Otimizando o cascateamento de MCU's em Videoconferências H.323

Thiago Curvelo dos Anjos[1], Lucídio dos Anjos Formiga Cabral[2], Guido Lemos de Souza Filho[1]

[1] Laboratório de Aplicações de Vídeo Digital – Universidade Federal da Paraíba (UFPB)

[2] Departamento de Estatística – Universidade Federal da Paraíba (UFPB)

{curvelo,guido}@lavid.ufpb.br, lucidio@de.ufpb.br

Abstract. *In this work we propose an approach to build and maintain a minimum latency topology, for a multipoint videoconference service overlay network, which uses multipoint control units (MCU's) cascading. However, setting MCU's and clients on the network brings a combinatorial intractable problem. In order to solve it, we developed a heuristic, based on the Simulated Annealing algorithm, which allows implementing a coordinator entity capable of maintain multiples conversation rooms, instanced in minimum latency trees of MCU's and clients. We also show an integer linear programming model for the problem.*

Resumo. *Neste trabalho é proposta uma abordagem para construção e manutenção da topologia de latência mínima, em uma rede overlay de um serviço de videoconferência multiponto, que utiliza o cascateamento das unidades de controle multiponto (MCU's). Todavia, a configuração de MCU's e clientes na rede gera um problema combinatório intratável. A fim de solucionar esse problema, desenvolveu-se uma heurística, baseada no algoritmo Simulated Annealing, a qual torna possível o desenvolvimento de uma entidade coordenadora capaz de manter múltiplas salas de conversação, instanciadas em árvores de MCU's e clientes, de mínima latência. Em tempo, é mostrada uma modelagem usando programação inteira para o problema.*

1. Introdução

Aplicações de videoconferência, devido ao tipo de mídia empregada – áudio e vídeo – demandam um alto custo de largura de banda. Este fato, associado à sensibilidade que as aplicações interativas em tempo real têm ao desempenho das redes comutadas por pacotes (sobretudo a latência), constituem nas principais barreiras para a popularização da videoconferência, sobretudo da conferência multiponto (MP) [Civanlar 2005].

Neste tipo de aplicação (videoconferência MP), onde há comunicação de grupo, o consumo de banda passante pode ser realmente impactante caso não haja um planejamento da infra-estrutura de comunicação para o uso racional deste recurso.

Sistemas de videoconferência que suportem comunicação multiponto distinguem-se, quanto a sua infra-estrutura de comunicação, em descentralizados, centralizados e híbridos [Leopoldino 2001].

No modelo descentralizado, a comunicação ocorre normalmente de duas formas: (1) utilizando-se *multicast* IP ou (2) através de conexões *unicast* entre os participantes. Apesar dos inegáveis ganhos do uso do *multicast* IP (como robustez e escalabilidade), sabe-se que a implantação do *multicast* em escala global não é uma realidade hoje em dia, o que impede esta opção de ser considerada uma solução viável [Yeo 2004]. Por outro lado, utilizando-se conexões *unicast* para interligar todos os terminais, obtém-se um cenário onde cada participante tem de enviar, receber e tratar $(n\text{-}1)^2$ fluxos. Isso gera uma limitação na escalabilidade tanto devido ao consumo de banda, quanto em processamento.

Existem esforços no sentido de melhorar a eficiência no consumo de banda por parte das videoconferências descentralizadas. Em [Civanlar 2005] é proposta uma arquitetura para implantação de videoconferência P2P, onde os pacotes de dados são transmitidos através de cadeias. Nestas cadeias, uma seqüência de *hosts* é pré-definida. O *host* na cabeça da cadeia, gerador do fluxo atual, envia seus dados para o seguinte, que retransmitirá o dado adiante, até o final da cadeia. Diferentes cadeias são definidas com cada um dos *hosts* na cabeça da estrutura. Deste modo todo nó está, a cada instante, enviando um fluxo e recebendo outro. Esta abordagem trata bem o problema de consumo de banda, por outro lado o atraso fim a fim – crítico às videoconferências – é negligenciado. Neste esquema de cadeias, o último nó de uma corrente receberá o dado oriundo *host* da cabeça após um período igual ao somatório dos atrasos de transmissão individual de cada nó para o seu seguinte da cadeia.

Em sistemas de videoconferência que implementem o modelo centralizado, os participantes se comunicam através de conexões *unicast* com uma unidade de controle multiponto (MCU), responsável por realizar um controle centralizado da conferência, como também por mesclar a informação audiovisual dos clientes e enviar a cada um deles um único fluxo com os dados mixados. Com este modelo não há a necessidade de suporte a *multicast* IP, mas ainda assim há um uso eficiente da rede, pois cada terminal envia apenas um fluxo único. Além disso, devido ao fato de terem de receber e tratar um fluxo único, é possível que *hosts* com menor poder computacional (ex. dispositivos móveis) utilizem-se do serviço, já que não será necessário que os terminais mesclem os fluxos. MCU's podem implementar adicionalmente outras funcionalidades, que agregam vantagens ao modelo, como negociação de capacidades com os terminais (*hosts* com necessidades diferentes recebem fluxos de qualidade diferente) e compartilhamento de documentos. Contudo, este modelo apresenta as desvantagens de possuir um ponto único de falha (a MCU), e uma baixa escalabilidade, devido à capacidade de a MCU ser fator limitante do número de usuários.

Há também o modelo híbrido que é uma mescla do modelo centralizado e do descentralizado. Normalmente é utilizado quando se tem acesso a uma infra-estrutura de comunicação *multicast*, mas ainda faz uso de uma MCU, por exemplo, para o controle da conferência e compartilhamento de documentos.

