PALMS: Um Protocolo ALM Simples para Distribuição de Conteúdo

Daniel Tetsuo Huzioka, Elias Procópio Duarte Jr.

Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brazil

{danielh,elias}@c3sl.ufpr.br

Abstract. *ALM Protocols are used to deploy multicast communication at the application layer, using end-host processing and upload capabilities. Several ALM protocols have been proposed, each one presenting varying features for varying applications. This paper proposes PALMS, a simple ALM protocol for content-distribution over the Internet. The protocol architecture consists of a server, responsible for content generation, a tracker, responsible for aiding nodes to join the system, and peers, which are clients that are also responsible for the core multicast transmissions. PALMS organizes peers in groups. Each group presents a tree topology and dictates which peer should send the content to which other peers. The server is the root of all groups, and sends one copy of the content for each group allowing its peers to disseminate the content within the group. PALMS was implemented using the PeerSIM simulator and we present results comparing PALMS with Narada, a popular application-layer multicast protocol. Results shows that our protocol achieves an acceptable server-to-peers delay and allows content distribution using fewer messages than Narada.*

Resumo. *Os protocolos ALM (*Application Layer Multicast*) são usados para implementar a difusão seletiva na camada de aplicação, utilizando a capacidade de processamento e comunicação dos nós da rede para o repasse de mensagens. Diversos protocolos ALM já foram propostos, cada um contendo características específicas diferentes para aplicações distintas. Este trabalho propõe o PALMS, um protocolo ALM Simples para a distribuição de conteúdo na Internet. A arquitetura do PALMS consiste de um* servidor *responsável pela geração de conteúdo, um* tracker *responsável pelo auxílio à entrada de nós na rede e pelos* peers*, os clientes da rede que também são responsáveis por grande parte das transmissões* multicast*. No PALMS, os* peers *são organizados em* grupos*, que agem como árvores de compartilhamento determinando a topologia de transmissão de conteúdo entre* peers*. Todos os grupos têm como raiz o servidor, que dissemina uma cópia do conteúdo para cada grupo. Um protótipo funcional do PALMS foi implementado no simulador PeerSIM e comparado com outro protocolo ALM bastante difundido, o Narada. Resultados mostram que nosso protocolo cumpre sua função de distribuição de conteúdo com um atraso aceitável entre* peers *e servidor utilizando uma quantidade menor de mensagens de controle que o Narada.*

1. Introdução

A transmissão por multicast permite que uma mensagem seja transmitida de uma só vez para um grupo de destinatários. Em redes de distribuição de conteúdo, a transmissão por multicast é a mais adequada quando há apenas uma fonte de conteúdo na rede. Diversas soluções multicast em nível de rede já foram propostas [Holbrook et al. 2006, Eriksson 1994]. Em especial, destaca-se o IP Multicast [Holbrook et al. 2006], que adiciona suporte multicast ao IPv4 (protocolo mais utilizado pelos roteadores da Internet). Devido a problemas de implantação, analisados detalhadamente em [Diot et al. 2000], o IP Multicast não é suportado por todos os roteadores da Internet, não podendo ser utilizado em sistemas cujo objetivo seja atender a usuários gerais da Internet.

Para contornar esses problemas de implantação, foram propostos diversos protocolos ALM (*Application Layer Multicast*, ou Difusão Seletiva em Nível de Aplicação, em português), que criam uma rede sobreposta para simular a distribuição multicast na camada de aplicação, sem a necessidade de modificação na camada de rede. Um protocolo ALM permite a comunicação multicast na Internet em sua estrutura atual. Em [Hosseini et al. 2007], os autores categorizam diversos protocolos ALM de acordo com algumas propriedades. Essas propriedades incluem, entre outras: *domínio da aplicação*, *organização dos peers* e *tipo de roteamento* de mensagens, descritas a seguir.

O *domínio da aplicação* é o objetivo final do sistema, descrevendo também o tipo de conteúdo distribuído por ele. Exemplos de *domínios de aplicação* de protocolos ALM são a distribuição de arquivos, distribuição de áudio e vídeo sob demanda, transmissão ao vivo em tempo real e videoconferência com múltiplos participantes. A *organização dos peers* inclui a topologia da rede, os métodos de inserção de novos nós, métodos de distribuição de conteúdo, entre outros. Exemplos de *organização dos peers* compreendem a forma como a topologia é criada, a existência ou não de níveis hierárquicos e a existência ou não de pontos de referência para a entrada dos *peers* na rede. O *roteamento de mensagens* define como as mensagens são transmitidas, dada a topologia da rede.

O protocolo ALM proposto neste trabalho pode ser categorizado de acordo com as propriedades mencionadas acima. Como domínio da aplicação, o protocolo inclui-se na categoria de transmissão de conteúdo proveniente de uma única fonte. Quanto ao tipo de conteúdo, este pode se apresentar como um conteúdo estático ou contínuo [Moraes et al. 2008]. Na distribuição de conteúdo estático, o conteúdo completo já se encontra disponível *a priori*, permitindo aos clientes receberem o conteúdo em qualquer ordem, podendo receber o final do conteúdo antes do início. Por outro lado, na distribuição de conteúdo contínuo este não se encontra totalmente disponível no início da transmissão, sendo gerado à medida que é transmitido [Li 2006]. Apesar da distribuição de conteúdo contínuo apresentar maiores restrições de banda e de tempo de recebimento, ele possibilita pouca interação do usuário sobre a execução do conteúdo (não permitindo, por exemplo, a execução de comandos de *retroceder* e *avançar*), ao contrário da distribuição de conteúdo estático. O protocolo proposto foi projetado para suportar as restrições da distribuição de conteúdo contínuo, embora seja possível a distribuição de conteúdo estático realizando pequenas modificações no sistema.

