AVALIAÇÃO DE ALGORITMOS ESCALÁVEIS PARA CONSTRUÇÃO DE MST EM ALM BASEADAS EM MESH-FIRST

Etienne César Ribeiro de Oliveira e Luiz Satoru Ochi

Instituto de Computação – Universidade Federal Fluminense
Niterói – Rio de Janeiro
{eoliveira, satoru}@ic.uff.br

RESUMO

Redes IP baseadas em *multicast* caracterizam-se por um suporte eficiente a comunicação entre membros de grupos, principalmente quando comparadas com redes *unicast*. No entanto, a implantação de redes puramente *multicast* requer o uso de mecanismos específicos para prover o roteamento e o encaminhamento de pacotes de dados, mecanismos esses que ainda não são plenamente suportados pelas redes atuais. A conseqüência imediata desta realidade é que o uso da abordagem *multicast* apresenta sérios problemas de escalabilidade.

A alternativa para os problemas relacionados ao uso de redes puramente *multcast* passa pela absorção da abordagem *multicast* pela camada de aplicação, criando redes *multicast* sobrepostas às redes físicas e eliminando, desta forma, a dependência de suporte a *multicast* nas camadas inferiores. Essas redes *multicast* sobrepostas são comumente identificadas por ALM (*Application Layer Multicast*).

Os protocolos distribuídos para ALM podem ser categorizados em estruturados e não estruturados. Os esquemas estruturados constroem redes sobrepostas baseadas em DHT (*Distributed Hast Table*). Já os esquemas não estruturados podem ser organizados através das abordagens *mesh-first* e *tree-first*. Especificamente na abordagem *mesh-first*, os membros do grupo *multicast* se organizam, inicialmente, de forma distribuída em uma topologia sobreposta em malha. Como nas redes em malha os nós dispõem de mais de um caminho para alcançar os demais nós da rede, torna-se necessário a aplicação de um protocolo para construção de uma árvore geradora mínima, normalmente denominado algoritmo MST (*Minimum Spanning Tree*). Quanto menor a profundidade da árvore geradora mínima, possivelmente menor será o atraso inserido no tempo total de transmissão e menor será o comprometimento dos recursos de rede. Os algoritmos MSTmf-Kruskal, MSTmf-Prim e MSTmf-Sollin foram desenvolvidos com base nessa heurística, visando a inserção de vértices que impliquem na maior quantidade de conexões para a árvore geradora mínima.

Este trabalho avalia e compara a métrica de profundidade das árvores geradoras mínimas produzidas pelos algoritmos Kruskal [Kruskal, 1956], Prim [Prim 1957] e Sollin [Sollin 1965] com os algoritmos propostos, denominados MSTmf (*Minimum Spanning Tree for mesh-first*), desenvolvidos especificamente para atender protocolos distribuídos e estruturados para ALM com abordagem *mesh-first*. Os resultados das

simulações demonstram que os algoritmos propostos apresentaram, consistentemente, árvores geradoras mínimas com profundidade inferior aos algoritmos originais.

1. Introdução

Redes IP baseadas em *multicast* caracterizam-se por um suporte eficiente a comunicação entre membros de grupos, principalmente quando comparadas com redes *unicast*. Para que membros de grupos em redes *unicast* possam simular um ambiente de redes *multicast* são necessárias múltiplas conexões ponto-a-ponto, já as redes puramente *multicast* são mais eficientes, pois cada pacote de dados é replicado somente em ramos da rede onde haja um membro de algum grupo interessado. Essa abordagem reduz os requerimentos de desempenho da rede como um todo, ou seja, os recursos próximos à estação fonte e nos enlaces de comunicação entre membros do grupo.

A implantação de redes puramente *multicast* requer o uso de mecanismos específicos para prover o roteamento e o encaminhamento de pacotes de dados. No entanto, esses mecanismos ainda não são plenamente suportados nas redes atuais. A conseqüência imediata desta realidade é que o uso da abordagem *multicast* apresenta sérios problemas de escalabilidade.

A alternativa para os problemas relacionados ao uso de redes puramente *multcast* passa pela absorção da abordagem *multicast* pela camada de aplicação, criando redes *multicast* sobrepostas às redes físicas e eliminando, desta forma, a dependência de suporte a *multicast* nas camadas inferiores. Essas redes *multicast* sobrepostas são identificadas, normalmente, por ALM (*Application Layer Multicast*) e todas as operações relacionadas ao serviço *multicast* como gerenciamento de grupos, replicação de pacotes e roteamento passam a ser suportadas pelos membros (nós) da rede. Redes P2P (*Peer to Peer*) constituem uma das aplicações mais populares de ALM em função da capacidade para compartilhamento de dados em larga escala, para a distribuição de conteúdo e para aplicações que implementem *multicast* na camada de aplicação.

Os protocolos para ALM podem ser categorizados em estruturados e não estruturados. Os esquemas estruturados constroem redes sobrepostas baseadas em DHT (*Distributed Hast Table*). Já os esquemas não estruturados podem ser organizados através das abordagens *mesh-first* e *tree-first*.

A busca de arquivos de áudio, vídeo ou simplesmente dados em redes ALM baseadas em DHT depende dos algoritmos de roteamento. O membro solicitante deve enviar alguma informação que possibilite a localização do conteúdo desejado e, como resposta, obter o endereço do nó responsável pelo conteúdo ou o próprio conteúdo solicitado. Chord [Stoica et al. 2001], Tapestry [Zhao et al. 2001] e Pastry [Rowstron e Druschel, 2001], entre outros, são exemplos de algoritmos de roteamento para redes ALM.

Na abordagem *mesh-first* os membros do grupo *multicast* se organizam, inicialmente, de forma distribuída em uma topologia sobreposta em malha. Como nas redes em malha os nós dispõem de mais de um caminho para alcançar os demais nós da rede, torna-se necessário a aplicação de um protocolo para construção de uma árvore

geradora mínima, normalmente denominado algoritmo MST (*Minimum Spanning Tree*). Desta forma, a construção das árvores de distribuição na abordagem *mesh-first* é realizada em duas fases: na primeira fase é construída uma rede em malha, que pode ser representada por um grafo com alta conectividade entre os membros; na segunda fase são construídas MSTs enraizadas nos nós responsáveis pela distribuição do conteúdo, de forma que sejam determinados os menores caminhos reversos para todos os membros. ESM [Chu et al. 2002] e TMesh [Wang et al. 2002], entre outros, são exemplos de protocolos que baseiam-se na abordagem *mesh-first*.

Na abordagem *tree-first*, os membros do grupo *multicast* constroem, de forma distribuída, uma árvore compartilhada e selecionam, a partir de um subconjunto de nós conhecidos, seu nó pai. Como a árvore gerada é um grafo acíclico, se um nó vier a falhar ou deixar repentinamente (sem informar seus vizinhos) um grupo, a árvore poderá ser particionada e os membros de partições distintas perderão a comunicação. Conseqüentemente, as abordagens *tree-first* requerem mecanismos para identificar e reparar particionamentos. CoopNet [Padmanabhan et al. 2002] e Yoid [Francis 2000a, Francis 2000b], entre outros, são exemplos de protocolos que baseiam-se na abordagem *tree-first*.

Métricas de desempenho dos algoritmos Kruskal, Prim, Sollin e do algoritmo proposto denominado MSTmf (*Minimum Spanning Tree for mesh-first*) são comparadas através de simulações. Para realizar essas simulações foi desenvolvido um programa específico com intuito de gerar topologias que simulem a realidade de redes P2P e todos os algoritmos supra citados foram implementados.

Este trabalho está organizado da seguinte forma: a seção 2 trata sucintamente de questões relativas às aplicações ALM, tais como desafios, classificação, protocolos de rede e protocolos de aplicação; a seção 3 aborda os algoritmos para construção de MST (*Minimum Spanning Tree*) utilizados nas simulações; a seção 4 descreve o ambiente desenvolvido para a realização das simulações, as métricas utilizadas nas comparações e apresenta os resultados obtidos; e, por fim, a seção 5 apresenta a conclusão final deste trabalho.