Em [Vasconcelos 2004] e [Souza Filho 2006] é proposto um sistema de videoconferência chamado *Dynavideo Conference System* (DCS), que apresenta uma arquitetura variante do modelo clássico videoconferência centralizada, que faz proveito do cascateamento de MCU's (previsto no padrão H.323 [ITU-T 2003]), no intuito de

prover uma maior escalabilidade para o modelo e minimizar o problema de ponto único de falha.

O padrão H.323 é parte da família de recomendações do ITU-T (*International Telecommunication Union – Telecommunication Standardization Sector*) da série H que trata de sistemas audiovisuais e multimídia, especificando comunicações multimídia em redes baseadas em pacotes sem garantia de qualidade de serviço. Muito embora o mercado esteja em contínuo crescimento, o H.323 ainda é o padrão predominante para aplicações de videoconferências sobre IP [Aposkitis 2005]. Nele são definidos alguns componentes, dentre eles os terminais e as MCU. Os terminais são os *hosts* participantes da videoconferência, com capacidade para comunicação audiovisual bidirecional; enquanto a MCU – um tipo especial de terminal – como já citada anteriormente, tem a função de controle da conferência, como também é responsável por mixar a informação audiovisual dos participantes em um fluxo único. Adicionalmente, o H.323 prevê uma característica opcional para as MCU's (embora não determine exatamente como): a possibilidade de cascateamento. Com essa função, duas ou mais MCU's trocam mensagens com o objetivo de se unirem e dividirem entre si o esforço de processamento de uma videoconferência.

A proposta do DCS é, através do cascateamento de MCU's, estender o modelo centralizado clássico de videoconferências multiponto de modo que, ao invés de uma MCU intermediando a comunicação entre os terminais, tenha-se uma "nuvem" de MCU's, cascateados sob demanda, como ilustra a Figura 1. Além disso, o DCS implementa a funcionalidade de redirecionamento de chamadas de clientes, que permite mover clientes entre MCU's, durante a conferência.

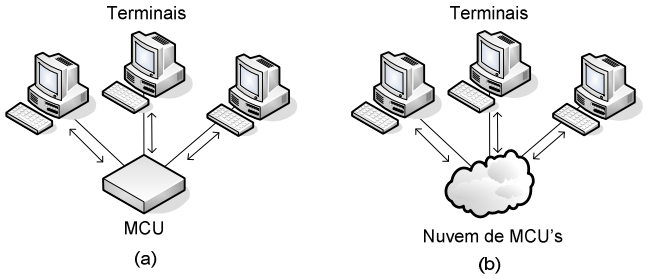

Figura 1. (a) modelo centralizado clássico; (b) modelo centralizado com cascateamento de MCU's

Com esta abordagem é possível assumir que, tendo-se múltiplas MCU's geograficamente espalhadas, estenda-se um cenário de um serviço de comunicação cliente-servidor, para um segundo cenário onde é utilizada uma infra-estrutura de *multicast* em nível de aplicação (*Application-Level Multicast* – ALM [Yeo 2004]).

A utilização de infra-estruturas de ALM constitui em uma das formas mais adotadas atualmente para suprir necessidades de comunicação de grupo na Internet, tanto para aplicações de fonte única (ex. IPTV) quanto de múltiplas fontes (ex. videoconferências). De modo geral, a abordagem para se construir uma arquitetura baseada em ALM envolve sondar as características da rede (latência, perdas de pacotes etc.) e construir topologias apropriadas, organizando os *hosts* em redes *overlay* de modo a obter uma distribuição eficiente dos dados. Esta topologia de distribuição, normalmente uma árvore, também é referenciada na literatura como *Application-Level*

Multicast Tree (ALMT) [Liu 2005] e, apesar das muitas variantes (fonte única, múltiplas fontes, limite de grau, mínimo atraso máximo etc.), quase sempre implicam em problemas complexos de natureza combinatória. A construção e manutenção destas árvores constituem uns dos principais objetos de estudo em comunicação em grupo *overlay*.

Os objetivos deste trabalho são (i) desenvolver uma formulação matemática usando programação linear inteira, e (ii) uma abordagem heurística, para o problema de construir e manter (através do redirecionamento de clientes) múltiplas árvores de distribuição de fluxos, com mínima latência, em um *backbone overlay* para um serviço de videoconferência. A topologia utilizada como guia é uma árvore-estrela, desenvolvida para o DCS (em detalhes na seção 3).

Apesar da ênfase em conferências H.323, as abordagens propostas neste trabalho podem ser adequadas a outras tecnologias de comunicação, desde que o conceito de MCU seja empregado e que se permita o redirecionamento de clientes participantes.

O artigo está organizado da seguinte forma: na seção 2 são mostrados alguns trabalhos correlatos encontrados na literatura. Na seção 3 são dados mais detalhes sobre a arquitetura e implementação do DCS; na seção 4 o problema posicionamento de MCU's e clientes é melhor elucidado; na seção 5 é apresentado o modelo matemático desenvolvido para o problema; a seção 6 trata da solução heurística; na seção 7 são mostrados os resultados computacionais obtidos; finalmente, na seção 8, são tecidos alguns últimos comentários sobre o trabalho realizado e sobre trabalhos futuros.

2. Trabalhos Relacionados

Diversos trabalhos tratam do problema de construção de árvores de *multicast* em nível de aplicação. Em [Yeo 2004] é apresentada uma análise das principais soluções de ALM propostas, com suas respectivas técnicas de manutenção da ALMT.

Na maioria dos trabalhos, os nós possuem um conhecimento local da rede, o que lhes permite melhorar a topologia apenas de uma forma gulosa, o que dificulta alcançar uma solução global de boa qualidade. Uma exceção é o ALMI [Pendarakis 2001], que utiliza uma entidade central, para construir uma árvore geradora de atraso mínimo entre os *hosts* participantes, e recalculá-la periodicamente.