Em relação à organização da rede, o protocolo proposto utiliza-se de *grupos* de *peers*, que efetivamente formam árvores de compartilhamento de conteúdo entre os nós. Como mecanismo de roteamento, o protocolo apresenta a técnica de roteamento a partir

de uma árvore geradora com raiz no servidor, que determina quais *peers* devem receber e transmitir conteúdo para outros *peers*. Protocolos ALM são intrinsecamente P2P (*par-a-par*, do inglês *peer-to-peer*), pois neles os *peers* desempenham funções tanto de receptores como transmissores dos dados. Protocolos ALM voltados à distribuição de conteúdo proveniente de uma única fonte, como é o caso deste protocolo, podem ser ainda categorizados como protocolos híbridos, uma vez que possuem a rede P2P mas também possuem um servidor, entidade que não recebe nem consome dados.

Quanto à arquitetura, o PALMS é composto por um *servidor*, um *tracker* e pelos *peers*. O *servidor*, também chamado de *fonte*, é o responsável pela geração de conteúdo e por ditar a periodicidade de disponibilização do fluxo. O *tracker* é o responsável por auxiliar os *peers* a entrarem na rede, além de manter uma lista contendo informações dos *peers* ativos. Os *peers* desempenham dois papéis importantes na rede. O papel de cliente, que recebe e consome o fluxo recebido, e o papel de retransmissor, que irá propagar o conteúdo recebido a outros *peers*. No PALMS, os *peers* são organizados em *grupos*, que agem como árvores de compartilhamento determinando a topologia da rede de transmissão de conteúdo entre *peers*. Todos os grupos têm como raiz o servidor, que transmite uma cópia do conteúdo por grupo. Um protótipo funcional do PALMS foi implementado no simulador PeerSIM e comparado com outro protocolo ALM bastante difundido, o Narada. Resultados mostram que nosso protocolo cumpre sua função de distribuição de conteúdo utilizando multicast em nível de aplicação, utilizando uma quantidade significativamente menor de mensagens de controle que o Narada e mantendo um nível aceitável de atraso entre o *servidor* e os *peers*.

O restante do artigo está organizado da seguinte maneira. A Seção 2 apresenta trabalhos relacionados, descrevendo outros protocolos ALM. A Seção 3 descreve a proposta do trabalho, o Protocolo ALM Simples. A Seção 4 apresenta resultados experimentais de uma implementação do PALMS no simulador PeerSIM e comparações com o protocolo NARADA [Chu et al. 2000]. A Seção 5 conclui o artigo.

2. Trabalhos Relacionados

Os trabalhos relacionados incluem outros protocolos ALM e sistemas de distribuição de conteúdo, voltados às mais diversas funções. Dentre os diversos trabalhos existentes, são descritos o protocolo Narada [Chu et al. 2000] e o protocolo Nice [Banerjee et al. 2002], além de menções a outros trabalhos importantes na área. O protocolo Narada foi um dos primeiros a utilizar usuários finais para a comunicação *multicast*. O protocolo Nice utiliza-se de uma rede hierárquica em *clusters*. Nesta seção, esses protocolos são apresentados, bem como uma visão geral de outros trabalhos relacionados recentes.

Em [Chu et al. 2000], é apresentado o sistema Narada para distribuição de conteúdo. Este sistema organiza os *peers* em uma rede sobreposta na camada de aplicação. Tendo um *rendezvous point* como uma entidade para auxiliar a entrada de nós no sistema, o Narada mantém sua estrutura de rede de maneira descentralizada através de duas redes sobrepostas. Uma árvore geradora mínima (*minimum spanning tree*), específica para cada nó, é utilizada para o roteamento das mensagens, e uma rede em malha (do inglês *mesh*) é utilizada para o controle dos membros. Quando um nó é inserido na rede, ele se conecta a alguns nós arbitrários, que propagam a informação de inserção aos demais nós. O Narada utiliza mensagens de *refresh* (ou atualização) com *número*

sequencial para manter a composição da rede atualizada, testando probabilisticamente uma conexão com um nó cujo *refresh* não foi recebido. O Narada utiliza também um mecanismo semelhante ao BGP [Bellovin and Zinin 2006] para a atualização de tabelas de roteamento, e utiliza um mecanismo de retransmissão por caminho reverso (do inglês *reverse-path-forwarding*) para propagação dos dados. Como mecanismos de otimização da rede, o Narada utiliza duas funções, o *grau de utilidade* e o *custo de concenso* de enlace. O *grau de utilidade* mede o ganho obtido por um nó na inserção de uma ligação direta entre um outro nó na rede sobreposta (tornando os dois vizinhos), enquanto o *custo de concenso* mede a importância de uma ligação existente entre dois nós. Cada nó mede periodicamente o *custo de concenso* com seus vizinhos e o *grau de utilidade* com nós arbitrários não-vizinhos, adicionando ou removendo ligações quando necessário.