2. ALM (*Application Layer Multicast*)

A distribuição de conteúdo multimídia pela Internet vem crescendo consideravelmente ao longo dos últimos anos, exigindo uma abordagem apropriada para o envio deste tipo de tráfego, principalmente quando o objetivo é alcançar múltiplos destinatários. A opção mais simples para efetuar esta operação é permitir que os servidores de conteúdo (fontes emissoras) enviem, em *unicast*, as mídias solicitadas por cada cliente. No entanto, arquivos de mídia consomem, normalmente, uma quantidade de recursos significante de banda de rede e, dependendo da quantidade de solicitações, certamente haveria um gargalho no servidor de conteúdo, ou seja, esta opção não é escalável. Redes de Distribuição de Conteúdo (CDNs – *Content Distribution Networks* e Redes P2P (*Peer to Peer*) são propostas interessantes de ALM que buscam resolver, de forma eficiente, o problema da distribuição de conteúdo, em contraste com IP *multicast* [Deering e Cheriton, 1990].

Nas Redes de Distribuição de Conteúdo (CDNs), o conteúdo é replicado do servidor principal do provedor de conteúdo para servidores de *proxy* ou para servidores de réplica distribuídos pela Internet, que devem estar localizados próximos aos clientes do provedor de conteúdo. Cabe, então, aos clientes, acessar o servidor com menor atraso e com menor congestionamento. No entanto, em situações de demanda excessiva e repentina (*flash crowds*), existe a possibilidade que os servidores de uma determinada região permaneçam congestionados durante um determinado período [Stravou et al. 2002]. A Figura 1 ilustra um exemplo possível para arquitetura de uma CDN e para uma rede P2P.

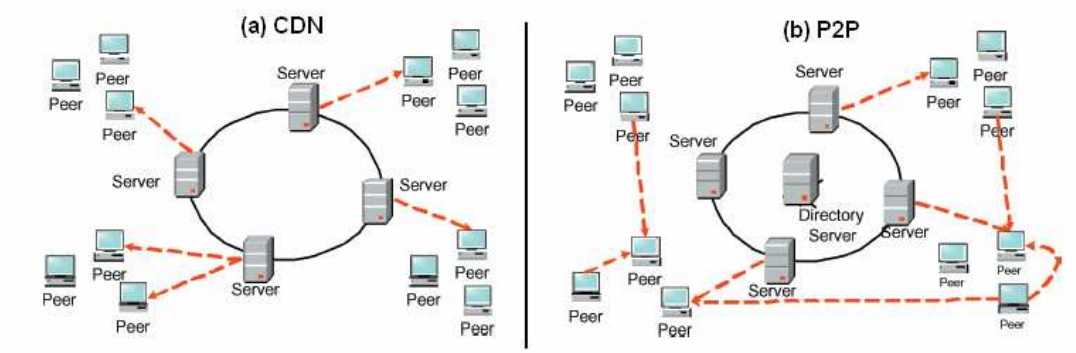

Figura 1 – Exemplo de arquitetura CDN e P2P [Tu et al. 2005]

Com intuito de melhorar a comunicação multidestinatária, [Deering e Cheriton, 1990] propuseram o IP *Multicast*. Nessa proposta, os dados passam uma única vez pelos enlaces e são replicados somente para as interfaces dos roteadores de onde existam clientes interessados. Embora seja eficiente, o IP *Multicast* requer recursos adicionais dos roteadores envolvidos neste processo, como a manutenção do estado dos grupos *multicast* envolvidos na transmissão e, além disso, alguns protocolos inundam periodicamente a rede com objetivo de construir a árvore de distribuição. Questões como confiabilidade fim-a-fim, segurança, privacidade, gerência de grupo etc ainda requerem estudos aprofundados na busca por soluções [Proença, 2006].

Recentemente, a tecnologia de redes P2P (*Peer to Peer*) vem despontando como uma alternativa viável para possibilitar o suporte necessário à distribuição de conteúdo multimídia ao vivo ou armazenado. Nas redes P2P os nós da rede são denominados de pares (*peers*) e formam uma rede sobreposta (*overlay*) sobre redes IP, ATM etc, redes essas que efetivamente encaminham os dados em *unicast.* Cabe, então, aos nós da rede (pares) prover a comunicação multiponto, replicando e retransmitindo os pacotes para os demais pares interessados. Os nós da rede sobreposta (pares) contribuem ainda com capacidade de armazenamento e ciclos de CPU (processamento). Na Figura 2, acima da linha horizontal, encontra-se a representação de uma rede sobreposta e, abaixo da linha horizontal, encontra-se a representação da rede física.

[Stoica et al. 2001] descrevem redes P2P como um conjunto de sistemas e aplicações que operam de forma distribuída, ou seja, sem qualquer tipo de controle centralizado e sem uma organização hierárquica, onde o software executado em cada nó equivale, em termos de funcionalidade, a todos os demais nós da rede P2P.

[Rocha et al. 2004] acrescentam que o uso da tecnologia de redes P2P possibilita que qualquer nó da rede possa acessar diretamente recursos que encontram-se disponibilizados em outros nós, sem a exigência de um controle centralizado. Essa cooperação entre os diversos nós participantes implica no uso eficiente de recursos da rede P2P que estejam disponíveis e sendo subutilizados, tais como processamento e capacidade de armazenamento.

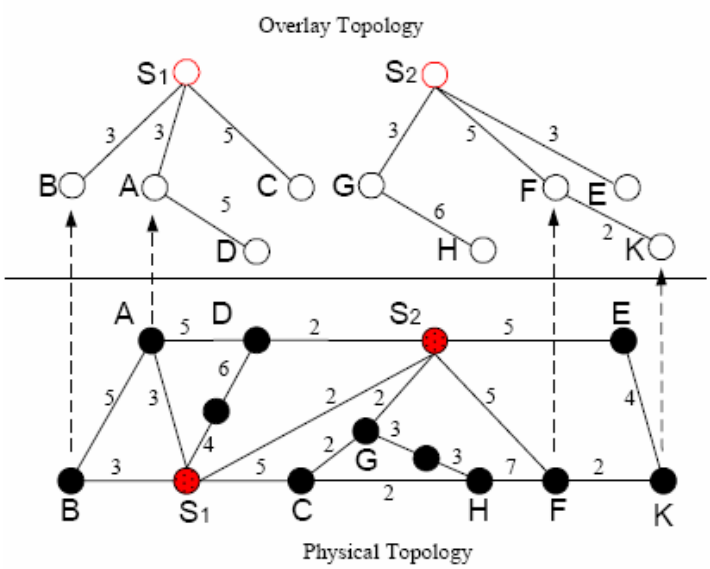

Figura 2 - Rede Sobreposta e Rede Física [Liao et al. 2006]

[Lua et al. 2004] ressaltam que o ambiente de computação e comunicação atual da Internet apresenta-se de forma significativamente mais complexa que os sistemas distribuídos clássicos, requerendo uma coordenação centralizada ou um controle hierárquico para manter-se organizada. O interesse emergente em redes P2P sobrepostas surge em função da capacidade para compartilhamento de dados em larga escala, para a distribuição de conteúdo e para aplicações que implementem *multicast* na camada de aplicação. A sub-seção 2.1 abordará detalhes das Redes P2P (*Peer to Peer*).

2.1 Redes P2P (*Peer to Peer*)

A arquitetura de redes P2P possibilita a criação de uma rede sobreposta à rede IP tradicional, onde o software executado em cada nó da rede P2P equivale, em termos de funcionalidade, a todos os demais nós. Ou seja, na arquitetura P2P os nós podem atuar como servidores e clientes ao mesmo tempo. Já no modelo tradicional da arquitetura cliente/servidor é possível determinar, com facilidade, quais dos nós assumirá o papel de servidor e de cliente durante um processo típico de comunicação.

