Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Programa de Pós-Graduação em Ciência da Computação

Bianca Portes de Castro

PALMS+: Protocolo ALM Baseado em Desigualdade Triangular para Distribuição de *Streaming* de Vídeo

Juiz de Fora

2014

Bianca Portes de Castro

PALMS+: Protocolo ALM Baseado em Desigualdade Triangular para Distribuição de *Streaming* de Vídeo

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computaçãoda Universidade Federal de Juiz de Fora, na área de concentração em Sistemas e Tecnologia da Computação, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientadora: Ana Paula Couto da Silva

Coorientador: Alex Borges Vieira

Juiz de Fora

2014

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Castro, Bianca Portes.
PALMS+: Protocolo ALM Baseado em Desigualdade Triangular para Distribuição de *Streaming* de Vídeo / Bianca Portes de Castro. – 2014.
47 f. : il.

Orientadora: Ana Paula Couto da Silva
Coorientador: Alex Borges Vieira
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2014.

1. Protocolo ALM. 2. Difusão seletiva. 3. *Streaming* de vídeo. I. Silva, Ana Paula Couto da, orient. II. Vieira, Alex Borges, coorient. III Título.

Bianca Portes de Castro

PALMS+: Protocolo ALM Baseado em Desigualdade Triangular para Distribuição de *Streaming* de Vídeo

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computaçãoda Universidade Federal de Juiz de Fora, na área de concentração em Sistemas e Tecnologia da Computação, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 26/08/2014

BANCA EXAMINADORA

Profa. Dra. Ana Paula Couto da Silva - Orientadora
Universidade Federal de Minas Gerais

Professor Dr. Alex Borges Vieira - Coorientador
Universidade Federal de Juiz de Fora

Professor Dr. Elias Procópio Duarte Jr
Universidade Universidade Federal do Paraná

Professor Dr. Heder Soares Bernardino
Universidade Federal de Juiz de Fora

A Deus em primeiro lugar. Aos meus pais, irmão, namorado e amigos pelo apoio incondicional.

AGRADECIMENTOS

Ao fim desta importante etapa em minha vida, sinto-me em débito com muitas pessoas que me sustentaram neste período.

Em primeiro lugar, agradeço a Deus pela força e oportunidade de crescimento que me proporcionou.

Agradeço aos meus pais, Sônia e Pedro, e ao meu irmão, Bruno. Sem os valores transmitidos e o apoio de vocês para que eu desse cada passo em busca dos meus objetivos, eu não conseguiria chegar aonde cheguei.

Ao meu namorado, Lucas, pela compreensão, incentivo e paciência em meus momentos mais difíceis.

Aos meus amigos e familiares pela capacidade de alentar minha alma com os risos e demonstrações constantes de carinho.

Aos professores que me guiaram nessa longa jornada, transmitindo-me ferramentas necessárias para a busca do conhecimento. Em especial, aos meus orientadores, Alex e Ana, por suportarem esta caminhada ao meu lado nesta etapa.

A Universidade Federal de Juiz de Fora e aos amigos que convivi nesses espaços ao longo desses últimos anos. Agradeço especialmente aos amigos do grupo de redes por compartilharem comigo seus conhecimentos e experiências.

A CAPES pelo suporte financeiro. Sem isto, meu caminho seria muito mais difícil.

"I have no special talent. I am only passionately curious."

Albert Einstein

RESUMO

Aplicações multimídia são muito populares na internet. Grande parte delas necessita de *multicast* para escalar. É sabido que *multicast* em nível de redes não foi implementado como desejado. Protocolos em nível de aplicação são a solução atual. Apesar do sucesso dos protocolos ALM (*Application Layer Multicast*), a maioria dos protocolos existentes são custosos e acarretam grande sobrecarga de controle à rede. Neste trabalho, apresentamos um novo protocolo de fluxo contínuo baseado em árvore, utilizando a desigualdade triangular entre cada três *peers* para gerenciamento dinâmico da topologia (o PALMS+). O novo protocolo é simples e com baixa sobrecarga. Mesmo assim, seu desempenho é tão bom quanto o estado da arte. Experimentos realizados na plataforma Oversim (OMNet++) demonstraram que o PALMS+ manteve desempenho tão bom quanto o estado da arte (e.g. protocolo NICE), mesmo quando submetido a alto *churn* em uma rede heterogênea. De fato, a sobrecarga nos *peers* do novo protocolo é menor que 10% da sobrecarga gerada pelo NICE. O protocolo PALMS+ entrega os dados em menos de 1,5s. O novo protocolo mostra-se adequado a vídeo ao vivo, escalando mesmo em cenários realistas e com alto *churn*.

Palavras-chave: *Multicast* na camada de aplicação. Difusão seletiva. *Streaming* de vídeo. Distribuição em árvore.

ABSTRACT

Multimedia applications are very popular on the internet. Many of these applications need multicast to scale. However, network layer multicast has not been implemented in the internet. Application layer multicast (ALM) protocols are a practical alternative. However, despite their popularity, many existing ALM protocols and mechanisms are expensive and bring a large overhead control on the network. In the present work, a new protocol is proposed for content distribution based on tree, using the triangular inequality between every three peers to dynamic topology control (the PALMS+). The new protocol is simple and with low overhead. Nevertheless, its performance as good as the state of the art. Experimental results conducted with the OverSim platform (OMNet++) suggest that PALMS+ improves the performance of a state-of-art implementation of ALM protocol when compared against the NICE protocol. Furthermore, the control message overhead at peers using the PALMS+ protocol is reduced by 10%, when compared with NICE. In the PALMS+ protocol, chunks are delivered up to 1,5s. Results confirm that proposed implementation of PALMS+ is very suitable to real-time video streaming, even when churn is high.

Key-words: Application Layer Multicast. Multicast. Video streaming. Distribution tree.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Nos últimos anos é possível observar um número crescente de aplicações na Internet que necessitam de meio compartilhado para transmitir uma quantidade elevada de dados. Destacam-se, neste contexto, aplicações de vídeo ao vivo, conferências multimídia ou jogos (*multiparty games*). Estas precisam transmitir a um grande número de participantes o conteúdo proveniente de poucas fontes (comumente uma única fonte). Em muitos casos, tais aplicações necessitam de mecanismos especiais para escalar e conseguir atender seus usuários.

No entanto, ainda nos dias atuais, as soluções de compartilhamento de recursos de rede na Internet raramente encontram-se disponíveis. Para que seja possível escalar aplicações colaborativas, a maioria das soluções adotadas baseia-se no compartilhamento de recursos utilizando protocolos na camada de aplicação (*Application Layer Multicast* - ALM). Mais precisamente, protocolos ALMs implementam a funcionalidade de encaminhamento *multicast* na camada de aplicação, utilizando os recursos disponíveis nos *end-hosts* [3].

Alguns pontos estimulam o uso e o sucesso de ALMs. Inicialmente, o uso de difusão seletiva (*multicast*) na camada de rede apresenta alguns problemas e depende profundamente da mudança da infraestrutura da rede [9]. Além disso, soluções nativas de *multicast*, como as oferecidas por IPV6, não são uma realidade. De fato, a migração para o IPv6 está levando mais tempo do que o esperado, fechando um crescimento de adoção de 3,8% no segundo trimestre de 2013. Este é o menor crescimento desde 2009, de acordo com dados divulgados pelo relatório da Akamai [1].

Apesar do sucesso e da grande utilização de protocolos ALM, muitos dos trabalhos existentes na literatura propõem mecanismos e protocolos custosos [24]. Mais ainda, como estes protocolos têm de lidar com a instabilidade e heterogeneidade de recursos dos *peers*, eles implementam mecanismos dependentes de aplicativos para tentar manter a qualidade desejável do serviço. Finalmente, estes protocolos funcionam na camada de aplicação e, por consequência, têm um desempenho limitado quando comparados com o *multicast* na camada de rede.

Neste trabalho é apresentado um novo protocolo, o PALMS+, uma versão com novas funcionalidades e melhorias a partir do protocolo PALMS proposto em [15, 14]. Originalmente, o PALMS é um protocolo para *multicast* na camada de aplicação baseado em uma arquitetura em árvore (*tree-based*). Ele foi concebido para ser o mais simples possível, permitindo a distribuição de fluxo contínuo de conteúdo a partir de uma única fonte na Internet. Além disso, o protocolo possui um recurso para refinamento dos acordos de retransmissão utilizando desigualdade triangular baseada no RTT (*Round-Trip-Time*). Suas principais características são a eficiência e a economia quanto ao uso de largura de banda com mensagens de controle. Para cenários com baixa rotatividade dos nós (*churn*),

os resultados comparativos para estas características mostraram em [15, 14] que o PALMS tem desempenho semelhante ao protocolo Narada, proposto em [5].

O novo protocolo se propõe a melhorar o tempo de recepção dos *peers* da rede em cenários com alto *churn*, adequando-o para aplicações de distribuição de *streaming* de vídeo. Contudo, pretende-se mantê-lo simples, atuando com *peers* não confiáveis no aspecto colaborativo da rede e tão eficiente quanto o estado da arte.

1.1 Contribuições

Este trabalho tem como objetivo estender o comportamento do protocolo PALMS, através da implementação e análise dos dados coletados em diferentes cenários. Conhecer o impacto sofrido a partir das simulações em alto *churn* e propor um novo protocolo simples e voltado para distribuição de *streaming* de vídeo, tão eficiente quanto o estado da arte em baixo *churn*. Neste sentido, as principais contribuições desta dissertação são:

- Identificar participantes mais estáveis que auxiliem na distribuição de carga na rede.
- Identificar outros problemas que afetem a escalabilidade do protocolo PALMS em cenários com alta rotatividade de nós.
- Propor um novo protocolo para aplicações de *streaming* de vídeo a partir do PALMS.
- Experimentalmente, validar o protocolo proposto (o PALMS+) em um cenário com alta rotatividade dos *peers*.

1.2 Organização da Dissertação

O restante deste trabalho é organizado da seguinte forma. No capítulo 2 são descritos os conceitos teóricos acerca da comunicação multiponto e protocolos ALM, importantes para a compreensão da proposta deste trabalho. Também no capítulo 2 são apresentados os trabalhos que serviram como base de conhecimento para as modificações propostas no protocolo original. O capítulo 3 descreve o protocolo proposto. Os métodos e ferramentas utilizados para efetuar os experimentos são descritos no capítulo 4. No capítulo 5 são discutidos os resultados obtidos com base nas métricas de desempenho descritas no capítulo 4. As conclusões do trabalho e a discussão sobre trabalhos futuros são apresentados no capítulo 6.

2 Conceitos teóricos e trabalhos relacionados

Neste capítulo, são apresentados os conceitos fundamentais para a distribuição de conteúdo em fluxo contínuo para múltiplos receptores de forma simultânea. Assim, a seção 2.1 descreve brevemente modelos de comunicação, com suas qualidades e restrições. Esta seção apresenta também a importância do uso estratégico de mecanismos para distribuição de conteúdo através da camada de aplicação e exemplos de protocolos que implementam esta funcionalidade. A seção 2.3 apresenta os trabalhos relacionados com esta dissertação e os principais protocolos de distribuição *multicast*.

2.1 Comunicação multiponto

Redes multiponto são aquelas em que mais de dois dispositivos estão interligados com o intuito de se comunicarem entre si. Os modelos de comunicação para múltiplos receptores estão intrinsecamente associados às necessidades da aplicação e, basicamente, podem ser divididos em três modelos: comunicação ponto-a-ponto (*unicast*), difusão (*broadcast*) e difusão seletiva (*multicast*). A figura 1 apresenta a diferença das estratégias na rede para estas comunicações, que são descritas a seguir.

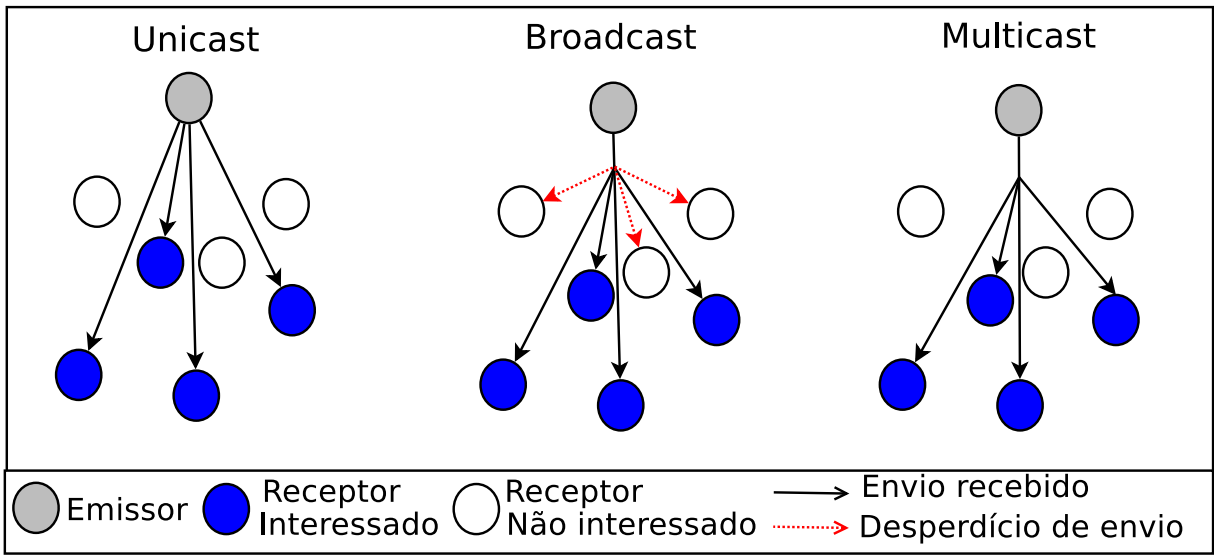


Figura 1 – Comparação entre comunicação por *unicast*, *broadcast* e *multicast* com suporte na camada de rede.