Outra característica comum a grande parte dos trabalhos correlatos é o fato de existirem apenas nós clientes na rede. Uma exceção é o OMNI [Banerjee 2003], que apresenta dois tipos diferentes de nós: os clientes e os MSN's – estes seriam equivalentes às MCU's aqui tratadas. Os MSN's também se arranjam em árvore, buscando minimizar o atraso. A otimização da topologia também ocorre de forma dinâmica, havendo direcionamento de clientes para MSN's de menor atraso, assim como neste trabalho. Todavia, não existe uma entidade que conheça toda a rede, restringindo as possibilidades de otimização da mesma. Também é apresentada uma formulação usando programação inteira para o OMNI

No Kudos [Jain 2002], apesar haver apenas clientes, estes se agrupam em *clusters* e um destes assume um papel de cabeça do grupo. Este nó cabeça é responsável por algumas funções de controle, e por repassar aos seus "filhos" o fluxo de dados. Há

também movimentos como migração de um cliente para outro *cluster* que apresente uma menor latência. Esta abordagem é focada em aplicações de fonte única.

Em [Liu 2005] é tratado o problema de construção de uma árvore *multicast,* que possui a restrição de os nós terem um grau limitado, de modo análogo ao MCU, que possui uma capacidade limitada de suportar clientes. Entretanto, devido à natureza da aplicação ser diferente (fonte única), a busca-se minimizar atraso-máximo, ao invés do atraso total.

Em [Civanlar 2005] propõe-se uma arquitetura para implantação de videoconferência P2P, por meio da retransmissão dos dados pelos próprios clientes, formando uma cadeia ou corrente. Apesar da eficiência no consumo da banda passante, a latência é penalizada, visto que o último *host* de uma cadeia receberá o pacote com um atraso igual ao somatório de todos os atrasos entre os integrantes da cadeia.

Apesar de algumas semelhanças, não se identificou em trabalhos correlatos, nenhuma proposta de construir e manter mais de uma instância de árvore de distribuição para comunicação em grupo.

3. Arquitetura do DCS

A solução empregada no *Dynavideo Conference System* [Vasconcelos 2004] [Souza Filho 2006] utiliza uma versão modificada do projeto OpenMCU [H.323plus 2007] – implementação em software e aberta de uma MCU H.323.

O OpenMCU usa o modo de presença contínua para exibir até 4 participantes simultaneamente. O primeiro participante é posto no quadrante superior esquerdo da tela, a partir daí os seguintes são dispostos conforme ilustra a Figura 2. Caso na conferência haja mais que 4 participantes, os restantes serão ocultos, sendo comutado para exibição apenas quando a MCU detectar que um dos ocultos está falando.

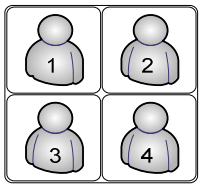

Figura 2. Disposição dos participantes na tela, pelo OpenMCU

A arquitetura desenvolvida para o DCS usa o cascateamento de até cinco MCU's simultaneamente. Uma delas, a MCU central, tem comunicação apenas com outras MCU's, nunca com os terminais. Neste modelo, cada MCU de periferia repassa aos terminais o vídeo gerado pela MCU central, que é uma combinação do vídeo das 4 MCU's periféricas. Assim, é possível a exibição da imagem de até 16 participantes simultaneamente. A Figura 3 ilustra a arquitetura do DCS e como a imagem é gerada.

Outra característica do DCS é a utilização de transferências de chamadas entre MCU's para redistribuir os participantes. Desta forma, torna-se possível mover os clientes, de modo que este seja atribuído a um servidor mais próximo de si, diminuindo a latência do sistema. A transferência de chamadas é uma característica prevista pelo padrão H.323 e, se for realizada entre as MCU's, é absolutamente transparente ao cliente.

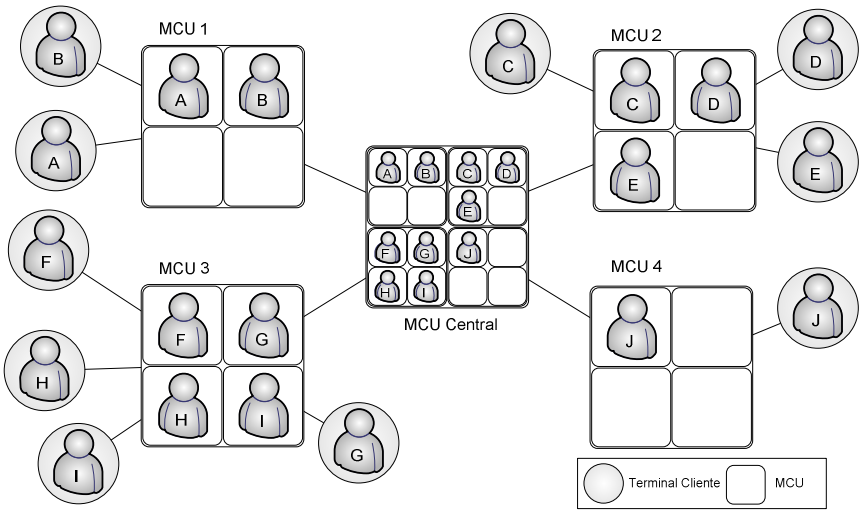

Figura 3. Arquitetura e geração da imagem no DCS

Além disso, é possível não apenas distribuir clientes entre as MCU's, como também alocar MCU's sob demanda. Desta forma, à medida que novos clientes entram ou saem da sala de videoconferência, MCU's diferentes podem ser alocadas ou liberadas. Assim, com a existência de uma entidade coordenadora, ciente do estado da rede, é possível reconfigurar o posicionamento dos nós de modo a diminuir a latência total do sistema.