Alguns desafios que devem ser vencidos para a construção de uma rede P2P eficiente para distribuição de conteúdo, tais como:

- construção de uma árvore de distribuição e gerência de nós (pares) – a maioria das redes P2P empregadas na distribuição de conteúdo multimídia baseia-se na construção de uma árvore de distribuição para enviar arquivos de mídia para uma grande quantidade de clientes (receptores). O grande desafio é como construir uma árvore de distribuição eficiente e,

principalmente, como gerenciar essa árvore com a inclusão e a exclusão constante de nós na rede;

- estabilidade – os nós têm total liberdade para entrar e sair da rede P2P a qualquer momento, além disso, alguns nós podem simplesmente falhar, por motivos alheios à sua vontade. Cabe, então, a rede P2P empregar uma técnica eficiente que possibilite a recuperação rápida e sem grandes transtornos aos clientes se alguma falha ocorrer, caso contrário, certamente haverá uma interrupção no serviço;
- adaptação dinâmica – as redes P2P são implementadas em redes dinâmicas, que podem sofrer com problemas de congestionamento e aumento na quantidade de perdas. Conseqüentemente, durante a exibição de um fluxo de mídia, existe a necessidade de implementação de alguma técnica capaz de se adaptar a esses fenômenos.

A subseção a seguir apresenta como as redes P2P encontram-se classificadas sob a ótica de vários autores e relaciona os principais protocolos de aplicação e de rede para redes P2P.

2.1.1 Classificação das Redes P2P

Existem inúmeras classificações propostas na literatura para o modelo de arquitetura de redes P2P. Entre as mais relevantes, é possível citar as seguintes:

De acordo com [P2P Architect Project 2002], existem dois tipos principais de arquiteturas para sistemas baseados em redes P2P que encontram-se brevemente descritas abaixo e representadas graficamente através da Figura 3:

- arquitetura descentralizada – cada nó (*peer*) na rede é tratado de forma igualitária, inexistindo alguma forma de controle centralizado, ou seja, os nós são autônomos, responsáveis pelo controle (gerenciamento) e troca dos recursos (poder computacional e dados). Em relação à forma de comunicação, (a) os nós podem ser comunicar diretamente; (b) podem se comunicar de forma indireta através de uma estrutura hierárquica; (c) ou podem se comunicar de forma indireta através de uma estrutura em grafos;
- arquitetura semi-centralizada – nessa arquitetura existe, ao menos, um nó de controle, que pode ser responsável por manter um controle rígido de toda a rede, ou simplesmente atuar como um ponto de referência central para os demais nós da rede. Em relação à forma de comunicação, (d) os nós podem se comunicar diretamente entre si, mas normalmente necessitam de alguma referência do servidor central; (e) os nós comunicam-se exclusivamente com o servidor central; (f) os nós podem se comunicar com o servidor central ou com os demais nós, desde que obtenhas as referências necessárias com o servidor central; (f) os servidores centrais podem se comunicar entre si.

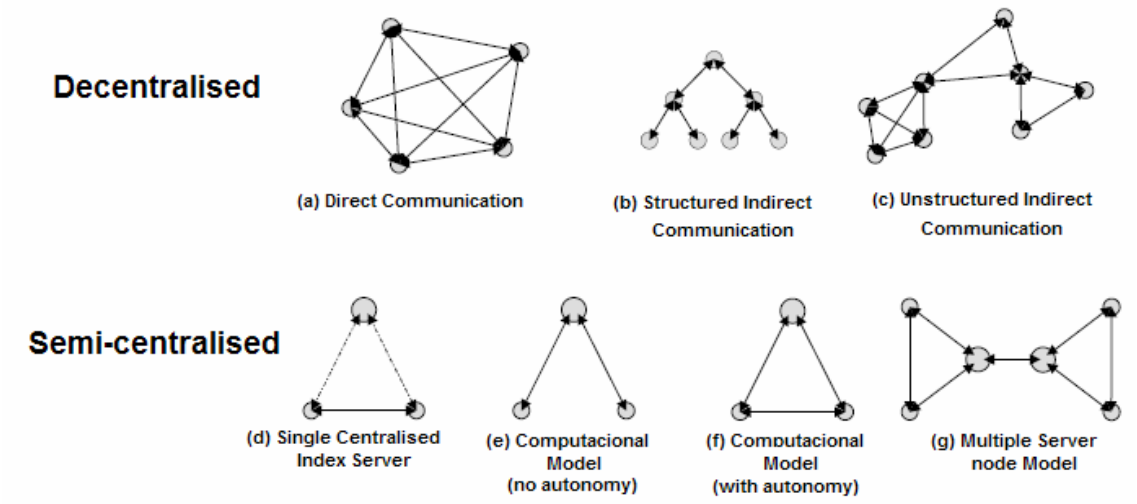

Figura 3 - Modelo de Arquiteturas [P2P Architect Project 2002]

De acordo com [Lv et al., 2002], existem três tipos principais de arquiteturas para sistemas baseados em redes P2P que encontram-se brevemente descritas abaixo:

- arquitetura centralizada – nesta arquitetura, os nós (pares) da rede P2P encaminham suas requisições para um servidor central de diretórios com objetivo de localizar nós que disponham dos arquivos solicitados. Como exemplo desta arquitetura é possível citar o Napster e outros sistemas similares que disponham de uma arquitetura de diretórios atualizada constantemente e hospedada em servidores centrais. Deve-se, entretanto, ressaltar que esta abordagem centralizada não é escalável e ainda sofre com o fato de possuir pontos de falha;
- arquitetura descentralizada e estruturada – os sistemas baseados nesta arquitetura são descentralizados, ou seja, não possuem uma arquitetura de diretórios centralizada, mas são, de certa forma, estruturados. Essa conotação estruturada define que a topologia da rede P2P é controlada e que os arquivos são dispostos em pontos específicos, tornando a sua localização uma tarefa fácil. O processo de distribuição e busca de arquivos é baseado na abordagem DHT (*Distributed Hash Table*), que é implementado pelos sistemas P2P Chord [Stoica et al. 2001], Pastry [Rowstron et Druschel, 2001], Tapestry [Zhao et al. 2001] etc;
- arquitetura descentralizada e não estruturada – nesta arquitetura não existe um servidor central e nem um controle preciso sobre a disposição de arquivos. Logo, para que um nó possa localizar um arquivo, ele deverá questionar seus vizinhos, sendo a inundação de solicitações aos vizinhos, até um certo limite da rede, a forma mais utilizada. Essa abordagem recupera-se rapidamente de entradas e saídas de nós na rede, mas o mecanismo de busca não é escalável. O Gnutella e KaZaA são exemplos desta arquitetura.

Segundo [Lao et al. 2005], os protocolos para ALM podem ser divididos em duas categorias: estruturados e não estruturados. Os esquemas estruturados constroem redes sobrepostas baseadas em DHT (*Distributed Hast Table*). Já os esquemas não estruturados podem ser organizados através das abordagens *mesh-first* e *tree-first*, que serão discutidas em seguida, conforme apresentado pela Figura 4.

[Albuquerque et al. 2006, Banerjee et Bhattacharjee 2002, Proença 2006, Zhao et al. 2006] afirmam que as propostas para ALM normalmente organizam os membros do grupo *multicast* em duas topologias: uma topologia para controle do grupo *multicast* e uma topologia para transmissão de dados. Na topologia de controle, os membros do grupo (pares) trocam, periodicamente, mensagens entre si, com objetivo de identificar e reparar possíveis particionamentos causados por saídas inesperadas de membros do grupo e falhas na rede física, além de otimizar a rede sobreposta. A topologia de dados é um subconjunto da topologia de controle e tem por função identificar os caminhos pelos quais os dados serão enviados através da rede sobreposta.

A topologia de dados é estruturada em forma de uma árvore (*tree*), já a topologia de controle, por dispor de maior conectividade entre os membros, é estruturada em malha (*mesh*). Desta forma, dependendo da seqüência de construção das topologias de dados e de controle, as ALMs podem ser classificadas em *tree-first* e *mesh-first*.