Na comunicação *unicast*, uma estação que deseja transmitir algum dado deve requerer o estabelecimento de conexão com cada uma das outras estações da rede e transmitir este dado separadamente. Assim, por exemplo, caso uma estação queira transmitir conteúdo contínuo de vídeo ao vivo para as outras N estações pertencentes a um grupo, N conexões serão necessárias para endereçar e transmitir, separadamente, dados idênticos ao grupo. O grupo é um subconjunto de nós que formam a rede. Esta é a única comunicação suportada por todos os roteadores da Internet [9]. Estes sistemas

de comunicação baseados na arquitetura cliente-servidor têm características inadequadas às necessidades de comunicação envolvendo múltiplos participantes (multiponto) devido, principalmente, à falta de escalabilidade.

O modelo *broadcast*, diferentemente do *unicast*, direciona um único fluxo de dados de uma estação para todas as estações da rede em um único canal de comunicação compartilhado por elas. Sendo assim, todos os participantes devem verificar todos os dados recebidos e observar se são o alvo da estação emissora; se não, o dado é ignorado. Neste método, há um desperdício de largura de banda ao inundar a rede com mensagens repetidas e sobrecarga dos nós intermediários da rede.

A comunicação *multicast*, por sua vez, permite que a entrega do conteúdo seja no esquema multiponto em aplicações, nas quais, várias estações necessitam se comunicar simultaneamente, como videoconferências, distribuição de vídeos sob demanda, atualizações de softwares e jogos on-line. Assim como o *broadcast*, o emissor se comunica a partir de uma única transmissão. Porém, em vez de enviar mensagens para todos os nós, apenas o subconjunto de máquinas interessadas em receber o dado será o alvo desta difusão seletiva. Este método é indicado para os casos em que o grupo interessado não é muito pequeno e a maioria dos participantes da rede não está interessada em receber a informação.

Inicialmente, a comunicação multiponto foi suprida pelos modelos de transmissão *unicast* e, localmente, em *broadcast* [8]. Naquela época, em aplicações de vídeo, um único servidor era capaz de disponibilizar conteúdo aos usuários, uma vez que o número de vídeos e usuários era pequeno. Entretanto, manter apenas sessões *unicast* torna-se inviável à medida que houve um grande aumento do número de usuários e conteúdos multimídia, isto devido ao alto custo imposto ao servidor e também ao grande desperdício por replicação de mensagens e sobrecarga dos nós intermediários da rede de transporte.

Como alternativa, Deering et al propuseram uma extensão para o IP *unicast*, o IP *Multicast* [7]. O IP *Multicast* tem o objetivo de fornecer comunicação *multicast* eficiente, mantendo a semântica do IP e permitindo a criação e manutenção de grupos de usuários dinamicamente, são os grupos de *multicast*. Com esta comunicação sob a infraestrutura IP, a replicação das mensagens é feita apenas quando necessária e a critério dos roteadores.

Todavia, esta não foi uma solução eficaz e o alcance do IP *Multicast* continua limitado nos dias atuais devido à necessidade de atualizar roteadores de toda a Internet para suportarem esta funcionalidade e lidar com o gerenciamento em tempo real de milhares de endereços multicast por aplicação. Além disso, muitos provedores de Internet (ISPs) simplesmente bloquearam ou desativaram esta funcionalidade devido a problemas econômicos e de segurança [16]. Tais restrições impedem que aplicações *multicast*, em geral, difundam o seu conteúdo com alcance global a centenas ou milhares de máquinas simultaneamente conectadas a partir desta solução, como mencionado em [12].

Ainda que o *IP Multicast* não seja largamente utilizado nos dias de hoje, existem aplicações especiais em redes IP privadas que o implementam. Vale ressaltar também que, diferentemente do IPv4, no qual o *multicast* foi introduzido apenas como uma extensão ao protocolo, o *multicast* é suportado nativamente pela versão 6 do protocolo IP (IPv6). Contudo, a versão 4 ainda é a mais utilizada atualmente na Internet e a transição para o IPv6 tem sido gradual [1]. Uma atualização radical para o IPv6, e não gradual, requer que cada *host* e roteador da internet tenha seu software atualizado para a nova versão, uma tarefa complexa ao se ponderar a quantidade de plataformas que suportam apenas o IPv4. Já a solução adotada para a migração permite a coexistência dos protocolos, permitindo que sistemas utilizando o IPv4 funcionem mesmo após ampla implantação do IPv6. Assim, soluções de âmbito global para *multicast* devem ser aplicáveis ao IPv4.

Como alternativa, passou-se a implementar uma comunicação *multicast* utilizando apenas os recursos da camada de aplicação (ALM - *Application Layer Multicast*) [23]. Esta comunicação independente da rede física pressupõe a construção de uma rede sobreposta (*overlay*) par-a-par onde a difusão seletiva é feita a partir da ligação lógica (sessão) entre os participantes na camada de aplicação. Desta forma, serviços como os de entrega de vídeo podem ser facilmente implantados na estrutura atual da Internet sem a necessidade de atualizar o *hardware* dos nós intermediários da rede (roteadores) [11].

2.2 Protocolos ALM

A ideia básica da comunicação *multicast*, utilizando apenas os recursos da camada de aplicação (ALM), é unir as máquinas interessadas em um determinado conteúdo e utilizá-las na própria alimentação do sistema de distribuição. Assim, os computadores, chamados *peers* (pares), podem alternadamente atuar como clientes (requerendo o dado) e como servidores (provendo o dado), de forma semelhante ao modelo *peer-to-peer* [3, 2].

Protocolos multicast podem ser classificados como *push-based* ou *pull-based* [19]. Estas duas formas definem a principal estratégia adotada para o encaminhamento dos dados na rede sobreposta, estão descritas a seguir.

Em sistemas *push-based*, também conhecidos como *tree-push* devido à organização da rede sobreposta de distribuição de conteúdo em forma de árvore, os *peers* encaminham os dados recebidos ao longo de uma ou mais árvores. Nesta estratégia, um fluxo advindo de uma fonte para os *peers* participantes é criado, sem a necessidade que os dados sejam explicitamente pedidos. Desta forma, um fluxo contínuo é entregue com um atraso mínimo.

Em sistemas *pull-based*, os *peers* devem requisitar explicitamente o conteúdo para recebê-lo. Protocolos em que os *peers* se organizam em forma de malha (*mesh*) são do tipo *pull-based*, já que cada *peer* requisita o dado desejado de seus parceiros.

Segundo [16], protocolos ALM de fluxo de vídeo ao vivo são, em sua grande

parte, *push-based* devido ao atraso mínimo que este encaminhamento de conteúdo permite, uma vez que os dados não necessitam ser anunciados ou requisitados antes do envio aos parceiros.

As métricas comumente utilizadas para a comparação de protocolos ALM na camada de aplicação referem-se à qualidade da rede sobreposta criada. Elas podem estar associadas às informações coletadas na camada de rede e que nem sempre estão disponíveis, tais como: taxa de *stress*, que mede a replicação de pacotes transmitidos por um mesmo nó na rede, tendo sempre valor 1 para os membros de uma rede cujo *multicast* é suportado na camada de rede; taxa de *stretch*, que é a razão entre o tamanho do caminho percorrido pelo pacote no protocolo ALM sobre o tamanho do caminho de uma mensagem *unicast* entre os nós, sendo sempre de valor 1 nas mensagens *unicast*. Outras métricas disponíveis podem ser coletadas na própria camada de aplicação, como a sobrecarga imposta aos nós e a diferença de latência fim-a-fim [14].

2.3 Trabalhos relacionados

Muitos protocolos *multicast* da camada de aplicação podem ser encontrados na literatura. Apesar disto, o problema de *multicast* continua sem uma solução definitiva e aplicável em larga escala [12, 17].

Entretanto, a demanda por este tipo de serviço continua a crescer, e com ela a necessidade de enquadrar o desenvolvimento de protocolos ALM que considerem fatores como a escalabilidade com baixa sobrecarga da rede, imprevisibilidade (contribuição e permanência na rede) dos nós e heterogeneidade entre suas configurações [14].

A seguir, apresentaremos típicos protocolos que utilizam a estratégia *push-based* para difusão de pacotes na rede: Narada [5] e NICE [3]. Além deles, estratégias mais recentes também foram descritas: PPM [10], *mTreebone* [25] e PALMS [14, 15].

2.3.1 Narada

O protocolo Narada [5] é um dos pioneiros na tarefa de utilizar *multicast* para distribuição de conteúdo. A difusão de conteúdo é feita de forma descentralizada. No entanto, existe uma entidade central - *rendezvous point* (RP) - responsável por manter informações sobre o estado de todos os *peers* que fazem parte da rede. Assim, quando um nó solicita entrada no grupo ao RP, ele recebe todos os nós pertencentes à rede e seleciona aleatoriamente um subconjunto de nós para estabelecer o conjunto de vizinhos com os quais trocará o conteúdo.

Na primeira etapa, a rede organiza os *peers* na forma de malha (*mesh*) para a transmissão de mensagens de controle. A partir das ligações existentes nesta rede *mesh*, cada nó constrói uma árvore de abrangência contendo todos os nós presentes na rede.

Assim, cada nó é capaz de emitir um dado a partir de sua árvore de distribuição. A figura 2 mostra a rede *mesh* (linha contínua), assim como a árvore de distribuição (linha tracejada) de um dado nó.

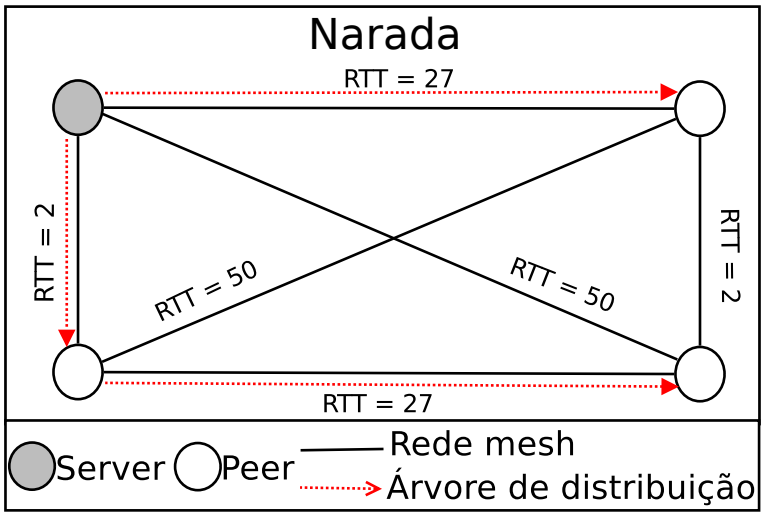


Figura 2 – Exemplo ilustrativo da rede sobreposta criada pelo protocolo Narada.

Após a etapa de construção das redes de transmissão de mensagens de controle e de dados, o protocolo executa as etapas de gerenciamento e refinamento das redes sobrepostas. O gerenciamento tem por intuito garantir a conectividade da rede a partir do envio periódico de mensagens de *refresh* (RM). Já o refinamento é dividido em duas etapas: (1) busca por novos nós arbitrários para melhorar a qualidade da conexão e (2) a avaliação da permanência das arestas existentes.

A existência de duas redes sobrepostas e a inundação por mensagens de controle para averiguar a consistência das mesmas tornam este protocolo complexo e pouco eficiente para grupos com grande número de *peers*, envolvendo uma sobrecarga de mensagens de controle total de $O(n^2)$, onde n é o número de *peers* da rede. Contudo, de forma mais simples, pretende-se construir uma única rede sobreposta com o foco na baixa sobrecarga de controle em que os *peers*, de forma distribuída e em pequenos grupos, gerenciem a consistência da mesma.

2.3.2 NICE

O protocolo NICE [3] foi desenvolvido para aplicações com baixo consumo de largura de banda e de transmissão para um grande conjunto de destinatários. O NICE mostrou desempenho equiparável ao Narada em relação ao tamanho do caminho para a entrega dos dados e a recuperação da rede sobreposta após falha dos *peers* [3].

No Nice, ao contrário do protocolo Narada, não há distinção entre as rotas de envio de dados e de mensagens de controle na rede sobreposta. Todas as mensagens são transmitidas dentro da mesma topologia criada. Esta organização dos participantes ocorre

de forma distribuída e conduz à formação de uma árvore de níveis (*layers*). Em cada nível, os nós juntam-se a outros participantes mais próximos criando grupos - *clusters*. Um nó com menor distância entre todos os outros do grupo é selecionado como líder, formando uma topologia de distribuição de dados local no formato estrela. Do conjunto de *clusters* de um dado nível, tem-se um conjunto de líderes que serão tratados como um novo *cluster* em um nível acima e assim por diante. A camada hierárquica mais acima é composta por um único nó, o *rendezvous point* (RP). A camada mais inferior, por sua vez, é constituída de todos os nós participantes da rede. Este modelo de organização está exemplificado na figura 3.

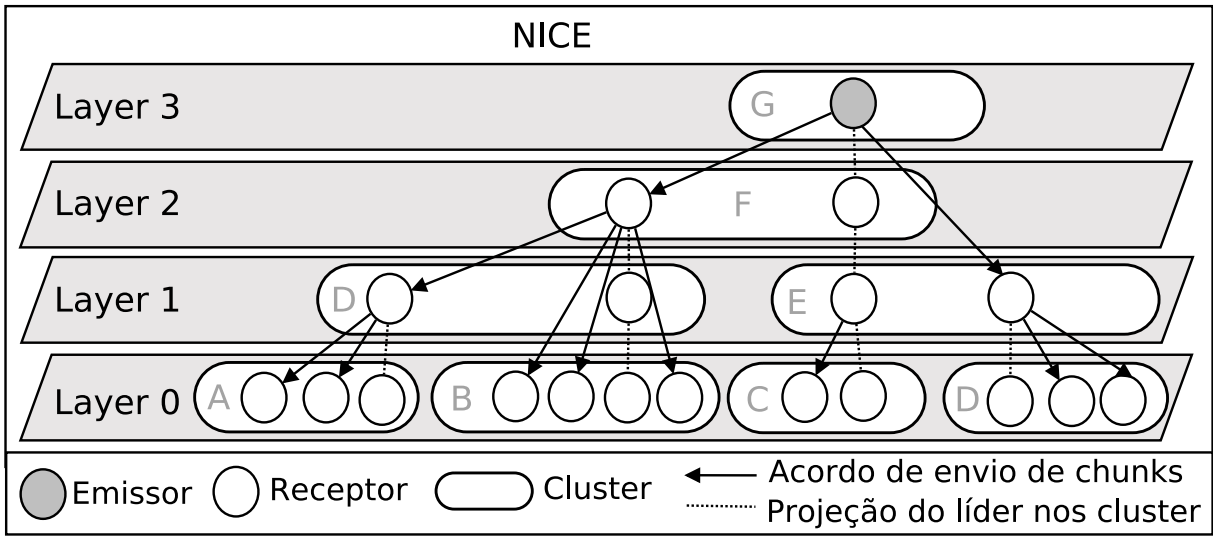


Figura 3 – Exemplo ilustrativo da árvore de níveis do protocolo NICE.