4. Alocação e Posicionamento de MCU's e Clientes

Para o problema tratado neste trabalho, modelamos a rede de MCU's como um grafo $G(V, E)$, completo e não-direcionado, onde V é o conjunto de MCU's e E o conjunto de arestas, que as conectam. A cada aresta (i, j) é associado um custo, que representa o atraso entre as MCU's i e j. A forma como estes custos são obtidos foge do escopo deste trabalho. Apesar de o método mais comum entre as aplicações de ALM ser através de medições de *round-trip time* (RTT) [Yeo 2004], existem propostas de outras formas mais eficientes de se estimar o atraso entre dois nós na Internet [Ziviani 2005].

Além das MCU's, temos também um conjunto de clientes, capazes de se conectar a todos os MCU's. Para cada par cliente-MCU tem-se também uma aresta com um custo associado, representando o atraso entre eles.

Os clientes estão agrupados em salas, onde os integrantes são participantes que desejam se comunicar entre si. Cada sala é composta por no máximo 16 clientes e 5 MCU's (limitações da versão atual do DCS).

O que se deseja obter é uma árvore de MCU's, para cada sala, de modo que se possa atribuir cada cliente da sala a uma MCU de periferia. Contudo, as imposições topológicas devem ser respeitadas, bem como a capacidade máxima suportada por cada MCU (devido à capacidade de processamento, largura de banda, etc.). Não obstante, o custo da latência total do sistema deve ser mínimo.

O custo total (ou função objetivo) que se deseja minimizar, é obtido pelo somatório, para cada sala, do custo individual de cada aresta utilizada, seja entre as MCU's, ou entre MCU's e clientes.

Além de construir árvores de latência mínima para as salas, deseja-se também dar manutenção a estas árvores. Em um cenário real, clientes podem a qualquer momento, entrar ou sair das salas existentes, ou ainda novas salas podem ser criadas, de modo que a configuração atual possa ser melhorada. Desta forma, mesmo no percurso de uma conferência, a configuração pode ser modificada, de modo transparente aos participantes, no intuito de minimizar a latência.

Apesar de a latência mínima ser desejada, obter o seu menor valor é uma tarefa inviável. Isto ocorre porque a complexidade do problema torna-o impossível de ser resolvido em tempo polinomial. Para ilustrar a complexidade, considere o problema de encontrar um subgrafo de latência mínima, contendo 5 MCU's de um *backbone* de tamanho *n,* sendo uma deles o nó central. Note que é um subproblema do que é proposto neste trabalho, e ainda assim tem-se um problema intratável, para instâncias grandes, visto que seria necessário testar $5 \cdot C_5^n$ combinações.

5. Modelo matemático

Nesta seção apresentaremos uma formulação usando programação linear inteira para o problema. Com este modelo é possível encontrar a solução ótima, executando-o em ferramentas como o LINGO [Lingo 2001]. No entanto, a complexidade é combinatorial, como já fora discutida.

Seja $x_{ijk} = 1$ se a conexão entre as MCU's *i* e *j* está sendo utilizada na sala *k* e $x_{ijk} = 0$ caso contrário; $t_{ij} = 1$ se o cliente *i* está conectado à conferência através da MCU *j* e $t_{ij} = 0$ caso contrário; $y_{ik} = 1$ se a MCU *i* foi selecionada na sala *k,* $y_{ik} = 0$ caso contrário; $z_{ik} = 1$ se a MCU *i* for nó central na sala *k,* $z_{ik} = 0$ caso contrário.

Além disso, dados que *m* o número de MCU's, *s* o número de salas, *c* o número de clientes, λ o limite de clientes por MCU em cada sala, μ o número máximo de MCU's por sala, W_{ij} o custo da aresta que conecta as MCU's *i* e *j,* V_{ij} o custo da aresta que conecta o cliente *i* à MCU *j,* S_{ik} a matriz que associa clientes e salas ($S_{ik} = 1$ se cliente *i* pertencer à sala *k,* $S_{ik} = 0$ caso contrário) e C_i a capacidade máxima suportada pela MCU *i*. Tem-se:

Minimize $$\sum_{k=1}^{s}\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\left(x_{ijk}\cdot W_{ij}\right)+\sum_{i=1}^{c}\sum_{j=1}^{m}\left(t_{ij}\cdot V_{ij}\cdot S_{ik}\right)\right)$$ (função objetivo)

Sujeito a

(i) $x_{iik} = 0$ $\forall i = 1,2,...,m; \forall k = 1,2,...,s$

(ii) $x_{ijk} + x_{jik} \leq 1$ $\forall i = 1,2,...,m; \forall j = 1,2,...,m; \forall k = 1,2,...,s$

(iii) $\sum_{i=1}^{m} y_{ik} \leq \mu$ $\forall k = 1,2,...,s$

(iv) $\sum_{i=1}^{m} z_{ik} = 1$ $\forall k = 1,2,...,s$

(v) $y_{ik} - z_{ik} \geq 0$ $\forall i = 1,2,...,m; \forall k = 1,2,...,s$

$$\text{(vi)} \quad \sum_{j=1}^{m} x_{ijk} + \sum_{j=1}^{m} x_{jik} \leq m \cdot y_{ik} \qquad \forall i = 1,2,...,m; \forall k = 1,2,...,s$$

$$\text{(vii)} \quad \sum_{j=1}^{m} x_{ijk} + \sum_{j=1}^{m} x_{jik} \geq y_{ik} \qquad \forall i = 1,2,...,m; \forall k = 1,2,...,s$$

$$\text{(viii)} \quad \sum_{j=1}^{m} x_{jik} \leq (\mu - 1) \cdot z_{ik} \qquad \forall i = 1,2,...,m; \forall k = 1,2,...,s$$

$$\text{(ix)} \quad \sum_{j=1}^{m} t_{ij} = 1 \qquad \forall i = 1,2,...,c$$

$$\text{(x)} \quad \sum_{j=1}^{c} t_{ji} + \sum_{j=1}^{m} \sum_{k=1}^{s} x_{jik} \leq C_i \qquad \forall i = 1,2,...,m;$$

$$\text{(xi)} \quad \sum_{j=1}^{c} t_{ji} \cdot S_{jk} \leq \lambda \cdot y_{ik} \qquad \forall i = 1,2,...,m; \forall k = 1,2,...,s$$

$$\text{(xii)} \quad \sum_{j=1}^{c} t_{ji} \cdot S_{jk} \leq \lambda \cdot (1 - z_{ik}) \qquad \forall i = 1,2,...,m; \forall k = 1,2,...,s$$

