de acordo com o seu nível de interatividade, e que ainda não tinham sido utilizados para parametrizar os modelos. Portanto, para cada um desses conjuntos de *logs* já classificados (conjunto de alta, média e baixa) foi emulada a classificação deles seguindo o procedimento de classificação descrito acima, sendo que para cada um foi variado o tamanho da amostra (X) entre 10 e 1000 segundos com o objetivo de verificar a fração de usuários classificados corretamente. Como resultado tem-se a fração de usuários de alta interatividade, Figura 4.6 (a), classificados como sendo de alta, média ou baixa; usuários de média interatividade, Figura 4.6 (b), classificados como sendo de média, alta ou baixa interatividade; e usuários de baixa interatividade, Figura 4.6 (c), como sendo de baixa, alta ou média interatividade.

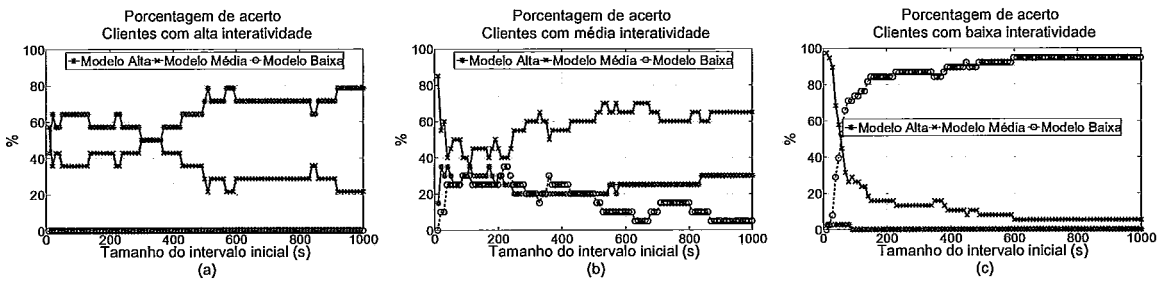

Figura 4.6: Porcentagem de acerto na caracterização do nível de interatividade do usuário.

O resultado da classificação dos usuários (*logs*) de alta interatividade, Figura 4.6(a), apresentou uma fração de acerto de 80% para $X > 500$ segundos, para $X < 500$ a fração máxima de acerto foi de 63%. O fato de alguns usuários (*logs*) terem sido classificados como sendo de média interatividade, quando o tamanho da amostra era menor que 500 segundos, é devido ao baixo número de ações realizadas no início do vídeo pelo usuário. Particularmente, para esses usuários as ações se concentram do meio para o fim da sessão, o que explica o percentual de 80% de acerto para tamanhos da amostra (X) maiores que 500 segundos.

Para os usuários (*logs*) de média interatividade, Figura 4.6(b), a fração de acerto chegou em média a 60%. Porém, apresentou no intervalo $40 < X < 250$ uma elevada fração de usuários classificados erroneamente, isto pode ter ocorrido por dois fatores: primeiro, elevado número de ações nesse intervalo de tempo, o que explica o modelo de alta ser mais compatível com o comportamento desses usuários; e segundo, podem ter ocorrido poucas ações dentro desse intervalo de tempo, levando o modelo de baixa interatividade a caracterizar melhor o comportamento desses usuários.

Já para os usuários (*logs*) de baixa interatividade, Figura 4.6(c), a fração de acerto ficou acima de 80% para $X > 140$ segundos. Para $X < 140$ tem-se grande proporção de usuários classificados erroneamente, principalmente como sendo de média interatividade. A classificação dos usuários (*logs*) como sendo de média interatividade para $X < 50$, pode ser explicada pelo elevado número de ações realizadas pelo usuário em um curto intervalo de tempo, o que pode fazer o modelo de média interatividade caracterizar melhor o comportamento desses usuários.

Esse experimento mostrou que o tamanho da amostra (X) escolhida depende do nível de interatividade, haja vista que, por exemplo, para um tamanho de amostra $X = 140$ para usuários de baixa interatividade, a classificação correta chega a alcançar uma fração igual a 80%, enquanto que para os usuários de média interatividade, para este mesmo tamanho de amostra, as classificações corretas alcançam 40%. Portanto, a escolha de um tamanho para a amostra entre $140 \leq X \leq 250$ segundos mostra-se relativamente eficiente para todos os níveis de interatividade.

Pode-se notar que a maioria dos usuários de alta e baixa interatividade classificados erroneamente foram classificados como sendo de média interatividade. Baseados nesta constatação foram executados experimentos com as propostas BIVoD, BIVoD-*Buffer* e de *Shah-Pâris*, cujos resultados são apresentados na Seção 5.4.5. Os objetivos dos experimentos são: avaliar o impacto dessa classificação errada dos usuários para as propostas que utilizam o modelo de previsão (BIVoD e BIVoD-*Buffer*); e comparar o desempenho entre as propostas que utilizam o modelo de previsão (BIVoD e BIVoD-*Buffer*), porém utilizando uma classificação incorreto do nível de interatividade dos usuários, com a proposta que não utiliza o modelo de previsão (*Shah-Pâris*).

Capítulo 5

Resultados

Neste capítulo são apresentados os resultados dos experimentos realizados com as propostas da literatura descritas no Capítulo 3 e as duas novas propostas descritas no Capítulo 4. Na Seção 5.1 são elencados os principais objetivos dos experimentos realizados. A Seção 5.2 discute as métricas utilizadas para a análise competitiva entre as propostas. Na Seção 5.3 são descritas as cargas reais e sintéticas, obtidas a partir do servidor multimídia RIO, que foram utilizadas nos experimentos e também no treinamento do modelo de comportamento do cliente. Por fim, na Seção 5.4 são apresentados os resultados dos experimentos realizados.

5.1 Objetivos

Para os diferentes cenários de interatividade e considerando o emprego das novas propostas, estes experimentos têm os objetivos principais enumerados a seguir:

1. Comparação dos resultados do modelo de simulação do protocolo *BitTorrent* com experimentos realizados com o protocolo *BitTorrent* em ambiente real. O modelo do protocolo *BitTorrent* serviu de base para os modelos dos algoritmos analisados neste trabalho. Portanto julga-se importante avaliar se as principais características do protocolo *BitTorrent* estão representadas no modelo

proposto e também avaliar suas limitações.

2. Avaliação do desempenho do protocolo *BitTorrent* quando empregado para a distribuição de vídeo sob demanda. Essa avaliação é importante para entender as deficiências que esse protocolo tem para a distribuição desse tipo de mídia. Esta avaliação permitirá a elaboração de uma proposta baseada no protocolo *BitTorrent*, visando a distribuição de vídeo sob demanda com eficiência e qualidade.

3. Determinar os valores ideais para os parâmetros das propostas. Será avaliado o desempenho das propostas considerando um vasto conjunto de parâmetros de forma a obter os valores que apresentem o melhor desempenho.

4. Realização de uma análise competitiva entre as novas propostas BIVoD e BIVoD-*Buffer* e as seguintes propostas da literatura: *BiToS*, de *Zhou-Chiu-Lui* e de *Shah-Pâris*. O propósito principal é identificar o algoritmo mais eficiente e, também, quantificar a diferença entre o desempenho delas.

5.2 Métricas

Para a realização das análises competitivas foram utilizadas as métricas: número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*.

- **Número médio de interrupções**: é definido como o número de blocos ausentes quando da tentativa de reproduzí-los. A Figura 5.1 ilustra a forma como esta métrica é contabilizada. A ausência de um bloco acarreta em uma interrupção e por seguinte a espera por ele. A principal característica dessa métrica é demonstrar a continuidade da reprodução. A sua média é estimada da seguinte forma: $I = \frac{\sum_{i=1}^{N} I_i}{N}$, onde I_i é o número de interrupções que ocorreram no usuário i e N é o total de usuários do *swarm*;

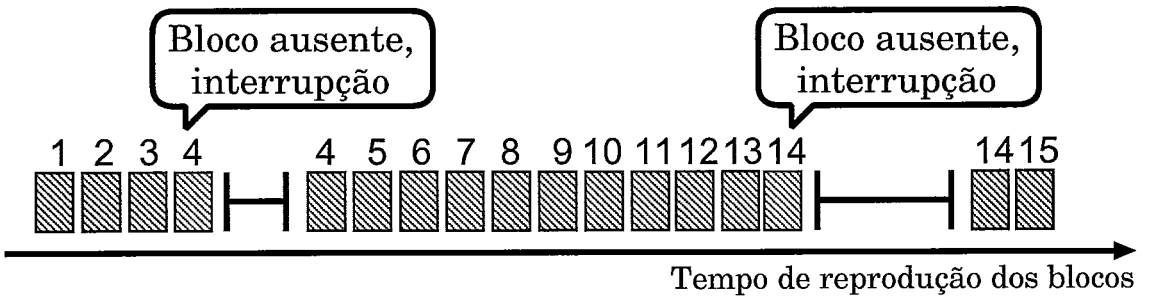

Figura 5.1: Procedimento para a contabilização da métrica número médio de interrupções.

- **Tempo médio de retorno**: tempo necessário para a retomada da reprodução após a ocorrência de uma interrupção, ocasionada pela ausência de um bloco no momento da tentativa de sua reprodução. A Figura 5.2 ilustra a forma como esta métrica é contabilizada. A principal característica dessa métrica é verificar se o protocolo consegue se recuperar rapidamente de uma interrupção. A sua média é igual a: $TR = \frac{\sum_{i=1}^{N} TR_i}{N}$, onde TR_i é o tempo médio de retorno relativo ao usuário i e N é o total de usuários do *swarm*;

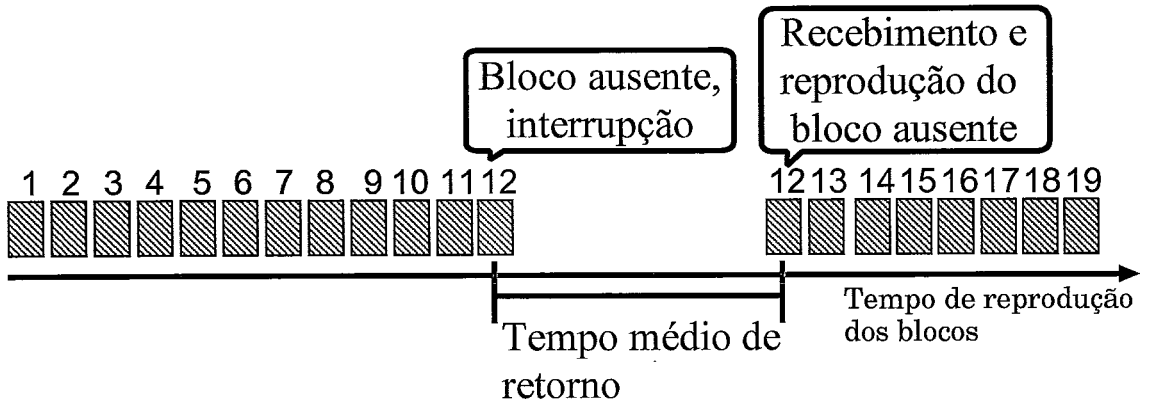

Figura 5.2: Procedimento para a contabilização da métrica tempo médio de retorno.

- **Tempo para iniciar a reprodução**: define o tempo que o usuário tem que esperar para reproduzir o primeiro bloco. A Figura 5.3 ilustra a forma como esta métrica é contabilizada. Sua média é estimada da seguinte forma: $TI = \frac{\sum_{i=1}^{N} TI_i}{N}$, onde TI_i é o tempo relativo ao usuário i;

- **Tempo de *download* - δ_d**: intervalo de tempo (segundos) entre o recebimento do primeiro bloco e o recebimento do último bloco pelo usuário;

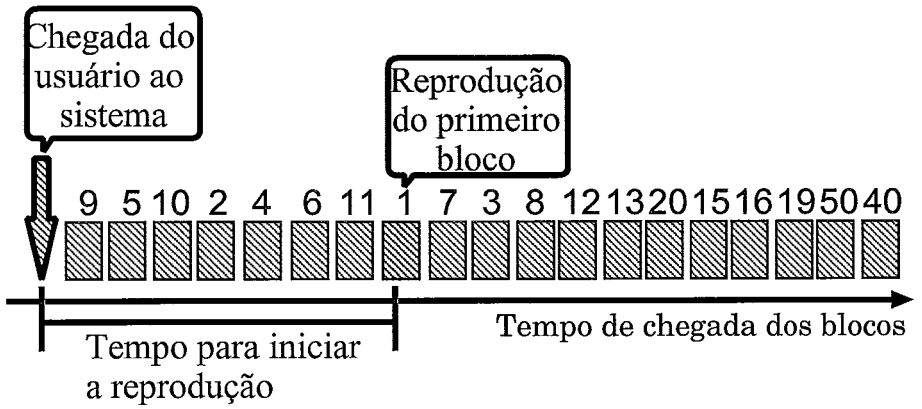

Figura 5.3: Procedimento para a contabilização da métrica tempo para iniciar a reprodução.

- **Tempo de *upload*** - δ_u: intervalo de tempo (segundos) entre o envio do primeiro bloco e o enviou do último bloco pelo usuário;
- **Taxa de *download***: número de blocos recebidos pelo usuário divido por δ_d e sua unidade é $KBytes/s$;
- **Taxa de *upload***: número de blocos enviados pelo usuário divido por δ_u e sua unidade é $KBytes/s$.

Sendo assim, as três primeiras métricas podem ser classificadas como medidas de qualidade da reprodução do vídeo, quantificando o número de interrupções e o tempo médio de retorno das interrupções durante a reprodução do vídeo. Enquanto que as quatro últimas, são métricas para quantificar o desempenho do sistema quanto a capacidade de distribuição do objeto entre os nós e a utilização da banda dos nós.

As métricas **número médio de interrupções** e **tempo médio de retorno** fazem parte das contribuições desse trabalho. Haja vista que, as propostas da literatura utilizam outras métricas para avaliar o seu desempenho, como: tempo para iniciar o reprodução e o índice de continuidade, que seria o número de blocos reproduzidos pelo número total de blocos do vídeo. Observou-se a necessidade de desenvolver novas métricas que pudessem medir com mais precisão o desempenho das propostas quando distribuindo vídeo sob demanda com interatividade.

5.3 Cargas

Foram utilizados *logs* reais de comportamento dos usuários do curso de tecnologia em sistemas de computação do CEDERJ (Centro de Educação Superior a Distância do Estado do Rio de Janeiro) [25], para emular o comportamento interativo dos usuários nas simulações, como também, para o treinamento do modelo de comportamento do usuário [4]. Os *logs* são registros de ações e acessos de usuários ao sistema multimídia do consórcio CEDERJ. O sistema que armazena, gerencia e disponibiliza o conteúdo do curso foi desenvolvido pelo Laboratório LAND da COPPE/UFRJ. O servidor multimídia RIO (*Random I/O System*) [67] é um sistema de armazenamento multimídia universal, que usa alocação aleatória e replicação de blocos. Para acessar o conteúdo armazenado no servidor, os usuários utilizam um *software* cliente, que possibilita a interação do usuário com o conteúdo que está sendo apresentado. É através do *software* cliente que os usuários podem realizar ações interativas, como: paralisar a exibição da aula; saltar para outro ponto da aula através de controles: *fast forward*, *fast rewind*, arrastar a barra de progresso ou clicar no índice de um *slide*; e parar a exibição do conteúdo a qualquer instante.

Do número total de *logs* disponíveis dos usuários do consórcio CEDERJ, foram selecionados os *logs* que apresentavam vídeos-aula com tempo de duração de pelo menos 1800 segundos e tempo de sessão, ou seja, tempo que o usuário permaneceu assistindo a vídeo aula, entre 900 e 1800 segundos. Além disso, foram eliminados *logs* que apresentam erros ou inconsistência nos dados gravados. Os *logs* selecionados foram classificados em três níveis de interatividade de acordo com o número de ações interativas (por exemplo: saltos para frente, para trás, pausas, etc.) realizadas pelo usuário: alta interatividade, aqueles que apresentam um número de ações entre 16 e 40 ($16 \leq X \leq 40$); média interatividade, aqueles que apresentam entre 6 e 15 ações ($6 \leq X \leq 15$); e baixa interatividade, os que apresentam entre 0 e 5 ações ($0 \leq X \leq 5$). Além destas três cargas com níveis de interatividade distintos, uma quarta carga mista, que apresenta número de ações entre 0 e 40 ($0 \leq X \leq 40$) também foi selecionada.

Tabela 5.1: Definição das variáveis

Variáveis	**Definição**
N	Número total de requisições
I	Número médio de requisições por sessão
L	Tamanho médio do segmento, medido em segundos
$Std(L)$	Desvio padrão de L
$Coef(L)$	Coeficiente de variação de L

Tabela 5.2: Estatísticas

	Alta		**Média**		**Baixa**		**Mista**	
Estatística	**Real**	**Sintética**	**Real**	**Sintética**	**Real**	**Sintética**	**Real**	**Sintética**
N	1752	1582	1205	1287	388	454	3346	2541
I	24.01	21.68	9.80	10.46	1.99	2.33	8.56	6.50
L	26.03	26.80	61.40	61.70	260.65	260.70	75.54	106.47
$Std(L)$	29	33	68	65	260	263	77	107
$Coef(L)$	1.14	1.23	1.11	1.05	1.00	1.00	1.02	1.00

Estes conjuntos de *logs* também foram utilizados para a parametrização e treinamento do modelo de emulação de comportamento do usuário [66, 4] e, assim, foram geradas quatro cargas sintéticas que apresentam características semelhantes às das cargas reais, como pode ser observado na Tabela 5.2. As quatro cargas sintéticas são utilizadas pelas propostas BIVoD e BIVoD-*Buffer* como sendo a previsão das futuras ações dos usuários, enquanto que as cargas reais são as ações de fato realizadas pelos usuários.

Estas quatro cargas garantem um significativo espectro de análise pois, são estatisticamente diferentes entre si. As variáveis usadas para denotar as estatísticas das cargas, reais e sintéticas, estão na Tabela 5.1, e seus valores correspondentes estão na Tabela 5.2.

A partir dessas tabelas, é possível notar que o nível de interatividade da carga sintética é entre 10% a 15% maior ou menor que o nível de interatividade da carga real. Já a diferença para o tamanho médio do segmento (L) é sempre abaixo de 3%, exceto para carga mista. A carga sintética mista, por ser uma carga heterogênea, apresenta uma diferença maior com relação à carga real.

5.4 Experimentos

Todos os experimentos foram realizados com a ferramenta Tangram-II [26]. Esta ferramenta constitui-se em um ambiente de modelagem e experimentação de sistemas computacionais e de comunicações, desenvolvido na Universidade Federal do Rio de Janeiro (UFRJ), com participação inicial da UCLA/USA, com os propósitos de pesquisa e educação. Este ambiente possui uma interface de usuário sofisticada e combina um paradigma de orientação a objeto para descrição do modelo, com diversas técnicas de solução para análise de desempenho e disponibilidade. O usuário especifica um modelo em termos de objetos que interagem por meio de um mecanismo de troca de mensagens. Uma vez compilado o modelo, pode ser resolvido analiticamente, caso seja Markoviano ou pertença a uma determinada classe de mo-

delos não Markovianos, ou resolvido via simulação. É possível obter soluções tanto para estado estacionário quanto para estado transiente.