A localização dos nós dentro dos grupos é feita com base no atraso entre eles. Assim, os nós mais próximos se encontram unidos num mesmo nível hierárquico, conferindo às árvores de distribuição um menor atraso médio. Por proximidade entende-se o menor tempo de ir-e-vir (RTT, do inglês *round-trip-time*) entre os nós da rede.

Novos nós solicitam sua ligação ao RP. Ele, por sua vez, entrega uma lista dos nós da camada imediatamente inferior na hierarquia. Assim, um novo nó verifica recursivamente qual destes participantes está mais próximo até encontrar o *cluster* no qual ele será inserido. Este algoritmo assegura a descoberta do *cluster* mais próximo em um número de passos que é logaritmicamente dependente do número de nós de um grupo.

Analogamente à inserção de nós, na desconexão ou falha de um líder de grupo, o procedimento de seleção de um novo líder que tenha a menor distância entre os membros do *cluster* é executada recursivamente em cada uma das camadas hierárquicas afetadas com a mudança na rede sobreposta.

Para evitar que os nós ingressantes recebam os dados com muito atraso, eles são temporariamente inseridos em *clusters* de maior hierarquia. Como destacado em [3], este procedimento de entrada de *peers* no NICE o tornou mais eficiente que o Narada. Assim, com baixa sobrecarga de mensagens de controle, o protocolo NICE converge rapidamente para valores estáveis de tempo de recepção de fatias de conteúdo (*chunks*).

No NICE, existe uma fase de manutenção da estrutura da rede sobreposta executada periodicamente. O tamanho do *cluster* deve estar entre k e $3k-1$, onde k é uma constante dada como parâmetro do protocolo. Por exemplo, sendo k igual a 3, o menor tamanho do *cluster* será 3 e o maior tamanho será 8. Nesta fase de manutenção, para manter a carga ideal de aproveitamento colaborativo da rede, um *cluster* que tem tamanho superior ao configurado é dividido em dois novos grupos. Analogamente, caso algum *cluster* esteja abaixo de sua capacidade, ele será unido a algum outro que esteja abaixo de sua capacidade máxima.

Em [13], um trabalho mais profundo sobre a percepção de performance do usuário em relação ao protocolo NICE foi apresentado. Neste trabalho, sob agressivo modelo de *churn*, a latência e a largura de banda consumida do usuário foi avaliada para grandes grupos de *multicast*. Os resultados apresentados destacaram que a latência da rede cresce moderadamente com o aumento do número de nós da rede, que o atraso da entrada na rede é otimizado com o passar do tempo e que a árvore de distribuição de dados surpreendemente se ajusta bem com alto *churn*. Tais resultados levaram o trabalho a concluir que NICE pode suprir as expectativas de usuários mesmo em grupos grandes e sob alto *churn*.

Contudo, o fato de os *clusters* aceitarem inicialmente ilimitados acordos de retransmissão de conteúdo para só então, periodicamente, executarem a fase de manutenção, pode corroborar com esta eficiência. Apesar deste relaxamento do tamanho dos *cluster* inviabilizar a aplicação do protocolo NICE em um cenário real devido à alta carga imposta aos *peers*, ele é um bom limite inferior entre os protocolos ALM para os parâmetros de sobrecarga de mensagens de controle e tempo de recepção de *chunks* devido à distribuição de carga aplicada aos *clusters* de maior hierarquia em períodos de pico de entrada de nós na rede.

Pretende-se com este trabalho obter resultados tão competitivos quanto o NICE, porém viáveis de aplicação em um ambiente real.

2.3.3 PPM

O PPM [10] é um protocolo híbrido que combina benefícios de propagação de dados dos mecanismos *pull* e *push*. O uso destes mecanismos está respectivamente relacionado a duas topologias: *mesh* e árvore. A construção destas topologias é feita em duas fases distintas descritas a seguir.

Na primeira fase, o mecanismo trabalha com a entidade *tracker* que tem conhecimento total da rede e a quem os *peers* solicitam a entrada no sistema. Assim que o *peer* solicita a entrada na rede, o *tracker* retorna uma pequena lista contendo alguns candidatos a vizinhos sorteada aleatoriamente. O novo *peer* contactará cada um destes candidatos, montando assim a rede *mesh*. Nesta rede, os dados faltantes podem ser solicitados aos vizinhos (*pull-based*).

Na segunda fase, cada *peer* escolhe entre os seus vizinhos aquele que possui menor número de saltos até o nó fonte na rede sobreposta e o escolhe como pai, construindo assim uma árvore dinâmica. Esta árvore é utilizada para distribuir o fluxo de dados na rede (*push-based*). Um exemplo de árvore (linhas tracejadas) está ilustrado na figura 4. Para definir as ligações desta árvore, considerou-se o menor número de saltos de cada nó até o nó fonte de acordo com as ligações da rede *mesh* (linhas contínuas).

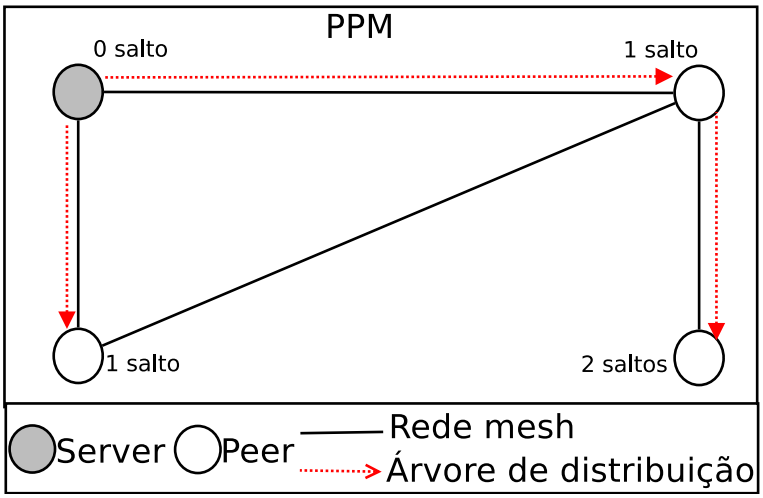


Figura 4 – Exemplo ilustrativo da rede sobreposta criada pelo protocolo PPM.

Contudo, podemos sinalizar que a rede *mesh* montada pode não ser adequada quanto a critérios de desempenho, como menor latência entre os *peers*, implicando logicamente em uma árvore inadequada em termos de desempenho. Outra questão é a organização da árvore de distribuição que considera apenas o menor número de saltos dentre os vizinhos até o nó fonte. Este critério pode exigir alta capacidade de provimento de informação de *peers* que participem do caminho de muitos nós, por exemplo.

Quanto aos critérios de ajuste lógico da rede sobreposta, a solução proposta para este trabalho é dinâmica e considera não o número de saltos como o PPM, mas a latência.

Além disso, pretende-se respeitar as limitações de recursos da rede, sem pressupor que todos os nós possuem garantias de banda de envio disponíveis e ilimitadas.

2.3.4 mTreebone

Outra estratégia possível é a inferência sobre a estabilidade dos nós na rede sobreposta com o intuito de diminuir a latência da propagação dos dados na rede após a saída de algum nó. O protocolo *mTreebone* [25] é um exemplo desta abordagem.

O protocolo *mTreebone* propõe que o desempenho de uma rede sobreposta com grande participação de usuários e voltada para transmissão de vídeo ao vivo depende de um pequeno conjunto de nós estáveis. A ideia é identificar um conjunto de nós estáveis para então construir uma árvore central chamada de *treebone*. O restante dos nós, considerados instáveis, são perifericamente dispostos nesta árvore.

O protocolo sugere que nós com uma idade mais elevada tendem a ficar mais tempo na rede. Para lidar com a questão crítica de como identificar os nós estáveis, utiliza-se um limiar de tempo de vida do nó e os nós acima deste limiar são escolhidos. Uma vez que o nó estável é escolhido, ele permanece nessa função até que saia da rede ou a sessão terminar. Contudo, a efetividade desta escolha nos períodos iniciais da rede ainda é crítica e foi contornada com a introdução de uma promoção aleatória no período inicial da sessão. Existe ainda a necessidade de um conhecimento geral da rede, tornando a solução custosa.

Como o *treebone* não é formado por nós absolutamente persistentes na rede, mas por uma estimativa a partir do tempo de permanência na rede, operações de reparação da estrutura da rede sobreposta não podem ser completamente eliminadas. Com o intuito de melhorar a resiliência e eficiência do *treebone*, todos os nós são organizados em um *mesh overlay* com o intuito de recuperar a árvore de transmissão e que pode vir a ser usado para requisição de dados em atraso pelos nós. Este projeto híbrido da forma *push-pull* é então nomeado como *mTreebone*.

Quando um nó instável deixa a rede, a propagação dos dados no sistema não é afetada ao longo do *treebone*. Por outro lado, se um nó estável deixa a rede (o que acontece com menos frequência), o impacto pode ser atenuado a partir da recepção de conteúdo provisório e rápida eleição de um novo provedor com a ajuda do *mesh overlay*. A figura 5 ilustra o antes e depois da saída de um nó estável e um nó instável da rede sobreposta.

Apesar da rede *mesh* melhorar a recuperabilidade da árvore de distribuição, ela possui o ônus com a troca periódica de informações do estado dos nós vizinhos entre os participantes, como o Narada. Contudo, pretende-se utilizar aqui esta solução de identificação de nós estáveis e com ela reduzir os prejuízos decorrentes de falha súbita de nós na rede.

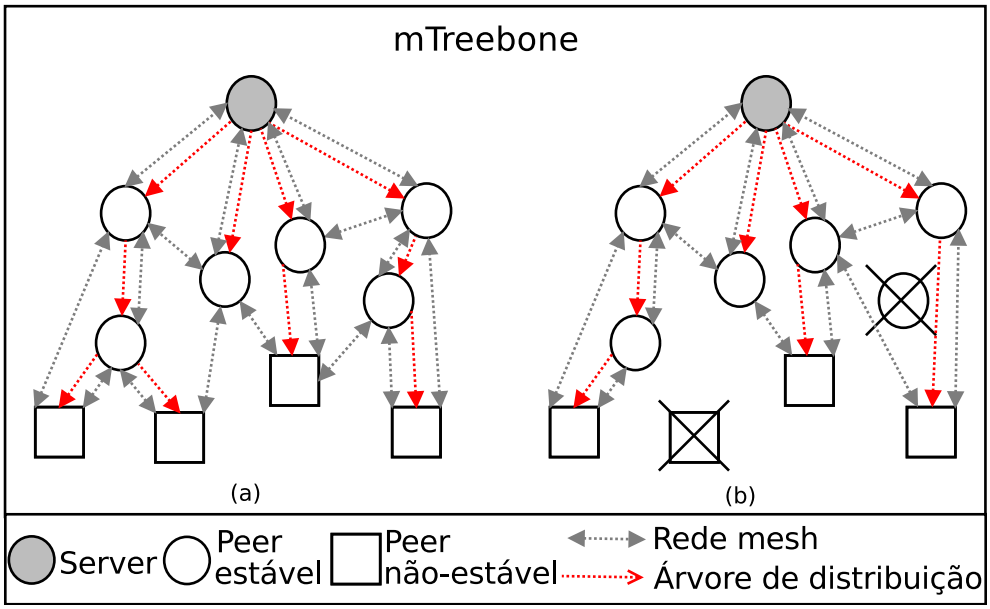


Figura 5 – Exemplo ilustrativo do *mTreeBone*.

2.3.5 PALMS

O protocolo base PALMS, proposto em [14, 15], organiza sua rede sobreposta em uma estratégia *push-based* de fluxo contínuo. Três entidades compõem este protocolo: servidor, *tracker* e *peers*. A seguir são descritas estas três entidades.

O servidor é responsável pela produção e distribuição inicial do conteúdo na rede. A organização do conteúdo se dá pela divisão do fluxo de dados em pedaços de tamanho fixo (*chunks*), lançados na rede sob um fluxo contínuo.

O *tracker* funciona de forma similar ao das redes Bittorrent [6] e aos *Rendezvous Point* (RP) de alguns protocolos ALM [3]. Esta entidade mantém, de forma centralizada, uma lista de *peers* participantes com capacidade de retransmissão. O *tracker* está acessível a qualquer *peer* e desempenha duas funções: (i) auxilia na inserção de novos *peers* e (ii) auxilia na organização da rede.

Um *peer* ingressante recebe do *tracker* um grupo de dados contendo três informações iniciais: fluxo de produção do conteúdo, lista de candidatos a provedores e endereço do servidor. O fluxo de produção do conteúdo auxilia o *peer* a detectar falhas de comunicação e a aplicar as medidas necessárias para a manutenção do fluxo contínuo de recepção e transmissão de conteúdo. A lista de candidatos contém outros *peers* provedores de conteúdo com os quais um nó pode tentar parceria. O endereço do servidor é necessário, já que, na ocorrência de falhas de provimento de informação, o *peer* requisita o conteúdo diretamente ao mesmo com o intuito de evitar interrupções significativas de recepção.

Para auxiliar na organização distribuída da rede, o *tracker* mantém um contador atribuído a cada um dos *peers* diretamente ligados ao servidor. O *peer* pode receber

do *tracker* um novo identificador ou adotar o identificador de seu provedor de conteúdo. Assim, formam-se grupos de nós com o mesmo identificador em cada subárvore com raiz no servidor. Esta informação é usada para evitar laços decorrentes de reingresso na rede, não permitindo acordos entre *peers* com identificadores de grupo idênticos.