O NICE é um protocolo ALM voltado à distribuição de *streams* a partir de uma única fonte. Tendo como topologia uma árvore, o NICE abstrai cada nó da árvore como um conjunto de *peers* (chamados *clusters*), que formarão o grupo multicast. Partindo dos *clusters* que na estrutura da rede são folhas, cada *cluster* possui um *peer* especial, chamado *cluster leader*, que recebe informações de outros níveis da árvore. Hierarquicamente, a árvore é montada criando-se um grupo acima da folha, que possui todos os *cluster leaders* dos *clusters* da folha. Estes grupos de *cluster leaders* também serão *clusters*, tendo seus próprios *cluster leaders* que formarão um *cluster* hierarquicamente superior. Desta forma, têm-se uma árvore cuja raiz é a origem do fluxo, que propaga o fluxo aos *cluster leaders* do primeiro nível, que por sua vez possuem *cluster leaders* pertencentes ao segundo nível e assim por diante. A inserção na rede se dá pela inserção do *peer* em um *cluster* da folha. Iniciando o processo pelos clusters de hierarquia mais alta (aqueles que recebem o fluxo diretamente da fonte), o *peer* verifica qual *cluster leader* mais se assemelha a ele (métricas como distância e latência podem ser utilizadas), verificando então qual dos *clusters leader* dos *clusters* inferiores a aquele *cluster* mais se assemelha a ele, seguindo até sua inserção em um *cluster* da folha. Em casos emergenciais, por exemplo quando um *peer* entra na rede e seu processo de inserção é lento, a fonte pode enviar o fluxo diretamente ao novo *peer*, até que um *cluster* seja encontrado para ele. A manutenção dos *clusters* é dada através de mensagens de *heartbeat* dentro do cluster, podendo haver demoções de líderes ou junções/separações de *clusters*, quando necessário.

Diversos outros protocolos relacionados foram apresentados recentemente, sendo muitos voltados à melhoria da organização da rede utilizando conhecimentos de localidade entre os *peers*. Em [Zhang et al. 2008], um espaço n-dimensional é utilizado para representar a distância dos nós entre si, onde n é o número de pontos de referência utilizados. Utilizando medidas como o *ping* entre os nós para determinar a distância entre eles, o *taonet* procura construir uma topologia onde nós dimensionalmente próximos estejam próximos também na rede sobreposta. Em [Zhang et al. 2009], *peers* são agrupados em *clusters* hierárquicos e utilizam membros do *cluster* como pontos de referência para determinar a proximidade. Em [Dai et al. 2009], o número de saltos é utilizado para criar um modelo de rede, permitindo diversos métodos de escolha de *peers* vizinhos. Outro protocolo, que não utiliza conceitos explícitos de localidade, é o sistema de *streaming* multimídia híbrido cliente-servidor assistido por *peers* não confiáveis, apresentado em [Mello 2009], onde *peers* realizam acordos de compartilhamento (de envio e recebimento) temporários par ao recebimento do conteúdo.

3. PALMS: Um Protocolo ALM Simples

O PALMS (Protocolo ALM Simples) é um protocolo ALM voltado à distribuição de conteúdo contínuo proveniente de uma única fonte pela rede, e permite a transmissão de dados entre processos de usuários finais utilizando multicast em nível de aplicação. Nesta seção, são descritas a arquitetura do sistema, a organização dos *peers* em grupos e o modelo de organização do conteúdo.

3.1. Arquitetura do Sistema

O sistema proposto é composto por três entidades, *servidor*, *tracker* e *peer*. Ao *servidor*, cabe a tarefa de gerar e transmitir os dados que serão distribuídos no sistema. Ao *tracker*, cabe a tarefa de permitir a entrada organizada de novos *peers* na rede. Aos *peers*, cabe a tarefa de recepção, execução e propagação do conteúdo a outros *peers*. A Figura 1 mostra a estrutura geral da arquitetura do protocolo. A seguir, cada um dos componentes da arquitetura é descrito.

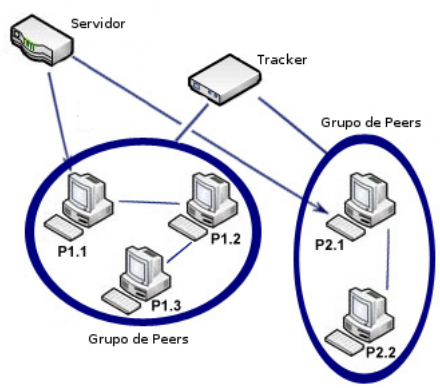

Figura 1. Arquitetura do protocolo PALMS.

3.1.1. O Servidor de Fluxo

O *servidor*, também chamado de fonte, é a entidade responsável pela produção do conteúdo e pela periodicidade em que eles são disponibilizados na rede. O conteúdo é gerado como um fluxo contínuo (em inglês, *stream*), e é descrito em detalhes na Seção 3.3.

É função do servidor determinar o intervalo de tempo em que cada novo elemento do fluxo será disponibilizado. Este intervalo é fixado no início da transmissão, mantendo-se constante até seu término.

Em relação à distribuição de conteúdo, o servidor desempenha a função de *seeder* das redes *torrent* [Cohen 2003], que diferencia-se de um *peer* comum tanto por não precisar receber conteúdo de nenhum outro *peer* como por não consumi-lo. O servidor pode enviar o conteúdo produzido a qualquer *peer* requisitante, respeitando seu limite de banda. As requisições de conteúdo são descritas na Seção 3.1.3.