Para facilidade de exposição, os resultados são apresentados de acordo com os objetivos elencados na Seção 5.1 e, salvo informado diferentemente, os cenários admitidos são:

- O tamanho do vídeo utilizado para as simulações é de 1800 segundos.
- Cada bloco equivale a 1 segundo de vídeo.
- O número de nós *seeds* é igual a 1.
- A banda de *upload* disponível para os nós *leechers* e para também o *seed* é de $100\,K\,bytes/s$.
- O tamanho da janela de previsão nas propostas BIVoD e BIVoD-*Buffer* foi definido em função do tamanho médio do segmento L, apresentando na Tabela 5.1. L é o tempo médio que o usuário permanece assistindo ao vídeo antes de realizar qualquer ação interativa, portanto é uma boa estimativa para a quantidade de blocos que a previsão deve recuperar, de forma a diminuir o risco de ocorrência de interrupções após um salto do usuário.
- A chegada dos usuários segue um processo de Poisson, onde foram definidos 2 cenários com taxas distintas: $\lambda = 4$ ou 0.008 usuários/segundo.
- Foram definidos 8 cenários distintos para a realização dos experimentos, que são: usuário com alta interatividade e com $\lambda = 4$ ou 0.008 usuários/s, usuários de média interatividade e com $\lambda = 4$ ou 0.008 usuários/segundo, usuário de baixa interatividade e com $\lambda = 4$ ou 0.008 usuários/segundo e usuários com alta, média e baixa interatividade (misto) e com $\lambda = 4$ ou 0.008 usuários/segundo. O número de nós em cada experimento variou entre 50 e 200;
- Para cada um dos experimentos foram realizadas 10 simulações e serão apresentados os seus valores médios, sendo o intervalo de confiança definido em 90%.

A Subseção 5.4.1 dedica-se aos resultados provenientes da validação do modelo do protocolo *BitTorrent*. A Subseção 5.4.2 é dedicada a apresentar os problemas do protocolo *BitTorrent* quando este é usado para a distribuição de vídeo sob demanda com interatividade. Posteriormente, a Subseção 5.4.3 é dedicada a avaliação dos parâmetros de todas as propostas. Na Subseção 5.4.4 é apresentada a primeira análise comparativa entre todas as propostas, com a finalidade de encontrar a proposta da literatura mais competitiva. A Subseção 5.4.5 é dedicada à realização de uma avaliação mais detalhada entre a proposta da literatura com o melhor desempenho e as propostas BIVoD e BIVoD-*Buffer*.

5.4.1 Comparação do modelo do Protocolo *BitTorrent* com testes em ambiente real

Como forma de avaliar o impacto das simplificações introduzidas no modelo do protocolo *BitTorrent* foram realizados experimentos reais no ambiente de teste *PlanetLab* [68, 69] para serem confrontados com os resultados gerados pelas simulações do modelo proposto.

O *PlanetLab* é um ambiente global para pesquisa e desenvolvimento de novas tecnologias e/ou serviços em redes de computadores e sistemas distribuídos. Fundado no começo de 2003, hoje conta com mais de 1000 instituições, sendo universidades, centros de pesquisas e indústrias, considerados membros ou associados, e que têm o utilizado para o desenvolvimento e pesquisa de novas tecnologias, tais como: armazenamento distribuído, compartilhamento de dados, sistemas *peer-to-peer*, etc. Atualmente, 1021 nós estão disponíveis para serem utilizadas pelos membros e associados do *PlanetLab*, distribuídos por todos os continentes, mas principalmente na América do Norte e Europa. Entretanto, apenas 250 nós foram utilizados para a realização desse estudo.

O cliente *BitTorrent* escolhido para ser utilizado nos experimentos foi o cliente oficial versão 4.0.2, que foi modificado nos trabalhos de Legout *et al* [65, 38]. As modificações têm o objetivo de fazer com que o cliente gere *logs* com o detalhamento

do conteúdo de todas as mensagens enviadas e recebidas e também informações importantes sobre o seu estado, como: taxa de *download* e *upload*, quais vizinhos estão desbloqueados e quais estão bloqueados, etc. Muitos dos parâmetros do cliente oficial não foram alterados, tais como: número mínimo de vizinhos para a requisição de mais vizinhos ao *tracker* (20), número máximo de vizinhos com os quais um nó pode manter conexão (50), número de vizinhos desbloqueados de acordo com a banda de *upload* disponível, incluindo o *optimistic unchocked* (1), tamanho dos blocos de 256 KB, tamanho dos pedaços de 16 KB e número de blocos recuperados na fase *random first* (4).

Para essa avaliação foram utilizados 4 cenários, onde se variou o número de nós *leechers* (50, 100, 150 e 200). Outros parâmetros tiveram os seguintes valores: capacidade de banda dos nós, ou seja, taxa máxima de *upload* de 100 KB/s, número de *seeds* igual a 1, tamanho do objeto distribuído de 25 MB, ou seja, 100 blocos de 256 KB e divido em 16 pedaços de 16 KB e a chegada dos nós ao *swarm* é em *flash-crowd*, ou seja, todos os nós chegam em um intervalo de tempo muito pequeno. O comportamento de cada *leecher* é de deixar o *swarm* assim que recuperar todo o objeto. Estes cenários foram usados para os experimentos no *PlanetLab* como também para as simulações do modelo.

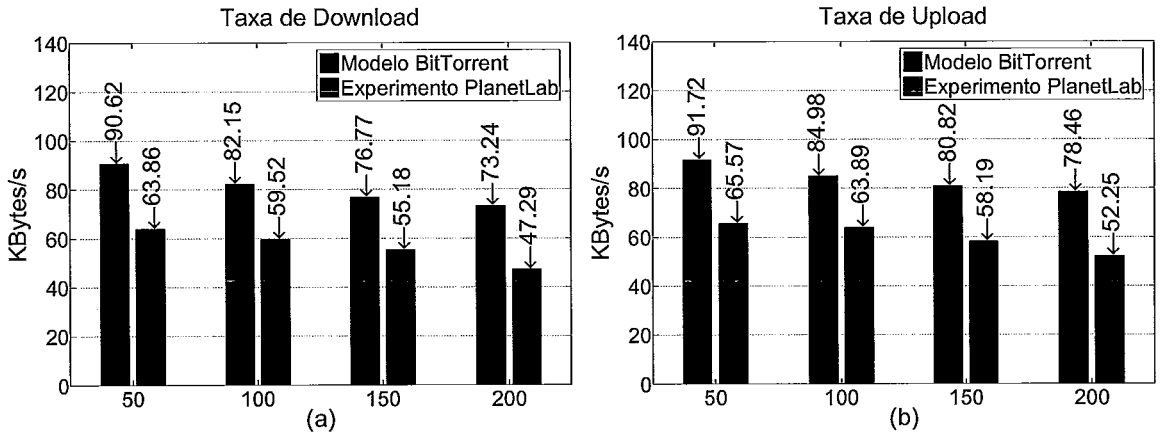

Figura 5.4: Comparação entre o modelo do *BitTorrent* e os experimentos reais realizados no *PlanetLab*.

Na Figura 5.4 são apresentadas as taxas médias de *download* e *upload* obtidas pelo modelo do protocolo *BitTorrent* e os experimentos reais no *PlanetLab* para os

4 cenários. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 5% em torno do valor da média. Em todos os 4 cenários os resultados do modelo do *BitTorrent* apresentaram taxas de *download* e de *upload* na média, 35% maior que as taxas obtidas com os experimentos realizados no *PlanetLab*. Essa diferença pode ser creditada a três fatores. Primeiro, no desenvolvimento do modelo de simulação, foram implementadas algumas simplificações com relação ao protocolo original, todas descritas no Seção 4.1. Segundo, não está representado no modelo características da rede como, perdas ou descartes de pacotes em roteadores ou o compartilhamento de *links* gargalos, foi modelado a transmissão de blocos entre os nós considerando que a rede opera em condições de baixa carga. Desta forma, o usuário não tem problemas na recuperação do objeto devido a ocorrência de perdas ou alto retardo no recebimento dos pacotes em sua conexão ou na de seus vizinhos. E terceiro, o *PlanetLab* por tratar-se de um ambiente real não pode ser controlado, isto é, não se pode reservar recursos, como banda e processamento das máquinas, exclusivamente para a realização de um experimento, outros usuários também estão utilizando-o e consumindo os seus recursos, o que pode também influenciar no desempenho desses experimentos. Entretanto, apesar da diferença entre os valores das taxas obtidos nos experimentos e os obtidos no modelo, o comportamento apresentado pelo modelo é semelhante ao encontrado no experimento real. No modelo a taxa média de *download* para os 50 nós é 19% maior que a taxa média de *download* para 200 nós. No experimento do *PlanetLab* esta taxa é 17% maior. Já a taxa média de *upload* é de 16% maior para 50 usuários no caso do modelo e de 20% maior no caso do experimento. Este comportamento demonstra que com o aumento da população no *swarm* ocorre um declínio nos valores médios de todas as métricas. Este declínio pode ser decorrente do cenário utilizado considerando um único *seed* e a saída dos *leechers* assim que terminam o *download* do objeto.

Na Figura 5.5 são apresentados os valores médios para as métricas taxa e tempo de *download* para cada um dos nós. Fica evidente que o modelo do *BitTorrent* captura muito bem a característica de equidade entre os usuários do sistema, demons-

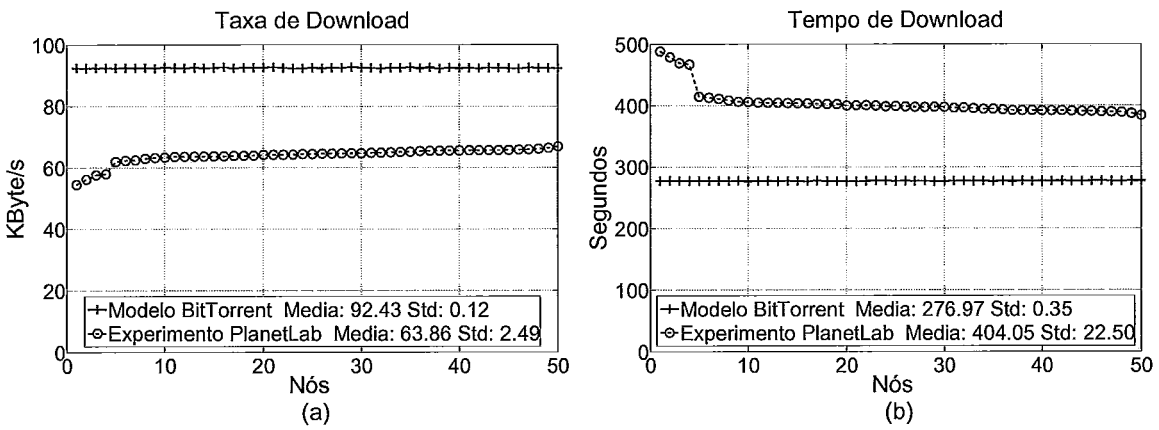

Figura 5.5: Validação do protocolo *BitTorrent* com 50 nós no *swarm* (cenário 1).

trando que não existem grandes variações de taxa/tempo entre os nós do *swarm*. Este é um resultado esperado pois, a equidade entre nós homogêneos é uma característica apresentada pelo protocolo *BitTorrent* em estudos da literatura [65, 38]. O experimento no *PlanetLab* demonstra uma pequena variabilidade nos valores de taxa/tempo médio entre os nós. Este resultado já era esperado haja vista que se trata de um experimento real e fatores como condições da rede de acesso do nó, carga da máquina executando o nó e banda disponível para o nó, podem afetar o desempenho do protocolo.

5.4.2 Problemas do protocolo *BitTorrent* para a distribuição de vídeo sob demanda com interatividade

O protocolo *BitTorrent* foi desenvolvido para a distribuição eficiente de conteúdo, esses podendo ser um CD de música, um ISO de uma distribuição linux, um vídeo, etc. Os seus algoritmos de seleção de blocos e seleção de vizinhos se mostram muito eficientes para essa tarefa. Porém, quando utilizado para a distribuição de vídeo ou áudio sob demanda com a reprodução imediata, ou seja, a mídia é exibida na medida em que é recuperada, o seu desempenho em relação à qualidade da reprodução, medido em termos de número de interrupções, tempo para o retorno de uma interrupção e principalmente o tempo para iniciar a reprodução é fortemente afetado. A razão para este baixo rendimento é principalmente o funcionamento do

seu algoritmo de seleção de blocos, que utilizando a política de seleção do bloco mais raro não privilegia nenhum bloco ou parte da mídia para ser recuperada antes de outras partes ou blocos menos necessários naquele momento. Isto faz com que ocorra um longo período de espera para o início da reprodução ou mesmo para o retorno de uma interrupção.

Foram realizadas algumas simulações com objetivo de mostrar o baixo desempenho do protocolo *BitTorrent* quando utilizado por uma aplicação de vídeo sob demanda. Para a realização desses experimentos foram utilizados diferentes níveis de interatividade e foi variado o número de nós no *swarm* entre 50 e 200.

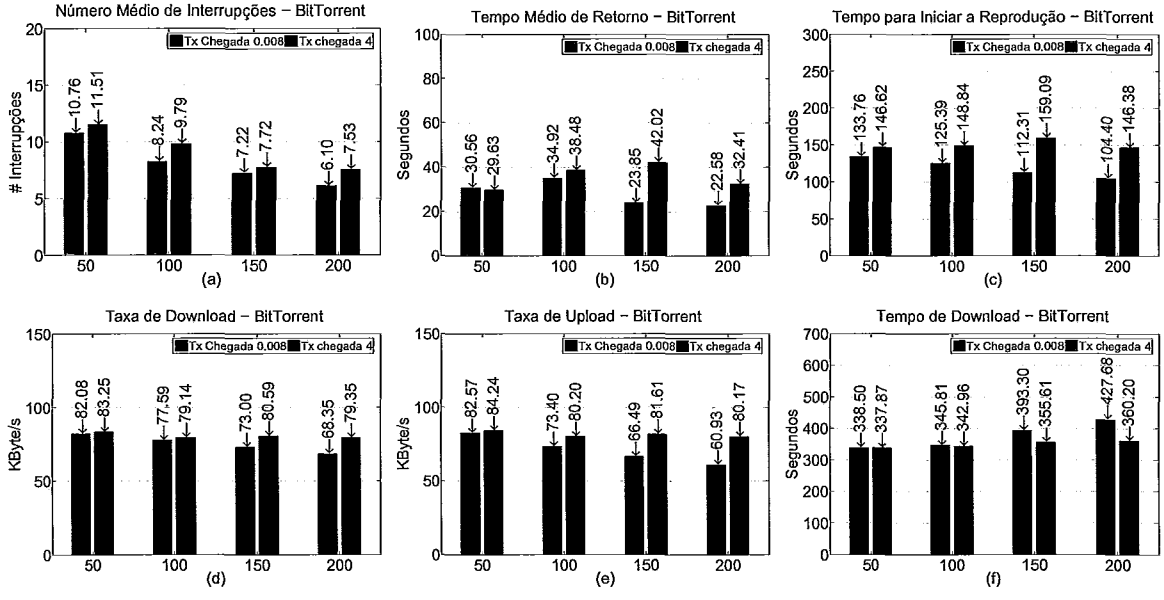

Figura 5.6: Avaliação do desempenho do protocolo *BitTorrent* variando o tamanho da população para carga mista.

Na Figura 5.6 são apresentados os resultados considerando carga mista e duas taxas de chegada. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 8%, 10%, 5%, 2%, 2% e 2% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*, respectivamente. Considerando todos os cenários esperou-se em média 135 segundos para ser iniciada a reprodução do vídeo (Figura 5.6 (c)),

enquanto que o tempo gasto para o *download* do vídeo completo foi em média de 362 segundos (Figura 5.6 (f)), ou seja, cada usuário esperou em média 40% do tempo total de download para iniciar a reprodução do vídeo. Claramente, é muito desagradável para o usuário ter que esperar todo este tempo para começar a assistir a um vídeo de 1800 segundos. Ainda na Figura 5.6 (a) é apresentado o resultado para a métrica número médio de interrupções, que apresenta a média de 8.7 interrupções. Esse é um resultado que pode ser considerado razoável, mas o tempo necessário para a recuperação de uma interrupção é muito alto. Como pode ser observado na Figura 5.6 (b), este tempo é em média de 32 segundos.

Considerando que um nó tenha sofrido 8.7 interrupções e que para se recuperar de cada uma ele precise de 32 segundos, ele ficou um total de 278 segundos parado esperando para retornar a reprodução. Somando esse tempo ao tempo para o início da reprodução que é de 135 segundos, tem-se um tempo de 413 segundos de espera, um tempo maior que o tempo gasto para o *download* do vídeo completo. Isto demonstra o quanto o protocolo *BitTorrent* pode ser ineficiente para aplicações de *streaming* de vídeo.

As métricas taxa de *download*, taxa de *upload* e tempo de *download* mostram a eficiência do protocolo *BitTorrent* com relação ao uso da capacidade dos nós do sistema para a distribuição do vídeo.

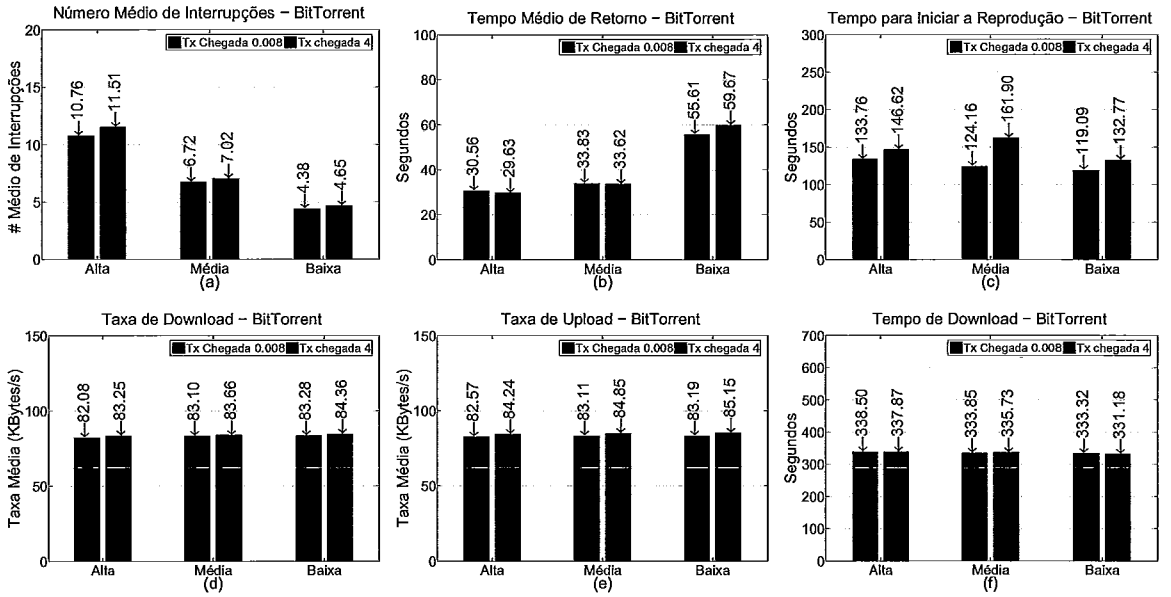

Figura 5.7: Avaliação do desempenho do protocolo *BitTorrent* variando o nível de interatividade da carga.

A Figura 5.7 apresenta o resultado da avaliação do protocolo *BitTorrent* para a distribuição de vídeo sob demanda considerado diferentes níveis de interatividade, para uma população de 50 usuários. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 8%, 11%, 5%, 2%, 2% e 2% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*, respectivamente. Os resultados seguem a mesma tendência do experimento anterior, apresentando longos períodos para iniciar a reprodução, Figura 5.7 (c), e também para retomar a reprodução após uma interrupção, Figura 5.7 (b), mas com excelentes taxas de *download* e *upload*. Já a métrica número médio de interrupções, Figura 5.7 (a), apresenta um comportamento relacionado ao nível de interatividade, onde com a diminuição no número de ações o número de interrupções tende a diminuir, como é o caso para a carga baixa que apresenta o menor número de interrupções enquanto que a alta apresenta o maior.

Este experimento mostra que a distribuição de vídeo sob demanda utilizando o protocolo *BitTorrent* apresenta um desempenho ruim com relação a qualidade

percebida pelo usuário. Modificações ou adaptações são necessárias para que se consiga distribuir este tipo de mídia com um mínimo de qualidade na reprodução.