Na abordagem *mesh-first* os membros do grupo *multicast* se organizam, inicialmente, de forma distribuída em uma topologia sobreposta em malha. Como nas redes em malha os nós dispõem de mais de um caminho para alcançar os demais nós da rede, torna-se necessário a aplicação de um protocolo para construção de uma árvore geradora mínima, denominado algoritmo MST (*Minimun Spanning Tree*). Desta forma, a construção das árvores de distribuição na abordagem *mesh-first* é realizada em duas fases: na primeira fase é construída uma rede em malha, que pode ser representada por um grafo com alta conectividade entre os membros; na segunda fase são construídas MSTs enraizadas nos nós responsáveis pela distribuição do conteúdo, de forma que sejam determinados os menores caminhos reversos para todos os membros. Essas árvores são criadas utilizando-se algoritmos RPF (*Reverse Path Forwarding*), como o protocolo DVMRP (*Distance Vector Multicast Routing Protocol*) [Waitzman et al. 1988] que é específico para redes *multicast*, entre outros [Proença 2006, Zhao et al. 2006]. ESM [Chu et al. 2002] e TMesh [Wang et al. 2002] são exemplos de protocolos que baseiam-se na abordagem *mesh-first*.

Na abordagem *tree-first*, os membros do grupo *multicast* constroem, de forma distribuída, uma árvore compartilhada e selecionam, a partir de um subconjunto de nós conhecidos, seu nó pai. Como a árvore gerada é um grafo acíclico, se um nó vier a falhar ou deixar repentinamente (sem informar seus vizinhos) um grupo, a árvore poderá ser particionada e os membros de partições distintas perderão a comunicação. Conseqüentemente, as abordagens *tree-first* requerem mecanismos para identificar e reparar particionamentos. CoopNet [Padmanabhan et al. 2002] e Yoid [Francis 2000a, Francis 2000b] são exemplos de protocolos que baseiam-se na abordagem *tree-first*.

A Figura 4 apresenta uma classificação dos protocolos para construção de árvores de distribuição.

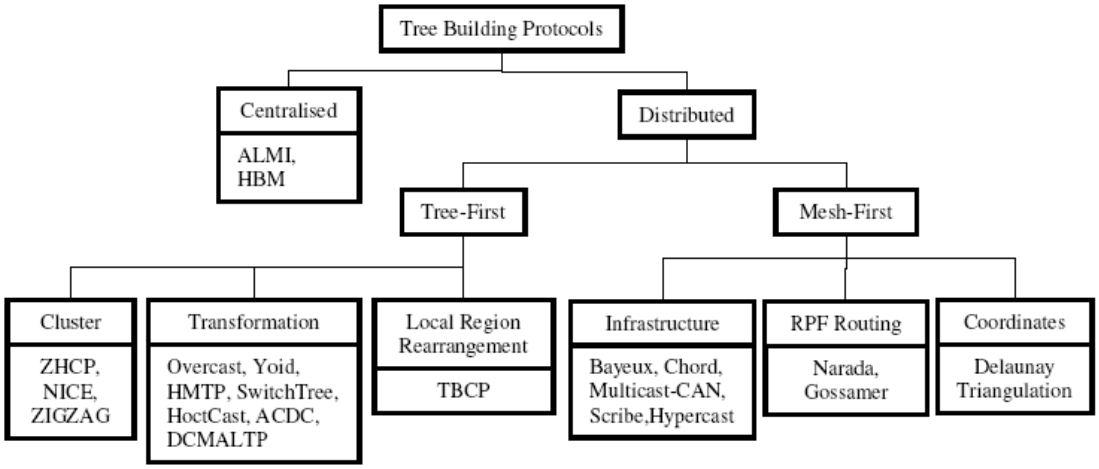

Figura 4 - Taxonomia de Protocolos para Construção de Árvores de Distribuição [Tan et al. 2003]

3 Algoritmos para Construção de MST

De acordo com [Graham et Hell, 1985], o problema de construção de uma árvore geradora mínima foi formulado, inicialmente, por Borůvka. Desde então, inúmeros artigos abordando o problema da Árvore Geradora Mínima (*Minimum Spanning Tree)* com propostas de resolução em tempo polinomial podem ser encontrados na literatura. Dentre as inúmeras propostas, os algoritmos Kruskal [Kruskal, 1956], Prim [Prim 1957] e Sollin [Sollin 1965] destacam-se por apresentarem uma complexidade de pior caso polinomial e por serem freqüentemente referenciados. Esses algoritmos serão descritos nas subseções a seguir e utilizados nas simulações na seção 4.

Considere que um grafo não direcionado $G = (N, A)$, onde $N = \{1, 2, 3, ..., N\}$ é um conjunto finito de vértices representando os membros de um determinado grupo da rede P2P sobreposta e onde $A = \{(i,j) \mid i,j \in N\}$ é um conjunto finito de arestas representando as conexões virtuais entre os membros desse determinado grupo da rede P2P sobreposta. Cada aresta tem um número real e positivo identificado por $C_{i,j}$ que determina o custo da aresta. Esse custo, dependendo da aplicação, pode ser denotado por distância, atraso de propagação, banda disponível etc.

Um subgrafo T de G é uma árvore se T for conexo e sem ciclos. O subgrafo T será uma árvore geradora de G se T contiver todos os nós de G, logo $|T| = N$. O subgrafo T será considerado uma Árvore Geradora Mínima se o custo de T for mínimo, ou seja, se o custo medido pela soma dos custos individuais de todas as arestas for o menor possível. A Figura 5 apresenta um grafo não direcionado completo e uma árvore geradora mínima extraída desse grafo.

As subseções seguintes descrevem os algoritmos utilizados nas simulações.

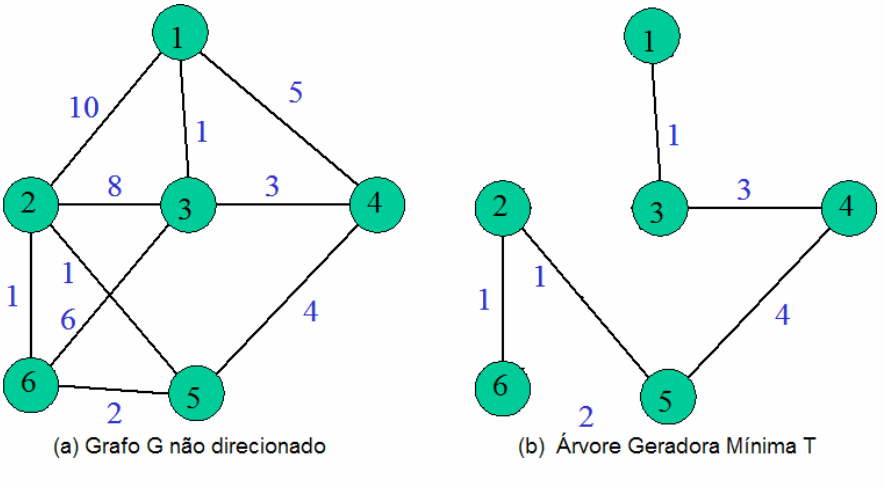

Figura 5 – Exemplo de Árvore Geradora Mínima

3.1 Algoritmo de Kruskal

Com objetivo de detalhar o funcionamento do algoritmo de Kruskal, vamos supor um grafo não direcionado $G = (N, A)$, onde $N = \{1, 2, 3, ..., N\}$ é um conjunto finito de vértices e onde $A = \{(i,j) \mid i,j \in N\}$ é um conjunto finito de arestas. De acordo com [Ahuja et al. 1993], a obtenção de uma árvore geradora mínima requer a execução dos seguintes passos:

(1) o algoritmo de Kruskal inicialmente ordena as arestas de forma não crescente em relação aos custos de cada aresta ($C_{i,j}$), inserindo-as em uma lista de arestas ordenada;

(2) em seguida, os $|N|$ vértices são inseridos na solução da árvore geradora mínima T;

(3) tenta-se inserir a primeira aresta da lista ordenada. Se a inserção formar um ciclo na árvore geradora mínima, essa aresta é descartada. Esse passo é repetido continuamente até que a condição disposta em (4) seja atendida;

(4) termina quando a quantidade de arestas inseridas na árvore geradora mínima for igual a $|N| - 1$

A Figura 6 ilustra, passo a passo, a construção de uma árvore geradora mínima baseada no algoritmo de Kruskal.

