Universidade Federal Fluminense

Tiago Silva Proença

Proposta e avaliação de uma arquitetura escalável para distribuição de TV na Internet

NITERÓI

2006

Universidade Federal Fluminense

Tiago Silva Proença

Proposta e avaliação de uma arquitetura escalável para distribuição de TV na Internet

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Processamento Paralelo e Distribuído.

Orientador:
Prof. Célio Vinicius Neves Albuquerque, Ph.D.

NITERÓI

2006

Proposta e avaliação de uma arquitetura escalável para distribuição de TV na Internet

Tiago Silva Proença

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Processamento Paralelo e Distribuído.

Aprovada por:

Prof. Célio Vinicius Neves Albuquerque, Ph.D. / IC-UFF
(Orientador)

Prof. Simone de Lima Martins, D.Sc. / IC-UFF

Prof. Luís Henrique Maciel Kosmalski Costa, Dr. / UFRJ

Niterói, Novembro de 2006.

Dedico esta dissertação aos meus pais,
pois nunca mediram esforços para que eu pudesse alcançar meus objetivos.
Nada disso seria possível sem o amor e a dedicação de vocês.

Agradecimentos

Agradeço a Deus, por me permitir mais uma vez aqui estar, vivenciando as diversas circunstâncias que nos permite evoluir nas vertentes de crescimento moral e intelectual.

De forma muito especial, agradeço aos meus pais, que sempre me apoiaram, incentivaram e, principalmente, acreditaram na minha capacidade. Serei eternamente grato pela educação que me deram e por não terem medido esforços para que eu pudesse alcançar os meus objetivos. Eu sei o quanto vocês trabalharam para que eu pudesse me qualificar melhor. Muito obrigado, papai e mamãe, por tudo que vocês fizeram por mim.

Agradeço com muito carinho à minha querida madrinha Erilce, que sempre me incentivou e apoiou em várias decisões importantes da minha vida. Com certeza, posso considerá-la minha segunda mãe, pois sempre me tratou com um filho. Sou muito grato a você por este carinho e consideração.

Agradeço também aos meus irmãos pelos momentos de conversas, desabafos e incentivos.

Agradeço de forma especial ao Prof. Célio Albuquerque pela orientação, paciência, persistência e por sempre acreditar que eu fosse capaz de realizar o trabalho necessário para defender o título de mestre. Forneceu equipamentos de ponta de linha para que eu pudesse realizar a minha pesquisa. Além disso possibilitou que eu fizesse parte do grupo de pesquisa do canal de retorno do projeto Sistema Brasileiro de Televisão Digital, o qual financiou em parte o meu mestrado. Além disso sempre me deu conselhos para superar minhas limitações e me acompanhou e incentivou durante todo o período do mestrado. Sem isso, eu não teria conseguido alcançar o objetivo. Obrigado por ter acreditado em meu potencial e por me dar uma ótima oportunidade para crescimento em minha carreira.

Aos professores do Instituto de Computação da Universidade Federal Fluminense que contribuíram para minha formação. Às secretárias Izabela, Ângela e Maria, pela atenção e apoio no que foi preciso.

Aos amigos Glauco e Renatinha por terem me acolhido no meu primeiro dia em Niterói.

Aos companheiros de república Cabelo, Marcelo, Sandro, Mokado, Leopoldo, Calado e Coca-cola pelos momentos de alegria e aprendizado vividos.

Aos colegas da UFF, Aline, Alexandre, Ary, Bruno Novelli, Cristiano, Cristiane, Daniela, Diego, Haroldo, Idalmis, Ivairton, Jacques, Jonivan, Kennedy, Luciana, Luciano, Luís, Rafael, Robson, Rodrigo, Sanderson, Stênio, Tiago, Viviane e Warley pelos agradáveis momentos de diversão, convivência e entretenimento. Em particular, aos amigos Janine, Etienne, Bruno e Nilmax pela paciência, atenção e cooperação nos trabalhos desenvolvidos.

Aos amigos Alexandro e Cristiane por terem me ajudado a revisar o texto.