5.4.3 Avaliação dos parâmetros das novas propostas e algumas da literatura

Nesta seção, inicialmente, será avaliado os valores para os parâmetros dos algoritmos propostos nesse trabalho, assim como os valores dos parâmetros dos algoritmos da literatura. O objetivo é avaliar o tamanho ideal da janela de *playback* das propostas BIVoD e BIVoD-*Buffer* e também o tamanho ideal do conjunto de alta prioridade das propostas *BiToS* [2] e de *Zhou-Chiu-Lui* [23], assim como o tamanho da janela deslizante da proposta de *Shah-Pâris* [3]. Além disso, para as propostas *BiToS* e *Zhou-Chiu-Lui* também é necessária a escolha do parâmetro p, que representa a probabilidade de escolher o conjunto de alta prioridade para a recuperação de seus blocos. E também para as propostas BIVoD e BIVoD-*Buffer* avaliar o tamanho ideal da janela de previsão. E ainda o tamanho ideal do *buffer* da proposta BIVoD-*Buffer*.

Uma primeira avaliação que pode ser feita entre todas as propostas é em relação ao número de parâmetros utilizado por cada uma. Um menor número de parâmetro torna a proposta mais simples, ao passo que um número maior pode tornar a proposta mais complexa. Entretanto, isto não implica em tornar a proposta com menor número de parâmetros mais eficiente ou com maior número de parâmetros menos eficiente. O julgamento pelo número de parâmetro serve, principalmente, para analisar a complexidade da proposta. A proposta de *Shah-Pâris* é a que apresenta o menor número de parâmetros, apenas um: tamanho da janela deslizante. Enquanto que as propostas *BiToS*, de *Zhou-Chiu-Lui* e a BIVoD tem dois parâmetros: tamanho do conjunto de alta prioridade e da probabilidade de escolha do conjunto de alta prioridade para as duas primeiras propostas e o tamanho das janelas de *playback* e de previsão para a proposta BIVoD. Já a proposta BIVoD-*Buffer* apresenta o maior número de parâmetros, três no total: tamanho das janelas de *playback* e de previsão

e o tamanho do *buffer*. Esta avaliação mostra que a proposta de *Shah-Pâris* é a mais simples e a proposta BIVoD-*Buffer* a mais complexa em relação ao número de parâmetros a serem avaliados, porém isto não tornar uma proposta mais eficiente do que a outra, o mais importante é avaliar o desempenho de cada uma na distribuição de vídeo sob demanda com interatividade.

Este estudo é necessário porque os valores dos parâmetros contidos nos trabalhos da literatura que apresentam as propostas *BiToS* [2], de *Zhou-Chiu-Lui* [23] e de *Shah-Pâris* [3] foram obtidos para acesso sequencial. Neste estudo é considerado o acesso interativo, portanto é importante obter os valores dos parâmetros mais adequados para este tipo de acesso. Em seguida, uma análise competitiva é realizada entre todas as propostas, com o objetivo de selecionar a proposta da literatura mais competitiva em comparação com as propostas desse trabalho.

Foi considerado uma população de 50 usuários no *swarm*, que partem dele assim que terminam as suas ações interativas, com três níveis de interatividade e duas taxas de chegada. O valor da janela (playback, conjunto de alta prioridade ou deslizante) utilizado para avaliação das propostas foi 2%, 8% e 20% do tamanho do vídeo, ou seja, 36, 144 e 360 segundos, respectivamente. Esses valores são baseados nos encontrados nos trabalhos das propostas *BiToS*, de *Zhou-Chiu-Lui* e de *Shah-Pâris*. Ainda, de acordo com aqueles trabalhos, valores entre 20% e 100% do tamanho do vídeo apresentam grande degradação na qualidade da reprodução do vídeo, principalmente por apresentar longos períodos de espera para iniciar a reprodução e a recuperação dos blocos ausentes. Isso é devido ao aumento no conjunto de blocos que podem ser recuperados. Resultados com tamanho de janela grande são semelhantes aos obtidos pelo protocolo *BitTorrent*.

Avaliação dos parâmetros da proposta *BiToS*

Na Figura 5.8, são apresentados os resultados da variação do tamanho do conjunto de alta prioridade para a proposta *BiToS*. De acordo com o trabalho de [2] o tamanho ideal do conjunto de alta prioridade é 8% do tamanho do objeto e a

probabilidade p igual a 0.80 para a escolha do conjunto de alta prioridade. Sendo assim, para realizar este primeiro experimento, variou-se o tamanho do conjunto de alta prioridade (2%, 8% e 20%) e fixou-se o valor da probabilidade $p = 0.80$. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 9%, 35% e 4% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno e taxa de *download*, respectivamente.

O resultado da métrica número médio de interrupções, Figuras 5.8 (a) e 5.8 (d), apresentou a maior variação entre o melhor e pior resultado para o cenário de carga baixa, onde o conjunto de tamanho 2% obteve um desempenho até 27% melhor que o conjunto de tamanho 20%. Entretanto, para as métricas tempo médio de retorno, Figuras 5.8 (b) e 5.8 (e), e taxa de *download*, Figuras 5.8 (c) e 5.8 (f), essa diferença não ultrapassou 15%, independente do cenário. Sendo assim, não foi observada significativa diferença entre os resultados quando foi variado o tamanho do conjunto de alta prioridade para todos os cenários analisados. Porém, o tamanho do conjunto igual a 8% mostrou um comportamento mais uniforme, ou seja, obteve resultados muito próximos aos melhores, quando não o obteve, e apresenta o melhor *trade-off* entre número médio de interrupções e tempo médio de retorno. Por isto, este é o valor escolhido para ser utilizado nos demais experimentos com a proposta *BiToS*.

Uma observação a ser feita é em relação ao desempenho do *BiToS* para cenários com diferentes níveis de interatividade. O melhor desempenho para a métrica número médio de interrupções foi para carga alta, porém a carga baixa apresentou um elevado número de interrupções, mas um tempo médio de retorno muito pequeno. Este fato por ser creditado ao problema de "toca e para", ou seja, um bloco ausente é recebido e tocado mas o bloco seguinte ainda não foi recuperado, gerando uma nova interrupção, porém por um curto intervalo de tempo.

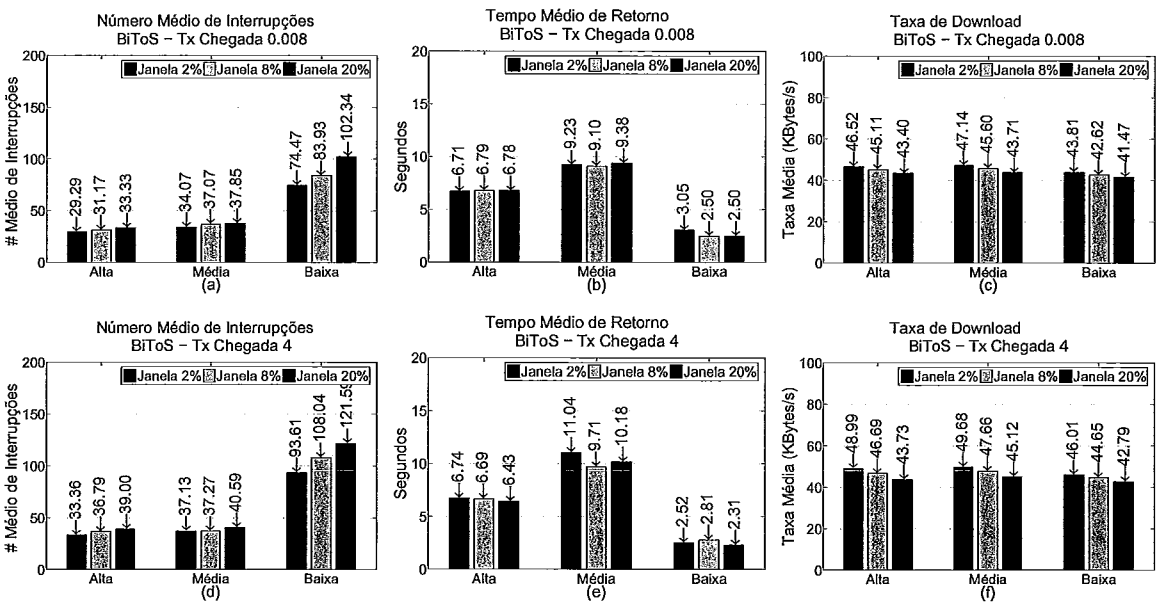

Figura 5.8: Avaliação do impacto na variação do tamanho do conjunto de alta prioridade para a proposta *BiToS*.

Na Figura 5.9, são apresentados os resultados da avaliação do parâmetro p da proposta *BiToS*. Esse parâmetro é a probabilidade de escolher um bloco do conjunto de alta prioridade para ser recuperado. Para esse experimento p assumiu três valores 0.70, 0.80 e 0.90, enquanto que o tamanho do conjunto de alta prioridade foi fixado em 8% do tamanho do objeto. Pelos resultados pode-se observar a mesma tendência do experimento da variação do tamanho da janela, onde também não se observa significativa diferença entre os resultados. Na métrica número médio de interrupções, Figuras 5.9 (a) e 5.9 (d), a maior diferença chegou a 17% mas em apenas um único cenário. Para a métrica tempo médio de retorno, Figuras 5.9 (b) e 5.9 (e), os resultados para $p = 0.90$ aparecem com mais destaque em relação aos demais, porém sua taxa de *download* é a mais baixa. Para $p = 0.70$, a taxa de *download* é a maior, mas o tempo médio de retorno também, isto pode ser justificado pela diminuição no número de blocos recuperados do conjunto de alta prioridade. Entretanto, $p = 0.80$ apresenta o melhor *trade-off* entre taxa de *download* e tempo médio de retorno, sempre apresentado resultados muito próximos ao melhor. Para os próximos experimentos com a proposta *BiToS*, os valores para o tamanho do conjunto de alta prioridade e do parâmetro p serão 8% e 0.80, respectivamente.

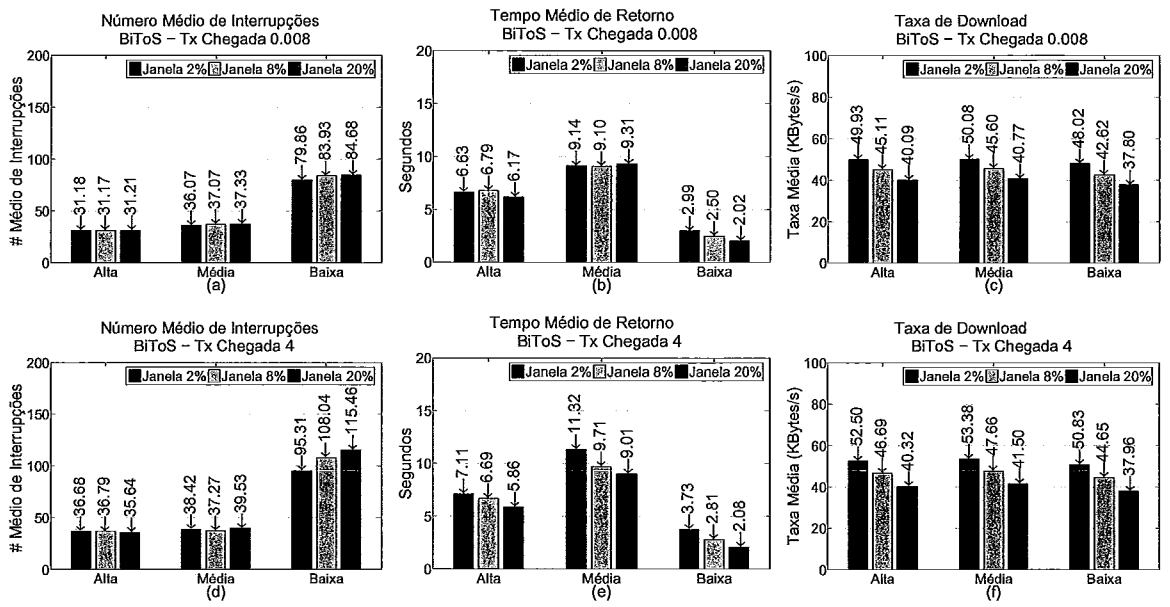

Figura 5.9: Avaliação do impacto na variação da probabilidade p na proposta *BiToS*.

Avaliação dos parâmetros da proposta *Zhou-Chiu-Lui*

Os resultados para a proposta de *Zhou-Chiu-Lui* estão demonstrados na Figura 5.10. No trabalho de *Zhou-Chiu-Lui* [23], o tamanho ideal para o conjunto de alta prioridade foi de 8% do tamanho do objeto e para a probabilidade $p = 0.80$. Para realizar esta avaliação foi fixado o valor da probabilidade p em 0.80, enquanto o tamanho do conjunto de alta prioridade foi variado. Todos os resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 9%, 35% e 5% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno e taxa de *download*, respectivamente.

Entretanto, nos experimentos realizados não foi verificada nenhuma diferença significativa de desempenho entre os três tamanhos do conjunto de alta prioridade. Sendo assim, será utilizado um conjunto de alta prioridade com tamanho de 8%, por apresentar o melhor *trade-off* entre número de interrupções e tempo médio de retorno, para os demais experimentos a serem realizados com a proposta *Zhou-Chiu-Lui*.

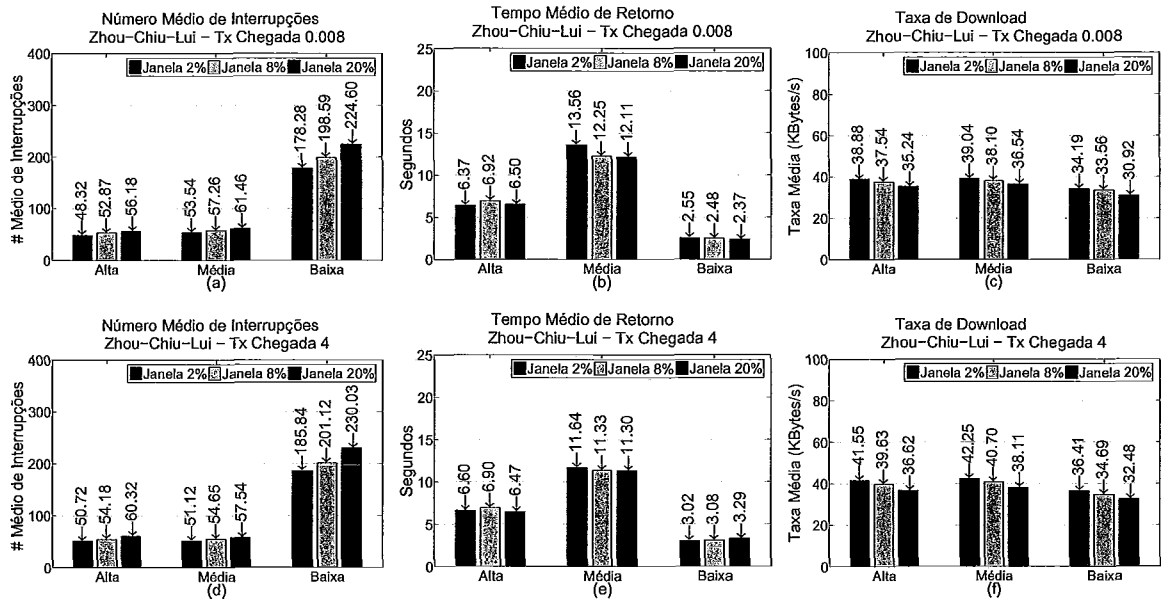

Figura 5.10: Avaliação do impacto na variação do tamanho do conjunto de alta prioridade para a proposta de *Zhou-Chiu-Lui*.

O comportamento da proposta de *Zhou-Chiu-Lui* nos diferentes cenários é muito semelhante ao encontrado na proposta *BiToS*, onde no cenário com carga baixa ocorre um elevado número de interrupções mas com um curto intervalo de tempo para o seu retorno, novamente em função do efeito "toca e para" da reprodução.

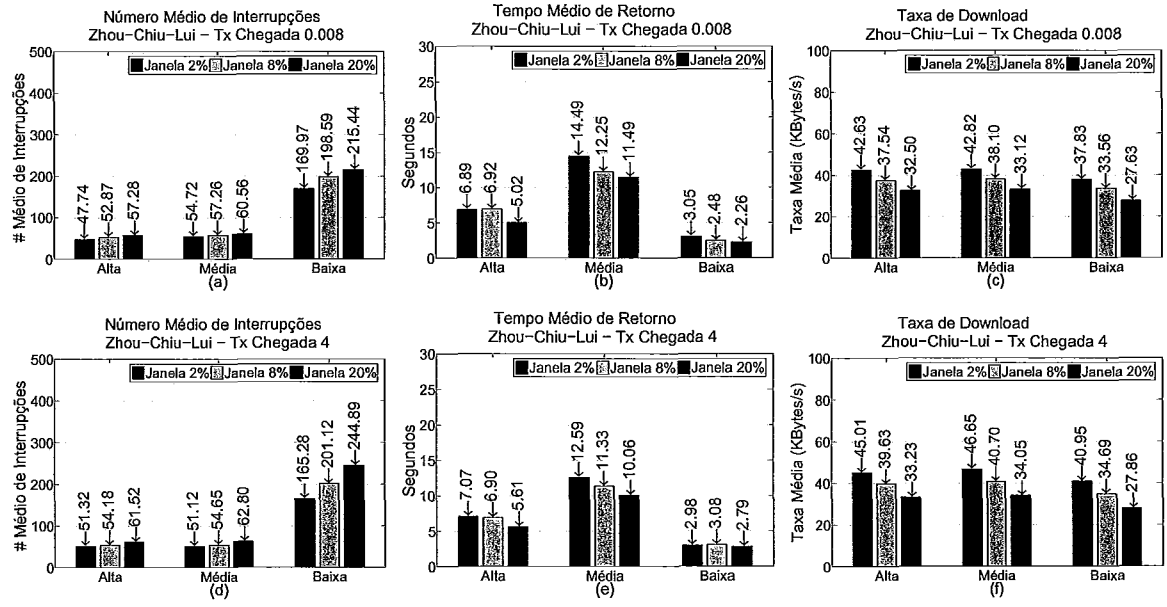

Figura 5.11: Avaliação do impacto na variação da probabilidade p na proposta de *Zhou-Chiu-Lui*.

Na Figura 5.11 é mostrado o desempenho da proposta de *Zhou-Chiu-Lui* quando o valor da probabilidade p assume os valores 0.70, 0.80 e 0.90, enquanto que o tamanho da janela foi fixado em 8%. Para a métrica número médio de interrupções, Figuras 5.11 (a) e 5.11 (d), a maior diferença entre o melhor e o pior desempenho foi de 32% nos cenários que envolvem a carga baixa. Nos demais essa diferença alcançou no máximo 18%, onde $p = 0.90$ apresenta em geral o pior desempenho. Entretanto, para a métrica tempo médio de retorno, Figuras 5.11 (b) e 5.11 (e), o cenário onde $p = 0.90$ foi o que obteve os melhores resultados, tendo ganhos da ordem de 27% em relação ao pior resultado. Já para $p = 0.70$ a proposta alcança as maiores taxas de *download*, mas o tempo médio de retorno é alto, o que é justificado pela diminuição da recuperação de blocos do conjunto de alta prioridade. Concluindo, $p = 0.80$ apresenta o melhor *trade-off* entre taxa de *download* e tempo médio de retorno, sempre apresentando resultados muito próximos ao melhor. Para os próximos experimentos com a proposta de *Zhou-Chiu-Lui*, os valores para o tamanho do conjunto de alta prioridade e do parâmetro p serão 8% e 0.80, respectivamente.