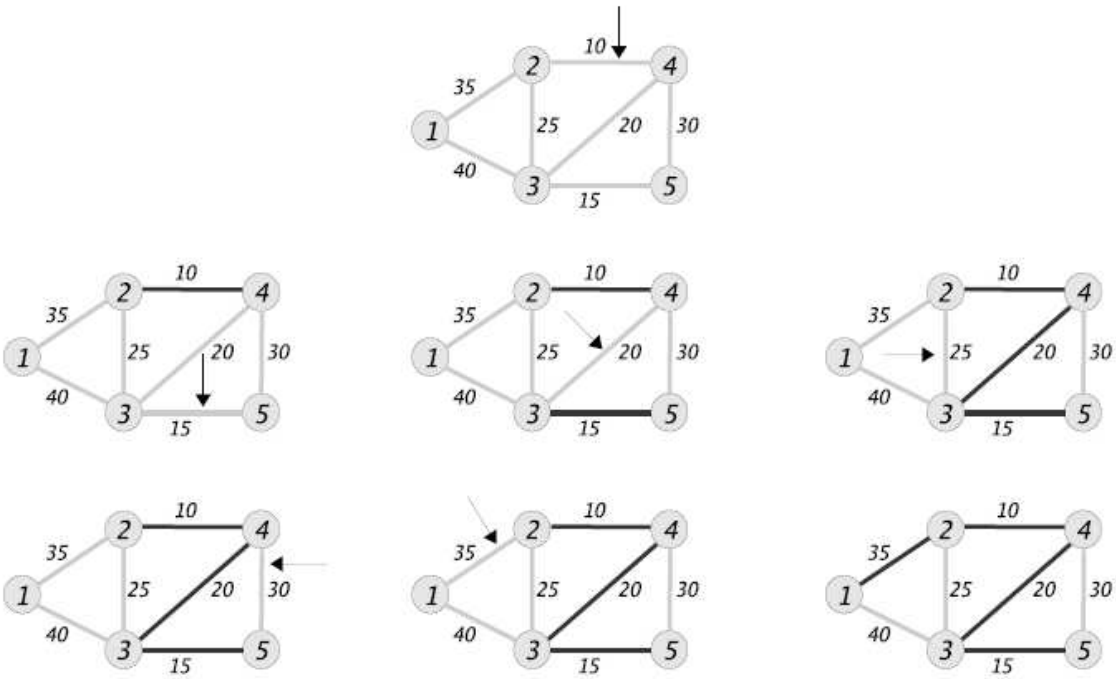

Figura 6 - Exemplo de aplicação do algoritmo de Kruskal [Ahuja et al. 1993]

3.2 Algoritmo de Prim

Com objetivo de detalhar o funcionamento do algoritmo de Prim, vamos supor um grafo não direcionado *G = (N, A)*, onde *N = {1, 2, 3, ..., N}* é um conjunto finito de vértices e onde *A = {(i,j) | i,j Є N}* é um conjunto finito de arestas. Segundo [Ahuja et al. 1993], a obtenção de uma árvore geradora mínima requer a execução dos seguintes passos:

(1) selecione, arbitrariamente, um vértice inicial *s* pertencente ao conjunto *N*;

(2) faça árvore geradora mínima *T´ := {s}*;

(3) enquanto existir algum vértice em *G* que não exista em *T´* faça;

(4) selecione a aresta de menor custo entre os vértices de *T´* e os vértices que não pertencem a *T´*;

(5) adicione a aresta selecionada e o vértice correspondente à *T´*;

(6) fim-enquanto

A Figura 7 ilustra, passo a passo, a construção de uma árvore geradora mínima baseada no algoritmo de Prim.

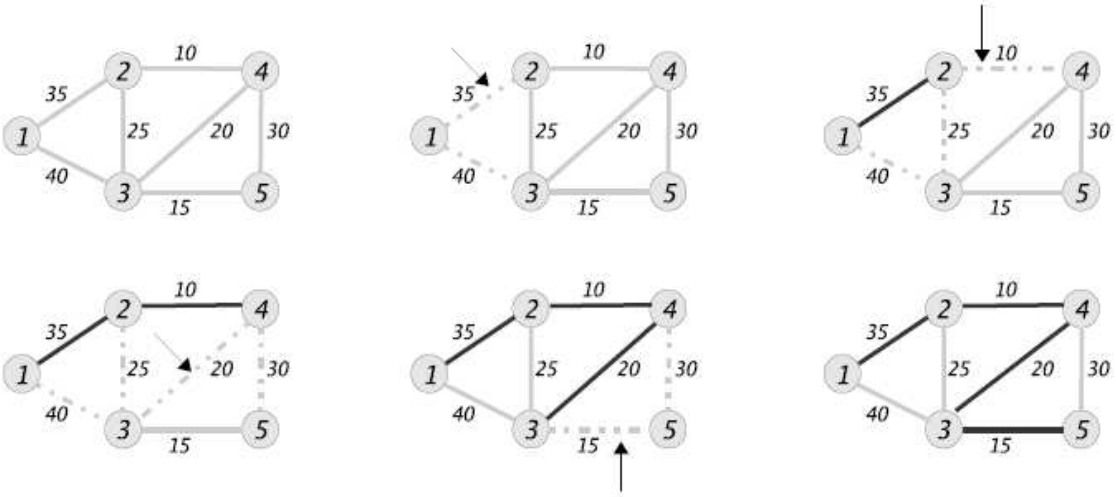

Figura 7 – Exemplo de aplicação do algoritmo de Prim [Ahuja et al. 1993]

3.3 Algoritmo de Sollin

Com objetivo de detalhar o funcionamento do algoritmo de Sollin, vamos supor um grafo não direcionado $G = (N, A)$, onde $N = \{1, 2, 3, ..., N\}$ é um conjunto finito de vértices e onde $A = \{(i,j) \mid i,j \in N\}$ é um conjunto finito de arestas. A cada iteração, o algoritmo de Sollin mantém um conjunto de árvores desconectadas e as conecta através da aresta que apresentar o menor custo para as demais árvores. O resultado de cada iteração é a geração de uma árvore geradora mínima. Vejamos, as seguir, os passos para execução deste algoritmo [Ahuja et al. 1993].

(1) para cada $i \in N$ faça $S_i = \{i\}$;

(2) faça árvore geradora mínima $T' := \{ \}$;

(3) enquanto $|T'| < (|N| - 1)$ faça;

(4) para cada árvore S_k encontre o vizinho mais próximo de $\{S_k, i_k, j_k\}$;

(5) para cada árvore S_k faça

(6) se os nós i_k e j_k pertencerem a árvores diferentes então

(7) unir $\{(i_k, j_k)\}$ em uma mesma árvore

(8) $T' = T' \cup \{(i_k, j_k)\}$;

(9) fim-para;

(10) fim-enquanto;

A Figura 8 ilustra, passo a passo, a construção de uma árvore geradora mínima baseada no algoritmo de Sollin.