Em relação à topologia da rede, o servidor não possui conhecimento de todos os nós presentes na rede, como também não participa da organização dos *peers*. Apenas *peers* que recebem conteúdo diretamente do servidor são conhecidos por ele.

3.1.2. O Tracker

O *tracker* é a entidade organizadora do sistema, e desempenha funções semelhantes às encontradas nos *trackers* das redes *torrent* [Cohen 2003] e aos *Rendezvous Point* de alguns protocolos ALM [Banerjee et al. 2002, Chu et al. 2000]. No PALMS, o *tracker* desempenha duas funções: (i) auxiliar na inserção de *peers* na rede e (ii) auxiliar na organização da rede.

É função do *tracker* disponibilizar o endereço do sistema na rede, possibilitando a entrada de novos *peers* a partir deste endereço. Ou seja, um novo *peer* conecta-se ao sistema através do *tracker*, e não através do servidor, como é o caso dos sistemas baseados no modelo cliente-servidor. Essa mudança retira a carga de inserção de novos *peers* do servidor, permitindo que ele execute apenas a função de distribuição de conteúdo.

O *tracker* mantém uma lista de *peers* ativos na rede, com informações em relação à disponibilidade de retransmissão de cada *peer*. Informações sobre disponibilidade de retransmissão podem ser enviadas pelo próprio *peer* ou por outros *peers* que detectem essa condição. Um *peer* é adicionado à lista quando ele é inserido com sucesso na rede (descrito em detalhes na Seção 3.1.3), sendo responsabilidade do *peer* reportar o sucesso de sua inserção. Um *peer* é removido da lista quando informa sua saída ao *tracker*.

Para auxiliar a inserção de novos *peers*, o *tracker* informa a cada novo *peer* duas informações iniciais: o endereço do servidor e um conjunto de *peers*. O endereço do servidor faz-se necessário pois na ocorrência de falhas na transmissão entre os *peers*, é feita uma requisição do conteúdo diretamente ao servidor, visando evitar interrupções significantes na execução do conteúdo. O conjunto de *peers* é um subconjunto da lista de *peers* mantida pelo *tracker*. De tamanho fixo, esse subconjunto é composto apenas por *peers* que se encontram disponíveis na rede, e seus integrantes são escolhidos aleatoriamente dentre todos os *peers* da lista. A escolha por um subconjunto aleatório baseia-se na premissa otimista que haverá pelo menos um *peer* disponível que consiga propagar o conteúdo ao novo *peer* e, como a cada requisição um subconjunto aleatório diferente é escolhido, a carga de propagação fica distribuída entre um grande número de *peers* na rede.

Para auxiliar na organização da rede, o *tracker* mantém um contador de *número de grupos* (*groupID*), que é incrementado sempre que um novo grupo é formado (os grupos são descritos na Seção 3.2). Quando um *peer* inicia sua inserção na rede, ele pode não conseguir se conectar a nenhum *peer* do conjunto de *peers* enviado pelo *tracker* (ou o conjunto pode estar vazio). Neste caso, o *peer* requisita ao *tracker* a criação de um novo grupo. Esse processo assegura a distinção entre os grupos, sem a necessidade de atualizações constantes pelo *tracker*.

3.1.3. Os Peers

Os *peers* são os processos dos usuários finais do sistema, e são responsáveis não somente por consumir o conteúdo, mas também auxiliar na propagação do mesmo pela rede. Há duas maneiras de um *peer* receber o conteúdo, diretamente do servidor ou de outro *peer*. Em ambas as situações, caso ele tenha disponibilidade de banda, esse *peer* poderá também

enviar o conteúdo recebido à outro *peer*, formando assim a rede P2P.

Quando um *peer* requisita sua entrada na rede, ele inicialmente recebe do *tracker* as informações sobre o servidor e uma lista de *peers*, descrita na Seção 3.1.2. O *peer* então dispara simultaneamente mensagens de requisição de inserção a todos os *peers* da lista. A aceitação ou não da requisição de inserção depende da disponibilidade de banda para retransmissão do conteúdo por parte dos *peers* receptores da mensagem. Apenas uma única resposta positiva é necessária para completar a inserção do *peer* na rede, sendo as demais descartadas. O disparo simultâneo paralelo se justifica pois, o *peer* que aceitar mais rapidamente à requisição será o *peer* mais indicado para a propagação do conteúdo ao novo *peer*, pois apresenta o menor RTT (*round-trip-time*) dentre os *peers* da lista recebida.

Quando o *peer* p_i dispara requisições a todos os *peers* da lista recebida pelo *tracker*, aqueles que se encontram disponíveis enviam seus respectivos identificadores de grupos a p_i. A primeira resposta recebida por p_i, proveniente do *peer* p_j pertencente ao grupo g_x, informa a p_i que há um *peer* p_j disponível no grupo g_x. P_i então requisita sua inserção ao *peer* p_j, que informa a p_i o sucesso de sua inserção no grupo g_x e passa a retransmitir o conteúdo ao *peer* p_i.