As restrições (i) e (ii) apenas evitam que uma MCU ligue-se a si mesma e que uma mesma aresta seja selecionada duas vezes na mesma sala. A restrição (iii) limita o número de MCU's selecionadas para cada sala. Em (iv) e (v) é assegurado que toda sala terá uma e apenas uma MCU central e que esta esteja dentre as MCU's selecionadas para a sala. As restrições (vi) e (vii) limitam o grau máximo de cada MCU, ao mesmo tempo em que garantem que sejam utilizadas apenas arestas entre MCU's selecionadas para a sala. A restrição (viii) garante a topologia em estrela. Em (ix) é assegurado que todo cliente esteja conectado a uma e apenas uma MCU. Em (x) é definido que a capacidade máxima de uma MCU é diminuída tanto devido à atribuição de clientes quanto de outras MCU's (quando aquela for nó central de alguma sala). Finalmente (xi) e (xii) asseguram que os clientes estejam conectados apenas a MCU's selecionadas, com exceção do nó central.

6. Abordagem Heurística

Em virtude da funcionalidade de redirecionamento de clientes, implementada pelo DCS, e pela possibilidade de adicionar e remover MCU's a uma sala de conversação, tem-se então os subsídios necessários à implementação de um algoritmo coordenador, capaz de reconfigurar a rede dinamicamente, de modo a obter uma melhor topologia para as múltiplas salas, minimizando a latência global do sistema de videoconferência.

Todavia, devido à complexidade do problema, optou-se por fazer uso de uma abordagem aproximada, baseada na metaheurística *Simulated Annealing* [Kirkpatrick 1983], que será tratada em detalhes na seção 6.1.

O *Simulated Annealing* se enquadra na família de heurísticas de refinamento, ou técnicas de busca local, que percorrem o espaço de soluções, baseando-se na noção de vizinhança [Souza 2006]. O espaço de soluções *S* consiste no conjunto de soluções possíveis de um problema de otimização. A vizinhança de uma solução $s \in S$ consiste no conjunto $N(s) \in S$, onde a partir de modificações sobre *s*, uma solução pertencente a

N(s) é alcançada. Ao conjunto de modificações (ou operações) possíveis sobre as soluções, dá-se o nome de movimentos.

Neste problema, uma solução é representada por uma lista de salas, onde cada uma contém um conjunto de clientes e um conjunto de MCU's. O conjunto clientes de uma sala representa os participantes da videoconferência que desejam conversar entre si. As MCU's das salas estão arranjadas entre si na topologia estrela já citada anteriormente. Uma das MCU's é indicada como MCU central. Além disso, cada cliente da sala está associado a uma e apenas uma MCU, com exceção da MCU central, que se associa apenas a outras MCU's.

Note que cada cliente pertence a apenas uma sala, em contrapartida uma MCU pode estar atendendo várias salas simultaneamente.

Como solução inicial, foi utilizada uma abordagem gulosa, de modo que fossem selecionadas MCU's suficientes para atender a demanda de clientes de cada sala, sem uma grande preocupação com a função objetivo, focando em obter uma solução válida.

Na seção 6.1. é apresentado o *Simulated Annealing* e em 6.2. são enumerados os movimentos que definem estrutura de vizinhança utilizada.

6.1. *Simulated Annealing*

A metaheurística *Simulated Annealing* consiste em uma técnica de busca local probabilística, proposta originalmente por [Kirkpatrick 1983], que se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos, operação conhecida como recozimento.

Esta técnica começa sua busca a partir de uma solução inicial qualquer. O procedimento principal consiste em um laço que gera aleatoriamente, em cada iteração, um único vizinho *s'* da solução corrente *s*.

Considerando um problema de minimização, seja Δ a variação de valor da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s') - f(s)$. O método aceita o movimento e a solução vizinha passa a ser a nova solução corrente se $\Delta < 0$. Caso $\Delta \geq 0$ a solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde T é um parâmetro do método, chamado de temperatura e que regula a probabilidade de se aceitar soluções de pior custo.

A temperatura T assume, inicialmente, um valor elevado T_0. Após um número fixo de iterações, a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_k \leftarrow \alpha \times T_{k-1}$, sendo $0 < \alpha < 1$. Com esse procedimento, dá-se, no início uma chance maior para escapar de mínimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de piora.

O procedimento para quando a temperatura chega a um valor próximo de zero e nenhuma solução de piora da solução corrente é mais aceita, isto é, quando o sistema está estável. A solução obtida quando o sistema encontra-se nesta situação evidencia o encontro de um mínimo local.