Avaliação dos parâmetros da proposta *Shah-Pâris*

Na Figura 5.12, é apresentado o resultado da avaliação do tamanho ideal para a janela deslizante da proposta de *Shah-Pâris* [3]. Segundo o trabalho de *Shah-Pâris* o tamanho ideal para a janela deslizante é de 20% do tamanho do objeto considerando acesso sequencial. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 12%, 29% e 4% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno e taxa de *download*, respectivamente. Entretanto, em todos os cenário dessa avaliação, o tamanho da janela igual a 20% apresentou valores até 20% superiores para a métrica número médio de interrupções, Figuras 5.12 (a) e 5.12 (d), e em até 72% maiores para a métrica tempo médio de retorno, Figuras 5.12 (b) e 5.12 (e), em relação aos melhores valores encontrados para estas métricas. Entretanto, para a métrica taxa de *download*, Figuras 5.12 (c) e 5.12 (f), obteve o melhor resultado para todos os cenários, chegando a valores em até 38% superiores aos obtidos pelos

demais tamanhos. Isto demonstra que para uma janela com tamanho de 20% do objeto a qualidade da reprodução é prejudicada pelo elevado número de interrupções e por longos períodos de espera para a retomada da reprodução, entretanto se mostra muito eficiente quanto a utilização dos recursos do sistema. Já para os tamanhos de 2% e 8%, os resultados demonstram que para as métricas número médio de interrupções e tempo médio de retorno não existe uma diferença expressiva que possa influenciar na qualidade de reprodução do vídeo, porém para a métrica taxa de *download*, o tamanho de 8% apresenta um resultado superior ao de 2% em todos os cenários. Apesar de não existir uma diferença entre os tamanhos de 2% e 8% que impactem na qualidade da reprodução do vídeo, um segundo fator que deve ser considerado é o uso eficiente da banda dos nós. Sendo assim, foi escolhido o tamanho de 8% para a janela deslizante, que será utilizado nos demais experimentos da proposta de *Shah-Pâris*.

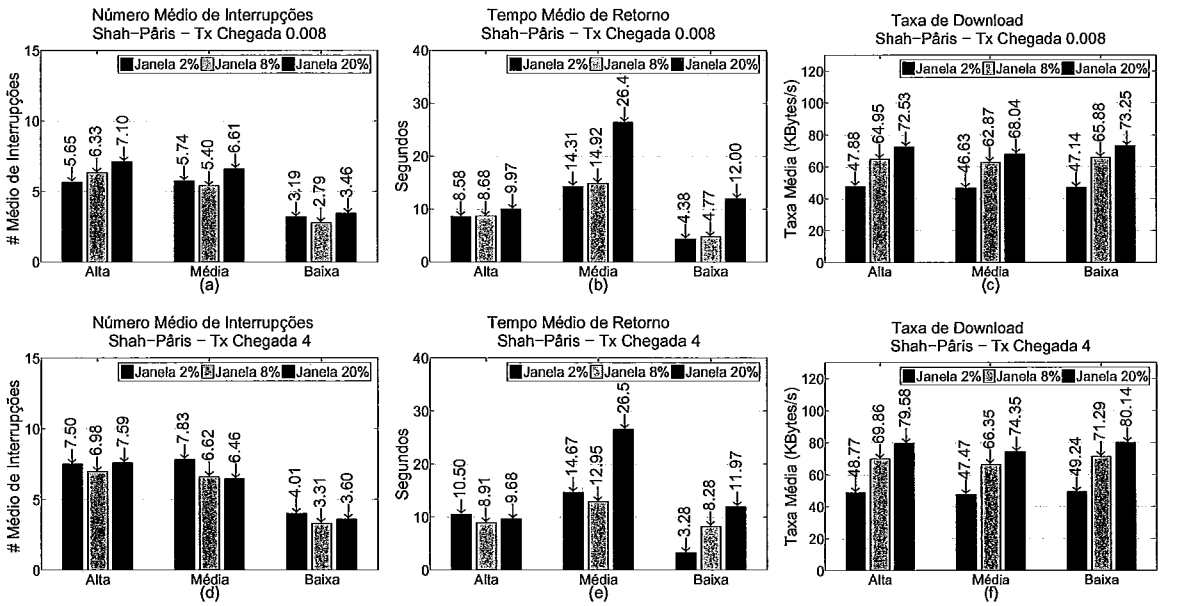

Figura 5.12: Avaliação do impacto na variação do tamanho da janela deslizante para a proposta de *Shah-Pâris*.

Avaliação dos parâmetros das novas propostas

Na Figura 5.13 são apresentados os resultados da avaliação do tamanho da janela de *playback* para a nova proposta BIVoD. Estes resultados são a média de 10 execu-

ções, tendo um intervalo de confiança de 90% variando no máximo 11%, 16% e 2% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno e taxa de *download*, respectivamente. Os resultados da métrica número médio de interrupções, Figuras 5.13 (a) e 5.13 (d), não apresentam significativa diferença considerando todos os cenários analisados. Porém, o tamanho de 8% apresentou resultados ligeiramente melhores que os demais em 5 dos 6 cenários. Observando-se, os resultados da métrica tempo médio de retorno, Figuras 5.13 (b) e 5.13 (e), nota-se uma significativa diferença de desempenho, podendo esta diferença chegar a 67% entre o melhor e o pior resultado. Claramente, o desempenho da janela com tamanho 20% é inferior ao da janela de 2% e 8%. Porém para a métrica taxa de *download*, Figuras 5.13 (c) e 5.13 (f), o desempenho apresentado pela janela de tamanho 20% é muito superior ao das demais, tendo o tamanho da janela de 2% apresentado o pior desempenho para esta última métrica.

A utilização de uma janela de *playback* muito grande ou muito pequena pode influenciar uma ou mais métricas positivamente e outras negativamente. Uma janela grande privilegia a recuperação e a distribuição do objeto em detrimento do tempo de retorno e do número de interrupções, enquanto que uma janela pequena prejudica a recuperação e distribuição, mas apresenta um tempo de retorno e número de interrupções menores. Portanto, utilizar uma janela com tamanho intermediário a estes dois mostra-se a melhor opção, o que foi comprovado pelos resultados apresentados na Figura 5.13, onde a janela com tamanho 8% apresentou o melhor desempenho em 5 dos 6 cenários para a métrica número médio de interrupções e desempenho sempre muito próximo ao melhor para as métrica tempo médio de retorno e taxa de *download*. Por isto, a escolha da janela de playback com tamanho de 8% parece ser a mais acertada para a proposta BIVoD.

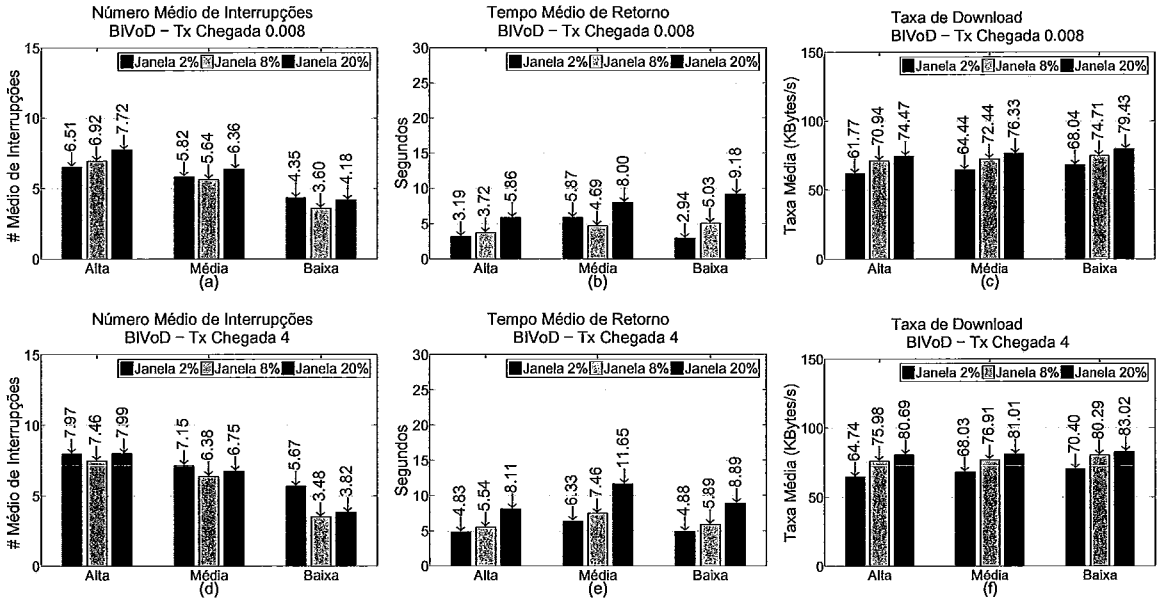

Figura 5.13: Avaliação do impacto na variação do tamanho da janela de *playback* para a proposta BIVoD.

O resultado da avaliação do tamanho da janela de *playback* para a proposta BIVoD-*Buffer* é apresentado na Figura 5.14. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 19%, 31% e 2% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno e taxa de *download*, respectivamente. Analisando o desempenho dessa proposta quando a janela de *playback* tem tamanho igual a 20%, pode-se verificar que a proposta atinge o melhor desempenho para todas as métricas no cenário carga baixa com as duas taxas de chegadas. Entretanto, nos demais cenários, apenas na métrica taxa de *download*, Figuras 5.14 (c) e 5.14 (f), consegue apresentar o melhor desempenho, enquanto que para as outras métricas apresenta um desempenho inferior. Por outro lado, a janela com tamanho 8% demonstra bons resultados para as métricas número médio de interrupções, Figuras 5.14 (a) e 5.14 (d), e tempo médio de retorno, Figuras 5.14 (b) e 5.14 (e). Em todos os cenários seu desempenho é muito próximo ao alcançado pelo da janela de 20%. Já a janela com tamanho 2% alcançou as mais baixas taxas de *download* em todos os cenários e seu desempenho para as outras métricas ficou próximo ao desempenho da janela de 8%. Portanto, será considerado o tamanho da janela de *playback* igual

a 8%, haja vista que esse tamanho apresenta o melhor desempenho para a grande maioria dos experimentos.

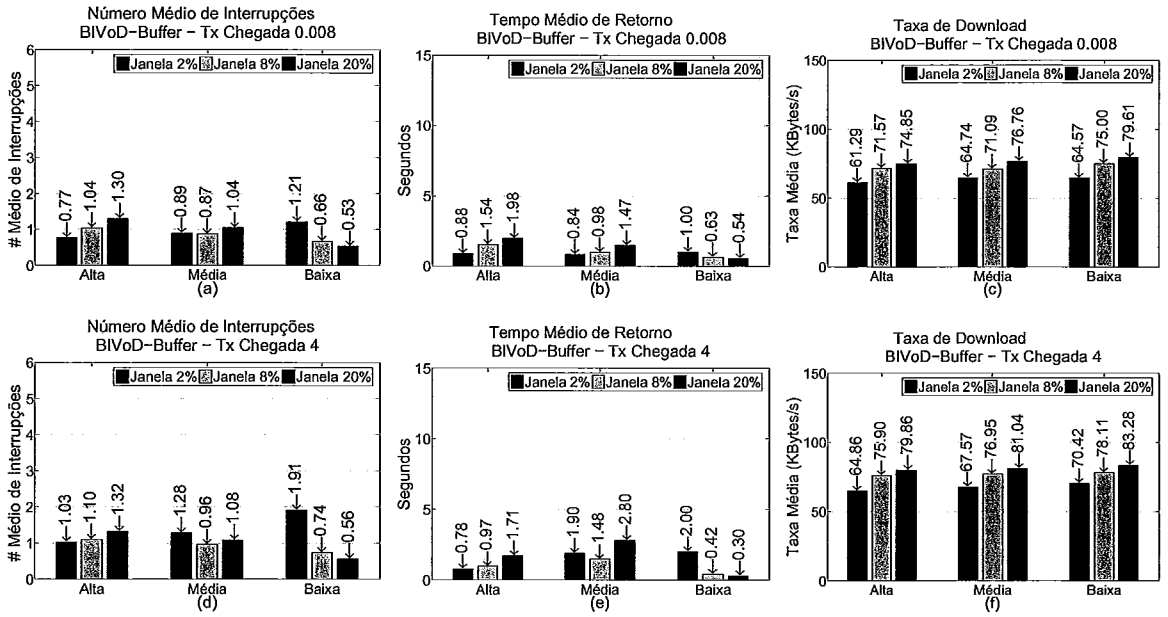

Figura 5.14: Avaliação do impacto na variação do tamanho da janela de *playback* para a proposta BIVoD-*Buffer*.

Avaliando o tamanho ideal para o *buffer* da proposta BIVoD-*Buffer*

Na Figura 5.15 é apresentado o resultado do experimento para a avaliação do tamanho ideal do *buffer* da proposta BIVoD-*Buffer*. Foram utilizados *buffers* com tamanho de 1, 5, 9, 15 e 20 blocos na realização desse experimento. Sendo considerado uma população igual a 50 usuários. E os resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 23%, 43% e 18% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno e tempo para iniciar a reprodução, respectivamente.

O resultado das métricas número médio de interrupções, Figura 5.15 (a) e 5.15 (d), e tempo médio de retorno, Figura 5.15 (b) e 5.15 (e), demonstram que quanto maior o tamanho do *buffer*, menor é o número de interrupções e também o tempo para o seu retorno. Entretanto, o tempo para iniciar a reprodução, Figura 5.15 (c) e 5.15 (f), aumenta com o aumento no tamanho do *buffer*.

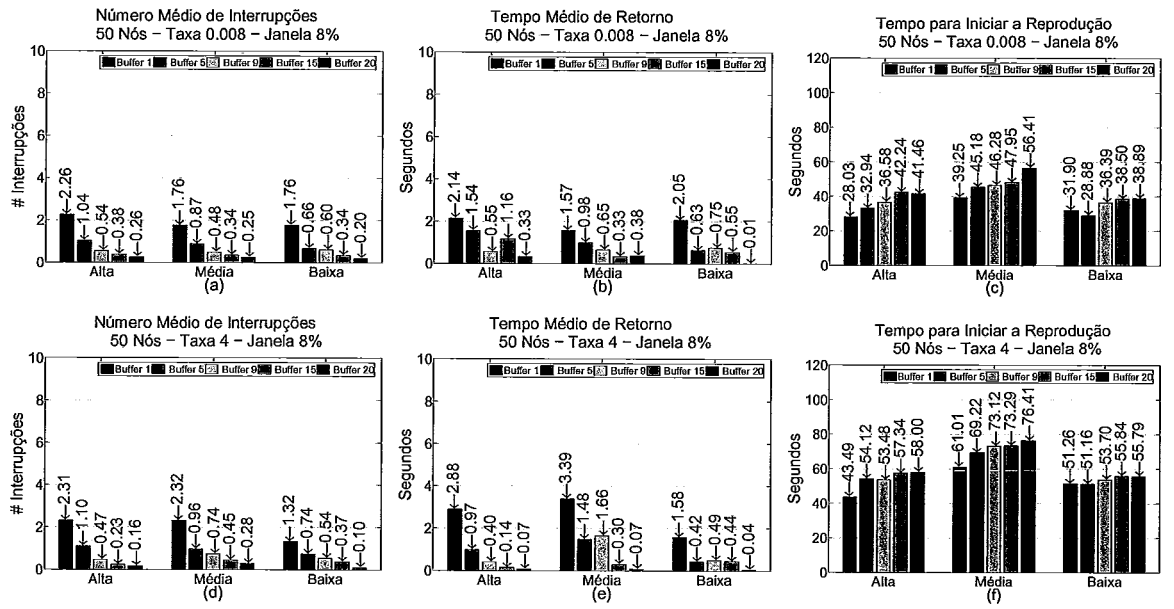

Figura 5.15: Avaliação do impacto na variação do tamanho do *buffer* da proposta BIVoD-*Buffer*.

O aumento no tamanho do *buffer* tende a diminuir o número de interrupções, principalmente as interrupções decorrentes do "toca e para". Já para o tempo médio de retorno, a tendência seria que com o aumento no tamanho do *buffer* ocorresse um aumento do tempo para retornar de uma interrupção, uma vez que o número de blocos a ser recuperado é maior. Porém, devido ao uso da janela de previsão, alguns desses blocos já foram recuperados previamente o que diminui o número de blocos a serem recuperados e por conseguinte o tempo de retorno.

Demais métricas, como taxa de *download*, taxa de *upload* e tempo de *download* apresentaram um comportamento uniforme, não tendo diferença superior a 5% nos valores alcançados.

Utilizar um *buffer* pequeno, com tamanho igual a 1, pode acarretar um número grande de interrupções e um tempo maior para seu retorno, ao passo que o início da reprodução é mais rápido. Porém, um tamanho igual a 15 ou 20 apesar de apresentar poucas interrupções e se recuperar rapidamente delas, apresenta um elevado tempo para iniciar a reprodução. Um valor intermediário, como 5 ou 9 se mostra uma escolha mais adequada, por apresentar um equilíbrio maior entre as métricas

número de interrupções, o tempo de retorno e o tempo para iniciar a reprodução. Porém, uma comparação entre esses dois tamanhos mostra uma pequena vantagem do tamanho 9 nas métricas número médio de interrupções e tempo médio de retorno para as cargas alta e média. Enquanto que o tamanho 5 apresenta o menor tempo para iniciar a reprodução para todas as cargas e o melhor resultado para as demais métricas para carga baixa. Portanto, o tamanho 5 por apresentar o melhor *trade-off* entre número de interrupções/tempo de retorno e tempo para iniciar a reprodução foi o escolhido para ser utilizando nos demais experimentos da proposta BIVoD-*Buffer*.

Avaliação do tamanho da janela de previsão

Para as proposta BIVoD e BIVoD-*Buffer* que utilizam a janela de previsão, convencionou-se que o tamanho dessa janela seria igual ao tamanho médio de segmento (L) da carga, descrito na Seção 5.3, ou seja, o intervalo de tempo que o usuário permaneceu assistindo ao vídeo antes de realizar qualquer tipo de ação. Entretanto, observou-se que esta métrica apresenta uma grande variabilidade, como pode ser verificado na Tabela 5.2, pelos valores do seu desvio padrão. Sendo assim, realizou-se um experimento onde o tamanho da janela de previsão é igual a $L + STD$, ou seja, o tamanho médio de segmento mais o seu desvio padrão. Sendo considerado uma população igual a 50 usuários. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 15%, 23% e 2% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno e taxa de *download*, respectivamente.

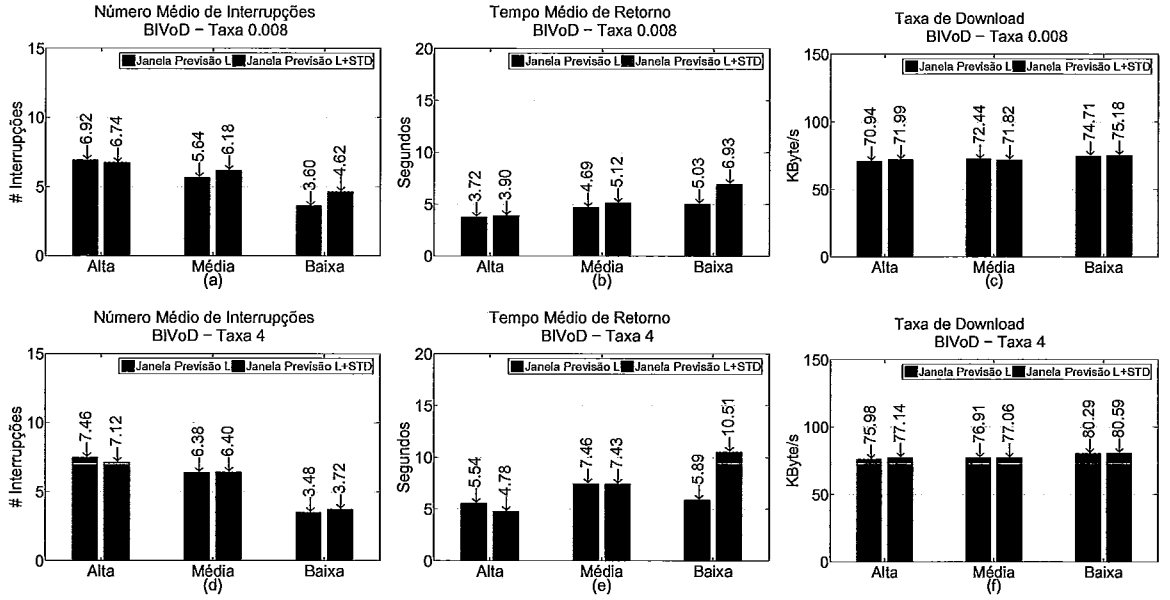

Figura 5.16: Avaliação do tamanho da janela de previsão para a proposta BIVoD.