Quando um *peer* p_k pretende deixar a rede, ele informa sua saída a todos os *peers* que recebem seu conteúdo diretamente. Cada um destes *peers* requisita uma nova criação de grupo ao *tracker*. O peer p_k deve ainda informar ao *tracker* de sua saída, para sua remoção da lista de *peers* do tracker. Caso receba conteúdo diretamente do servidor, p_k deve informar também ao servidor, que encerra a transmissão de conteúdo a p_k. Um parâmetro do sistema informa aos *peers* o tempo máximo de atraso entre o recebimento das mensagens de conteúdo dos *peers*. Caso este tempo seja excedido, o *peer* receptor assume que seu *transmissor* tenha deixado a rede, tornando-se um *peer* órfão.

Caso um *peer* encontre-se em estado órfão e não esteja retransmitindo conteúdo a nenhum outro *peer* (ele era um nó folha na árvore de compartilhamento), ele requisita ao *tracker* uma nova lista de *peers* para realizar acordos. Caso contrário, o *peer* órfão cria um novo grupo.

A criação de novos grupos pelos *peers* agora órfãos faz com que eles recebam conteúdo diretamente do *servidor* (ver Seção 3.2 para detalhes). A atualização para um novo grupo ocorre do *peer* orfão até todos os seus filhos, que propagam a informações a todos os filhos até os nós folhas. Essa técnica permite a rápida recuperação do sistema a uma saída súbita de um *peer*, e leva em consideração a natureza não-confiável dos *peers*.

3.2. Organização dos Peers em Grupos

Os grupos são utilizados para separarem os *peers* em árvores de compartilhamento distintas. A topologia interna de um *grupo* é representada por uma árvore, onde a relação pai-filho corresponde a uma transmissão do nó pai ao nó filho. Cada grupo possui um *peer* que recebe os dados diretamente do servidor (o primeiro *peer* do grupo), que fica responsável por propagá-los aos seus *peers* filhos, que por sua vez propagam para seus próprios *peers* filhos.

Quando um *peer* p_x deseja criar um novo grupo, por exemplo o primeiro *peer* de todo o sistema, ele primeiro envia uma requisição ao *tracker*. O *tracker* então incrementa seu contador de *id de grupos*, anteriormente gid_x, para gid_{x+1} e envia esse novo valor a

p_x. Ao receber a resposta, p_x muda seu próprio id de grupo para gid_{x+1}. Qualquer outro *peer* que se conectar à p_x será pertencente ao grupo gid_{x+1}.

Quando um novo grupo é criado pelo *tracker*, o primeiro *peer* inserido no grupo deverá requisitar o conteúdo diretamente ao servidor. Caso o primeiro *peer* inserido no grupo tenha nós conectados a ele (a criação foi proveniente de um rompimento de um grupo anterior), o id do novo grupo é propagado a estes *peers*, que propagam aos *peers* conectados a eles e assim sucessivamente.

Quando um *peer* conecta-se a um grupo já existente, ele se conecta a apenas um único *peer* do grupo (ao qual ele enviou a requisição), passando a receber conteúdo diretamente desse *peer*. Este *peer* recém inserido pode apresentar recursos para outras retransmissões de conteúdo, informando ao *tracker* de sua disponibilidade e possibilitando futuras inserções de novos *peers*. A topologia da rede resultante desse processo forma uma árvore interna aos *grupos* para o compartilhamento de conteúdo. A topologia de compartilhamento em árvore é apresentada na Figura 2. As setas incidentes nos grupos indicam quais *peers* recebem fluxo diretamente do servidor, sendo estes os únicos *peers* conhecidos pelo servidor. Internamente aos grupos, o *peer* $p_{2.1}$ está conectado diretamente ao servidor, e não está conectado nem possui informações sobre nenhum outro *peer*. Já o *peer* $p_{1.1}$ possui acordos de envio com os peers $p_{1.5}$ e $p_{1.3}$. $P_{1.3}$ possui um acordo de recebimento de $p_{1.1}$ e um acordo de envio para $p_{1.2}$. $P_{1.5}$ possui um acordo de recebimento com $p_{1.1}$ e dois acordos de envio, um com $p_{1.6}$ e outro com $p_{1.4}$. Neste cenário, $p_{1.2}$ não têm conhecimento de $p_{1.1}$, nem de $p_{1.5}$ e seus filhos, sendo o inverso também verdadeiro.

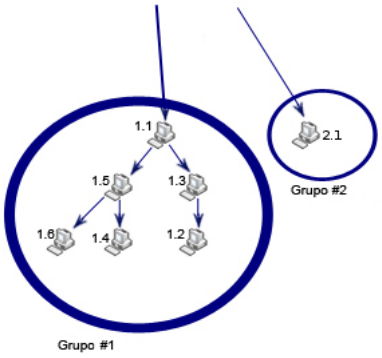

Figura 2. Topologia interna ao grupo.