Tão logo um *peer* se junte a um grupo, caso possua recursos, ele se reporta ao *tracker* como candidato a provedor. O *tracker* não verifica a situação dos *peers* considerados ativos, sendo responsabilidade da rede reportar sempre que considerar um *peer* inapto a prover conteúdo. *Peers* podem possuir tantos acordos de envio quanto sua capacidade permitir. Contudo, apenas um único acordo de recepção é permitido.

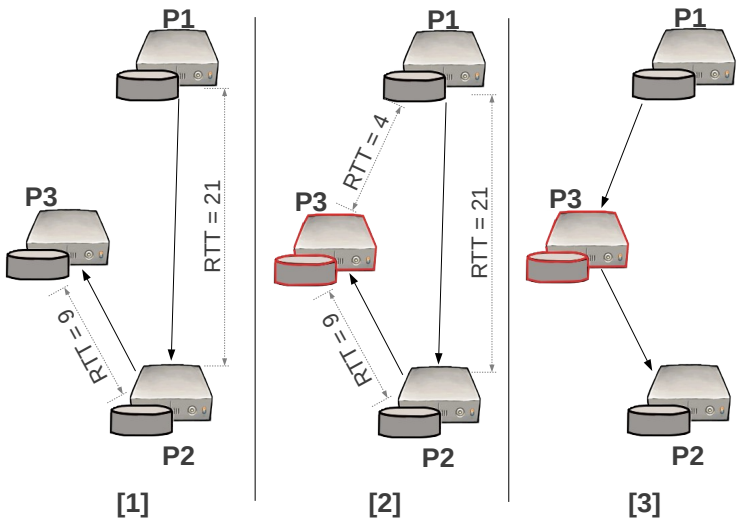


Figura 6 – Execução do mecanismos de desigualdade triangular pelo nó P3.

Periodicamente, cada *peer* requisita atualizações das distâncias, representado pelo tempo de ir-e-vir (RTT - *Round-Trip-Time*), entre ele e seu pai (RTT($p3$,$p2$)), ele e seu avô (RTT($p3$,$p1$)), e seu avô e seu pai (RTT($p1$,$p2$)). De posse destas distâncias, o *peer-filho* julga localmente se neste espaço topológico a desigualdade triangular se verifica, ou seja, se a distância entre seu pai e seu avô é menor que à soma das outras duas distâncias, conforme a equação 2.1:

$$RTT_{(p1,p2)} < RTT_{(p3,p1)} + RTT_{(p3,p2)}. \tag{2.1}$$

A figura 6 ilustra uma condição onde não se verifica tal desigualdade. Nesse caso, articula-se as trocas dos atuais acordos a fim de melhorar a qualidade da comunicação.

Caso o pai não responda a requisição de atualização do RTT por um número fixo de vezes consecutivas definido por parâmetro, o *peer* reporta seu pai ao *tracker* como inapto a provedor de conteúdo, finaliza seu acordo de recepção de dados com ele (declara-se órfão) e busca um novo provedor a partir da lista de candidatos. A esta sequência de passos chamou-se o mecanismo de detecção de órfão. O *tracker*, tão logo receba a informação de

que um dado *peer* está inapto a prover, o retira do grupo de *peers* candidatos a provedores de conteúdo.

Para manter o fluxo de recebimento de conteúdo e amenizar prejuízos na recepção decorrentes de falhas na rede, como a saída não anunciada de provedores de conteúdo, o PALMS possui um mecanismo distribuído chamado *notify-emergency*. Neste mecanismo, um *peer* que não receber o conteúdo do seu provedor no tempo correto (tempo de produção de conteúdo do servidor), o tempo de primeira espera, coloca-se em estado de emergência. Neste estado, o *peer* solicita o conteúdo em atraso ao servidor, envia alertas de atraso aos seus filhos e configura um tempo de segunda espera por conteúdo.

Apesar da solicitação de *chunks* em atraso ao servidor reduzir o tempo de espera, esta solução tende a sobrecarregar o servidor à medida que a taxa de falha na rede aumenta. Os alertas, enviados aos filhos para notificá-los sobre o atraso, evitam que os descendentes se declarem como órfãos de forma sistemática. Após receber algum conteúdo, o *peer* sai do estado de emergência. Mesmo que o conteúdo recebido não seja o conteúdo em atraso, novas requisições não serão feitas.

Para redes com baixo fluxo de entrada e saída de nós (baixo *churn*), esta organização entre o *tracker*, o servidor e os *peers* mostrou-se eficiente, com baixa sobrecarga e com desempenho semelhante ao protocolo Narada em relação ao tamanho do caminho das mensagens da rede sobreposta e a redundância de informações sobre um enlace.

Contudo, para manter a escalabilidade do protocolo em alto *churn*, existem alguns pontos desta organização que devem ser revistos e serão detalhados no capítulo 3.

3 Um novo protocolo ALM - PALMS+

Muitos protocolos ALM atuais possuem dependência de entidades como *peers* confiáveis ou entidades centrais que gerenciam e aglomeram conhecimento global da topologia da rede, criando pontos críticos de falhas. Outro problema observado consiste em desconsiderar dinamicamente a latência entre os *peers* e recursos dos mesmos, podendo gerar um impacto negativo no desempenho do sistema, como possíveis gargalos de transmissão na rede. Ainda, para um ambiente dinâmico como a Internet, também se considera crítico protocolos que supõem que todos os *peers* contribuem para a rede, quando, na verdade, não existe garantia de banda disponível em todos os nós [21].

Segundo [20], no futuro a Internet deve estar apta a distribuir vídeo em alta qualidade em um caminho eficiente, flexível e personalizável, além de um ambiente heterogêneo e dinâmico. O protocolo apresentado nesta dissertação, nomeado de PALMS+, tem por finalidade atender de forma escalável esta demanda com uma solução simples, de pouco custo e articulada em uma rede sobreposta que pode ou não ter a presença de *peers* confiáveis.

Neste trabalho, criamos um novo protocolo ALM que ultrapassa os limites existentes no PALMS. Alem disso, aumenta escalabilidade, mesmo em alto *churn*. O novo protocolo é simples e eficiente, tem eficiência comparável aos protocolos PALMS e NICE, como será mostrado no capítulo 5.

Como mencionado na seção anterior, o foco das mudanças está em tratar a obsolescência da lista de candidatos a provedores, reduzir a sobrecarga indireta do *server* ao prover conteúdo em atraso e recuperar os *chunks* que por ventura não foram recebidos.

3.1 Lista de candidatos a provedores

No protocolo PALMS, os *peers* que ingressam com sucesso na rede, reportam-se ao *tracker* como ativos. Estes *peers* ativos são considerados pelo *tracker* como potenciais provedores de conteúdo. Contudo, não há referências do comportamento dos *peers* que não possuem capacidade colaborativa de distribuição. Para o novo protocolo proposto, apenas os *peers* com capacidade colaborativa se reportam ao *tracker*.

Sempre que um novo *peer* solicita seu ingresso na rede, o *tracker* efetua um sorteio de 32 candidatos que considera ativos. O tamanho desta lista foi estipulado experimentalmente para a aplicação simulada, com o intuito de não haver reenvio da lista durante a simulação. O sorteio dos *peers* é aleatório, sem qualquer política de seleção adicional. Entretanto, caso a rede esteja submetida a alto *churn*, há um aumento da probabilidade dos candidatos contidos nesta lista não pertencerem mais à rede com passar do tempo.

Desta forma, um *peer* órfão, que tente encontrar um novo candidato apto em sua lista, tem como única alternativa ligar-se ao servidor. Para evitar que esta lista de candidatos se torne obsoleta para diferentes cenários de simulação, não é viável seguir o exemplo do protocolo original e definir um tamanho de lista que se adeque a cada aplicação simulada. Assim, duas alterações são propostas para tornar dinâmica esta adequação do tamanho da lista ao cenário simulado: 1) redução da probabilidade de sorteio de *peers* recém inseridos na rede como candidatos a provedores de outros *peers*; 2) possibilitar envios sob demanda da lista contendo novos candidatos.

Para o PALMS+, considerando o comportamento de usuários em sistemas de distribuição de vídeo, estipulou-se diferentes probabilidades de sorteio entre *peers* recentes e antigos para compor a lista de candidatos. *Peers* antigos possuem 9 vezes mais chances de sorteio do que *peers* recentemente inseridos na rede. Esta diferenciação entre recente e antigo foi definida a partir de um limiar referente ao tempo de atividade do *peer* na rede. Desta forma, *peers* mais estáveis possuem uma alta probabilidade de serem escolhidos, proporcionando uma maior estabilidade na formação topológica da rede.

Esta proposta de diferentes probabilidades de sorteio prioriza os nós há mais tempo na rede. É uma solução mais aceitável que o proposto pelo protocolo *mTreebone*, por exemplo, que descarta a capacidade colaborativa de *peers* com grande disponibilidade de recursos (largura de banda) apenas por estarem a um curto período na rede.

3.2 Sobrecarga indireta do *server* e recuperação de *chunks* em atraso

A sobrecarga no *server* com a distribuição de conteúdo pode ser dividida em duas naturezas: carga direta e carga indireta.

A carga direta é o trabalho na distribuição de conteúdo aos *peers* que se ligam diretamente ao *server*. Contudo, existe uma diminuição desta carga com a adoção de *peer* mais estáveis para proverem o conteúdo e a possibilidade de requerer uma nova lista de candidatos atualizada proposto na seção 3.1.

A carga indireta é decorrente do provimento de conteúdo em atraso advindo de requisições de *peers* da rede que por ventura não receberam seu conteúdo no tempo correto. Estas requisições têm por intuito amenizar os prejuízos decorrentes de falhas súbitas na rede e são parte do mecanismo *notify-emergency*, relacionado ao estado emergencia.

Foram feitas modificações neste mecanismo para aumentar o aspecto colaborativo do protocolo, reduzir as requisições de informações em atraso ao *server* e reduzir o índice de perda de conteúdo devido a falhas na rede. Todas estas diferenciações entre o PALMS e o PALMS+ estão apresentadas na figura 7 e são descritas em detalhes abaixo.

Para aumentar o aspecto colaborativo de difusão de conteúdo em atraso do protocolo, foram selecionados alguns *peers* especiais que auxiliem nessa tarefa. Contudo, como este

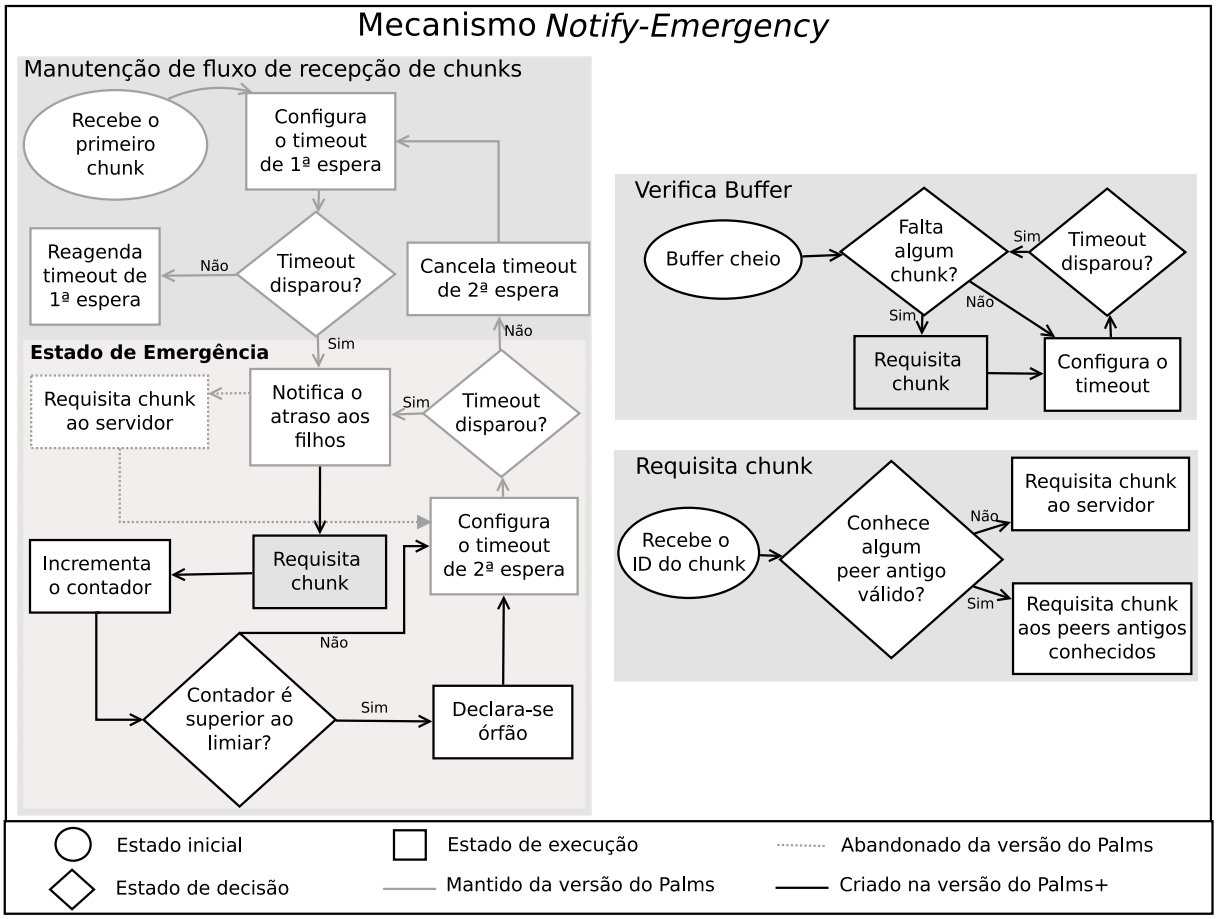


Figura 7 – Mudanças efetuadas no protocolo PALMS para aumentar o aspecto colaborativo do protocolo PALMS+

protocolo tem por premissa atuar sob *peers* não confiáveis, não é prudente remover o *server* desta tarefa. Assim, optou-se por direcionar as requisições a *peers* ativos há mais tempo na rede por provavelmente possuírem um mapa de *chunks* mais completo. Caso as requisições não sejam atendidas, elas são redirecionadas ao *server*.