A Figura 4 mostra o pseudocódigo do procedimento *Simulated Annealing*. Os parâmetros são: a função objetivo ($f(.)$); a função que determina a vizinhança de uma

solução ($N(.)$), baseada nos movimentos possíveis a partir dela; a razão de resfriamento (α); o número máximo de iterações para cada temperatura ($SAmax$); a temperatura inicial (T_0); e a solução inicial a ser refinada (s).

```
procedimento SA( f(), N(), α, SAmax, T0, s)
1  s*← s;            {melhor solução obtida até então}
2  IterT ← 0;        {núm. de iterações na temperatura T}
3  T ← T0;           {temperatura corrente}
4  enquanto (T > 0) faça
5          enquanto (IterT < SAmax) faça
6                  IterT ← IterT + 1;
7                  Gere um vizinho qualquer s' ∈ N(s);
8                  Δ = f(s') – f(s);
9                  se (Δ < 0)
10                         então
11                                 s ← s';
12                                 se ( f(s') < f(s*) ) então s* ← s';
13                         senão
14                                 Tome x ∈ [0,1];
15                                 se (x < e^(-Δ/T) ) então s ← s';
16                 fim-se;
17         fim-enquanto;
18         T ← α×T;
19         IterT ← 0;
20 fim-enquanto;
21 s ← s*;
22 retorne s;
Fim SA;
```

Figura 4. Procedimento *Simulated Annealing*

6.2. Movimentos e Vizinhança

Um movimento consiste em uma modificação sobre uma solução viável do espaço de soluções, resultando em outra solução válida. Os movimentos definem a estrutura de vizinhança, necessária para o emprego do *Simulated Annealing*. Foram definidos dois conjuntos de movimentos para o domínio do problema. Um conjunto de movimentos realizados sobre os terminais clientes (subseções 6.2.1 e 6.2.2) e outro com movimentos realizados sobre as MCU's (subseções 6.2.3, 6.2.4 e 6.2.5).

6.2.1. Mover participante

Este movimento consiste em mover um participante que esteja alocado a uma MCU para outra. Para tanto, a MCU destino precisa possuir disponibilidade para acomodar o participante. A Figura 5 (a) ilustra o movimento de mover participante.

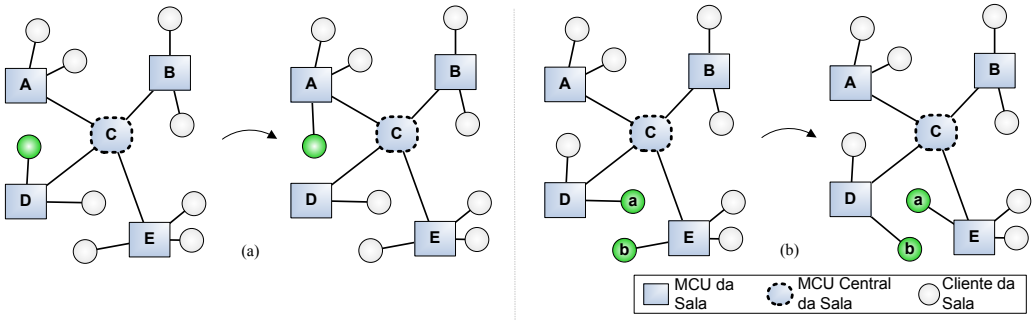

Figura 5. Movimentos sobre clientes. (a) Mover participantes. (b) Permutar participantes

6.2.2.Permutar participantes

Este movimento consiste em permutar dois participantes que estejam alocados em MCU's diferentes. É similar a dois movimentos de mover participante (6.2.1), com a diferença que não é necessário que as MCU's possuam disponibilidade sobrando. A Figura 5 (b) ilustra o movimento de permutar participantes.

6.2.3.Adicionar MCU

Este movimento consiste em adicionar à sala uma MCU que não esteja participando da mesma. Para que esta operação seja realizada, a MCU central deve possuir disponibilidade para acomodar a nova MCU (no caso da arquitetura do DCS, não deve haver mais que 3 MCU's além do nó central) e a nova MCU não pode estar com toda sua capacidade comprometida. A Figura 6 (a) ilustra o movimento de adicionar MCU.

6.2.4. Remover MCU

Este movimento consiste em remover da sala uma MCU que esteja participando. Para que esta operação seja realizada, a MCU não deve possuir clientes participantes alocados a si. A Figura 6 (b) ilustra o movimento de remover MCU.

6.2.5. Permutar MCU's

Para este movimento, temos 2 situações. A primeira consiste em permutar uma MCU (não-central) que esteja alocada à sala com outra que não esteja. Para que isso ocorra, a MCU de fora deve ter capacidade livre suficiente para comportar os clientes atualmente atribuídos a MCU que se deseja permutar. A Figura 6 (c) ilustra este movimento.

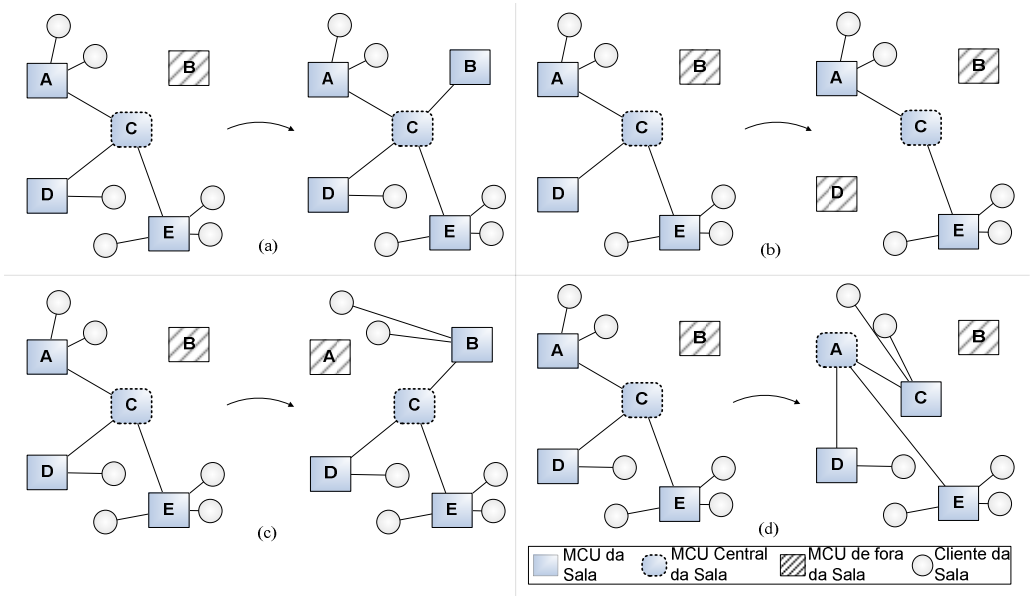

Figura 6. Movimentos sobre MCU's. (a) Adicionar MCU (b) Remover MCU (c) Permutar MCU's I (d) Permutar MCU's II (central com periferia)