3.3. Organização do Conteúdo

A organização do conteúdo compreende o seu mecanismo de geração e como ele é distribuído. O fluxo é inicialmente dividido em pedaços, chamados fatias, de tamanhos idênticos e é distribuído de acordo com a árvore de compartilhamento interna a cada *grupo*. Para cada par pai-filho na árvore, denomina-se *acordo de envio* o acordo que o pai tem para com o filho de retransmitir o fluxo, e *acordo de recebimento* o acordo que o filho tem para com o pai de receber o fluxo. A Figura 3 demonstra um exemplo desse conjunto de acordo entre os *peers*. Nesta figura, pode-se observar a propagação da fatia 1, transmitida pelo servidor aos *peers* $p_{1.1}$ e $p_{2.1}$. Os peers $p_{1.1}$, $p_{1.2}$ e $p_{1.3}$ pertencem ao

grupo g_1, enquanto os *peers* $p_{2.1}$ e $p_{2.2}$ pertencem ao grupo g_2. Quando $p_{1.1}$ recebe a fatia 1, ele a propaga para $p_{1.2}$, que por sua vez propaga para $p_{1.3}$. O mesmo ocorre no grupo g_2, onde $p_{2.1}$, ao receber a fatia 1, propaga-a para $p_{2.2}$.

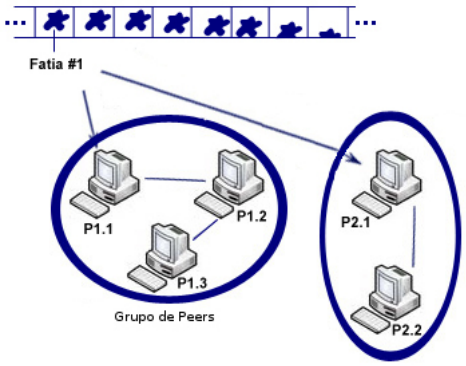

Figura 3. Distribuição de fluxo segundo acordos realizados.

4. Avaliação Experimental

Esta seção descreve a avaliação experimental realizada através de simulação do protocolo proposto. As simulações foram realizadas utilizando o simulador Java PeerSim [Jelasity et al.], no qual as entidades do sistema são representadas por objetos Java. Apesar de executadas localmente, o ambiente de simulação reflete a Internet. A topologia da rede simulada leva em consideração a distribuição de graus de nodos *power law* apresentados em [Faloutsos et al. 1999, Siganos et al. 2003, Bu and Towsley 2002], assim como conceitos de redes *small world* apresentadas em [Watts and Strogatz 1998]. Como falha entre os roteadores foram desconsideradas, a rede de roteadores gerada apenas auxilia na medição de latência entre os nós terminais.

Em todas as simulações, o comportamento dos nós foi simulado baseando-se em estudos sobre distribuição de fluxos presente em [Sripanidkulchai et al. 2004]. Sobre esse estudo, foi considerado que grande parte dos *peers* se conectam no início da transmissão, grande parte deles permanecem durante toda a transmissão e as inserções ocorrem segundo o efeito *flash mob*, onde num instante não há qualquer inserção e no instante seguinte podem ocorrer inserções múltiplas.

4.1. Experimento 1

O primeiro experimento objetiva demonstrar como a rede se comporta em um ambiente com *churn* (entrada e saída constante de *peers* na rede). O limite máximo de *peers* na rede foi de 256. A latência entre os roteadores foi estabelecida entre 5 e 15ms, sendo este valor adiciona em cerca de 10ms dos roteadores aos *peers*. Cada *peer* quando inserido foi designado a um roteador arbitrário, e se manteve conectado ao mesmo roteador até sua saída da rede. O número de *identificadores de peers* enviados pelo *tracker* foi limitado em 5 por requisição.

Este experimento teve a duração total de 600 segundos. A cada 1 segundo um novo elemento de conteúdo foi gerado e transmitido pela rede. A taxa de transmissão do conteúdo foi de 64Kbps, sendo a largura de banda dos *peers* suficiente para receber o

conteúdo em sua totalidade. Para cada *peer* foi designada uma banda de envio entre 512, 256, 128, 64 ou 0 (e neste caso o *peer* entra na rede como *leecher*, que não contribui para a propagação de conteúdo). Modificações na rede ocorreram em intervalos de 5 segundos, podendo acarretar na inserção de até 20 membros, na remoção de até 10 membros ou na manutenção no número de *peers*. A Figura 4 mostra a quantidade de *peers* ativos na rede durante a execução da simulação. A Figura 5 mostra a distribuição dos grupos no fim da execução.

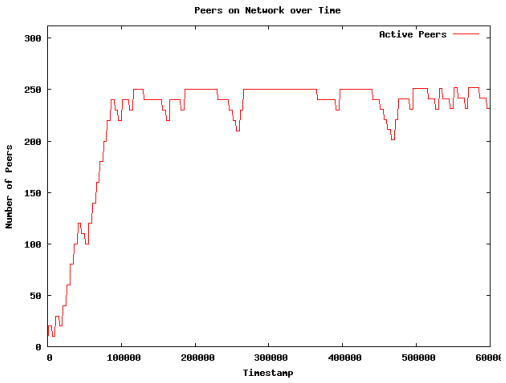

Figura 4. Peers por período de tempo.

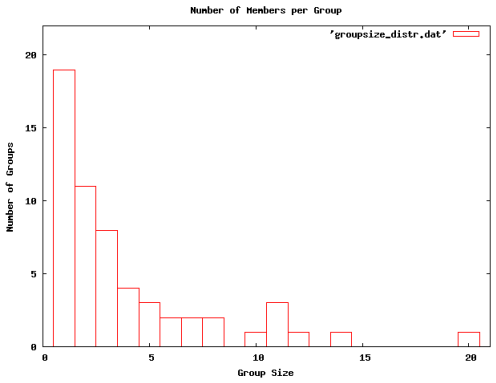

Figura 5. Distribuição de grupos por tamanho.