O *tracker* é responsável por compor essa lista de *peers* especiais que serão alvos de requisição. A composição da lista se faz por sorteio entre os *peers* antigos, sendo enviada ao ingressante junto às três informações iniciais.

Para os casos em que esta lista estiver vazia ou as requisições em atraso do estado de emergência não forem respondidas por um número fixo de vezes consecutivas (limiar do estado de emergência), o *server* passa a ser o alvo da requisição desse conteúdo emergencial. Além disso, os *peers* antigos são reportados ao *tracker* como inaptos a abastecer conteúdo na rede e uma nova solicitação da lista de *peers* especiais é efetuada. Na figura 8 é possível ver como os *chunks* em atraso eram requisitados no PALMS e como são requisitados no PALMS+.

Como a requisição do conteúdo em atraso foi redirecionada a *peers* não confiáveis, outra função foi acrescida ao mecanismo *notify-emergency* para evitar possíveis prejuízos

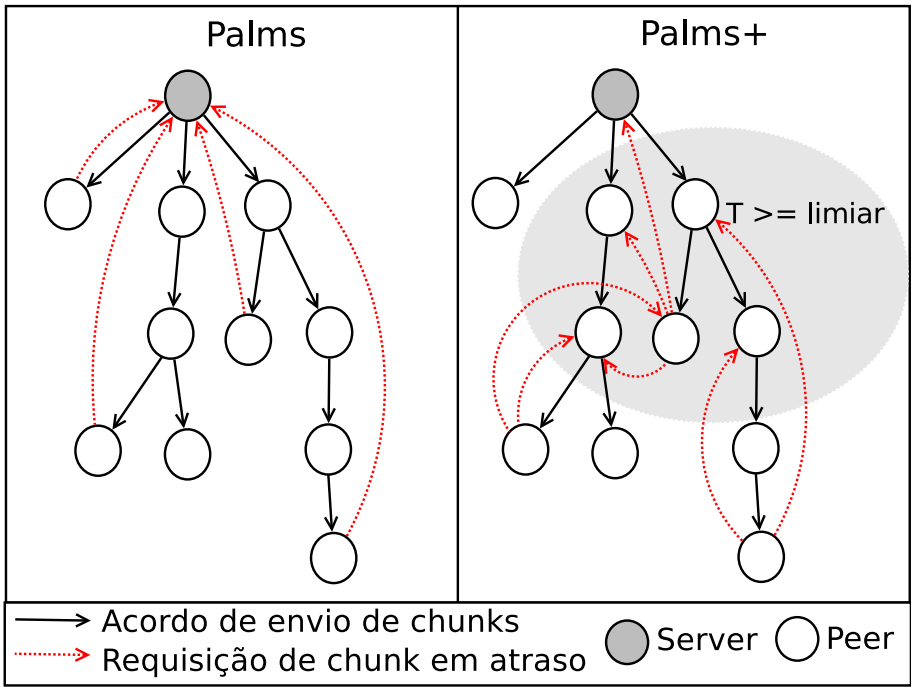


Figura 8 – PALMS versus PALMS+.

decorrentes da não recepção de conteúdo neste modelo. Acrescentou-se um mecanismo de constante verificação de *chunks* faltantes no *buffer*. Nele, caso se constate a falta de algum *chunk* na janela de transmissão, o mesmo é requisitado à lista de *peers* antigos ou ao *server*.

Por fim, incluiu-se um segundo mecanismo de detecção de estado de órfão. Agora, além de um nó declarar-se órfão quando não obtiver a resposta de atualização do RTT de seu pai, ele também declara-se órfão após o limiar do estado de emergência ser ultrapassado. Com isto espera-se melhorar a recuperabilidade após falhas súbitas na rede.

4 Metodologia de avaliação

Este capítulo descreve a metodologia utilizada para coleta de dados e análise dos resultados obtidos. O simulador utilizado é descrito na seção 4.1. As métricas de desempenho são descritas na seção 4.2. Os cenários utilizados nas simulações são apresentados na seção 4.3. A metodologia para coleta de dados é descrita na seção 4.4. Finalmente, na seção 4.5 os parâmetros do protocolo PALMS+ são ajustados em um cenário de *churn* controlado, visando o balanceamento entre sobrecarga de controle e manutenção da rede sobreposta em tempo real. Os valores para os parâmetros foram utilizados na configuração do protocolo PALMS.

4.1 OMNET++

O OMNET++[1] é um simulador de eventos discretos, gratuito, multiplataforma e com código fonte aberto. Sua arquitetura é baseada em componentes desenvolvidos em C++. O OMNET++ é indicado para simulações onde eventos discretos ocorrem, como na modelagem de tráfego e filas de redes de telecomunicações, modelagem de multiprocessamento e outros sistemas de *hardware* distribuído, e validação de protocolos de rede.

Devido a sua construção modular, esta plataforma permite a abstração de eventos que contribuem para a melhor validação dos resultados obtidos nas simulações. Por exemplo, a entrada e saída de *peers* (*churn*) em protocolos ALM pode ser feita a partir de arquivos de entrada originados de sistemas reais ou a partir de modelos pré-definidos.

O OverSim[2] é um dos principais modelos de simulação baseado no OMNET++. Mantido por um grupo independente de pesquisa, o *Institute of Telematics*[3], ele permite a implementação de sistemas P2P e protocolos de uma rede sobreposta. Com um esquema flexível de configuração da rede real, os sistemas implementados nesta plataforma podem utilizar de um simples e rápido modelo de construção da topologia real (*SimpleUnderlay*), com a latência entre os participantes modelada com base em medições reais advindas do projeto CAIDA/Skitter[4]); ou uma rede totalmente ajustada (*INETUnderlay*), com larguras de banda, atrasos e perdas de pacotes configurados na simulação.

As simulações efetuadas neste trabalho foram obtidas através de experimentos realizados na plataforma de simulação OverSim. Os protocolos PALMS e PALMS+ foram implementados nesta dissertação e o protocolo NICE por [13].

[1] http://www.omnetpp.org/
[2] http://www.oversim.org/
[3] http://telematics.tm.kit.edu
[4] http://www.caida.org/tools/measurement/skitter/

4.2 Métricas de desempenho

Nesta seção formalizamos as métricas de desempenho utilizadas para comparar os protocolos ALM estudados nesta dissertação.

O conjunto de nós de uma rede sobreposta é representado através de um grafo orientado $G = (V, E)$, em que os participantes de uma transmissão são representados pelos vértices (V) e as conexões e parcerias entre eles são as arestas (E).

Considere os vértices $u, v \in V(G)$. Um caminho l é um passeio sem repetição de vértices, onde $\forall i \neq j, v_i \neq v_j$, e $l \in L(G)$. Assim, se existe um passeio $P = (v_i, v_j)$, então os vértices v_i e v_j são os extremos de P e v_j é alcançável por v_i a partir de uma sequência finita de vértices conectados par a par por arestas. O conjunto de arestas pertencentes a P é denotado por $E(P)$.

Nos grafos ponderados, o indicador de distância entre dois vértices v_i e v_j é obtido pelo custo da conexão entre eles a partir da função peso, $w : E \longrightarrow \mathbb{R}$. O peso de uma aresta e, $w(e)$, é utilizado para caracterizar a importância desta aresta no grafo. Caso $P = (v_i, v_j)$ possua mais de uma aresta interna, o custo do caminho $l(v_i, v_j)$ será:

$$w(l(v_i, v_j)) = \sum_{\forall e \in E(P)} w(e) \tag{4.1}$$

Diversas informações podem ser usadas para definição de custo do enlace, como largura de banda em uso, taxa de perda de pacotes e latência.

Como o critério adotado pelo protocolo para definir o caminho percorrido pelos dados influencia na qualidade do serviço fornecido e no impacto do uso dos recursos da rede. Associado aos caminhos criados, é possível comparar a qualidade de protocolos baseado no custo médio da informação associada ao enlace na rede.

Seja u um vértice pertencente a $V(G)$ que alcança todos os demais vértices v_i também pertencentes a $V(G)$. O custo médio de uma topologia de distribuição de dados *pushed-based* pode ser definido como a soma dos pesos de todos estes caminhos l_i, tal que $l_i = l(u, v_i)$ e $1 \leq i \geq |L|$.

$$CustoMedio = \sum_{\forall l \in L} w(l_i)/|L| \tag{4.2}$$

Outra questão relacionada a uma rede sobreposta é o custo para manutenção da topologia criada. O custo de manutenção atribuído a um nó v será a soma dos pesos das arestas adjacentes a este nó, alvos do envio de mensagens de controle. O conjunto de vértices adjacentes a v é denotado vizinhança de v, representado por $S(v)$. A soma dos pesos das arestas adjacentes a v é denominada grau do vértice v, que é notado por $d(S(v))$.

O custo total da rede com envio de mensagens de controle é:

$$CustoControle = \sum_{\forall v \in V} d(S(v)) \tag{4.3}$$

A figura 9 mostra os custos das arestas de um grafo G em árvore. A partir desta figura, pode-se dizer que o custo médio ($CustoMedio$) de alcance a todos os nós a partir do nó 0, considerando que cada aresta possui custo igual a 1, é 1.8 para o protocolo 1 e 2.8 para o protocolo 2. Contudo, caso a análise desse grafo representasse os recursos consumidos pela rede com mensagens de controle, o custo ($CustoControle$) seria igual a 5 para os protocolos 1 e 2.

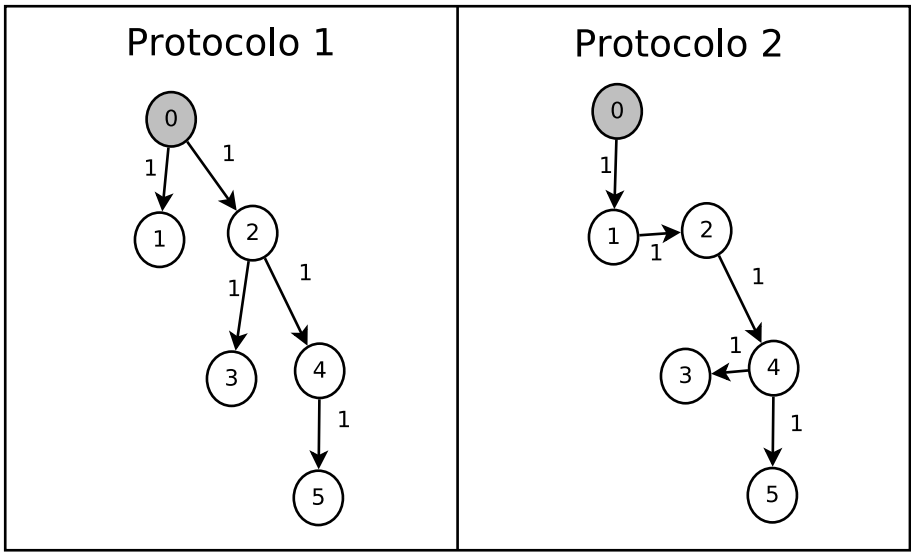


Figura 9 – Grafos de custos de duas redes em árvore diferentes.

A informação a ser transmitida na rede sobreposta normalmente é dividida em fatias de conteúdo de tamanho fixo (*chunks*). O conjunto de *chunks* gerados é aqui nominado de C. Para cada recepção de *chunk* c_k com sucesso em um nó v_j pertencente ao grafo G, tal que $1 \leq k \leq |C|$ e $k \in \Re$, existe um nó emissor v_i em que $j \neq i$.

A quantidade de vezes que um nó receptor v_j recebeu uma fatia c_k é dada pela função $rv_j(c_k)$. $R(c_k)$ representa a quantidade total de vezes que um *chunk* foi recebido de forma não única, ou seja, a quantidade de vezes que algum nó receptor v recebeu um dado *chunk* de forma repetida.

$$R(c_k) = \sum_{\forall v_j \in V} rv_j(c_k) - 1, \forall rv(c_k) > 0 \tag{4.4}$$

O alcance de um *chunk* c_k na rede, $A(c_k)$, é dado pelo total de nós v diferentes que o receberam:

$$A(c_k) = (\sum_{\forall v_j \in V} r_v j(c_k)) - R(c_k) \tag{4.5}$$

Para a avaliação dos protocolos PALMS, PALMS+ e NICE, as métricas latência média ($CustoMedio$), sobrecarga de mensagens de controle total da rede ($CustoControle + R(c)$) e alcance dos *chunks* ($A(c)$) foram utilizadas.

A latência refere-se à média de tempo da primeira recepção de cada *chunk* pelos *peers*. Cada *chunk* tem seu tempo inicial definido no momento de criação pelo servidor e o tempo final no momento da recepção por cada *peer*. Para que a qualidade desta métrica não seja comprometida por eventuais retransmissões que distorçam a média, o *buffer* armazena apenas os últimos *chunks* recebidos. Os demais são contados como *chunks* repetidos.

A sobrecarga de controle que a rede sobreposta é submetida, em KBytes, consiste na carga média de envio de mensagens de controle ($CustoControle$) e no montante de *chunks* repetidos recebidos (soma de todos os $R(c_k)$).

O alcance de um *chunk* na rede mede a quantidade de *peers* que o receberam ao menos uma vez durante a simulação. Ao analisar cada *chunk*, mede-se sua confiabilidade na entrega dos dados e a escalabilidade da rede.

4.3 Cenários avaliados

As comparações do PALMS+ com o PALMS e NICE consideram dois cenários de *churn* distintos: um cenário sintético de *churn* controlado (cenário 1) e um com *churn* baseado em um ambiente real (cenário 2).

O primeiro cenário, com entrada e saída de *peers* controladas, foi utilizado para ajustar os parâmetros dos protocolos PALMS e PALMS+. O NICE não foi ajustado nesta estapa. Os parâmetros utilizados no NICE foram extraídos do trabalho [13] que o ajustou para alto *churn*. Este cenário também foi utilizado para comparativamente analisar o comportamento de todos os protocolos. O *churn* controlado permite observar os tempos de entrada, de saída e de estabilidade (sem entrada ou saída) da rede.

O segundo cenário foi utilizado para analisar o desempenho dos protocolos, observando a escalabilidade e confiabilidade da entrega dos dados especificamente sob alto *churn*.