Para a proposta BIVoD, Figura 5.16, não foi observada uma diferença significativa entre os valores encontrados para as métricas número médio de interrupções, taxa de *download* e tempo médio de retorno, exceto para essa última métrica no cenário com carga baixa, onde ocorreu uma diferença de aproximadamente 44% entre os tempos encontrados. Este comportamento pode ser atribuído ao tamanho muito grande da janela de previsão, neste caso igual a 520 blocos. Com este aumento considerável na janela de previsão o número de blocos que podem ser recuperados aumenta, o que também explica a alta taxa de *download* alcançada nesse cenário, porém isto acaba aumentando o tempo de retorno quando ocorre uma interrupção pois, os blocos recuperados estão mais dispersos dentro da janela de previsão e este fato aliado a raridade igual de muitos deles, torna a recuperação praticamente aleatória.

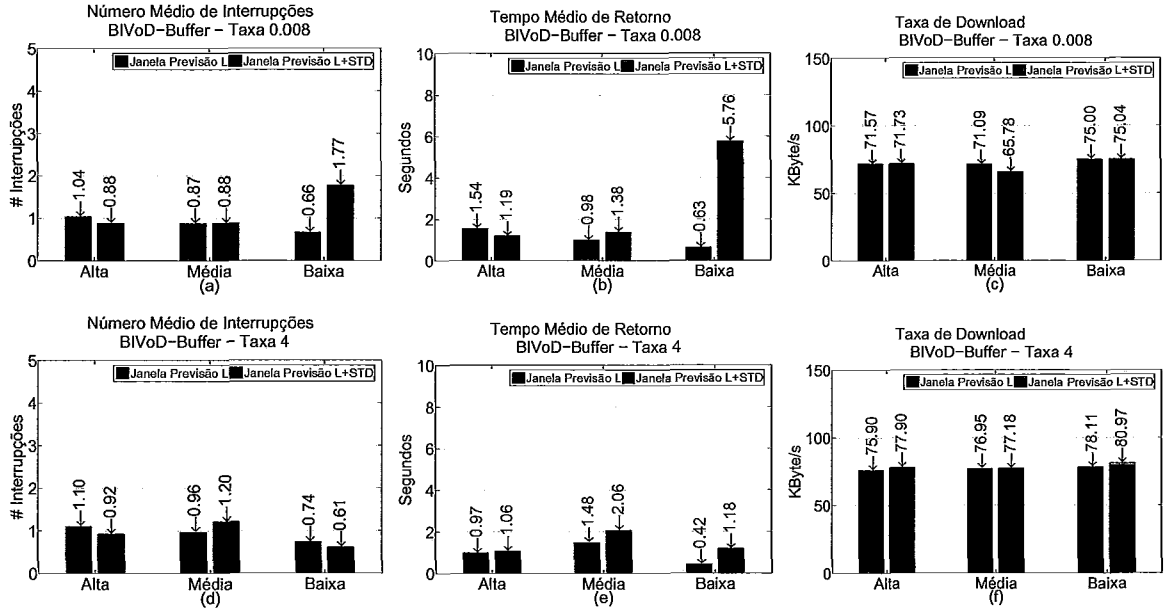

Figura 5.17: Avaliação do tamanho da janela de previsão para a proposta BIVoD-*Buffer*.

A proposta BIVoD-*Buffer*, Figura 5.17, apresentou resultado semelhante ao da proposta BIVoD, Figura 5.16. Porém para o cenário com carga baixa, além do aumento do tempo médio de retorno para a janela de previsão com tamanho $L + STD$, houve aumento também no número de interrupções. Este aumento pode ser verificado no cenário carga baixa e taxa de chegada $\lambda = 4$, que apresenta 63% mais interrupções quando o tamanho é $L + STD$, isso ocorre devido ao comportamento de "toca e para", o que não aconteceu com os demais cenários.

Esse experimento mostrou que as propostas BIVoD e BIVoD-*Buffer* podem ter seu desempenho afetado quando utilizada uma janela de previsão muito grande, como foi o caso do cenário para carga baixa. Porém, variações no tamanho da janela de previsão que não ultrapassem 8% do tamanho total do vídeo não tem grande influência no desempenho dessas propostas, como ocorreu com os cenários para carga alta e média. Ficou claro que a utilização do tamanho da janela de previsão igual a L é uma escolha adequada.

5.4.4 Avaliação comparativa entre as propostas

O objetivo de fazer essa primeira análise comparativa entre todas as propostas é eleger a melhor proposta da literatura dentre as estudadas nesse trabalho, *BiToS*, de *Zhou-Chiu-Lui* e de *Shah-Pâris*, para serem realizados novos experimentos comparativos entre esta proposta selecionada da literatura e as duas novas propostas, BIVoD e BIVoD-*Buffer*.

Na realização desses experimentos foram utilizados os valores para os parâmetros encontrados no experimento anterior. São eles: o valor para o tamanho da janela de *playback* das propostas BIVoD e BIVoD-*Buffer* igual a 8%, o valor do tamanho do conjunto de alta prioridade das propostas *BiToS* e de *Zhou-Chiu-Lui*, também igual a 8% e da janela deslizante da proposta de *Shah-Pâris* com o mesmo valor. Além disso para as propostas *BiToS* e de *Zhou-Chiu-Lui* também foi analisado o valor ideal para o parâmetro p e convencionou-se $p = 0.80$. E para a proposta BIVoD-*Buffer* o tamanho do *buffer*, que convencionou-se em 5 segundos.

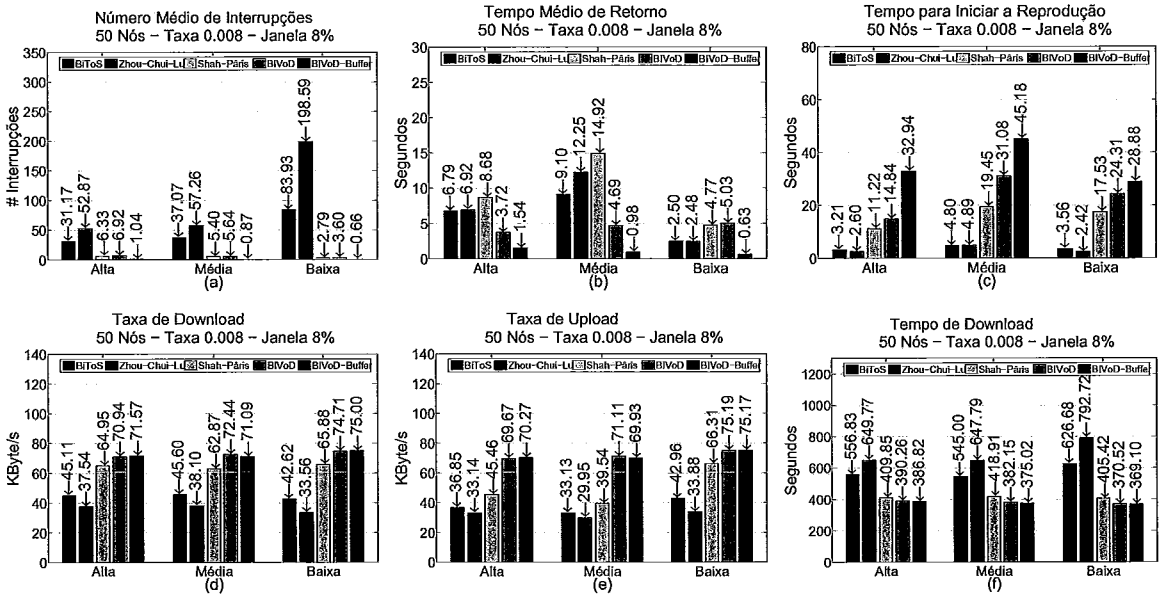

Figura 5.18: Avaliação comparativa entre todas as propostas, 50 usuários e $\lambda = 0.008$ usuários/segundo.

Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 12%, 28%, 39%, 4%, 6% e 3% em torno do valor da média

para as métricas número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*, respectivamente.

Nas Figuras 5.18 (a) e 5.19 (a) são apresentados os resultados para a métrica número médio de interrupções e pode-se observar que a proposta *Zhou-Chiu-Lui* apresenta o pior desempenho em todos os cenários, principalmente para carga baixa, e seguindo a mesma tendência de desempenho tem-se a proposta *BiToS* que também apresenta um elevado número de interrupções, enquanto que as demais propostas não chegam a 8 interrupções em média. As razões para a proposta de *Zhou-Chiu-Lui* apresentar um desempenho tão ruim são devido a: utilização da política de seleção de blocos sequencial no conjunto de alta prioridade, além ser uma política pouco eficiente, ela colabora para a ocorrência do problema de "toca e para". Fazendo uma comparação com a proposta *BiToS* que é muito semelhante, mas utiliza a seleção do bloco mais raro no conjunto de alta prioridade, foi observado que o seu desempenho é aproximadamente 50% superior ao da proposta *Zhou-Chiu-Lui*. Outro fator que influencia, tanto a proposta *Zhou-Chiu-Lui* como também a proposta *BiToS*, é a classificação dos blocos em dois conjuntos com prioridades diferentes. Os blocos do conjunto de baixa prioridade são selecionados baseando-se no critério do bloco mais raro o que acaba acarretando em uma baixa qualidade de reprodução. Por outro lado, quando se recuperaram blocos distantes do ponto de reprodução utilizando algum tipo de critério que privilegie a qualidade da reprodução, como é o caso das propostas BIVoD e BIVoD-*Buffer* que utilizam a janela de previsão, os resultados podem ser muito satisfatórios, como os apresentados. Por fim, a proposta de *Shah-Pâris* mostrou-se competitiva com relação a métrica número médio de interrupções.

A métrica tempo médio de retorno tem seus resultados apresentados nas Figuras 5.18 (b) e 5.19 (b). Esta métrica apresenta bom desempenho para as propostas *BiToS* e de *Zhou-Chiu-Lui* em todos os cenários. Esse comportamento para estas propostas já era esperado, principalmente para a proposta de *Zhou-Chiu-Lui* que recupera sequencialmente os blocos do conjunto de alta prioridade. Com esta política o tempo para a recuperação do bloco ausente é bastante curto. Da mesma forma,

existe uma característica particular da proposta *BiToS*, que também ajuda na recuperação das interrupções muito rapidamente. Durante a recuperação dos blocos do conjunto de alta prioridade, se dois blocos apresentarem o mesmo número de cópias, aquele que estiver mais próximo de ser reproduzido será o escolhido. Por outro lado, na proposta de *Shah-Pâris* se ocorrer situação semelhante a escolha será totalmente aleatória. Já para as propostas BIVoD e BIVoD-*Buffer* seu bom desempenho para esta métrica pode ser creditado a utilização da janela de previsão, a recuperação desses blocos auxilia na redução do tempo de espera, principalmente após um salto, onde poderá ocorrer o deslocamento ou não da janela de *playback*, e como parte dos blocos dessa nova janela já foram recuperados pela janela de previsão, a tendência é que a recuperação dos blocos ausentes seja mais rápida.

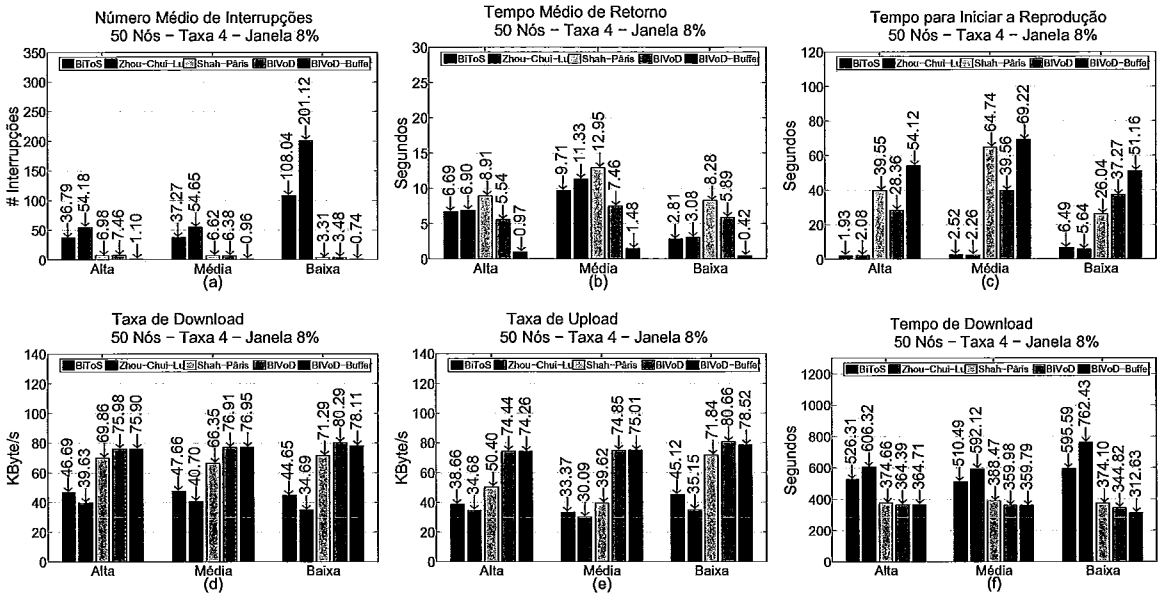

Figura 5.19: Avaliação comparativa entre todas as propostas, 50 usuários e $\lambda = 4$ usuários/segundo.

O desempenho das propostas para a métrica tempo para iniciar a reprodução são diretamente influenciadas pela forma como são recuperados os blocos. Os resultados dessa métrica são apresentados nas Figuras 5.18 (c) e 5.19 (c), e claramente as propostas BIVoD, BIVoD-Buffer e de *Shah-Pâris* apresentam os maiores tempos para iniciarem a reprodução. Por outro lado, as propostas *BiToS* e de *Zhou-Chiu-Lui* são as que iniciam mais rapidamente a reprodução do vídeo. A proposta *Zhou-Chiu-Lui*

recupera os blocos do conjunto de alta prioridade sequencialmente, o que acarretará em um bom desempenho para essa métrica. Já na proposta *BiToS*, no início da recuperação do vídeo, o número de cópias dos blocos deve ser igual, o que forçará a recuperação dos primeiros blocos, caracterizando assim uma "pseudo" recuperação sequencial, o que será muito bom para o desempenho dessa métrica. As demais propostas por utilizarem a política de seleção do bloco mais raro, inicialmente poderão ter um comportamento totalmente aleatório, o que poderá influenciar negativamente no desempenho dessa métrica.

O desempenho das propostas para as métricas taxa de *download*, Figuras 5.18 (d) e 5.19 (d), taxa de *upload*, Figuras 5.18 (e) e 5.19 (e), e tempo de *download*, Figuras 5.18 (f) e 5.19 (f), sofrem forte influência das políticas de classificação e recuperação de blocos, isso é evidente quando analisado o desempenho das propostas *BiToS* e de *Zhou-Chiu-Lui* que apresentaram as menores taxas, enquanto que as demais propostas apresentam resultados muito superiores.

As propostas *BiToS* e de *Zhou-Chiu-Lui* apresentaram algum resultado competitivo somente para as métricas tempo médio de retorno e tempo para iniciar a reprodução, para as demais métricas, principalmente para o número médio de interrupções, os seus desempenhos foram extremamente inferiores aos das outras propostas. A qualidade da reprodução fica consideravelmente prejudicada devido ao excessivo número de interrupções, mesmo que a sua recuperação seja muito rápida. Além disso, essas duas propostas se mostram extremamente deficientes quanto a utilização da banda dos nós, apresentado taxas em até 50% menores as alcançadas pelas propostas BIVoD e BIVoD-*Buffer*.

5.4.5 Avaliação comparativa entre as propostas BIVoD, BIVoD-*Buffer* e a de *Shah-Pâris*

Os experimentos anteriores mostraram que a proposta de *Shah-Pâris* é a mais competitiva entre as propostas da literatura. Portanto essa proposta será utilizada para uma análise comparativa mais detalhada com as duas novas propostas, BIVoD

e BIVoD-*Buffer*.

Os próximos experimentos têm o objetivo de avaliar o desempenho dessas propostas, considerando o aumento no número de usuários no *swarm*; e o aumento no tamanho do vídeo, com características de alta interatividade. Avaliar também a robustez das propostas BIVoD e BIVoD-*Buffer* quando os usuários são classificados incorretamente no seu nível de interatividade. E ainda avaliar os efeitos da utilização de um *buffer*, igual ao da proposta BIVoD-*Buffer*, na proposta de *Shah-Pâris*. Outra característica avaliada é a equidade (*fairness*) que as propostas proporcionam aos usuários. Mesmo utilizando cenários onde todos os nós tem a mesma capacidade, um algoritmo pode prover mais ou menos justiça entre os usuários com relação as métricas que indicam a qualidade da reprodução.

Variando a população

Uma importante avaliação é quanto à escalabilidade do algoritmo, ou seja, como o algoritmo se comporta com o aumento do número de usuários do *swarm*. Para isto, foram realizados experimentos com populações com tamanho de 50, 100, 150 e 200 usuários, todos com a carga mista. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 13%, 19%, 31%, 2%, 4% e 2% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*, respectivamente.

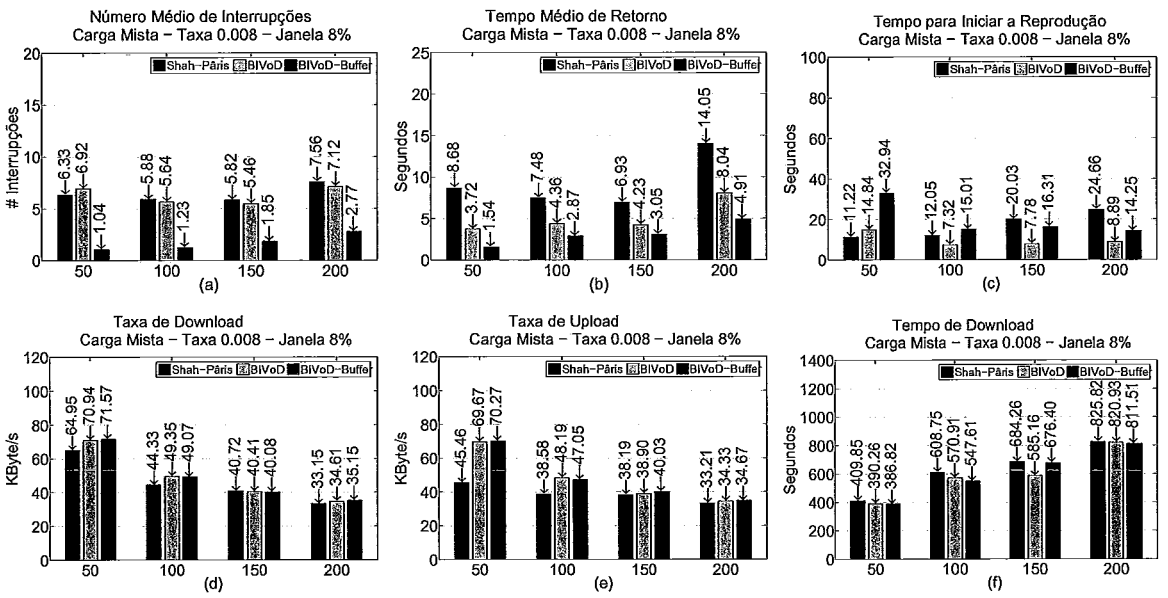

Figura 5.20: Avaliação comparativa entre as propostas de *Shah-Pâris*, BIVoD e BIVoD-*Buffer* com cenário carga mista e $\lambda = 0.008$ usuários/segundo.

As Figuras 5.20 (a) e 5.21 (a) apresentam o resultado deste experimento para a métrica número médio de interrupções, onde a proposta BIVoD-*Buffer* claramente tem o melhor desempenho para ambas as taxas de chegada, enquanto que as propostas de *Shah-Pâris* e BIVoD apresentam um desempenho muito semelhante, não tendo diferença significativa entre elas. Uma tendência observada é o aumento no número de interrupções devido ao aumento no número de usuários. Isso se deve principalmente ao cenário utilizado para o experimento, onde só existe um nó *seed* e os nós *leechers* partem do *swarm*, assim que terminam de assistir ao vídeo. Este tipo de comportamento era esperado pois, com o aumento do número de usuários, existe um aumento no número de requisições e por conseguinte no número de interrupções.