Pode-se observar pela Figura 5 que há uma grande quantidade de grupos pequenos. Isso se deve principalmente à saída de nós intermediários dos grupos durante a execução, o que acarreta a criação de um novo grupo pelos nós desconectados. Entretanto, espera-se que os novos grupos sejam preenchidos por outros *peers* a medida que estes são inseridos na rede, devido à aleatoriedade de escolha do conjunto de *peers* pelo *tracker*. Por outro lado, grupos demasiadamente grandes são também repartidos em grupos menores quando um *peer* pertencente ao grupo deixa a rede, resultando na formação de grupos menores.

4.2. Experimento 2

O segundo experimento têm por objetivo demonstrar como a simplicidade do PALMS resulta em comparação a técnicas de refinamento e controle do protocolo Narada.

Para este experimento, foi utilizada uma rede gerada pelo mesmo método do experimento 1, contendo no entanto 384 peers. Vale ressaltar que o protocolo Narada resulta na soma agregada de $O(N^2)$ (send N o número de nós na rede) mensagens de controle [Banerjee et al. 2002], dificultando simulações com grandes números de *peers*.

A taxa de geração e a largura de banda dos *peers* foram as mesmas adotadas no experimento 1, assim como a proporção de nós *leecher*. A taxa de ocorrência de inserção/remoção foi de 5 segundos, mas neste experimento foram realizadas inserções aleatórias de até 50 membros simultâneos e saída de até 5 membros simultâneos. A proporção entre entrada e saída se manteve igual ao do experimento 1.

Para o Narada, foram considerados os parâmetros de taxa de atualização (*refresh*) em 10 segundos, tempo mínimo de *timeout* em 20 segundos, tempo máximo de *timeout* em 30 segundos, taxa de checagem de utilidade em 20 segundos, taxa de atualização de rotas em 10 segundos, custo de consenso mínimo 5, utilidade mínima 40 e checagem de

utilidade em 5 outros *peers* por vez. Além disso, o *peer* de identificador 1 foi considerado a fonte de dados.

Para o PALMS, foram considerados os parâmetros de quantidade de nós enviada pelo *tracker* em 5, e largura máxima de banda em 512 Kbps(como no experimento 1).

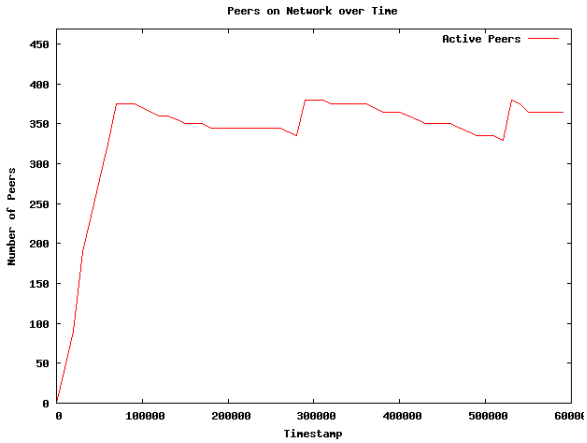

Figura 6. Peers por período de tempo.

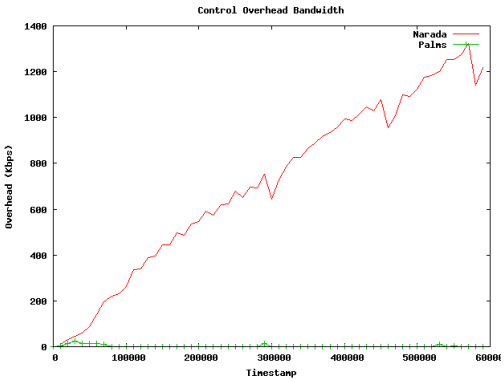

Figura 7. Banda utilizada para mensagens de controle.

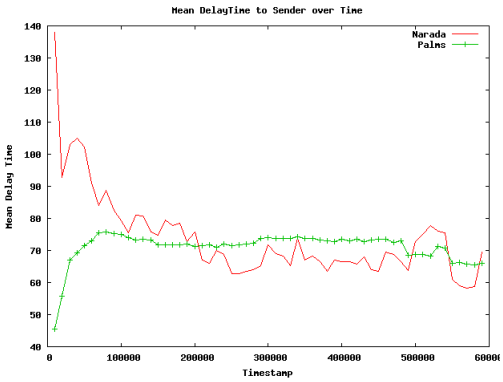

Figura 8. Atraso médio dos peers para a fonte de dados.

A Figura 6 mostra a quantidade de *peers* ativos durante a simulação. Como no experimento anterior, é necessário algum tempo até que a rede se torne estável pois ela é iniciada com nenhum nó presente. Após esse período, as inserções e remoções arbitrárias mantêm a rede próxima de seu limite.

A Figura 7 mostra a quantidade de banda utilizada para a troca de mensagens de controle na rede. No caso do PALMS, as mensagens de controle possuem papel importante no processo de entrada e saída dos nós, mantendo-se quase nulas enquanto os nós estiverem transmitindo conteúdo de maneira estável. Já no Narada, atualizações constantes e verificações de utilidade e custo de enlaces fazem com que a quantidade de mensagens de controle cresça até um ponto onde ela estabiliza-se. Caso a rede apresente *churn* constante (como é o caso desta simulação), os nós irão constantemente verificar por novas conexões e alertar sobre possíveis interrupções nos caminhos escolhidos.