Aos membros da banca pelas contribuições apresentadas.

Agradeço a todos os outros que porventura foram omitidos por um esquecimento momentâneo, mas nunca por falta de importância.

A todos vocês, meu MUITO OBRIGADO.

Sumário

Lista de Figuras

Lista de Siglas e Abreviaturas

ALMI	:	*Application Level Multicast Infrastructure*
BRITE	:	*Boston University Representative Internet Topology Generator*
BTP	:	*Banana Tree Protocol*
CAN	:	*Content Addressable Network*
CDF	:	*Cumulative Distribution Function*
CDN	:	*Content Distribution Network*
CONSER	:	*Collaborative Simulation for Education and Research*
CoopNet	:	*Cooperative Networking*
DARPA	:	*Defense Advanced Research Projects Agency*
DHT	:	*Distributed Hash Table*
DVMRP	:	*Distance Vector Multicast Routing Protocol*
HMTP	:	*Host Multicast Tree Protocol*
IP	:	*Internet Protocol*
ISI	:	*Information Sciences Institute*
LBNL	:	*Lawrence Berkeley National Laboratory*
MDC	:	*Multiple Description Coding*
MST	:	*Minimum Spanning Tree*
NAT	:	*Network Address Translation*
NS-2	:	*Network Simulator 2*
NSF	:	*National Science Foundation*
P2P	:	*Peer-to-Peer*
PARC	:	*Palo Alto Research Center*
PDF	:	*Probability Density Function*
PRA	:	Penalidade Relativa ao Atraso
PVR	:	*Personal Video Recorder*
RP	:	*Rendezvous Point*
RPF	:	*Reverse Path Forwarding*
RTO	:	*Retransmission Timeout*
RTT	:	*Round Trip Time*

SAMAN	:	*Simulation Augmented by Measurement and Analysis for Networks*
TCP	:	*Transmission Control Protocol*
TV	:	Televisão
UCB	:	*University of California at Berkeley*
UFF	:	*Universidade Federal Fluminense*
URN	:	Uso de Recursos Normalizado
USC	:	*University of Southern California*
VINT	:	*Virtual InterNetwork Testbed*
XML	:	*Entensible Markup Language*

Resumo

Esta dissertação aborda o problema de distribuição de fluxos de vídeo para um grande número de usuários de forma escalável. Especificamente é considerado o problema de atender um grande volume de requisições em um cenário de Internet TV, o qual é caracterizado por possuir somente um emissor transmitindo conteúdo para um grande número de receptores. Como solução, é proposta uma arquitetura baseada em redes P2P, chamada TVoIP (Televisão sobre IP), na qual os clientes formam uma rede sobreposta cuja função é receber e repassar o conteúdo de vídeo, gerando uma árvore de distribuição cujo propósito é realizar comunicação *multicast* em nível de aplicação. Resultados de simulação comparam o desempenho desta proposta com alguns trabalhos encontrados na literatura em relação a várias métricas de desempenho tais como saturação dos enlaces, penalidade relativa ao atraso, balanceamento de carga, tempo de entrada na rede e taxa de perda de pacotes.

Palavras-chave: *Multicast* na camada de aplicação, redes sobrepostas, distribuição de vídeo, redes *peer-to-peer*.

Abstract

This dissertation deals with the problem of video streaming distribution for a large number of users in a scalable manner. Specifically, it is considered the problem of handling a high volume of requests in an Internet TV scenario, which is characterized by only one node sending video content to a large number of receivers. As a solution, it is proposed a P2P network-based architecture, herein defined as TVoIP (Television over IP), in which clients make an overlay network whose function is to receive and forward the video content, generating a distribution tree whose purpose is to perform application-level multicast communication. Simulation results compare the performance of this proposal with the state of the art found in literature considering several performance metrics such as link stress, relative delay penalty, load balance, join time and packet loss ratio.

Keywords: Application layer multicast, overlay networks, video distribution, peer-to-peer networks.