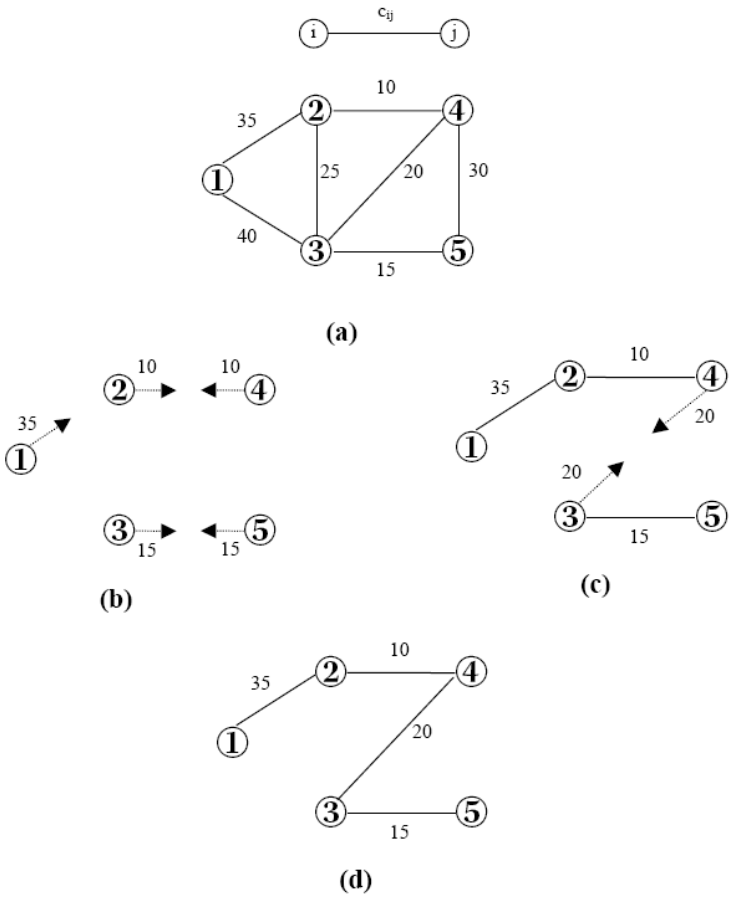

Figura 8 – Exemplo de aplicação do algoritmo de Sollin [Ahuja et al. 1993]

3.4 Algoritmos MSTmf (*Minimum Spanning Tree for Mesh First*)

Os algoritmos denominados MSTmf-Kruskal, MSTmf-Prim e MSTmf-Sollin foram constituídos dos respectivos algoritmos originais com a aplicação da seguinte heurística: árvores com menor profundidade entre a raiz e seus sucessores de maior nível tendem a manter um menor atraso e, no caso de redes de computadores, tendem ainda a consumir menos recursos. Conseqüentemente, o uso preferencial de vértices que estejam conectados a maior quantidade possível de outros vértices aumenta a possibilidade de redução da profundidade da árvore geradora mínima produzida pelos algoritmos.

Essa heurística foi adotada nos três algoritmos modificados nos casos em que existiam mais de uma alternativa (vértice ou aresta) com o mesmo custo a ser selecionada para inclusão na árvore geradora mínima. As subseções a seguir detalham as modificações produzidas.

3.4.1 MSTmf-Kruskal

Em relação ao algoritmo original, exposto na seção 3.1, foi incluído o item (2). Caso existam arestas com mesmo custo, o algoritmo original não determina como essas arestas devem ser tratadas. No algoritmo MSTmf-Kruskal, arestas com o mesmo custo

são reordenadas de forma que as arestas cujos vértices apresentem uma maior quantidade de arestas sejam posicionados preferencialmente.

(1) o algoritmo de Kruskal inicialmente ordena as arestas de forma não crescente em relação aos custos de cada aresta ($C_{i,j}$), inserindo-as em uma lista de arestas ordenada;

(2) caso existam arestas com mesmo custo, reordenar de forma que as arestas cujos vértices apresentem uma maior quantidade de arestas sejam posicionados preferencialmente;

(3) em seguida, os $|N|$ vértices são inseridos na solução da árvore geradora mínima T;

(4) tenta-se inserir a primeira aresta da lista ordenada. Se a inserção formar um ciclo na árvore geradora mínima, essa aresta é descartada. Esse passo é repetido continuamente até que a condição disposta em (4) seja atendida;

(5) termina quando a quantidade de arestas inseridas na árvore geradora mínima for igual a $|N| - 1$

3.4.2 MSTmf-Prim

Em relação ao algoritmo original, exposto na seção 3.2, o item (1) foi modificado e foi incluído o item (5). Em relação à modificação do item (1), ao invés de selecionar aleatoriamente um vértice para dar início à construção da árvore geradora mínima, o algoritmo MSTmf-Prim seleciona sempre o vértice que apresentar a maior quantidade de arestas. Já em relação à inclusão do item (5), caso existam arestas com mesmo custo, o algoritmo original não determina como essas arestas devem ser tratadas. No algoritmo MSTmf-Prim, arestas com o mesmo custo são tratadas de forma diferenciada, possibilitando que as referidas arestas cujos vértices estejam conectados a uma maior quantidade de arestas sejam selecionadas preferencialmente.

(1) selecione o vértice inicial *s com a maior quantidade de arestas*, pertencente ao conjunto N;

(2) faça árvore geradora mínima $T' := \{s\}$;

(3) enquanto existir algum vértice em G que não exista em T' faça;

(4) selecione a aresta de menor custo entre os vértices de T' e os vértices que não pertencem a T';

(5) caso existam duas ou mais arestas de menor custo entre os vértices de T´ e os vértices que não pertencem a T, selecionar a aresta cujos vértices apresentem a maior quantidade de arestas;

(6) adicione a aresta selecionada e o vértice correspondente à T';

(7) fim-enquanto

3.4.3 MSTmf-Sollin

Em relação ao algoritmo original, exposto na seção 3.3, o item (4) foi modificado, possibilitando a aplicação da heurística que norteia os algoritmos MSTmf propostos.

(1) para cada $i \in N$ faça $S_i = \{i\}$;

(2) faça árvore geradora mínima $T' := \{\ \}$;

(3) enquanto $|T'| < (|N| - 1)$ faça;

(4) para cada árvore S_k encontre o vizinho mais próximo de $\{S_k, i_k, j_k\}$, *minimizando o custo e maximizando a quantidade de arestas conectadas à* i_k *e* j_k;;

(5) para cada árvore S_k faça

(6) se os nós i_k e j_k pertencerem a árvores diferentes então

(7) unir $\{(i_k, j_k)\}$ em uma mesma árvore

(8) $T' = T' \cup \{(i_k, j_k)\}$;

(9) fim-para;

(10) fim-enquanto;

4 Simulações e Resultados

O objetivo deste trabalho é analisar os resultados obtidos através de simulações com os algoritmos Kruskal [Kruskal, 1956], Prim [Prim 1957], Sollin [Sollin 1965] e com os algoritmos MSTmf-Kruskal, MSTmf-Prim e MSTmf-Sollin desenvolvidos especificamente para atender protocolos distribuídos e estruturados para ALM com abordagem *mesh-first*.

O estudo comparativo foi baseado na análise da profundidade da árvore geradora mínima produzida pelos referidos algoritmos, que foram implementados na linguagem C ANSI. Para esse estudo comparativo foi desenvolvido um programa específico para a geração de topologias que simulem a realidade de redes P2P.

As subseções a seguir apresentam a metodologia utilizada para a realização das simulações e a análise dos resultados obtidos.

4.1 Metodologia

Com o intuito de analisar o comportamento dos algoritmos Kruskal, Prim, Sollin, MSTmf-Kruskal, MSTmf-Prim e MSTmf-Sollin foram realizadas 200 simulações com 100, 200, 300, 500, 750 e 1000 vértices para cada um dos algoritmos, para cada cenário de avaliação. Entende-se que a avaliação através de 200 topologias distintas, para cada cenário de avaliação, seja suficiente para eliminar possíveis distorções provenientes de alguma topologia que não caracterize adequadamente as redes P2P. O programa gerador de topologia produz, para cada uma das simulações, um arquivo de entrada formatado que será utilizado por todos os algoritmos, de forma que o resultado final seja justo.