A segunda situação consiste em permutar a MCU central da sala com uma de periferia. Neste caso, a demais MCU passarão a referenciar o novo nó central, e os clientes anteriormente atribuídos à MCU periférica passarão a se ligar à antiga MCU central. A Figura 6 (d) ilustra este movimento.

7. Resultados

Os resultados desta seção foram obtidos a partir da execução da implementação heurística e do modelo matemático no otimizador LINGO [Lingo 2001]. A máquina utilizada foi um Pentium D 2.8GHz, 512MB de memória, Windows XP para os testes com o LINGO e Linux *kernel* 2.6.17 para a heurística.

Para os experimentos foram geradas 10 instâncias, variando-se o número de MCU's, clientes e salas em cada uma. O método de geração das instâncias baseou-se no posicionamento aleatório de MCU's e clientes em um plano, e o custo das arestas obtido pela distância euclidiana entres os pontos. A Tabela 1 mostra as instâncias utilizadas.

Tabela 1. Instâncias utilizadas

Instância	i0	i1	i2	i3	i4	i5	i6	i7	i8	i9
#MCU's	5	5	10	10	15	15	30	30	40	40
#Clientes	16	16	36	36	50	50	150	150	200	200
#Salas	1	2	3	4	5	10	15	25	20	30

Para obter as soluções de referência, foram executadas as cinco primeiras instâncias (i0 – i4) no LINGO, até que a solução ótima fosse encontrada. As demais (i5 – i9) foram executadas por um período de três horas (10800s), e a melhor solução encontrada nessa janela de tempo foi então registrada para servir de referência.

Para a parametrização do *Simulated Annealing*, a temperatura inicial foi obtida pelo procedimento descrito em [Souza 2006], e fixada em 4000. O *SAmax,* que determina o número de iterações em cada nível de temperatura, teve seu valor fixado em 1000. Para a razão de resfriamento (α) foram adotados dois valores – 0,9 e 0,95. Cada instância foi executada 50 vezes, para ambos os valores de α. Os resultados relatados correspondem à média das funções objetivo resultantes.

A Tabela 2 mostra o quadro comparativo entre os resultados obtidos como referência e a heurística. A primeira coluna identifica as instâncias. A segunda e terceira colunas mostram, respectivamente, a função objetivo de referência e o tempo necessário para obtê-la. A quarta coluna mostra o valor obtido pela heurística para α_1 (0,9), a quinta coluna mostra tempo para obtê-lo e a sexta coluna mostra o seu *gap*, que indica percentualmente, quão longe está da solução de referência. Analogamente as três últimas colunas indicam o valor, o tempo gasto e o *gap* da heurística para α_2 (0,95).

Tabela 2. Resultados dos experimentos

Instância	**Sol Ref.**	**T Ref.**	**Sol α_1**	**T α_1**	***Gap* α_1**	**Sol α_2**	**T α_2**	***Gap* α_2**
i0	3342	1s	3342	1,46s	0%	3342	2,98s	0%
i1	3453	1s	3453	1,52s	0%	3453	3,11s	0%
i2	5550	13s	5631,04	2,56s	1,46%	5582,46	5,7s	0,58%
i3	6863	258s	7032,48	2,78s	2,47%	6959,70	6,06s	1,4%
i4	7605	29093s	7924,02	2,84s	4,19%	7837,98	10,89s	3,06%
i5	9097	10800s	9036,10	7,07s	-0,07%	8961,68	14,48s	-1,48%
i6	31290	10800s	25966,32	16,92s	-17%	25751,88	34,47s	-17,7%
i7	33625	10800s	29308,92	20,48s	-12,8%	28952,32	41,70s	-13,89%
i8	-	10800s	35718,54	22,88s	-	35366,38	46,59s	-
i9	-	10800s	35854,74	26,24s	-	35178,32	53,51s	-

Nos experimentos realizados, não foi possível encontrar soluções viáveis para as instâncias i8 e i9 no tempo estabelecido. Portanto, não se obteve valores de referência, tampouco *gap* para elas. Para as instâncias i0 e i1, a heurística alcançou a solução ótima

em todas as execuções, tanto para α_1 quanto para α_2. Nas instâncias i2, i3 e i4, há uma pequena perda de qualidade em relação à solução ótima, entretanto é evidente a disparidade no tempo resposta, sobretudo em i4, que requer mais de 8 horas para alcançar o ótimo. Para as instâncias i5, i6 e i7, onde não se tem a solução ótima como valor de referência, houve ganho sobre o valor de referência em todas elas.

Analisando os resultados obtidos para α_1 e α_2, é fácil perceber a influência que razão de resfriamento exerce sobre os testes. Quanto maior o α, mais lentamente ocorre o resfriamento, consequentemente maiores são os tempos. Por outro lado, a busca é mais refinada, levando a soluções de melhor qualidade. Deste modo, caso se queira pagar o preço de piorar o tempo de resposta, pode-se obter soluções de melhor qualidade.