Para a métrica tempo médio de retorno, Figuras 5.20 (b) e 5.21 (b), a proposta de *Shah-Pâris* somente conseguiu apresentar resultados próximos ao da proposta BIVoD para os cenários com a taxa de chegada igual a $\lambda = 4$ e número de nós no *swarm* igual a 100 e 150, do contrário chegou a apresentar tempo até 55% superior ao da proposta BIVoD. Já a proposta BIVoD-*Buffer* continua a apresentar os melhores resultados também para essa métrica, porém, a diferença para a BIVoD diminuiu, mostrando que apesar de a proposta BIVoD ter um número médio de interrupções

superior, os usuários conseguem se recuperar rapidamente das interrupções. Portanto, a utilização de um *buffer* e também da previsão das futuras ações do usuário fazem com que as propostas BIVoD e BIVoD-*Buffer* apresentem uma rápida recuperação quando ocorre uma interrupção. Além de diminuir o número de interrupções, a previsão auxilia na disseminação do vídeo dentro do *swarm*, uma vez que a busca por blocos não fica restrita a apenas uma parte do vídeo, como é o caso da proposta de *Shah-Pâris* que recupera somente blocos da janela deslizante. A busca por blocos em outra parte do vídeo, no caso a janela de previsão, auxilia na disseminação dos blocos pelo *swarm*.

A proposta BIVoD obteve os melhores tempos para a métrica tempo para iniciar a reprodução (Figuras 5.20 (c) e 5.21 (c)), enquanto que as propostas de *Shah-Pâris* e BIVoD-*Buffer* obtiveram tempos mais elevados. Essa métrica é fortemente afetada pelo algoritmo de recuperação de blocos, principalmente nos dois casos descritos a seguir. O primeiro caso ocorre quando os primeiros usuários se conectam no *swarm*. Neste caso o número de cópias dos blocos é igual, ou seja, os blocos têm a mesma raridade, portanto a recuperação é totalmente aleatória, porque não existe nenhuma prioridade na recuperação dos blocos iniciais, como por exemplo na proposta *BiToS*. A segunda situação ocorre quando os primeiros blocos do vídeo têm um número de cópias maior que os blocos mais distantes da parte inicial. Esta situação pode ocorrer em um certo tempo após a criação do *swarm* porque, neste caso, os primeiros usuários a chegarem no *swarm* já recuperaram os blocos iniciais, elevando o seu número de cópias e assim somente os blocos mais distantes serão recuperados inicialmente por serem os mais raros. Nas duas situações descritas acima, o tempo para iniciar a reprodução pode ser elevado.

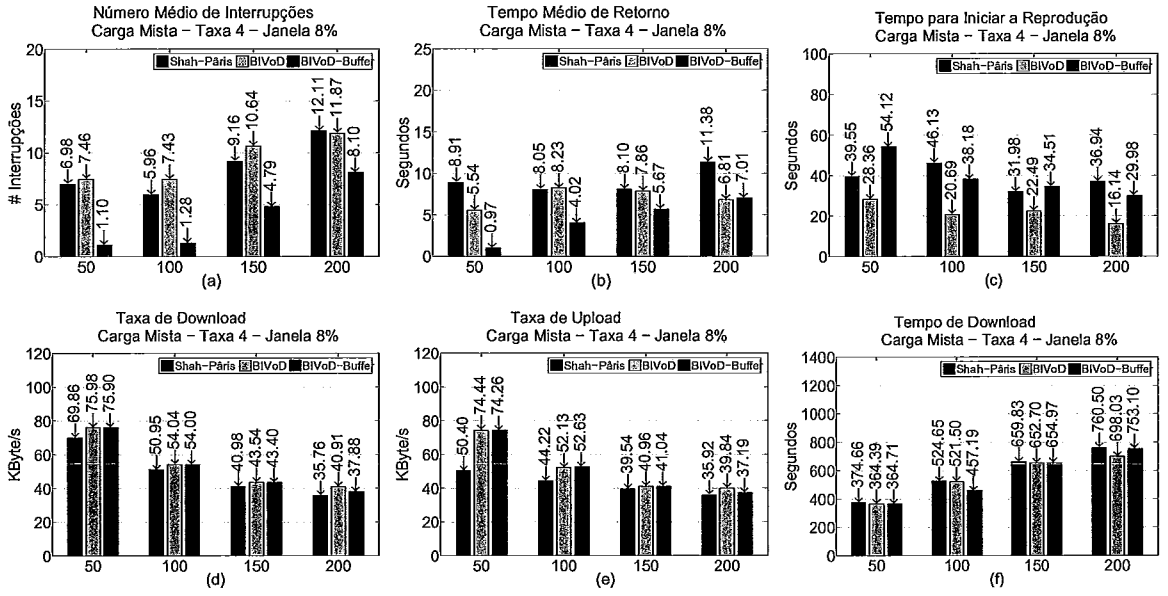

Figura 5.21: Avaliação comparativa entre as propostas de *Shah-Pâris*, BIVoD e BIVoD-*Buffer* com cenário carga mista e $\lambda = 4$ usuários/segundo.

O desempenho da proposta de *Shah-Pâris* é fortemente afetado pelas duas situações descritas acima, que podem ocorrer dependendo do momento da entrada do usuário no *swarm*, ou seja, os primeiros irão ter problemas devido aos blocos possuírem a mesma raridade, enquanto que os intermediários e últimos serão afetados pela raridade menor dos blocos iniciais. A explicação para o melhor desempenho da proposta BIVoD é o fato da busca por blocos não se limitar à janela de *playback*, como na proposta de *Shah-Pâris*. A busca é feita também na janela de previsão, o que distribui mais uniformemente a raridade dos blocos. Já a proposta BIVoD-*Buffer* não apresenta o mesmo desempenho da BIVoD pois, um *buffer* deve ser preenchido antes do início da reprodução.

Para as métricas que medem o desempenho com relação ao uso da capacidade do sistema como, taxa de *download*, Figuras 5.20 (d) e 5.21 (d), taxa de *upload*, Figuras 5.20 (e) e 5.21 (e), e tempo de *download*, Figuras 5.20 (f) e 5.21 (f), as propostas BIVoD e BIVoD-*Buffer* apresentaram os melhores resultados em todos os cenários, enquanto que a proposta de *Shah-Pâris* teve um desempenho um pouco abaixo. Nesses experimentos foi observada a tendência de queda das taxas médias de *download* e *upload* com o crescimento da população, como ocorreu no experimento

real realizado no *PlanetLab* (Subseção 5.4.1).

Nessa primeira parte da avaliação comparativa o desempenho das propostas BIVoD e BIVoD-*Buffer* foi superior ao desempenho da proposta de *Shah-Pâris*. Foi possível observar o ganho que a utilização do algoritmo de previsão pode proporcionar ao desempenho da aplicação para a distribuição de vídeo sob demanda com interatividade. Este ganho foi tanto na qualidade de reprodução, quanto na eficiência na recuperação e distribuição do vídeo.

Avaliando o comportamento das propostas para um vídeo longo

Nos experimentos anteriores foi utilizado um vídeo com tamanho de 1800 segundos. Nestes experimentos é considerado vídeos-aula de 4200 segundos, usuários chegando com duas taxas de chegada $\lambda = 4$ e $\lambda = 0.008$ usuários/segundo e com nível de interatividade alto. Estes resultados são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 20%, 29%, 30%, 2%, 3% e 2% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*, respectivamente.

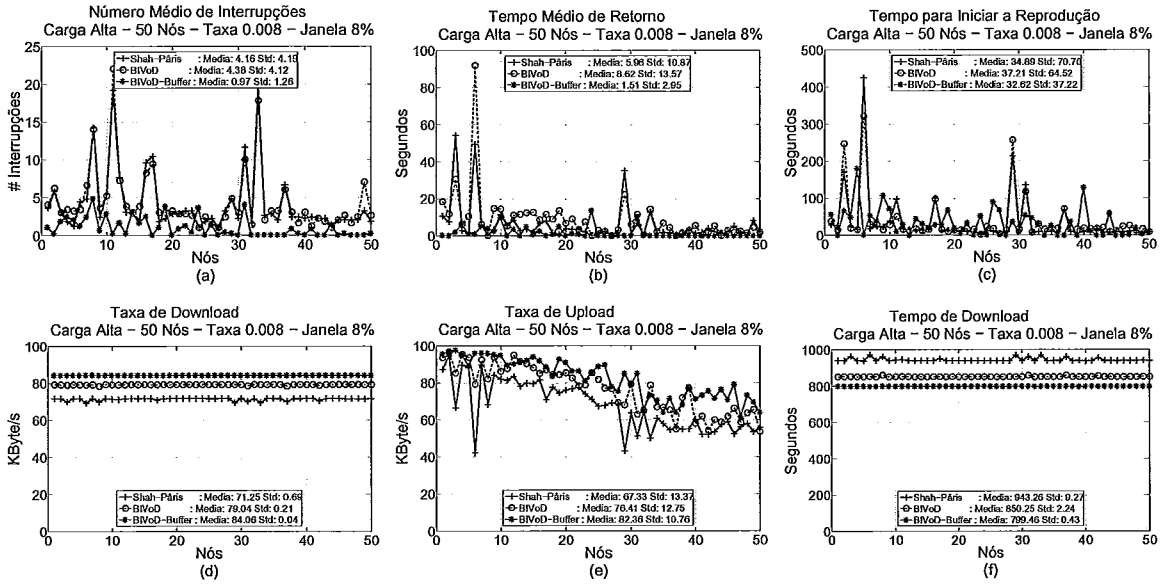

Figura 5.22: Avaliação comparativa entre todas as propostas, com vídeo de 4200 segundos, 50 usuários e $\lambda = 0.008$ usuários/segundo.

Para o cenário com taxa de chegada igual a $\lambda = 0.008$ usuários/segundo, Figura 5.22, a proposta BIVoD-*Buffer* obteve o melhor resultado para todas as métricas e apresentou baixa variabilidade entre o desempenho dos nós, o que comprova a sua equidade. Enquanto que a proposta BIVoD obteve tempos ligeiramente maiores do que os da proposta de *Shah-Pâris*, para as métricas tempo médio de retorno e tempo para iniciar reprodução, mas ainda continua com bom desempenho para a distribuição e recuperação do vídeo, como pode ser observado nas métricas taxa de *download*, *upload* e tempo de *download*.

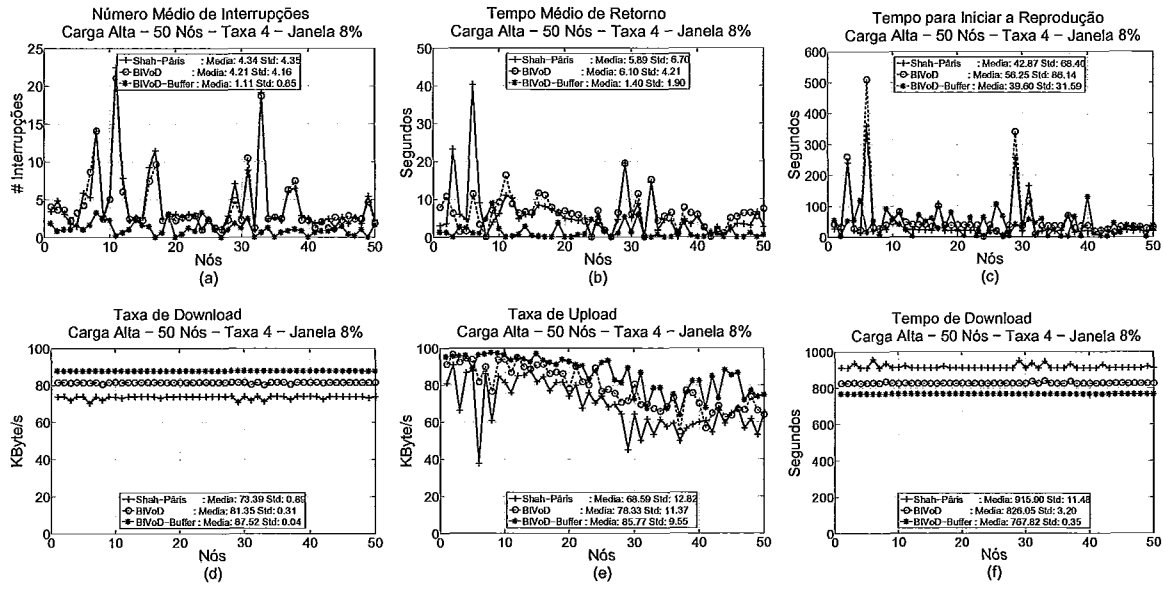

Figura 5.23: Avaliação comparativa entre todas as propostas, com vídeo de 4200 segundos, 50 usuários e $\lambda = 4$ usuários/segundo.

Quando a taxa de chegada é igual a $\lambda = 4$ usuários/segundo, Figura 5.21, o resultado não foi muito diferente do que o para chegada $\lambda = 0.008$ usuários/segundo. A proposta BIVoD obteve um número médio de interrupções ligeiramente melhor que o da proposta de *Shah-Pâris*, mas por outro lado teve o maior tempo para iniciar a reprodução, chegando a 56 segundos em média para o usuário começar a assistir ao vídeo. Este resultado foi fortemente afetado pelo desempenho de 4 nós que apresentaram tempo para iniciar a reprodução muito elevado. Concluindo, a proposta BIVoD-*Buffer* apresentou o melhor desempenho, independentemente da taxa de chegada e também do tamanho do vídeo.

Experimento com usuários classificados incorretamente

Este experimento foi realizado para verificar os efeitos da classificação errada do nível de interatividade do usuário pelo modelo de comportamento do usuário, descrito na Seção 4.2.2. Foram selecionados todos os usuários de alta e baixa interatividade classificados incorretamente pelo modelo como sendo de média interatividade, e foram executados os algoritmos BIVoD e BIVoD-*Buffer* usando um modelo de

usuário de média interatividade. Foram utilizados 50 *logs* de usuários para realizar esse experimento. Esses *logs* serão denominados como carga errada.

Para comparar os resultados da classificação errada com os da classificação correta, foram plotados nas Figuras 5.24 e 5.25 os resultados obtidos previamente para os algoritmos quando todos os usuários são classificados corretamente. Também foram plotados os resultados do algoritmo de *Shah-Pâris* quando este é executado utilizando a carga errada, porém este não faz uso da previsão. O objetivo é verificar se mesmo classificando os usuários incorretamente, as propostas BIVoD e BIVoD-*Buffer* apresentam ou não melhor desempenho que o algoritmo de *Shah-Pâris*, haja vista que nos experimentos anteriores os usuários são classificados corretamente.

Os resultados obtidos são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 16%, 25%, 23%, 3%, 3% e 3% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*, respectivamente.

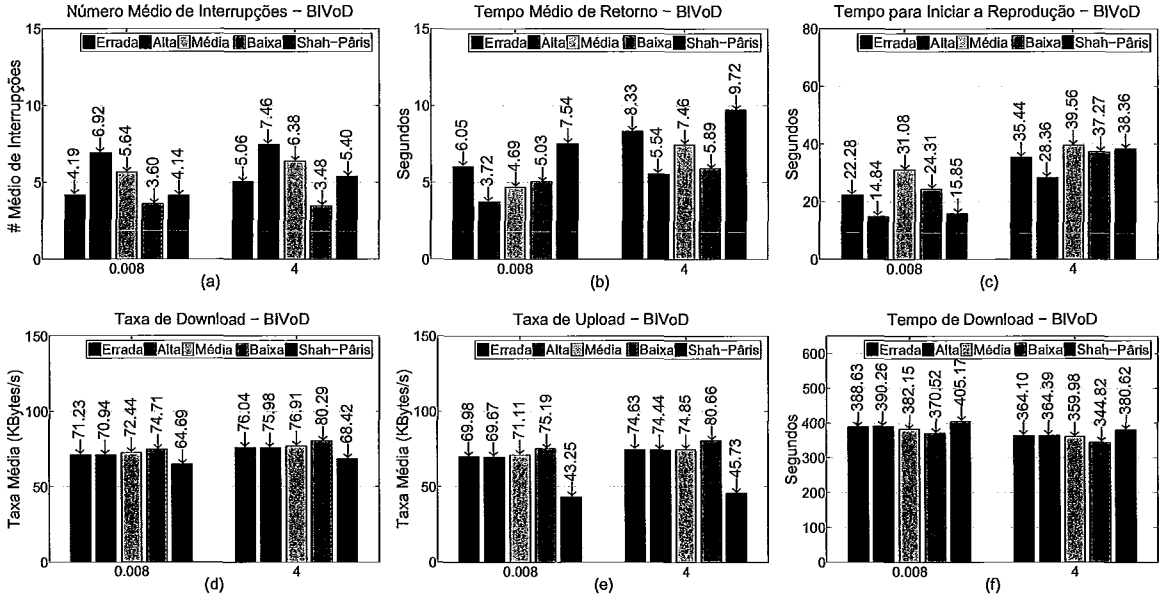

Figura 5.24: Avaliação do desempenho da proposta BIVoD quando os usuários são classificados erroneamente.

O desempenho das propostas BIVoD e BIVoD-*Buffer* quando utilizam a carga

errada mostra-se satisfatório para todas as métricas. O pior resultado obtido foi para a métrica tempo médio de retorno, Figuras 5.24 (b) e 5.25 (b). Os resultados mostram que o tempo médio de retorno para usuários classificados incorretamente, pode ser até 45% superior ao tempo obtido quando o usuário é classificado corretamente. Isto confirma que a classificação dos usuários incorretamente e consequentemente a utilização da previsão imprecisa influencia diretamente no desempenho dessa métrica.

Os resultados para a métrica número médio de interrupções, Figuras 5.24 (a) e 5.25 (a), mostram que usuários de alta e baixa interatividade classificados incorretamente teriam uma pequena vantagem. O mesmo comportamento é observado a partir dos resultados da métrica tempo para iniciar a reprodução. Como a diferença entre os resultados obtidos com a classificação correta e a classificação incorreta é pequena, não se pode afirmar que a classificação incorreta terá uma influência significativa no valor dessas métricas.

As métricas taxa de *download*, Figuras 5.24 (d) e 5.25 (d), taxa de *upload*, Figuras 5.24 (e) e 5.25 (e), e tempo de *download*, Figuras 5.24 (f) e 5.25 (f), não apresentam diferença significativa nos resultados obtidos. O que mostra que a classificação errada não interfere no desempenho da proposta quanto à eficiência no uso dos recursos dos nós para a distribuição do vídeo.

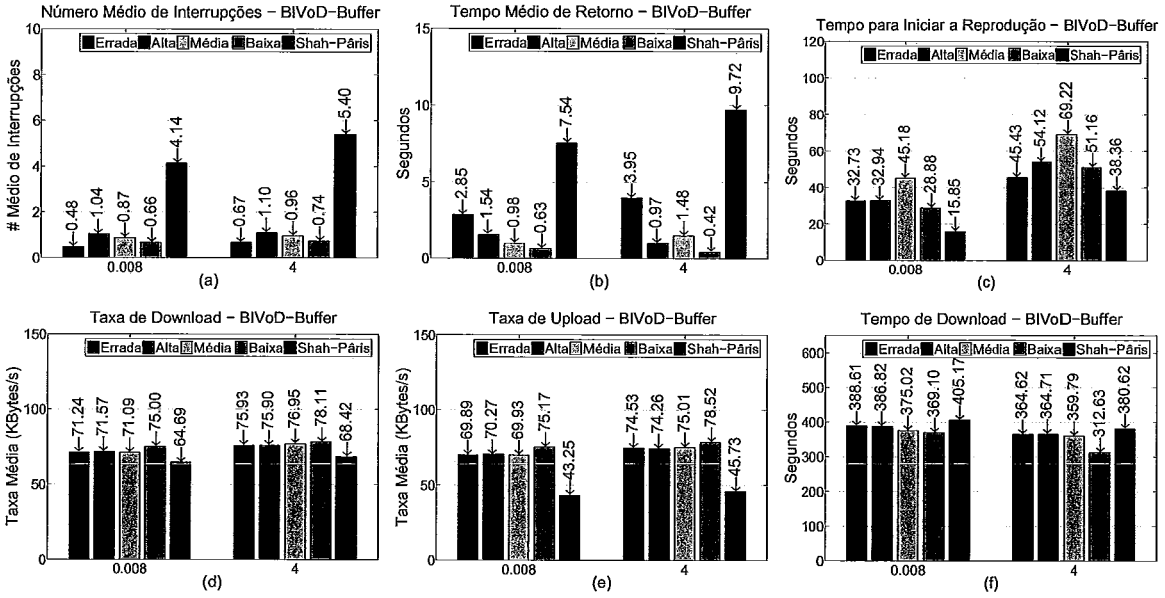

Figura 5.25: Avaliação do desempenho da proposta BIVoD-*Buffer* quando os usuários são classificados erroneamente.