O valor associado ao custo de cada aresta (enlace de comunicação) e a quantidade de arestas de cada vértice (membro do grupo P2P) são gerados de forma

aleatória. As topologias são armazenadas em um arquivo de histórico, possibilitando que não haja repetição de topologia entre as 200 simulações para cada quantidade de vértices.

O custo mínimo associado a cada aresta e a quantidade de arestas por vértice variaram de acordo o cenário de simulação. Foram realizadas simulações com seis cenários distintos, a saber:

1) Com no máximo 4 arestas por vértice e com custo máximo de cada aresta limitado a 5;
2) Com no máximo 4 arestas por vértice e com custo máximo de cada aresta limitado a 10;
3) Com no máximo 4 arestas por vértice e com custo máximo de cada aresta limitado a 20;
4) Com no máximo 8 arestas por vértice e com custo máximo de cada aresta limitado a 5;
5) Com no máximo 8 arestas por vértice e com custo máximo de cada aresta limitado a 10;
6) Com no máximo 8 arestas por vértice e com custo máximo de cada aresta limitado a 20;

Embora a métrica de desempenho seja essencial para avaliação de todo e qualquer algoritmo, esse trabalho considera somente a profundidade (nível ou altura) da árvore geradora mínima produzida pelos algoritmos avaliados como métrica de avaliação. Para aplicações ALM, especificamente para aplicações de redes P2P, árvores com menor profundidade entre a raiz e seus sucessores de maior nível tendem a manter um menor atraso. Este atraso refere-se à latência inserida pela propagação do sinal nos enlaces de comunicação (arestas) e pelo atraso inserido pelo processamento nos nós intermediários (vértices). Essa heurística foi considerada na construção dos algoritmos MSTmf.

4.2 Resultados

Os resultados das simulações demonstram, claramente, que os algoritmos propostos apresentaram árvores com menor profundidade que os respectivos algoritmos originais na imensa maioria das avaliações. A Figura 9 apresenta um resumo do desempenho de todos os algoritmos, possibilitando a comparação, cenário a cenário, do comportamento dos algoritmos Kruskal, Prim, Sollin e dos algoritmos propostos MSTmf-Kruskal, MSTmf-Prim e MSTmf-Sollin.

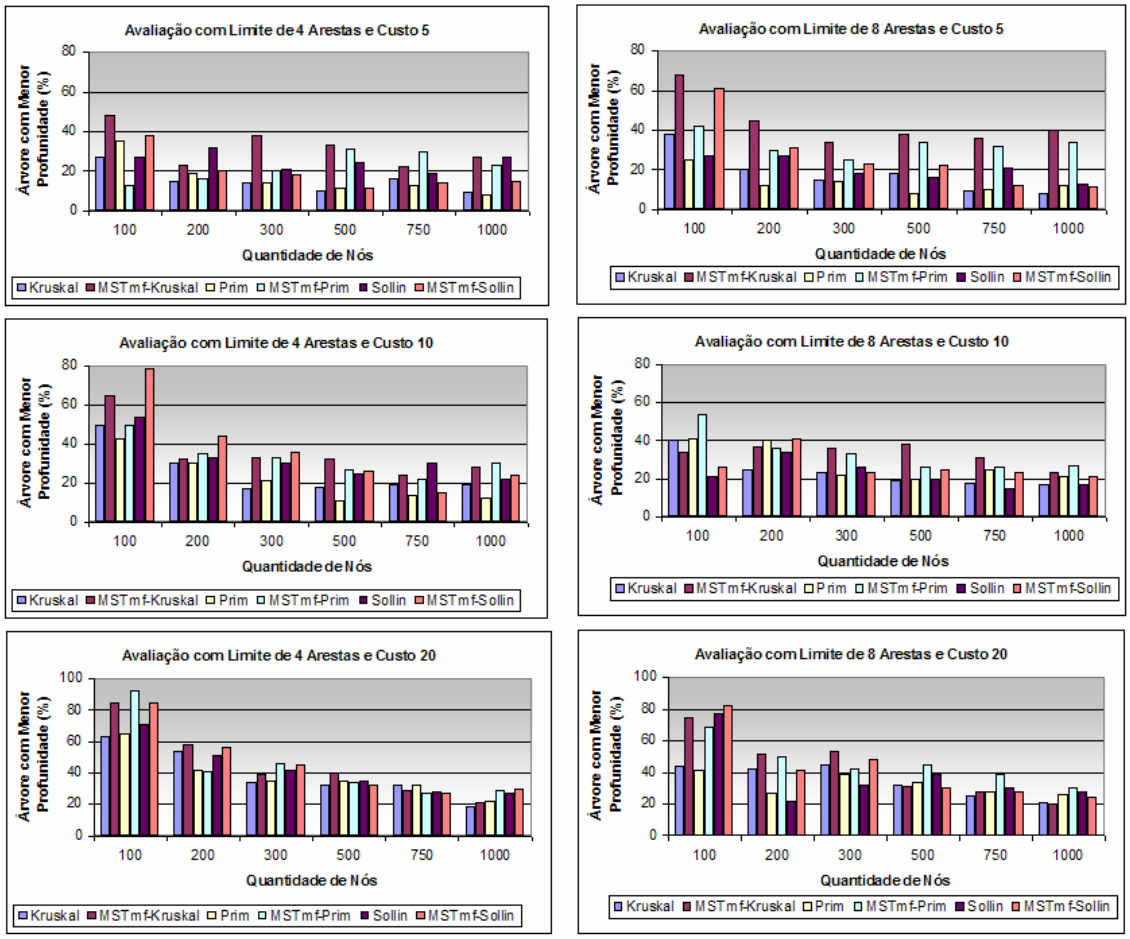

Figura 9 – Árvore com menor profundidade

A Figura 10 apresenta uma comparação entre cada um dos algoritmos propostos e os respectivos algoritmos originais. Os gráficos demonstram o percentual de vezes que os algoritmos propostos apresentaram melhores resultados que os algoritmos originais, ou seja, o percentual de vezes que os algoritmos propostos conseguiram gerar uma árvore geradora mínima com profundidade menor que os respectivos algoritmos originais. Os gráficos da Figura 10 também apresentam o percentual de vezes que os algoritmos propostos e os algoritmos originais obtiveram árvores geradoras mínimas com a mesma profundidade.

Figura 10 - Comparação com Algoritmos Originais

A Figura 11 apresenta um gráfico com o percentual de simulações onde pelo menos um algoritmo MSTmf obteve, de forma exclusiva, uma árvore com menor profundidade que a gerada por todos os outros algoritmos originais. Nos cenários iniciais, ou seja, com poucas arestas (no máximo quatro), os resultados obtidos pelos algoritmos assemelham-se bastante. No entanto, à medida que se aumenta o intervalo de variação do custo (passando de cinco para vinte), os algoritmos MSTmf começam a se destacar. Esse destaque é realçado com o aumento do intervalo da quantidade de arestas por vértice, passando de quatro para oito.

Os algoritmos MSTmf apresentam resultados significativamente melhores em cenários com muitos vértices e com intervalos de quantidade de arestas e de custo maiores.