4.3.1 Cenário 1: cenário de *churn* controlado

No cenário 1, proposto por [14], foram utilizados 1024 *peers*, incluindo o *server* e o *tracker*. Os *peers* foram inseridos e removidos em rajadas e em intervalos distintos. O evento de inserção ocorreu nos primeiros 200 segundos, no qual 128 nós foram inseridos a cada 8s até que a rede atingisse o seu máximo pretendido. O evento de remoção, por sua vez, ocorreu aos 346s com a retirada súbita de 128 peers aleatórios. Após estes dois eventos, a simulação seguiu até os 3600s com os 896 *peers* restantes.

Neste cenário, cada rodada de simulação teve duração de 1 hora, no qual os eventos de inserção, quebra e reestabelecimento/normalização da rede foram divididos ao longo da simulação. O isolamento do *churn* permite avaliar melhor o comportamento e parametrização do protocolo.

4.3.2 Cenário 2: cenário de *churn* real

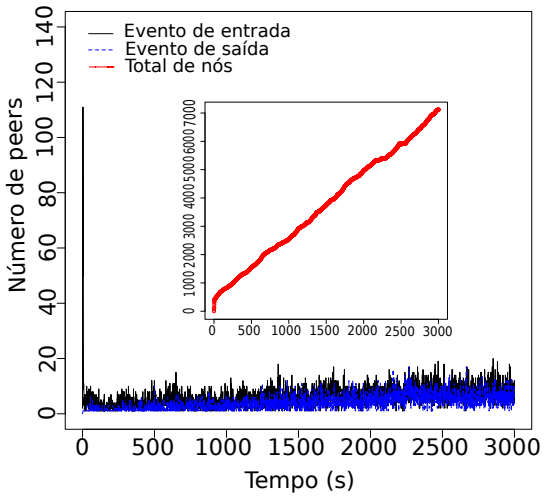


Figura 10 – *Churn* do cenário real utilizado.

O cenário 2 utilizado neste trabalho segue o registro de tráfego coletado de uma transmissão de vídeo ao vivo em *http* realizada em 2013 por um importante canal de TV brasileira.

A Figura 10 apresenta os processos de entrada e saída de *peers* referentes ao cenário real (Copa das Confederações). Observamos um número maior de *peers* que se junta ao sistema, em comparação aos que abandonam. De fato, a figura interna mostra uma crescente população de *peers* no sistema. A taxa de *churn* médio de entrada é igual a 6,19 *peers*/segundo e de saída igual a 3,82 *peers*/segundo. Ao final da simulação, a população total de *peers* ativos foi de 7.123.

A escolha deste cenário está relacionada à alta demanda de interesse dos usuários pelo conteúdo transmitido, uma vez que se tratava de um evento esportivo ocorrido na Copa das Confederações - torneio de futebol organizado pela FIFA[5] (Federação Internacional de Futebol Associado) entre as seleções nacionais a cada quatro anos.

4.4 Coleta de dados

Cada cenário simulado foi executado 10 vezes. Variou-se a semente de 0 a 9 em cada execução para evitar que os resultados decorrentes de sorteios fiquem enviesados nas simulações. A coleta dos dados foi efetuada a cada 6 segundos e os resultados apresentados correspondem aos valores médios destas coletas com confiança de 95% quando necessário.

O intervalo entre *chunks* no servidor é igual a 1,5 s para que os *peers* tenham tempo suficiente para recepção e requisição de *chunks* antes da produção de novos, como proposto em [14, 15]. Durante a simulação, caso os 1,5 s se esgotem e o *peer* não receba o *chunk* esperado, o mesmo ficará em estado de emergência por mais alguns segundos aguardando o *chunk*. O tempo que o *peer* fica nesta segunda espera está relacionado ao parâmetro *emergency timeout* e será definido em simulação na seção 4.5. Sobre a permanência em segunda espera antes de declarar-se órfão, estipulou-se o limiar de duas vezes consecutivas. Este valor foi escolhido para conceber maior recuperação ao *peer* em uma rede sobreposta com alto *churn*.

As configurações gerais de todas as rodadas de simulação em relação à capacidade de recepção e envio de *chunks* pelos *peers* são baseadas nas capacidades próximas a ambientes heterogêneos como a Internet. Assim, ao *peer* recém criado é atribuído um grau limite de envio uniformemente distribuído no intervalo de variação de 0 a 8 filhos de acordo com medições presentes em [22]. O protocolo não lida com o fato do servidor estar acima de sua capacidade de provimento de dados, sendo este o único nó da rede que possui capacidade ilimitada de envio de dados.

Para a simulação de perdas de *chunks*, um *chunk* é descartado com probabilidade de 1% no momento da recepção do mesmo. Este é um valor aceitável de perda de pacotes para situações normais. Valores entre 2% e 4% são classificados como alarmantes e acima de 4% como críticos, segundo dados fornecidos pela Keynote Network [18]. Mensagens de controle são confiáveis e não são perdidas.

4.5 Definição de parâmetros dos protocolos

A escolha deste conjunto de parâmetros tem como intuito realizar um estudo comparativo, por meio de protocolos presentes na literatura que possuam resultados

[5] http://www.fifa.com/

competitivos, em relação à sobrecarga da rede e tempo de recepção dos dados (e.g. protocolo NICE e PALMS).

O protocolo NICE foi ajustado para alto *churn* em [13] com foco em tempo de recepção e sobrecarga na rede. Apenas os parâmetros do protocolo PALMS+ serão ajustados experimentalmente neste trabalho e os valores obtidos serão replicados ao protocolo PALMS.

Os parâmetros que impactam diretamente na sobrecarga de controle e tempo de recepção do conteúdo do PALMS+ são o *update* e o *emergency timeout* e serão definidos experimentalmente na seção 4.5.1.

O *update*, relacionado com a verificação da desigualdade triangular, é responsável pelo intervalo de medição dos RTTs entre os nós. Seu ajuste tem influência direta na quantidade de mensagens de controle disparadas e, consequentemente, no uso de banda para o envio das mesmas.

Já o *emergency timeout*, relacionado com a requisição de *chunks* em atraso e detecção de estado de orfandade no estado de emergência. Este parâmetro define o tempo do estado de emergência.

define o tempo no estado de emergência, onde *chunks* em atraso são requisitados e alertas de atraso de envio *chunks* são disparados aos *peers* descendentes. Desta forma, seu ajuste tem influência no tempo de recepção de *chunks* e no montante de largura de banda usado para envio de mensagens de controle.

Um parâmetro acrescido aos protocolos foi o tamanho do *buffer*. Segundo [4], um *buffer* deve guardar somente entre 10 e 100 segundos de vídeo. Sendo assim, o tamanho do *buffer* para uma taxa de distribuição de 1.5 s deve ficar entre 7 e 67 posições de armazenamento de *chunks*. Como os protocolos trabalham com transmissão de fluxo contínuo, não são indicados armazenamentos longos. Contudo, para captar possíveis repetições de recepção de *chunks* na rede, optou-se por um tamanho de *buffer* de 32 posições. Assim, sempre que um novo *chunk* é recebido, o *chunk* com o identificador mais antigo é removido e o novo adicionado no *buffer*. Caso o *chunk* já tenha sido recebido antes, ele entrará na contagem de sobrecarga da rede.

O PALMS+ ainda verifica constantemente se existem *chunks* ausentes em seu *buffer*. A taxa de verificação foi estipulada em duas vezes o tempo da taxa de transmissão do servidor, uma vez que um *chunk* não recebido já é requisitado imediatamente após a primeira espera (taxa de transmissão do servidor) tanto no PALMS quanto no PALMS+.

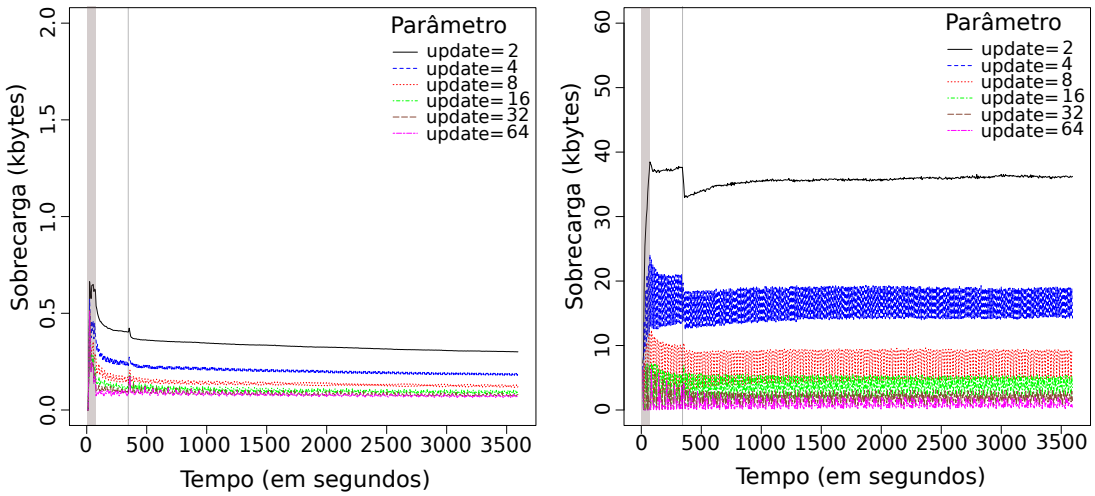
O conteúdo propagado nas simulações foi composto de dois tipos de mensagens: *chunks* de dados e mensagens de controle. Os *chunks* foram representados por mensagens de tamanho fixo de 1KB [6]. Eles contêm um marcador temporal para capturar o tempo médio de recebimento, um ID sequencial para diferenciá-los e um parâmetro para contagem

de saltos visando definir a altura da árvore na rede sobreposta. As mensagens de controle possuem tamanho variado, calculado segundo o mínimo de campos necessários para o envio de todas as informações relevantes.

4.5.1 Definição de parâmetros do protocolo PALMS+ com o cenário 1

A seguir, definimos experimentalmente os melhores valores dos parâmetros que impactam diretamente na sobrecarga de controle e latência do conteúdo: *update* e *emergency timeout*. O *update* é responsável pelo intervalo de medição dos RTTs entre os nós e a posterior execução do mecanismo de desigualdade triangular (MDT). Seu ajuste tem influência direta na quantidade de mensagens de controle disparadas e, consequentemente, no uso de banda para o envio das mesmas. Já o *emergency timeout* define o tempo no estado de emergência, onde *chunks* em atraso são requisitados e alertas de atraso de envio *chunks* são disparados aos descendentes. Desta forma, seu ajuste tem influência no tempo de recepção de *chunks* e no montante de largura de banda usado para envio de mensagens de controle.

4.5.1.1 Parametrização do *update*



(a) Sobrecarga nos *peers*. (b) Sobrecarga no servidor.

Figura 11 – Sobrecarga de banda usada com mensagens de controles.

O parâmetro *update* está relacionado à execução da desigualdade triangular (MDT), sendo portanto um importante parâmetro de ajuste para a resposta do sistema às variações de latência, como as ocasionadas pela sobrecarga de um *peer*. Entretanto, apesar de intervalos curtos de atualização do RTT conferirem maior sensibilidade ao MDT, eles também implicam em maior sobrecarga de uso de largura de banda com mensagens de controle. A figura 11 mostra a sobrecarga de controle no sistema para diferentes valores do parâmetro *update*, segundo uma função exponencial de base 2.

A tabela 1 mostra as piores medições de desempenho no tempo de recepção de *chunks* obtidas nos eventos de entrada e saída de *peers* na rede para cada um dos valores do parâmetro *update*. É possível perceber um ponto inflexão de desempenho para o parâmetro igual a 8 no evento de entrada de nós na rede, sendo este o melhor parâmetro para este evento. Já no evento de saída dos nós, há uma relação inversa entre o crescimento do *update* e a queda de desempenho no evento de saída, atingindo uma variação de tempo de até 548,18%. Esta relação é justificável, uma vez que a falha de atualização do RTT é um mecanismo usado para detecção da falta de recebimento de informação e posterior declaração como órfão por parte do *peer*.

Update(s)	Latência máxima (ms)			
	Evento de chegada		Evento de saída	
	Média	IC 95%	Média	IC 95%
2	1555,9	1555,1-1556,7	430,4	430,4-430,5
4	1190,1	1189,2-1191,0	440,0	440,0-440,1
8	1172,2	1171,3-1173,0	537,1	537,0-537,2
16	1610,8	1609,7-1611,9	728,0	727,8-728,2
32	1526,9	1525,8-1528,0	1370,4	1370,0-1371,0
64	1606,7	1605,8-1607,5	2790,0	2789,0-2791,1

Tabela 1 – Tempo máximo de recepção de *chunks*.

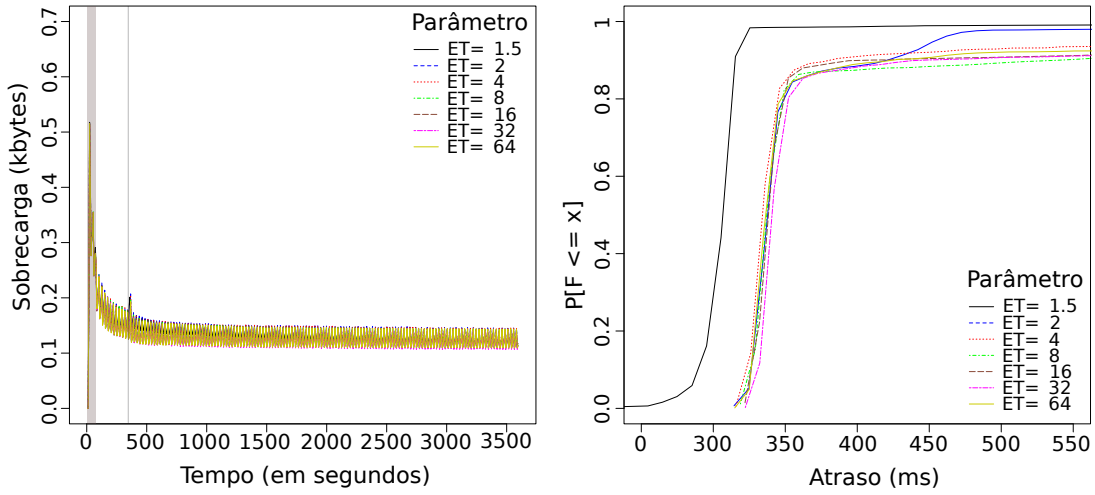
Para o ajuste dos parâmetros do protocolo com o foco em adaptá-lo a um cenário real, consideramos o período de *churn* do cenário. Tendo em vista este período e sua relação com o desempenho em latência na entrega dos *chunks*, a escolha do *update* ficaria entre os valores 2, 4 e 8 no evento de saída e com o valor 8 no evento de entrada. Contudo, ao observar a sobrecarga de controle (figura 11), o valor 8 mostrou melhores resultados. Assim, segundos os experimentos realizados, o melhor valor para o parâmetro de *update* é de 8 segundos.