Na Figura 5.24 pode-se observar a comparação de desempenho entre as propostas BIVoD e de *Shah-Pâris* ambas utilizando a carga errada. Para a métrica número médio de interrupções (Figura 5.24 (a)) o desempenho de ambas é muito semelhante. Porém, para a métrica tempo médio de retorno (Figura 5.24 (b)) a proposta BIVoD apresenta um tempo até 20% menor que a proposta de *Shah-Pâris*. Esse desempenho é o resultado da utilização da previsão, que mesmo pouco precisa, consegue impactar na diminuição do tempo de retorno de uma interrupção. Para as métricas taxa de *download* (Figura 5.24 (d)), taxa de *upload* (Figura 5.24 (e)) e tempo de *download* (Figura 5.24 (f)), a proposta BIVoD continua a apresentar valores até 40% melhores que os da proposta de *Shah-Pâris*.

A Figura 5.25 apresenta o resultado da comparação entre as propostas BIVoD-*Buffer* e de *Shah-Pâris* ambas para carga errada. Para as métricas número médio de interrupções (Figura 5.25 (a)), tempo médio de retorno (Figura 5.25 (b)), taxa de *download* (Figura 5.25 (d)), taxa de *upload* (Figura 5.25 (e)) e tempo de *download* (Figura 5.25 (f)) fica evidente a superioridade da proposta BIVoD-*Buffer*, com diferenças de até 90% nos valores alcançados por estas métricas. Somente para a métrica tempo para iniciar a reprodução (Figura 5.25 (c)) é que a proposta de *Shah-Pâris*

consegue ter um desempenho melhor que a proposta BIVoD-*Buffer*, apresentado um tempo até 52% menor. Este resultado já era esperado, haja vista que em todos os experimentos realizados com a proposta BIVoD-*Buffer* o seu desempenho para a métrica tempo para iniciar a reprodução era o maior dentre todas as propostas analisadas, fato este que ocorre devido à necessidade de preencher o *buffer* antes de iniciar a reprodução.

Pode-se concluir desse experimento que o maior impacto da classificação errada de um usuário nas propostas BIVoD e BIVoD-*Buffer* é na métrica tempo médio de retorno. Este resultado era esperado pois esta métrica é a que mais se beneficia do uso do modelo de comportamento do usuário, isto é, a utilização da previsão auxilia na diminuição do tempo de retorno, isto porque ela consegue recuperar previamente blocos que no futuro serão necessários ao usuário. Os resultados mostram também que o uso dos algoritmos propostos mesmo com a classificação errada do usuário é mais vantajoso que a proposta de *Shah-Pâris*.

Avaliação da proposta de Shah-Pâris com *buffer*

Este experimento foi realizado com o objetivo de analisar a influência no desempenho da proposta de *Shah-Pâris* quando utilizado um *buffer* dentro da janela deslizante. Esse *buffer* apresenta as mesmas características e funções que o utilizado pela proposta BIVoD-*Buffer*. Os cenários utilizados nesse experimento foram de 50 usuários de alta interatividade, com taxa de chegada igual a $\lambda = 4$ e 0.008 usuários/segundo e se variou o tamanho do *buffer* em 5, 15 e 20 blocos.

Os resultados obtidos são a média de 10 execuções, tendo um intervalo de confiança de 90% variando no máximo 16%, 25%, 23%, 3%, 3% e 3% em torno do valor da média para as métricas número médio de interrupções, tempo médio de retorno, tempo para iniciar a reprodução, taxa de *download*, taxa de *upload* e tempo de *download*, respectivamente.

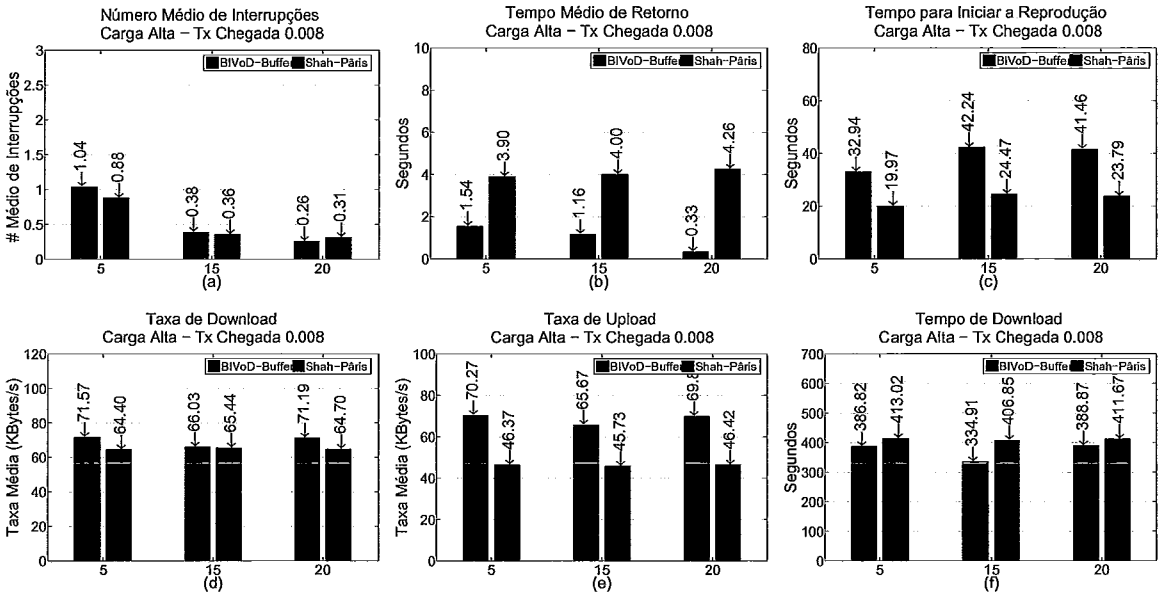

Figura 5.26: Avaliação do desempenho da proposta de *Shah-Pâris* utilizando um *buffer* com $\lambda = 0.008$ usuários/segundo.

Para a métrica número médio de interrupções, Figuras 5.26 (a) e 5.27 (a), não observa-se diferença significativa no desempenho entre as propostas de *Shah-Pâris* com *buffer* e BIVoD-*Buffer* independente do tamanho do *buffer*. Este comportamento já era esperado, haja vista que também ocorreu na comparação entre as propostas BIVoD e de *Shah-Pâris* sem *buffer*. Para esta métrica a influência da utilização do modelo de previsão pelas propostas BIVoD e BIVoD-*Buffer* é muito incipiente. Entretanto, para a métrica tempo médio de retorno, Figuras 5.26 (b) e 5.27 (b), o desempenho da proposta BIVoD-*Buffer* é muito superior ao da proposta de *Shah-Pâris* com *buffer*, que apresentou um tempo 96% menor que os alcançados pela proposta de *Shah-Pâris*. Este desempenho é devido a utilização do modelo de previsão pela proposta BIVoD-*Buffer*, este comportamento também foi observado nos experimentos anteriores entre as propostas BIVoD e *Shah-Pâris* sem *buffer*. Portanto, fica claro que o modelo de previsão influência fortemente o desempenho da métrica tempo médio de retorno.

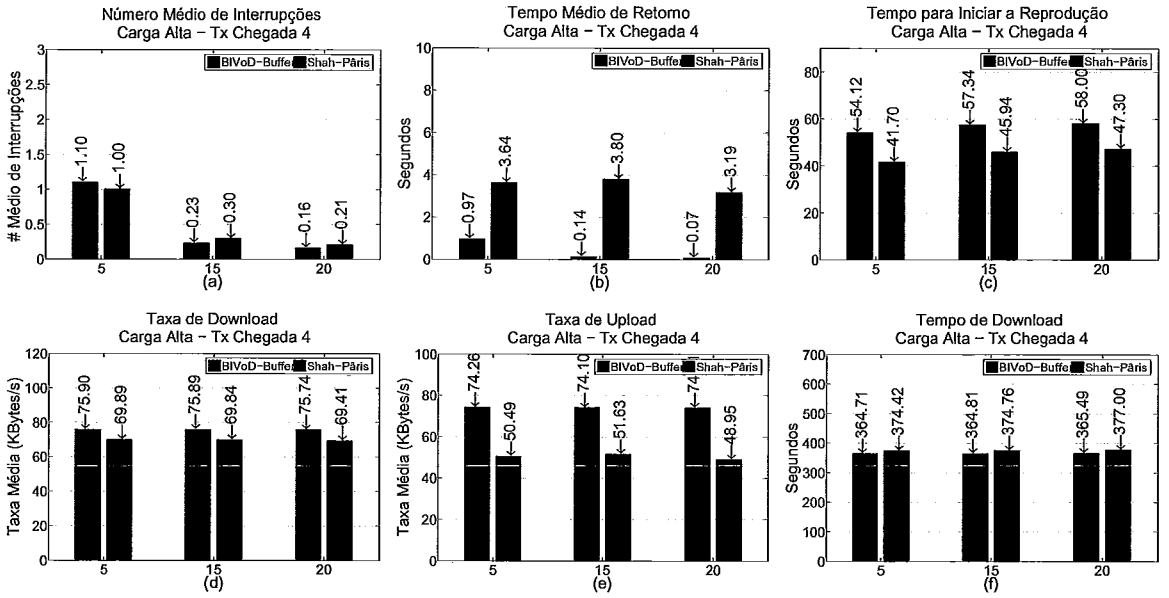

Figura 5.27: Avaliação do desempenho da proposta de *Shah-Pâris* utilizando um *buffer* com $\lambda = 4$ usuários/segundo.

Para a métrica tempo para iniciar a reprodução, Figuras 5.26 (c) e 5.27 (c), a proposta de *Shah-Pâris* se mostrou mais eficiente que a proposta BIVoD-*Buffer*. Esse desempenho da proposta BIVoD-*Buffer* pode ser creditado a alta variabilidade dessa métrica, causada pela recuperação alternada em duas janelas (*playback* e previsão) de blocos distintos. Entretanto, para as métricas taxa de *download*, Figuras 5.26 (d) e 5.27 (d), taxa de *upload*, Figuras 5.26 (e) e 5.27 (e), e tempo de *download*, Figuras 5.26 (f) e 5.27 (f), a proposta BIVoD-*Buffer* apresenta um ganho de 12%, 50% e 10% respectivamente, em relação a proposta de *Shah-Pâris* utilizando *buffer*. Este desempenho demonstra que a proposta de *Shah-Pâris*, mesmo utilizando *buffer*, ainda sofre com o baixo desempenho de alguns usuários, por apresentarem elevado número de saltos com distância muito grande entre eles, enquanto que a proposta BIVoD-*Buffer* não sofre tanto com esses nós, devido a utilização do modelo de previsão, que minimiza esses efeitos no desempenho do sistema.

Avaliando a equidade

Uma importante característica dos algoritmos para a distribuição de dados que utilizam a arquitetura *Peer-to-Peer* é o nível de equidade que o algoritmo mantém entre os nós que participam da recuperação/distribuição de um objeto. Um exemplo é a política *tit-for-tat* utilizada pelo protocolo *BitTorrent*, que tenta manter um alto nível de equidade entre os nós participantes do *swarm*, ao mesmo tempo em que desestimula a participação de nós "egoístas". Além disso, para o caso da distribuição de vídeo sob demanda é importante avaliar se todos os nós estão conseguindo ter o mesmo nível de qualidade na reprodução do vídeo.

Para determinar o grau de equidade de cada proposta foram utilizadas as seguintes medidas: a média ($\bar{x}$); a variância das métricas (σ^2); e a diferença entre o nó que alcançou o maior e o nó que alcançou o menor valor para determinada métrica (d). Estas medidas foram calculadas sobre as métricas: número médio de interrupções, tempo médio de retorno e taxa de *download*. As duas primeiras para avaliar a qualidade da reprodução e a última o desempenho com relação à utilização dos recursos dos nós. Para este experimento foi considerada uma população de 50 nós no *swarm*.

Tabela 5.3: Equidade para número médio de interrupções.

Carga	Algoritmo	**Número de Interrupções**		
		$\bar{x}$	σ^2	d
Alta	*Shah-Pâris*	6.3280	24.7539	16.1000
	BIVoD	6.4380	26.2942	16.3000
	BIVoD-Buffer	1.0880	2.1639	5.9000
Média	*Shah-Pâris*	5.3980	12.5112	15.8000
	BIVoD	5.6440	13.2857	15.2000
	BIVoD-Buffer	0.8720	0.8290	3.2000
Baixa	*Shah-Pâris*	2.7920	1.4938	5.2000
	BIVoD	3.6000	2.3771	7.4000
	BIVoD-Buffer	0.6640	0.5248	2.6000

Na Tabela 5.3 são apresentados os valores da média, da variância e da diferença d para a métrica número médio de interrupções. Pela variância, a proposta BIVoD-*Buffer* mostra-se a com a menor variabilidade em todos os cenários, o que é confirmado por d, com a menor distância entre o nó com valor máximo e o com valor mínimo. Por outro lado, as propostas BIVoD e de *Shah-Pâris* apresentam valores muito próximos para $\bar{x}$, σ^2 e d, tendo a proposta de *Shah-Pâris* uma pequena vantagem. Comparando as três propostas pode-se concluir que a que oferece a maior equidade entre os usuários para a métrica número médio de interrupções é a proposta BIVoD-*Buffer*.

Tabela 5.4: Equidade para tempo médio de retorno.

Carga	Algoritmo	**Tempo Médio de Retorno (segundos)**		
		$\bar{x}$	σ^2	d
Alta	*Shah-Pâris*	8.6845	244.7302	77.3542
	BIVoD	3.8936	8.5564	13.3812
	BIVoD-Buffer	3.1974	11.8429	13.2975
Média	*Shah-Pâris*	14.9167	447.2422	91.8489
	BIVoD	4.6895	14.5362	17.0427
	BIVoD-Buffer	0.9828	2.8140	7.8960
Baixa	*Shah-Pâris*	4.7738	28.5057	22.6871
	BIVoD	5.0327	8.1829	13.4389
	BIVoD-Buffer	0.6294	1.2920	5.9867

Para a métrica tempo médio de retorno, Tabela 5.4, a proposta de *Shah-Pâris* apresentou a maior variância e a maior diferença (d), o que mostra a menor equidade entre os nós quando se trata do tempo necessário para retornar de uma interrupção. Por outro lado, as propostas BIVoD e BIVoD-*Buffer* apresentaram valores para a variância e para d bem menores que os apresentados pela proposta de *Shah-Pâris*, o que demonstra a sua forte tendência em promover um equilíbrio no desempenho dos usuários, isto pode ser creditado a utilização da previsão. A previsão afeta fortemente o desempenho das propostas BIVoD e BIVoD-*Buffer* para a métrica tempo médio de retorno, apresentando sempre tempos muito inferiores aos alcançados pela proposta de *Shah-Pâris* e proporcionando maior equidade entre os usuários.

Tabela 5.5: Equidade para taxa de *download*.

Carga	Algoritmo	**Taxa de *Download* (K Bytes/s)**		
		$\bar{x}$	σ^2	d
Alta	Shah-Pâris	66.2474	115.1405	41.7507
	BIVoD	71.4166	3.7014	11.0006
	BIVoD-Buffer	71.6644	2.5196	8.0444
Média	Shah-Pâris	64.1252	180.6644	45.0483
	BIVoD	73.8928	0.3031	1.9834
	BIVoD-Buffer	73.5746	0.4002	2.3274
Baixa	Shah-Pâris	67.1952	74.6804	41.4553
	BIVoD	76.2083	0.0091	0.4503
	BIVoD-Buffer	76.5045	0.0090	0.5185

A Tabela 5.5 mostra a proposta de *Shah-Pâris* com elevados valores para a variância e para d, para a taxa de *download*. A baixa equidade com relação a esta métrica na proposta de *Shah-Pâris* é o resultado do baixo desempenho de alguns nós, que apresentam elevado número de saltos em um curto intervalo de tempo, sendo esses salto em grandes distâncias. Entretanto, estes mesmos nós não influenciam no desempenho das outras duas propostas devido a utilização da previsão por elas.

Resumindo, a proposta BIVoD-*Buffer* promove a melhor equidade entre os usuários com relação a todas as métricas analisadas. Da mesma forma, a proposta BIVoD também proporciona uma boa equidade entre os nós. O desempenho das propostas BIVoD e BIVoD-*Buffer* é fortemente influenciada pela utilização da previsão, e por isto também promove um elevado nível de equidade entre os usuários. Já a proposta de *Shah-Pâris* apresenta pouca equidade para as métricas tempo médio de retorno e taxa de *download*. O cálculo da equidade para cenários com uma população maior também foi realizado e foi observada a mesma tendência que os resultados apresentados acima.

Capítulo 6

Conclusão e Trabalhos Futuros

6.1 Conclusões

A distribuição de vídeo sob demanda na Internet tem aumentado consideravelmente nos últimos anos. Dados de Julho de 2009 mostram que foram assistidos mais de 21 bilhões de vídeos na Internet, ou seja, um aumento de mais de 80% em relação ao mesmo período de 2008 [70]. Somente o *YouTube* foi responsável por 9 bilhões de visualizações, com uma média de 74 vídeos por pessoa. Esses números reforçam a importância no desenvolvimento de novos protocolos e/ou aplicações, com o objetivo de melhorar a qualidade da reprodução do vídeo distribuído sob demanda, aumentar a escalabilidade dos sistemas de distribuição e, por conseguinte, na diminuição de custos inerentes a esta distribuição. Uma forma de distribuir vídeo sob demanda na Internet, que apresente alta escalabilidade, qualidade e baixo custo, é a arquitetura *Peer-to-Peer* (P2P). O Protocolo *BitTorrent*, desenvolvido para a distribuição de dados, mostra o quanto é eficiente a arquitetura P2P. Porém, ainda se faz necessários estudos para a utilização dessa arquitetura para a distribuição de vídeos sob demanda.

A contribuição fundamental desse trabalho é o desenvolvimento de duas novas propostas: BIVoD e BIVoD-*Buffer*, para a distribuição de vídeo sob demanda com

interatividade.

Inicialmente, foi realizado um estudo das características da arquitetura P2P e do funcionamento do protocolo *BitTorrent*, cujo objetivo foi de identificar as deficiências do protocolo *BitTorrent* e da arquitetura P2P para a distribuição de vídeo sob demanda com interatividade. Após entendimento de toda a estrutura de seu funcionamento, foi desenvolvido um modelo de simulação do protocolo *BitTorrent* utilizando a ferramenta Tangram-II.

Em seguida, um abrangente estudo bibliográfico foi realizado sobre as propostas para a distribuição de vídeo sob demanda, que utilizam a arquitetura P2P, mais especificamente o protocolo *BitTorrent*. O objetivo foi a identificação das propostas mais relevantes para serem utilizadas em uma avaliação comparativa de desempenho com as duas novas propostas. Três propostas foram as escolhidas: *BiToS* [2], de *Zhou-Chiu-Lui* [23] e de *Shah-Pâris* [3]. Todas utilizam o protocolo *BitTorrent* com algumas adaptações para a distribuição de vídeo sob demanda.