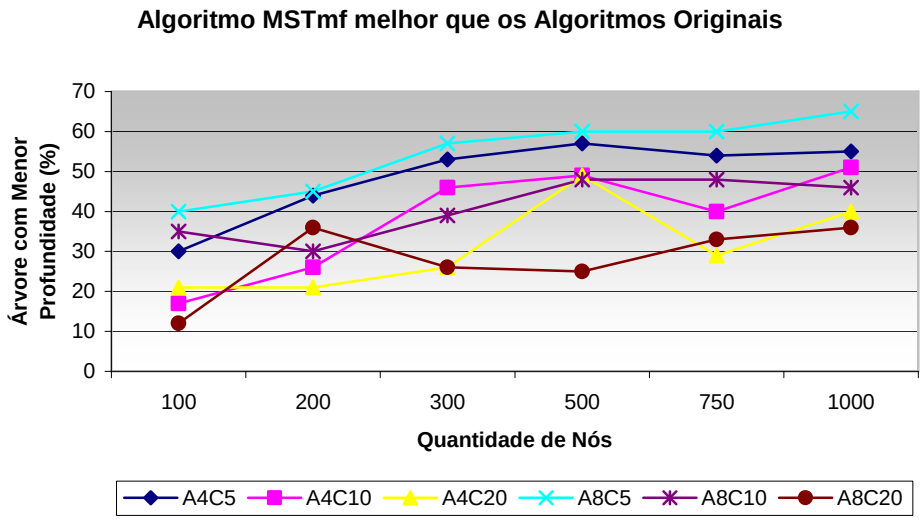

Figura 11 - Algoritmo MSTmf melhor que os Algoritmos Originais

5 Conclusão

Este trabalho tem por objetivo avaliar e comparar a métrica de profundidade das árvores geradoras mínimas produzidas pelos algoritmos Kruskal [Kruskal, 1956], Prim [Prim 1957] e Sollin [Sollin 1965] com os algoritmos propostos, denominados MSTmf (*Minimum Spanning Tree for mesh-first*), desenvolvidos especificamente para atender protocolos distribuídos e estruturados para ALM com abordagem *mesh-first*.

Embora a métrica de desempenho seja essencial para avaliação de todo e qualquer algoritmo, esse trabalho considera somente a profundidade (nível ou altura) da árvore geradora mínima produzida pelos algoritmos avaliados como métrica de avaliação. Para aplicações ALM, especificamente para aplicações de redes P2P, árvores com menor profundidade entre a raiz e seus sucessores de maior nível tendem a manter um menor atraso. Este atraso refere-se à latência inserida pela propagação do sinal nos enlaces de comunicação (arestas) e pelo atraso inserido pelo processamento nos nós intermediários (vértices). Essa heurística foi considerada na construção dos algoritmos MSTmf.

Os resultados das simulações demonstram que os algoritmos propostos MSTmf-Kruskal, MSTmf-Prim e MSTmf-Sollin apresentaram, consistentemente, árvores geradoras mínimas com profundidade inferior aos algoritmos originais, atendendo, satisfatoriamente, ao propósito.

Referências Bibliográficas

[Ahuja et al., 1993] Ahuja, R., Magnanti, T., et Orlin, J. **Network Flows : Theory, Algoritms and Applications**, Prentice Hall, 1993, ISBN: 0-13-617549-X

[Albuquerque et al., 2006] Albuquerque, C., Proença, T. et Oliveira, E. **TVoIP: TV sobre IP – Arquiteturas para Transmissão em Larga Escala**, Minicurso da SBRC, 2006, maio, cap. 3.

[Banerjee et Bhattacharjee, 2002] Banerjee S., et Bhattacharjee, B. **A Comparative Study of Application Layer Multicast Protocols**, Univ. Maryland, College Park, Technical Report, 2002.

[Chu et al., 2002] Chu, Y., Rao, S., et al. **A Case for End System Multicast**, IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast, 2002, outubro, vol. 20(8), p. 1456-1471.

[Deering e Cheriton, 1990] Deering, S., et Cheriton, D. **Multicast routing in datagram internetworks and extended LANs**, ACM Transactions on Computer Systems, 1990, maio, vol. 8(2), p. 85–110.

[Francis, 2000a] Francis, P. **Yoid: Extending the Internet Multicast Architecture**, Unrefereed report, abril, p. 1-38.

[Francis, 2000b] Francis, P. Yoid. **Tree Management Protocol (YTMP) Specification**, Specification, abril, p. 1-60.

[Graham et Hell, 1985] Graham, L. et Hell, P. **On the History of the Minimum Spanning Tree Problem**, Annals of the History of Computing, 1985, vol. 7, p. 43-57.

[Kruskal, 1956] Kruskal, B. **On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem**, *Proc. Amer. Math. Soc.* **7**, 1956, p. 48-50.

[Lao et al., 2005] Lao, L., Cui, J., Gerla, M., et Maggiorini, D. **A Comparative Study of Multicast Protocols: Top, Bottom, or in the Middle?**, IEEE INFOCOM, 2005, março, vol. 4, p. 2809-2814.

[Liao et al., 2006] Liao, X., Jin, H., Liu, Y. et al. **AnySee: Peer-to-Peer Live Streaming**, IEEE INFOCOM, 2006, abril, p. 1-10.

[Lua et al., 2004] Lua, E., Crowcroft, J., Pias, M., Sharma, R., et Lim, S. **A Survey and Comparison of Peer-to-Peer Overlay Network Schemes**, IEEE Communications Survey and Tutorial, 2004, março, vol. 7(2), p. 72- 93.

[Lv et al., 2002] Lv, Q., Cao, P., Cohen, E. et al. **Search and Replication in Unstructured Peer-to-Peer Networks**, 16th ACM International Conference on Supercomputing, 2002 (ICS'02), junho, p. 84-95.

[P2P Architect Project 2002] P2P Architect Project. **Ensuring Dependability of P2P Applications at Architectural Level - Comprehensive Survey of Contemporary P2P Technology**, http://www.atc.gr/p2p_architect/results/0101F05_P2P Survey.pdf.

[Padmanabhan et al., 2002] Padmanabhan, V., Wang, H., Chou, P., et Sripanidkulchai, K. **Distributing Streaming Media Content Using Cooperative Networking**, 12th NOSSDAV - International Workshop on Network and Operating Systems Support for Digital Audio and Video, 2002, maio, p. 177–186.

[Prim, 1957] Prim, R. **Shortest Connection Networks and Some Generalizations**, Bell System Technical Journal, 1957, p. 1389-1401.

[Proença, 2006] Proença, T. **Proposta e avaliação de uma arquitetura escalável para distribuição de TV na Internet**, Dissertação de Mestrado, Universidade Federal Fluminense (UFF), 2006, Novembro.

[Rocha et al., 2004] Rocha, J., Domingues, M., Callado, A., Souto, E., Silvestre, G., Kamienski, C., et Sadok, D. **Peer-to-Peer: Computação Colaborativa na Internet**, Minicurso da SBRC, 2004, cap. 1, p. 1-45.

[Rowstron et Druschel, 2001] Rowstron, A. et Druschel, P. **Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems**, 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, outubro, p. 329-350.

[Sollin, 1965] Sollin, M., **Le trace de canalisation**. Programming, Games, and Transportation Networks, ed. C. Berge e A. Ghouilla-Houri, 1965.

[Stoica et al., 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, M., et Balakrishnan, H. **Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications**, ACM SIGCOMM, 2001, agosto, p. 149-160.

[Stravrou et al., 2002] Stavrou, A., Rubenstein, D., et Sahu, S. **A lightweight, robust P2P system to handle flash crowds**, IEEE International Conference on Network Protocols (ICNP), 2002, novembro, p. 226-235.

[Tan et al., 2003] Tan, S., Waters, G. et Crawford, J. **A Survey and Performance Evaluation of Scalable Tree-based Application Layer Multicast Protocols**, Technical Report 9-03, University of Kent, 2003, julho.

[Waitzman et al., 1988] Waitzman, D., Partridge, C., et Deering, S. **Distance Vector Multicast Routing Protocol**, RFC 1075, 1988, novembro.

[Wang et al., 2002] Wang, W., Helder, D., et al. **Overlay Optimizations for End-host Multicast**, Proceedings of 4th International Workshop on Networked Group Communication (NGC), 2002, outubro.

[Zhao et al., 2001] Zhao, B., Kubiatowicz, J, et Joseph, A. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing, University of California at Berkeley, 2001.

[Zhao et al. 2006] Zhao, Q., He, Y., et Zhang, J. **A Hybrid Approach for Overlay Multicast**, IEEE 1st International Multi-Symposiums on Computer and Computational Sciences, 2006 (IMSCCS), 2006, junho, vol. 1, p. 496- 502.