A Figura 8 mostra o atraso médio de envio e recebimento das mensagens pelos *peers* até ou para o nó fonte (primeiro nó no Narada e servidor no Palms). É impor-

tante ressaltar que no protocolo Narada o atraso médio nesta simulação apresenta-se alto pois a rede ainda está em fase de descoberta. É importante observar também que após a considerável estabilização no número de *peers* na rede, o Narada inicia seus mecanismos de otimização da rede, melhorando constantemente a topologia da rede. Entretanto, é possível observar que o PALMS mantém o atraso médio constante e relativamente próximo aos apresentados pelo Narada, mesmo com a presença de *churn* na rede.

5. Conclusão

Este artigo apresentou uma solução simples para multicast em nível de aplicação. O protocolo PALMS permite a distribuição de conteúdo contínuo pela Internet a partir de uma única fonte. Utilizando técnicas P2P para realizar a difusão seletiva em nível de aplicação, o protocolo consegue diminuir a carga imposta ao servidor na distribuição de conteúdo dividindo os *peers* em *grupos*, construindo árvores de compartilhamento entre os *peers* pertencentes ao *grupo*. Experimentos demonstraram que o sistema impõe uma baixa carga de controle sobre os nós, e tira vantagem do *churn* (característico das redes de compartilhamento) para controlar o tamanho de seus *grupos*. A arbitrariedade dos *peers* enviados pelo *tracker* aumenta a efetividade de balanceamento dos grupos. Além disso, a pesquisa simultânea entre os *peers* enviados pelo *tracker* melhora a qualidade do caminho escolhido localmente, uma vez que o melhor *peer* (entre os enviados) é escolhido para retransmitir o tráfego.

Trabalhos futuros incluem a modificação na escolha dos *peers*, permitindo um maior equilíbrio entre o tamanho dos *grupos* e o refinamento dos *grupos* existentes pelos *peers* participantes. O simulador utilizado é altamente configurável, sendo possível realizar simulações com comportamento dos *peers* mais próximos à realidade, levando em consideração a localização dos *peers* e a interação entre os mesmos na rede se necessário.

Referências

Banerjee, S., Bhattacharjee, B., and Kommareddy, C. (2002). Scalable Application Layer Multicast. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 205–217, New York, NY, USA. ACM.

Bellovin, S. and Zinin, A. (2006). Standards Maturity Variance Regarding the TCP MD5 Signature Option (RFC 2385) and the BGP-4 Specification. RFC 4278 (Informational).

Bu, T. and Towsley, D. F. (2002). On distinguishing between internet power law topology generators. In *INFOCOM*.

Chu, Y.-h., Rao, S. G., and Zhang, H. (2000). A Case for End System Multicast (keynote address). In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA. ACM.

Cohen, B. (2003). Incentives Build Robustness in BitTorrent.

Dai, L., Cao, Y., Cui, Y., and Xue, Y. (2009). On Scalability of Proximity-aware Peer-to-Peer Streaming. *Comput. Commun.*, 32(1):144–153.

Diot, C., Levine, B. N., Lyles, B., Kassem, H., and Balensiefen, D. (2000). Deployment Issues for the IP Multicast Service and Architecture. *Network, IEEE*, 14(1):78–88.

Eriksson, H. (1994). MBONE: The Multicast Backbone. *Commun. ACM*, 37(8):54–60.

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, NY, USA. ACM.

Holbrook, H., Cain, B., and Haberman, B. (2006). Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast. RFC 4604 (Proposed Standard).

Hosseini, M., Ahmed, D. T., Shirmohammadi, S., and Georganas, N. D. (2007). A Survey of Application-Layer Multicast Protocols. *Communications Surveys & Tutorials, IEEE*, 9(3):58–74.

Jelasity, M., Montresor, A., Jesi, G. P., and Voulgaris, S. The Peersim simulator. `http://peersim.sf.net`.

Li, J. (2006). Peer-to-Peer Multimedia Applications. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 3–6, New York, NY, USA. ACM.

Mello, S. L. V. (2009). Hybrid Client-server Multimedia Streaming Assisted by Unreliable Peers. *The 15th International Conference on Distributed Multimedia Systems*, pages 1–6.

Moraes, I. M., Campista, M. E. M., Duffles, M., Rubinstein, M., Costa, L. H., and Duarte, O. C. M. B. (2008). Distribuição de Vídeo sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios.

Siganos, G., Faloutsos, M., Faloutsos, P., and Faloutsos, C. (2003). Power laws and the as-level internet topology. *IEEE/ACM Trans. Netw.*, 11(4):514–524.

Sripanidkulchai, K., Maggs, B., and Zhang, H. (2004). An analysis of live streaming workloads on the internet. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, New York, NY, USA. ACM.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10.

Zhang, X., Li, X., Luo, W., and Yan, B. (2009). An Application Layer Multicast Approach Based on Topology-Aware Clustering. *Comput. Commun.*, 32(6):1095–1103.

Zhang, X., Li, Z., and Wang, Y. (2008). A Distributed Topology-Aware Overlays Construction Algorithm. In *MG '08: Proceedings of the 15th ACM Mardi Gras conference*, pages 1–6, New York, NY, USA. ACM.