4.5.1.2 Parametrização do *emergency timeout*

Para manter o fluxo correto de recepção de conteúdo, como em períodos de falha na rede por saída não anunciada de nós provedores de conteúdo, é intuitivo que o parâmetro *emergency timeout* (ET) esteja sincronizado ao valor da taxa de produção de *chunks* pelo servidor. Contudo, é necessário verificar o impacto em sobrecarga de controle dessa configuração. Assim, além do valor inicial referente à taxa de produção de *chunks*, para uma gama maior de análise da influência deste parâmetro, considerou-se também valores esparsos obtidos a partir de uma função exponencial de base 2.

A figura 12(a) apresenta a sobrecarga de controle nos *peers*. As áreas destacadas entre os tempos 0 e 64 segundos e 346 segundos de simulação indicam os períodos de inserção e remoção dos *peers* no sistema, respectivamente. É importante notar que a

solicitação de *chunks* em atraso não é um fator impactante na sobrecarga de mensagens de controle. A sobrecarga de controle se manteve muito próxima em todos os períodos da simulação: inferior a 0,6 Kbytes no evento de entrada dos 1024 *peers*, aproximadamente 0,2 Kbytes no evento de saída dos 128 *peers* e se estabilizando abaixo de 0,2 Kbytes no período estável, com 896 *peers* no sistema.



(a) Sobrecarga nos *peers*. (b) Tempo de recepção de *chunks*.

Figura 12 – Sobrecarga de controle e tempo médio de recepção de conteúdo na rede (estado de emergência).

Assim, a escolha da configuração do parâmetro ET fica a cargo apenas da latência de recepção dos *chunks*. Na figura 12(b) verifica-se que este parâmetro é melhor configurado segundo o intervalo de produção de *chunks* (ET=1,5s), atingindo um tempo médio de recepção de *chunks* de 309,83ms com desvio padrão de 0,3. Este valor é 50,52% melhor que o pior caso.

Este valor de configuração de 1,5 s é intuitivo, uma vez que se deseja com este parâmetro definir o tempo de requisição de *chunks* em atraso e a manutenção do fluxo de distribuição de conteúdo em períodos de instabilidade do sistema. Assim, no caso do *peer* responsável por prover conteúdo não anunciar a sua saída, o estado de emergência em que os filhos se colocam deve tentar reproduzir o fluxo correto de transmissão, requisitando os *chunks* em atraso até que um novo provedor seja encontrado.

5 Resultados experimentais

Os experimentos a seguir foram realizados conforme a metodologia de captura descrita na seção 4. Nesta seção, avaliamos o desempenho dos protocolos PALMS, PALMS+ e NICE ao prover o serviço de *streaming* em um cenário controlado e baseado em um cenário real.

5.1 Avaliação do Protocolo PALMS+

A avaliação do protocolo foi dividida em duas fases: cenário de *churn* controlado e cenário de *churn* real. Em ambos os casos, comparamos o PALMS+ com o PALMS e NICE. No cenário controlado é possível avaliar separadamente os desempenhos dos eventos de entrada e saída *peers*, e estabilidade da rede. O cenário real permite submeter ambos os protocolos a variáveis estressantes, tais como um grupo grande de *peers* em alto *churn* com o intuito de observar a robustez do protocolo.

5.1.1 Resultados do cenário controlado

A figura 13 compara o tempo de recepção de *chunks* entre PALMS, PALMS+ e o NICE. O PALMS obteve uma média de latência de recepção dos *chunks* de 0,28s, o PALMS+ de 0,34s e o NICE de 0,20s. Neste sentido, mesmo o NICE obtendo o melhor resultado, os valores absolutos das médias não refletem diferenças significativas entre os três protocolos neste cenário. Analisando ainda a diferença das médias, com 95% de confiança, a latência do protocolo PALMS+ fica entre 0,06s e 0,14s a mais do que o protocolo PALMS. Já o comparando ao NICE, esta diferença fica em aproximadamente 0,14s. Apesar das diferenças de latência, os valores do PALMS+ são aceitáveis.

Por outro lado, a partir da figura 14, pode-se observar que o PALMS e PALMS+ possuem os menores picos de sobrecarga nos períodos de entrada de *peers*. As médias absolutas de sobrecarga da rede em Kbytes para os *peers* foi de 0,91 no NICE, 2,10 no PALMS e 2,28 no PALMS+, já para o server foi de 1,82 no NICE, 7,44 no PALMS e 7,88 no PALMS+. O pior desempenho do NICE para esta métrica pode estar associado com a verificação recursiva para a criação inicial dos *clusters*. Assim, existindo muitos nós e poucos *clusters*, mais comparações na busca de menores tempos entre os nós serão feitas.

Quanto ao período sem *churn* na rede e em relação à sobrecarga nos *peers*, o PALMS e o NICE destacam-se por rapidamente estabilizarem em baixa sobrecarga. O PALMS+, por outro lado, tem um ligeiro crescimento de sobrecarga no período sem *churn*. Pode-se atribuir este ligeiro crescimento do PALMS+ em relação à sua versão anterior, o PALMS, a mudança da política de requisição de *chunks* em atraso. Isto pode indicar que, em baixo *churn* da rede, o servidor é capaz de sozinho atender estas requisições. Estas

requisições, neste período sem *churn* são decorrentes de troca de acordos de colaboração entre os *peers* após a execução do mecanismo de desigualdade triangular.

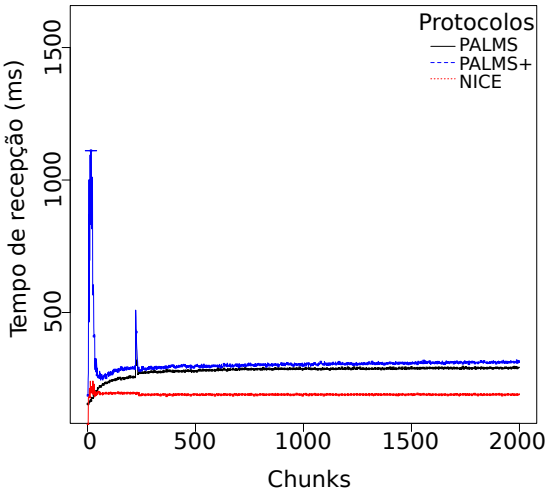
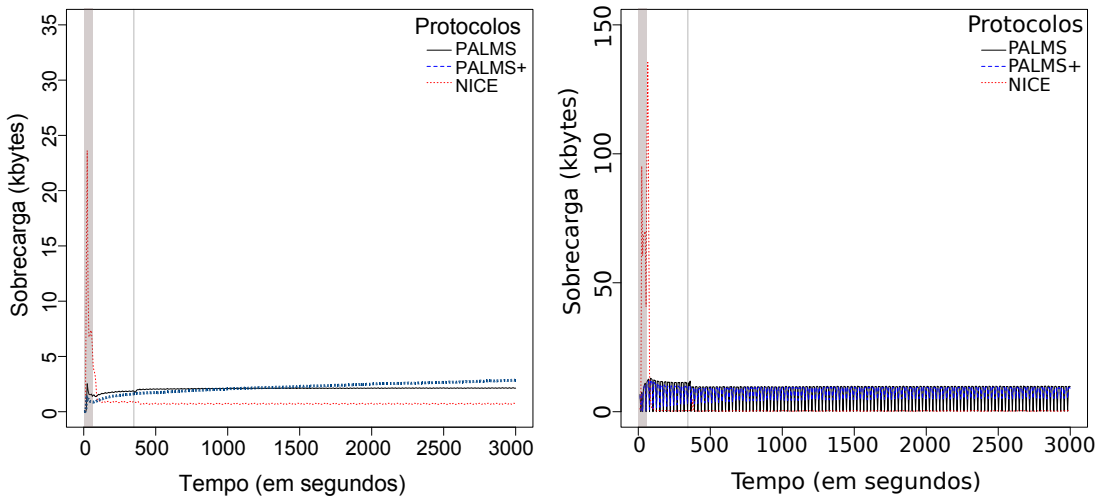


Figura 13 – Cenário Controlado - Tempo médio de recepção de *chunks*



(a) Sobrecarga nos *peers*. (b) Sobrecarga no *server*.

Figura 14 – Cenário Controlado - Sobrecarga de banda com mensagens de controle.

5.1.2 Resultados do cenário real

Devido à natureza de melhor esforço da Internet, descartes de pacotes são inevitáveis devido a congestionamentos. Contudo, perdas de pacotes em rajadas têm um efeito devastador sobre a qualidade de *streaming* de vídeo. Na figura 15 é possível perceber que, exceto o PALMS, os protocolos avaliados conseguem atender os *peers* da rede com a entrega dos *chunks* produzidos. Isto indica que os protocolos PALMS+ e NICE conseguem

escalar ao manter uma taxa de fluxo de vídeo crescente em uma rede de participantes também crescente. O PALMS, por sua vez, não é capaz de escalar em qualidade de entrega de *chunks*. O máximo alcance de *peers* conseguido por um *chunk* foi de 2685.

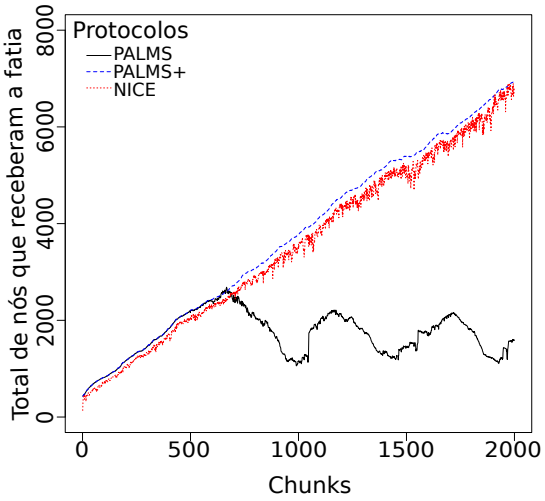
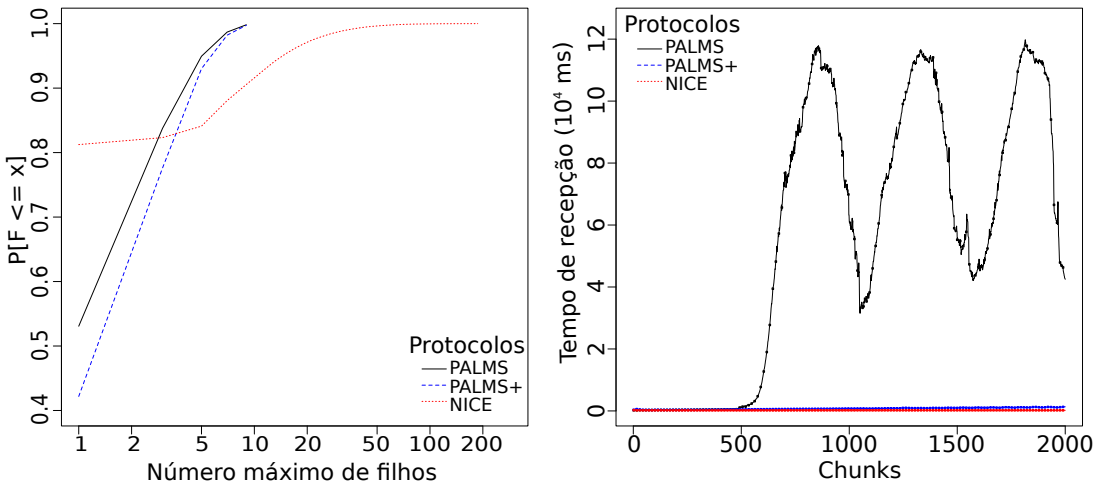


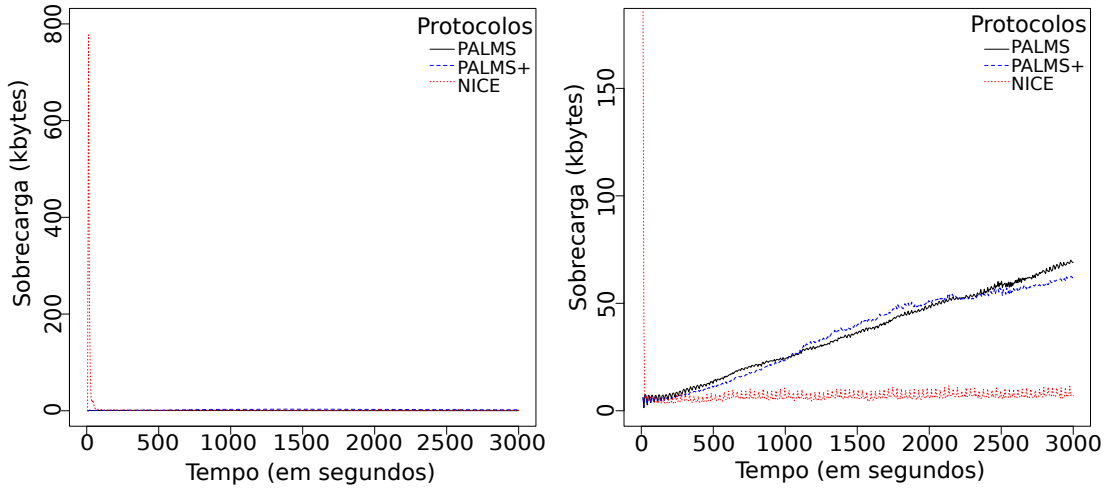
Figura 15 – Alcance dos *chunks* na rede.