Mesmo para o fator de resfriamento mais lento e para as maiores instâncias, o tempo de resposta obtido pela heurística é baixo e o *gap* é pequeno, não chegando a 5% em nossos testes, onde o ótimo era conhecido. Isso nos leva concluir que a estratégia desenvolvida é escalável, e viável para solucionar o problema proposto.

8. Conclusões e Trabalhos Futuros

Neste trabalho foi abordado o problema de minimizar a latência total em um serviço de videoconferência, através do desenvolvimento de um algoritmo capaz de coordenar a construção e manutenção dinâmica de múltiplas topologias, para distribuição dos fluxos audiovisuais. O algoritmo foi planejado para atuar como coordenador sobre a arquitetura desenvolvida pelo *Dynavideo Conference System*, aproveitando-se do cascateamento de MCU's e redirecionamento de clientes implementados pelo mesmo.

Apesar da particularidade do problema, foi feita uma análise na literatura em busca de trabalhos semelhantes, mas apesar do aprendizado proporcionado, não fora encontrado nenhum trabalho com as mesmas características apresentadas neste artigo.

Desenvolveu-se um algoritmo aproximativo, baseado na metaheurística *Simulated Annealing*, em virtude da intratabilidade do problema, como também uma modelagem matemática usando programação linear inteira, capaz de encontrar a solução ótima de instâncias menores, para fins de validação e comparação com a heurística.

Os resultados obtidos mostram que a heurística apresenta um tempo de resposta baixo, mesmo para as instâncias maiores, mostrando que a abordagem é uma opção viável, apesar da perda da qualidade característica dos algoritmos aproximativos. Também pôde se observar que é possível diminuir o *gap* em detrimento do tempo, pela parametrização da razão de resfriamento do *Simulated Annealing*.

Embora a heurística tenha se mostrado escalável, esta característica pode ser melhorada através da paralelização e distribuição da mesma em um *grid* de coordenadores mestre-escravo, onde o mestre "comunicaria" a solução a ser implantada na rede *overlay*. Similarmente, o problema de ponto único de falha pode ser suprimido através um mecanismo de *heartbeat* que detectaria falha no coordenador mestre e ativaria uma eleição de líder. Sugerem-se estudos neste sentido para trabalhos futuros.

Sugere-se ainda, estender a topologia para uma árvore generalizada, de modo que a árvore-estrela utilizada aqui seja um caso particular dessa generalização. Outro ponto a ser abordado em trabalhos futuros é uma comparação da implementação do *Simulated Annealing* com outras metaheurísticas.

9. Referências

Aposkitis, C. et al. (2005) "Videoconferencing Cookbook". Versão 4.1. http://www.vide.net/cookbook/cookbook.en/, Novembro, 2007.

Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S. (2003) "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications". In: Proceedings of INFOCOM, Abril, 2003.

Civanlar, M. R., Özkasap, Ö., Çelebi, T. (2005) "Peer-to-peer multipoint videoconferencing on the Internet". In: *Signal Processing: Image Communication* 20, 743–754, Elsevier.

H.323plus (2007). Open Source H.323. http://www.h323plus.org, Dezembro, 2007.

ITU-T (2003), ITU-T Recommendation H.323 V5.

Souza, M. J. F. (2006) Notas de Aula. Universidade Federal de Ouro Preto, http://www.decom.ufop.br/prof/marcone/, Setembro, 2006.

Jain, S., Mahajan, R., Wetherall, D., Borriello, G., Gribble, S.D. (2002) "A comparison of large-scale overlay management techniques", Technical Report UV-CSE 02-02-02, University of Washington, Fevereiro, 2002.

Kirkpatrick, S., Gellat, D.C., Vecchi, M.P. (1983) "Optimization by Simulated Annealing". In *Science* 220, p. 671-680.

Leopoldino, G. M., Moreira, E. S. (2001) "Modelos de Comunicação para Videoconferência", In: *RNP – NewsGeneration*, volume 5, número 3, Maio de 2001 http://www.rnp.br/newsgen/0105/video.html, Novembro, 2007.

Lingo, Extended rel7 (2001) LINDO Systems Inc., http://www.lindo.com.

Liu, F., Lu, X., Peng, Y. (2005) "An Efficient Heuristic Algorithm for Constructing Delay- and Degree-Bounded Application-Level Multicast Tree". In: Lecture Notes in Computer Science. ISBN: 978-3-540-30510-1. p. 1131-1142. SpringerLink.

Pendarakis, D., Shi, S., Verma, D., Waldvogel, M., (2001) "ALMI: an Application Level Multicast Infrastructure". In: Proceedings of the Third Usenix Symposium on Internet Technologies and Systems (USITS), Março, 2001.

Souza Filho, G. L. ; Vasconcelos, M. M. ; Anjos, T. C. ; Cabral, L. A. F (2006) "Aplicação da Metaheurística Simulated Annealing para Otimização do Posicionamento de MCU's em uma Rede de Videoconferência Multiponto Centralizada", In: *Anais do VI ERMAC R3*, João Pessoa, 2006.

Vasconcelos, M. A. V. M.; Souza Filho, G. L. (2004) "Dynavideo Conference System, Um Sistema de Videoconferência H.323". In: *Webmedia & LA-Web 2004 Joint Conference*, Ribeirão Preto, SP. Anais. ISBN 8576690101. p. 319- 320.

Yeo, C.K., Lee, B.S., Er, M.H. (2004) "A survey of application level multicast techniques". In: Computer Communications 27, p. 1547–1568. Ed. Elsevier.

Ziviani, A, Duarte, O. C. M. B. (2005) "Metrologia na Internet". In: Minicursos do XXIII Simpósio Brasileiro de Redes de Computadores, SBRC'2005, p. 285-329. Sociedade Brasileira de Computação (SBC).