O desenvolvimento das propostas BIVoD e BIVoD-*Buffer* está fundamentado em dois conceitos principais. O primeiro é a utilização do protocolo *BitTorrent* para a distribuição de vídeo sob demanda com interatividade. Na literatura poucos trabalhos abordam o problema da interatividade, utilizando a arquitetura P2P, como base e os poucos que fazem, não utilizam o protocolo *BitTorrent*. Estas duas propostas podem ser consideradas as primeiras neste sentido. O segundo conceito é a utilização do modelo de emulação do comportamento do usuário, com o objetivo de recuperar os blocos necessários no futuro pelo usuário devido a saltos que ele possa vir a realizar.

Por fim, uma série de experimentos foram realizados para avaliar o desempenho das propostas, com as seguintes conclusões:

- O protocolo *BitTorrent*, sem adaptações, mostrou-se inviável para a distribuição de vídeo sob demanda com interatividade.

- Dentre as três propostas da literatura analisadas e utilizadas nos experimentos,

a proposta de *Shah-Pâris* apresentou o melhor desempenho.

- A proposta BIVoD-*Buffer* apresentou o melhor desempenho em todos os cenários. Entretanto, para a métrica tempo para iniciar a reprodução o seu desempenho não foi o melhor que os das demais propostas, comportamento este já esperado pelas características da proposta.

- A utilização da previsão, pelas propostas BIVoD e BIVoD-*Buffer*, apresentou excelentes resultados. A sua utilização contribui fortemente para a redução do tempo de retorno de uma interrupção como também para o bom desempenho das métricas taxa de *download*, taxa de *upload* e tempo de *download*.

- As propostas BIVoD e BIVoD-*Buffer* se mostram muito robustas quanto ao seu desempenho. Isto porque, mesmo o sistema classificando os usuários incorretamente, ou seja, utilizando uma previsão diferente do nível de interatividade real do usuário, o desempenho dessas duas propostas ainda foi melhor que o da proposta de *Shah-Pâris*.

- A proposta de *Shah-Pâris* apresentou um desempenho pior do que o da proposta BIVoD-*Buffer*, mesmo utilizando um *buffer* dentro da janela deslizante, como o utilizado pela proposta BIVoD-*Buffer*.

- As propostas BIVoD e BIVoD-*Buffer* alcançaram um alto índice de equidade, isto é, os usuários obtém um desempenho uniforme, tanto para a qualidade da reprodução, quanto para utilização da banda. Este alto nível de equidade pode ser atribuído a utilização da previsão por estas duas propostas. Enquanto que a proposta de *Shah-Pâris*, que não utiliza nenhuma previsão, apresentou um baixo índice de equidade em todos os cenários.

6.2 Trabalhos futuros

Durante o desenvolvimento desse trabalho muitas idéias surgiram, sem, no entanto, haver tempo hábil para a conclusão de todas elas. Abaixo, são listadas

algumas destas idéias que poderão ser futuramente implementadas:

- Analisar o comportamento de todas as propostas em um ambiente heterogêneo, onde exista uma grande diferença de capacidade entre os nós.
- Analisar o comportamento do algoritmo de seleção de vizinhos em um ambiente heterogêneo, uma vez que este pode influenciar no desempenho dos usuários. E se necessário modificá-lo com o objetivo de minimizar seus efeitos negativos no desempenho dos usuários, principalmente os de baixa capacidade.
- Analisar o desempenho da proposta BIVoD-*Buffer* utilizando o algoritmo de seleção de blocos sequencial para recuperar os blocos da janela de *playback*. Com a utilização do *buffer* talvez os efeitos negativos da seleção sequencial possa ser minimizado e principalmente o tempo para iniciar a reprodução seja reduzido.
- Reclassificar o nível de interatividade dos usuários constantemente e não somente na sua chegada ao sistema. Alguns usuários podem alterar o seu comportamento de acordo com o que esta sendo reproduzido, ou seja, em um intervalo de tempo ele pode apresentar um comportamento de alta interatividade e em um momento seguinte de médio ou mesmo de baixa interatividade, principalmente quando se trata de vídeo-aulas, onde o aluno pode realizar muitas ações a fim de buscar ou entender melhor um conceito, mas somente em um determinado ponto do vídeo.
- Utilizar mais nós *seed* nos experimentos, seja pela adição deles desde o inicio ou pela permanência de nós que já terminaram de assistir ao seu vídeo e se tornam *seed*.
- Analisar as propostas de distribuição de vídeo com taxas variáveis de codificação (vídeo em camadas).
- Analisar cenários onde clientes podem deixar o sistema antes de finalizar a sessão.

Referências Bibliográficas

[1] KUROSE, J. F., ROSS, K. W., *Computer Networking: A Top-Down Approach (4th Edition)*. Addison Wesley, Março 2007.

[2] YOUTUBE LLC, "Historia do Youtube", Disponível em: http://www.youtube.com/t/about. Acessado em Março de 2009.

[3] GRAHAM, J., "YouTube: 100 million videos served in one month", Disponível em: http://blogs.usatoday.com/technologylive/youtube/. Acessado em Abril de 2009.

[4] ZINK, M., SUH, K., GU, Y., et al., "Watch Global, Cache Local: YouTube Network Traces at a Campus Network - Measurements and Implications". In: *Multimedia Computing and Networking Conference - MMCN*, San Jose, CA, USA, January 2008.

[5] CARTER, L., "Web could collapse as video demand soars", Disponível em: http://www.telegraph.co.uk/news/uknews/1584230/Web-could-collapse-as-video-demand-soars.html. Acessado em Abril de 2009.

[6] YEN, Y.-W., "youtube looks for the money clip", Disponível em: http://techland.blogs.fortune.cnn.com/2008/03/25/youtube-looks-for-the-money-clip/. Acessado em Abril de 2009.

[7] RODRIGUES, C. K. S., LEÃO, R. M. M., "Bandwidth usage distribution of multimedia servers using Patching", *Computer Networks*, v. 51, pp. 569–587, February 2007.

[8] EAGER, D., VERNON, M., ZAHORJAN, J., "Bandwidth skimming: A technique for cost-effective video-on-demand". In: *SPIE Conf. On Multimedia Computing and Networking - MMCN*, pp. 206–215, San Diego, CA, USA, June 2000.

[9] RODRIGUES, C. K. S., FILHO, L. J. H., LEÃO, R. M. M., "On Scalable Interactive Video-On-Demand Services", *European Journal of Scientific Research*, v. 21, n. 4, pp. 662–686, September 2008.

[10] RODRIGUES, C. K. S., FILHO, L. J. H., LEÃO, R. M. M., "Buffering para Otimizacão de Sistemas Interativos de VoD". In: *XXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 941–954, Belém, PA, Brasil, Maio 2007.

[11] COHEN, BRAM, "Incentives Build Robustness in BitTorrent". In: *First Workshop on Economics of Peer-to-Peer Systens*, Berkeley, CA, USA, June 2003.

[12] GNUTELLA, "The Gnutella Protocol Specification v0.4", Disponível em: http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf. Acessado em Abril de 2009.

[13] GUO, Y., SUH, K., KUROSE, J., et al., "P2Cast: peer-to-peer patching for video on demand service", *Multimedia Tools and Applications*, v. 33, n. 2, pp. 109–129, May 2007.

[14] VLAVIANOS, A., ILIOFOTOU, M., FALOUTSOS, M., "BiToS: Enhancing BitTorrent for Supporting Streaming Applications". In: *9th IEEE Global Internet Symposium*, Barcelona, Spain, April 2006.

[15] ZHANG, X., LIU, J., LI, B., et al., "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming". In: *24th IEEE International Conference on Computer Communications - INFOCOM*, v. 3, pp. 2102–2111, Miami, Flórida, USA, July 2005.

[16] WANG, F., XIONG, Y., LIU, J., "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast". In: *27th International Conference on Distributed Computing Systems - ICDCS*, p. 49, Toronto, Ontario, Canada, June 2007.

[17] CHENG, X., LIU, J., "NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing". In: *28th IEEE International Conference on Computer Communications - INFOCOM*, April 2009.

[18] FILHO, L. J. H., RODRIGUES, C. K. S., LEÃO, R. M. M., "Acesso interativo para aplicações P2P de streaming de vídeo". In: *XVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 599–612, Recife, PE, Brasil, Maio 2009.

[19] PARVEZ, N., WILLIAMSON, C., MAHANTI, A., et al., "Analysis of bittorrent-like protocols for on-demand stored media streaming". In: *ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 301–312, Annapolis, MD, USA, June 2008.

[20] ZHOU, Y., CHIU, D. M., LUI, J. C. S., "A Simple Model for Analyzing P2P Streaming Protocols". In: *IEEE International Conference on Network Protocols - ICNP*, pp. 226–235, Beijing, China, October 2007.

[21] SHAH, P., PÂRIS, J.-F., "Peer-to-Peer Multimedia Streaming Using BitTorrent". In: *IEEE International Performance, Computing, and Communications Conference - IPCCC*, pp. 340–347, New Orleans, Louisiana, USA, April 2007.

[22] CARLSSON, N., EAGER, D. L., "Peer-assisted On-demand Streaming of Stored Media using BitTorrent-like Protocols". In: *IFIP/TC6 Networking*, pp. 570–581, Atlanta, GA, USA, May 2007.

[23] DE SOUZA E SILVA, E., LEÃO, R. M. M., SANTO, A. D., et al., "Multimedia Supporting Tools for the CEDERJ - Distance Learning Initiative applied to

the Computer Systems Course". In: *22th ICDE World Conference on Distance Education*, pp. 1–11, Rio de Janeiro, RJ, Brasil, September 2006.

[24] DE VIELMOND, C. C. L. B., LEÃO, ROSA M. M. E DE SOUZA E SILVA, E., "Um modelo HMM hierárquico para usuários interativos acessando um servidor multimídia". In: *Simpósio Brasileiro de Redes de Computadores - SBRC*, v. I, Belém, Pará, Brasil, maio 2007.

[25] DE SOUZA E SILVA, E., FIGUEIREDO, D., LEÃO, R. M. M., "The TANGRAM-II Integrated Modeling Environment for Computer Systems and Networks", *ACM SIGMETRICS Performance Evaluation Review*, v. 36, n. 4, pp. 45–65, March 2009.

[26] PPLIVE INC., "PPLive", Disponível em: http://www.pplive.com. Acessado em Dezembro de 2008.

[27] PPSTREAM INC., "PPStream", Disponível em: http://www.ppstream.com. Acessado em Dezembro de 2008.

[28] SETI@HOME, "SETI@home", Disponível em: http://setiathome.berkeley.edu/index.php. Acessado em Abril de 2009.

[29] UC BERKELEY COMPUTER SCIENCE DIVISION, "The OceanStore Project", Disponível em: http://oceanstore.cs.berkeley.edu/. Acessado em Abril de 2009.

[30] SKYPE TECHNOLOGIES S.A., "Skype", Disponível em: www.skype.com. Acessado em Abril de 2009.

[31] ANDROUTSELLIS-THEOTOKIS, S., SPINELLIS, D., "A survey of peer-to-peer content distribution technologies", *ACM Computing Surveys*, v. 36, n. 4, pp. 335–371, December 2004.

[32] LI, J., "On peer-to-peer (P2P) content delivery", *Peer-to-Peer Networking and Applications*, v. 1, n. 1, pp. 45–63, March 2008.

[33] BRILLIANT DIGITAL ENTERTAINMENT INC., "Kazaa", Disponível em: http://www.kazaa.com/. Acessado em Abril de 2009.

[34] STOICA, I., MORRIS, R., KARGER, D., et al., "Chord: A scalable peer-to-peer lookup service for internet applications". In: *Conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM*, v. 31, n. 4, pp. 149–160, San Diego, CA, USA, August 2001.

[35] DRUSCHEL, P., ROWSTRON, A., "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility". In: *HotOS VIII*, pp. 75–80, Schloss Elmau, Germany, May 2001.

[36] NAPSTER LLC, "Napster", Disponível em: http://free.napster.com. Acessado em Abril de 2009.

[37] LEGOUT, A., LIOGKAS, N., KOHLER, E., et al., "Clustering and sharing incentives in BitTorrent systems". In: *ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 301–312, San Diego, California, USA, June 2007.

[38] SOPCAST.COM, "SopCast", Disponível em: http://www.sopcast.com. Acessado em Dezembro de 2008.

[39] JOOST N.V., "Joost", Disponível em: http://www.joost.com. Acessado em Dezembro de 2008.

[40] DANA, C., LI, D., HARRISON, D., et al., "BASS: BitTorrent Assisted Streaming System for Video-on-Demand". In: *7th Workshop on Multimedia Signal Processing*, pp. 1–4, Shanghai, China, November 2005.

[41] CHOE, Y. R., SCHUFF, D. L., DYABERI, J. M., et al., "Improving VoD Server Efficiency with BitTorrent". In: *15th international conference on Multimedia - MULTIMEDIA*, pp. 117–126, Augsburg, Germany, September 2007.

[42] LIU, Y., GUO, Y., LIANG, G., "A survey on Peer-to-Peer video streaming systems", *Peer-to-Peer Networking and Applications*, v. vol. 1, pp. 18–28, March 2008.

[43] MORAES, I. M., CAMPISTA, M. E. M., MORREIRA, M. D. D., et al., "Distribuição de Vídeo Sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios", 1st ed., v. 1, chap. 3, pp. 115–171, n. 1, Sociedade Brasileira de Computação: Rio de Janeiro - RJ - Brasil, Maio 2008.

[44] CHU, Y.-H., RAO, S. G., ZHANG, H., "A Case for End System Multicast". In: *ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 1–12, Santa Clara, California, USA, June 2000.

[45] JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., et al., "Overcast: reliable multicasting with on overlay network". In: *4th conference on Symposium on Operating System Design & Implementation - OSDI*, pp. 14–20, San Diego, CA, USA, August 2000.

[46] MIRANDA, M., FIGUEIREDO, D., "A Preferential Attachment Model for Tree Construction in P2P Video Streaming". In: *XXIX Congresso da Sociedade Brasileira de Computação - VIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, Bento Gonçalvez, RS, Brasil, Julho 2009.

[47] TRAN, D. A., HUA, K., DO, T., "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming". In: *22th IEEE International Conference on Computer Communications - INFOCOM*, v. 2, pp. 1283–1292, San Francisco, California, USA, April 2003.

[48] CASTRO, M., DRUSCHEL, P., KERMARREC, A.-M., et al., "SplitStream: high-bandwidth multicast in cooperative environments". In: *9th ACM symposium on Operating systems principles - SOSP*, pp. 298–313, Bolton Landing, NY, USA, October 2003.

[49] MAGHAREI, N. E REJAIE, R., "PRIME: Peer-to-Peer Receiver-drIven MEsh-Based Streaming". In: *26th IEEE International Conference on Computer Communications - INFOCOM*, pp. 1415–1423, Anchorage, Alaska, USA, May 2007.

[50] ZHANG, M., TANG, Y., ZHAO, L., et al., "Gridmedia: A Multi-Sender Based Peer-to-Peer Multicast System for Video Streaming". In: *IEEE International Conference on Multimedia and Expo - ICME*, pp. 614–617, Amsterdam, Nederland, August 2005.

[51] LI, B., XIE, S., KEUNG, G., et al., "An Empirical Study of the Coolstreaming+ System", *IEEE Journal on Selected Areas in Communications*, v. 25, n. 9, pp. 1627 – 1639, December 2007.

[52] LIAO, X., JIN, H., LIU, Y., et al., "AnySee: Peer-to-Peer Live Streaming". In: *25th IEEE International Conference on Computer Communications - INFOCOM*, pp. 1–10, Barcelona, Catalynya, Spain, April 2006.

[53] BANERJEE, S., BHATTACHARJEE, B., KOMMAREDDY, C., "Scalable Application Layer Multicast". In: *2th ACM SIGCOMM conference on Internet measurement*, pp. 205–217, Pittsburgh, PA, USA, October 2002.

[54] TRAN, D. A., HUA, KIEN E DO, T., "A Peer-to-Peer Architecture for Media Streaming", *IEEE Journal on Selected Areas in Communications, Special Issue on Advances in Service Overlay Networks*, v. 22, n. 1, pp. 121–133, January 2004.

[55] PAI, V., KUMAR, K., TAMILMANI, K., et al., "Chainsaw: Eliminating Trees from Overlay Multicast". In: *4th International Workshop Peer-to-Peer Systems, IPTPS*, pp. 127–140, Ithaca, NY, USA, February 2005.

[56] SENTINELLI, A., MARFIA, G., GERLA, M., et al., "Will IPTV ride the peer-to-peer stream?" *IEEE Communications Magazine*, v. 45, n. 6, pp. 86–92, 2007.

[57] RODRIGUES, C. K. S., *Mecanismos de Compartilhamento de Recursos para Aplicacões de Mídia Contínua na Internet*, Ph.D. Thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Abril 2006.

[58] HU, A. H., "Video-on-demand Broadcasting Protocols: A Comprehensive Study". In: *20th IEEE International Conference on Computer Communications - INFOCOM*, pp. 508–517, Anchorage, Alaska, USA, April 2001.

[59] GAO, L., TOWSLEY, D., "Threshold-Based Multicast for Continuous Media Delivery", *IEEE Transactions on Multimedia*, v. 3, n. 4, pp. 405–414, 2001.

[60] DO, T. T., HUA, K. A., TANTAOUI, M. A., "Robust video-on-demand streaming in peer-to-peer environments", *Comput. Commun.*, v. 31, n. 3, pp. 506–519, 2008.

[61] GUO, Y., SUH, K., KUROSE, J., et al., "DirectStream: A directory-based peer-to-peer video streaming service", *Journal of Computer communications - COMCOM*, v. 31, n. 3, pp. 520–536, 2008.

[62] GUO, Y., MATHUR, S., RAMASWAMY, K., et al., "PONDER: Performance Aware P2P Video-on-Demand Service". In: *IEEE Global Telecommunications Conference - GLOBECOM*, pp. 225–230, Washington, DC, USA, November 2007.

[63] BITTORRENT, "Cliente Principal BitTorrent 4.0.2", Disponível em: http://download.bittorrent.com/dl/archive/. Acessado em Abril de 2007.

[64] BHARAMBE, A. R., HERLEY, C., PADMANABHAN, V. N., "Analyzing and Improving a BitTorrent Networks Performance Mechanisms". In: *25th INFOCOM*, pp. 1–12, Barcelona, Catalunya, Espanha, April 2006.

[65] LEGOUT, A., URVOY-KELLER, G., MICHIARDI, P., "Rarest first and choke algorithms are enough". In: *6th ACM SIGCOMM conference on Internet measurement*, pp. 203–216, Rio de Janeriro, RJ, Brazil, October 2006.

[66] DE VIELMOND, CAROLINA C. L. B., *Um Modelo HMM Hierárquico para Usuários Interativos Acessando Um Servidor Multimídia*, Master's Thesis, Universidade Federal do Rio de Janeiro - COPPE - Programa de Engenharia de Sistemas e Computação, Rio de Janeiro - RJ, Novembro 2007.

[67] NETTO, B. C. M., AZEVEDO, J. A., DE SOUZA E SILVA, E., et al., "Servidor Multimídia RIO em Ensino à Distância". In: *Proc. 6th International Free Software Forum*, Porto Alegre, RS, Brasil, June 2005.

[68] PLANETLAB, "PlanetLab An open platform for developing, and accessing planetary-scale service", Disponível em: http://www.planet-lab.org/. Acessado em Abril de 2008.

[69] CHUN, B., CULLER, D., ROSCOE, T., et al., "PlanetLab: an overlay testbed for broad-coverage services", *SIGCOMM Comput. Commun. Rev.*, v. 33, n. 3, pp. 3–12, 2003.

[70] COMSCORE INC., "21,4 bilhões de vídeos assistidos em julho", Disponível em: http://comscore.com/index.php//Press_Events/Press_Releases/2009/8/U.S._Online_Video_Market_Soars_in_July_as_Summer_Vacation_Drives_Pickup_in_Entertainment_and_Leisure_Activities_Online. Acessado Agosto de 2009.