(a) Capacidade máxima de *upload*/*peer*. (b) Tempo médio de recepção de *chunks*

Figura 16 – Cenário Real - Paralelo entre o tempo de recepção e a carga de upload exigida dos *peers*

Na figura 16(a) é possível observar o número máximo de filhos por peer exigido durante a simulação. No NICE, 9,27% dos *peers*, em algum momento da simulação, aceitaram prover conteúdo a um número superior ao limite de filhos estipulado para a simulação (8 filhos). Um único peer, por exemplo, chegou a prover conteúdo a 209 filhos, o que seria proibitivo em um sistema real. Já o PALMS e PALMS+ não apresentaram *peers* acima de sua capacidade de distribuição em nenhum momento da simulação.



(a) Sobrecarga nos *peers*. (b) Sobrecarga no *server*.

Figura 17 – Cenário Real - Sobrecarga de banda usada com mensagens de controle.

Como discutido anteriormente, em baixo *churn*, o protocolo PALMS+ mantém um tempo de recepção próximo ao PALMS e NICE com baixa sobrecarga de controle. Entretanto, como pode ser observado na figura 16(b), o tempo médio de recebimento dos *chunks* neste cenário com alto *churn* teve algumas mudanças, exceto o NICE que manteve sua média de 0,2s. O PALMS+ obteve uma média de aproximadamente 0,8 s, em valores absolutos, ele obteve resultados 71 vezes melhores que o PALMS, com 56,8 s. Além disso, é importante frisar que o tempo máximo se manteve inferior a 1,5 s, o que ainda torna o PALMS+ apropriado para aplicações de vídeo ao vivo.

Um dos principais resultados obtidos em [13] ressalta que a latência do protocolo NICE cresce moderadamente à medida que se aumenta o número de *clusters* e que, apesar de ser um protocolo de distribuição em árvore, o NICE é capaz de ajustar-se de forma surpreendente a alto *churn*. Pode-se atribuir este comportamento de destaque à forma não realista com que o NICE lida com a capacidade colaborativa de seus *peers*, mesmo que momentaneamente. Apesar disto, os seus bons resultados podem ser usados como um bom limite inferior para os protocolos ALM que lidem de forma realista com distribuição de carga entre os participantes da rede.

Apesar desta vantagem colaborativa listada, a oscilação da função de entrega de *chunks* sofrida pelo protocolo NICE na rede (figura 15) mostra que ele não possui uma entrega confiável, não sendo indicado a serviços altamente sensíveis a perdas. Em compensação, o protocolo PALMS+ mostrou comportamento similar à função de crescimento da rede (cenário descrito na seção 4.3), entregando em média um total de 7.525.451 novos *chunks*, 5,6% superior ao NICE (7.107.605 *chunks*) e 64,21% superior ao PALMS (2.693.158 *chunks*).

O PALMS+ obteve uma sobrecarga média nos *peers* de 2,33 KBytes contra 3,43 KBytes do observado no NICE e 0,36 no PALMS (figura 17(a)). Esta sobrecarga superior ao PALMS+ em relação ao PALMS pode ser justificada pela quantidade de *chunks* em atraso solicitados e recebidos de forma repetida.

Já a sobrecarga média no servidor, em relação à sobrecarga nos *peers*, foi superior: o PALMS+ obteve sobrecarga média de 35,89 KBytes, o PALMS 36,40 KBytes e o NICE 7 Kbytes (figura 17(b)). A sobrecarga do servidor nos protocolos PALMS e PALMS+ foram aproximadas, contudo muito superiores ao NICE. Uma possível justificativa para estes valores altos pode ter relação com a relativamente pequena altura média da árvore de distribuição, 6,46. Assim, o servidor é alvo de verificações periódicas do RTT para o cálculo do mecanismo de desigualdade triangular como pai ou avô de *peers*. Por outro lado, a baixa sobrecarga do servidor no protocolo NICE pode ser associada ao fato que todos os *peers* colaborarem com a rede, inclusive atingindo em alguns momentos uma carga irreal de colaboração em alguns pontos no caso dos líderes.

Assim, apesar do ganho em sobrecarga e tempo de recepção do NICE serem ligeiramente superiores ao PALMS+, o mesmo não lida de forma realista com a capacidade colaborativa de seus *peers*. Além disso, ao contrário do PALMS, o PALMS+ mostrou escalar adequadamente em alto *churn*, entregando *chunks* a um número maior de *peers* na rede.

5.1.3 Resumo dos resultados obtidos

A partir dos resultados obtidos nos experimentos com cenários controlado e real, é descrito um resumo nas tabelas 2 e 3.

Experimento 1	**NICE**	**PALMS**	**PALMS+**
Latência	0,20 s	0,28 s	0,34 s
Sobrecarga nos peers	0,91 Kbytes	2,10 Kbytes	2,28 Kbytes
Sobrecarga no servidor	1,82 Kbytes	7,44 Kbytes	7,88 Kbytes

Tabela 2 – Resultados comparativos obtidos para cada um dos protocolos no cenário experimental controlado.

O NICE mostrou ser um protocolo com resultados excelentes, chegando inclusive a reduzir em aproximadamente 5 vezes a sobrecarga imposta ao servidor. Contudo, a carga colaborativa dos *peers* participantes nessa rede obteve resultados proibitivos em um cenário real. Em um dado momento da simulação, um único *peer* chegou a prover conteúdo a outros 209 filhos nessa rede. Assim, apesar de seu excelente desempenho, seu uso em aplicações reais não é indicado. Porém, indicamos o seu uso para fins comparativos de melhor caso.

Experimento 2	**NICE**	**PALMS**	**PALMS+**
Latência	0,20 s	56,8 s	0,8 s
Sobrecarga nos peers	3,43 Kbytes	0,36 Kbytes	2,33 Kbytes
Sobrecarga no servidor	7 Kbytes	36,40 Kbytes	35,89 Kbytes
Carga upload/peer acima do estipulado	9,27%	0%	0%
Alcance das fatias	7.107.605	2.693.158	7.525.451

Tabela 3 – Resultados comparativos obtidos para cada um dos protocolos no cenário experimental real.

Em relação ao PALMS e o protocolo proposto, PALMS+, concluímos que o PALMS é indicado para cenários com baixa carga de *churn* e o PALMS+ para cenários com alta carga de *churn*. Neste segundo cenário, o PALMS+ chega a obter uma latência 71 vezes melhor e alcance das fatias na rede 64,21% superior.

6 Conclusões e trabalhos futuros

Neste trabalho, apresentamos o protocolo PALMS+, um protocolo para multicast na camada de aplicação baseado em um protocolo chamado PALMS de arquitetura em árvore (*tree-based*). Ele foi concebido para ser o mais simples possível, permitindo a distribuição de fluxo contínuo de conteúdo a partir de uma única fonte na Internet. Além disso, o protocolo possui um recurso para refinamento dos acordos de retransmissão utilizando desigualdade triangular baseada no RTT (*Round-Trip-Time*).

Suas principais características são a eficiência e economia quanto ao uso de largura de banda com mensagens de controle, escalabilidade mesmo em cenário de agressivo *churn* e baixa latência de recepção de conteúdo.

Nossos experimentos, realizados na plataforma Oversim (OMNet++), demonstraram que o PALMS+ manteve desempenho tão bom quanto o estado da arte, mesmo quando submetido a alto *churn* em uma rede heterogênea. De fato, com 95% de confiança, o tempo médio de recepção de *chunks* do novo protocolo em um cenário controlado difere em apenas 0,14s mais do utilizado pelo NICE e PALMS. Mesmo o tempo médio de recepção de dados sendo maior, o novo protocolo entrega os dados com uma média de 0,34 s.

Finalmente, o novo protocolo mostra-se adequado a vídeo ao vivo, mantendo-se com um atraso de entrega inferior a 1,5s e 98,49% melhor que sua versão anterior, PALMS. Além disso, o novo protocolo escala com melhor qualidade de entrega em 5,6% mais que o NICE e em 64,21% mais que sua versão anterior em cenários realistas e com alto *churn*.

Como principais trabalhos futuros, pode-se implementar o protocolo e avaliar seu desempenho em *testbeds* como o PlanetLab e observar se o seu ganho se mantém. Mais ainda, pode-se avaliar o desempenho do PALMS+ em diferentes aplicações, como aplicações colaborativas que exijam baixa taxa de transferência de rede contudo alta rotatividade dos nós.

Finalmente, sugere-se articular o protocolo proposto com arestas ponderadas não apenas pela latência entre os *peers*, mas incluir outros valores como capacidade colaborativa, largura de banda ou idade na rede.

REFERÊNCIAS

[1] AKAMAI. The state of the Internet Report, 2013. Disponível em: <http://www.akamai.com/stateoftheinternet/>.

[2] ANDROUTSELLIS-THEOTOKIS, S.; SPINELLIS, D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, ACM, v. 36, n. 4, p. 335-371, 2004.

[3] BANERJEE, S.; BHATTACHARJEE, B.; KOMMAREDDY, C. Scalable application layer multicast. *SIGCOMM*. v. 32, n. 4, 2002.

[4] CHEN, Y.; ZHANG, B.; CHEN, C.; CHIU, D. M. Performance Modeling and Evaluation of Peer-to-Peer Live Streaming Systems Under Flash Crowds. *Networking, IEEE/ACM Transactions on*, IEEE, PP, 2013.

[5] CHU, Y.; RAO, S. G.; ZHANG, H. A case for end system multicast. *ACM SIGMETRICS Performance Evaluation Review*, ACM, v. 28, n. 1, p. 1-12, 2000.

[6] COHEN, B. Incentives build robustness in BitTorrent. *Workshop on Economics of Peer-to-Peer systems*, v. 6, p. 68-72, 2003.

[7] DEERING, S. E.; CHERITON, D. R. Multicast Routing in Datagram Internetworks and Extended LANs. textitACM Trans. Comput. Syst., ACM, New York, NY, USA, v. 8, n. 2, p. 85-110, maio 1990. ISSN 0734-2071.

[8] FORGIE, J. W. IEN 119: ST–A Proposed Internet Stream Protocol. *Unpublished Memorandum*, MIT Lincoln Laboratory, 1979.

[9] GANJAM; A. ZHANG, H. Internet multicast video delivery. *Proceedings of the IEEE*, v. 93, n. 1, p. 159-170, 2005.

[10] GHANBARI, A.; RABIEE, H. R.; KHANSARI, M. SALEHI, M. PPM - A Hybrid Push-Pull Mesh-Based Peer-to-Peer Live Video Streaming Protocol. *International Conference on Computer Communications and Networks*. v. 21, p. 1-8, 2012.

[11] HEI, X.; LIU, Y.; ROSS, K. W. IPTV over P2P streaming networks: the mesh-pull approach. *Communications Magazine*, IEEE, v. 46, n. 2, p. 86-92, 2008.

[12] HOSSEINI, M.; AHMED, D. T.; SHIRMOHAMMADI, S.; GEORGANAS, N. D. A survey of application-layer multicast protocols. *IEEE Communications Surveys and Tutorials*, v. 9, n. 1-4, p. 58-74, 2007.

[13] HÜBSCH, C.; MAYER, C. P.; WALDHORST, O. P. User-perceived performance of the nice application layer multicast protocol in large and highly dynamic groups. *Measurement, Modeling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, Springer Berlin Heidelberg, v. 5987, p. 62-77, 2010.

[14] HUZIOKA, D. T. Um protocolo ALM baseado em desigualdade triangular para distribuição de conteúdo. Masters thesis - Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná, 2010.

[15] HUZIOKA, D. T.; DUARTE JR, ELIAS P. Um protocolo ALM baseado em desigualdade triangular para distribuição de conteúdo. *WP2P 2012: Anais do VIII Workshop de Redes Dinâmicas e Sistemas P2P. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, SBRC2012, 2012.

[16] LI, B.; WANG, Z.; LIU, J.; ZHU, W. Two decades of internet video streaming: A retrospective view. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, ACM, v. 9, n. 1s, p. 33, 2013.

[17] LIU, J.; RAO, S. G., LI, B.; ZHANG, H. Opportunities and challenges of peer-to-peer internet video broadcast. *Proceedings of the IEEE*, v. 96, n. 1, p. 11-24, 2008.

[18] NETWORK, K. On-demand performance monitoring and load testing for Web and mobile sites, jan 2013. Disponível em: <http://www.keynote.com/>.

[19] PELTOTALO, J.; HARJU, J.; JANTUNEN, A.; SAUKKO, M.; VAATAMOINEN, L.; CURCIO, I.; BOUAZIZI, I.; HANNUKSELA, M. Peer-to-peer streaming technology survey. *Seventh International Conference on Networking*, IEEE, p. 342-350, 2008.

[20] RAMZAN, N.; PARK, H.; IZQUIERDO, E. Video streaming over P2P networks: Challenges and opportunities. *Signal Processing: Image Communication*, Elsevier, v. 27, n. 5, p. 401-411, 2012.

[21] SRIPANIDKULCHAI, K.; GANJAM, A.; MAGGS, B.; ZHANG, H. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. *ACM SIGCOMM Computer Communication Review*, ACM, v. 34, n. 4, p. 107-120, 2004.

[22] SRIPANIDKULCHAI, K.; MAGGS, B.; ZHANG, H. An analysis of live streaming workloads on the internet. *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ACM, p. 41-54, 2004.

[23] TALPADE, R.; AMMAR, M. H. Single connection emulation (SCE): An architecture for providing a reliable multicast transport service. *Proceedings of the 15th International Conference on Distributed Computing Systems*, IEEE, p. 144-151, 1995.

[24] ULLAH, I.; DOYEN, G.; BONNET, G.; GAITI, D. A survey and synthesis of user behavior measurements in p2p streaming systems. *Communications Surveys & Tutorials*, IEEE, v. 14, n. 3, 2012.

[25] WANG. F.; XIONG, Y.; LIU, J. mTreebone: A collaborative tree-mesh overlay network for multicast video streaming. *Parallel and Distributed Systems*, IEEE, v. 21, n. 3, p. 379-392, 2010.