Capítulo 1

Introdução

Nos últimos anos, a popularização de serviços de distribuição de vídeo na Internet tem se tornado cada vez maior, devido à alta disponibilidade de banda passante no núcleo da rede e ao rápido crescimento do número de usuários com acesso à Internet em banda larga. Dentre os serviços mais populares, estão os de vídeo sob demanda, os de transmissão de vídeo ao vivo e o emergente serviço de notificação e envio automático de vídeos conhecido como vídeo *podcast*. Em um serviço de vídeo sob demanda o conteúdo é distribuído de forma assíncrona de modo que o servidor realiza a transmissão de diferentes partes de um mesmo fluxo para diferentes requisições de clientes. Exemplos do uso deste tipo de serviço incluem sítios de trailers de filmes como Yahoo [58] e Apple Movie Trailers [5], sítios de vídeo clips como MTV [35] e Yahoo Music Videos [59] e sítios de compartilhamento de vídeos de usuários como youTube [60] e Google Video [22]. O serviço de transmissão ao vivo refere-se à distribuição sincronizada de conteúdo para todas as requisições de clientes. Exemplos incluem sítios de notícias como CNN [13], sítos de transmissão de eventos esportivos como Globo.com [20] e até mesmo informações sobre moedas como o sítio ForexTV [18]. O recente serviço de vídeo *podcast* agrega a arquitetura *download-and-play*[1] à tecnologia de notificação de novos conteúdos conhecida como *feeds* XML. Neste tipo de serviço, os clientes recebem todo o conteúdo de vídeo para posterior visualização, semelhante a idéia de um PVR (*Personal Video Recorder*), com o diferencial dos clientes serem notificados sobre novos vídeos pelos *feeds* XML. Vários sítios já incluem suporte ao *podcast*, principalmente os provedores de notícias e os *blogs* de usuários. Também estão surgindo comunidades específicas para compartilhamento de vídeos através desta tecnologia, como é o caso do projeto Democracy [15].

Os serviços acima descritos incluem-se em um grupo chamado de maneira genérica e abrangente de Internet TV. Aplicações de Internet TV normalmente são caracteriza-

[1]Nesta arquitetura, um cliente recebe todo o conteúdo para posterior execução.

das por possuírem somente um emissor transmitindo conteúdo para um grande número de receptores. Realizar a entrega de serviços Internet TV de forma escalável com uma qualidade de vídeo aceitável tem sido um grande desafio, devido à inerente natureza de alto requisito de banda na transmissão de fluxos de vídeo. A solução empregada por provedores de conteúdo de pequeno e médio porte faz uso de servidores que transmitem fluxos de vídeo utilizando conexões ponto a ponto com cada um dos clientes requisitantes. Este tipo de solução é obviamente não escalável, pois com o aumento de popularidade do arquivo de mídia, o servidor acaba se tornando o gargalo.

Muitas propostas têm sido exploradas para resolver o problema de escalabilidade de um serviço de transmissão de vídeo. Soluções utilizadas por provedores de conteúdo de grande porte incluem o uso de *caches* em servidores *proxy* e o uso de redes de distribuição de conteúdo (*Content Distribution Network* - CDN). Neste caso, o conteúdo é replicado do servidor principal para os servidores *proxy* ou servidores réplica da CDN localizados próximos aos clientes. Espalhando os servidores estrategicamente pela Internet, os clientes podem acessar o conteúdo do servidor que esteja menos congestionado e que possua o menor atraso. Embora este método consiga aliviar o problema de escalabilidade, ele não o resolve por completo. Existe a possibilidade de um servidor de determinada região ser momentaneamente sobrecarregado por um alto número de requisições de clientes que estejam em sua vizinhança, como acontece em situações de repentina demanda (*flash crowds* [37, 47]).

Deering [14] propôs o IP *multicast* para melhorar a eficiência na comunicação multi-destinatária de um-para-muitos e muitos-para-muitos na Internet. Nesta arquitetura, os dados só passam por um enlace uma única vez e são replicados nos roteadores para os clientes que desejam receber o conteúdo. O IP *multicast* é a forma mais eficiente para realizar comunicação em grupo, pois é capaz de reduzir a replicação de pacotes em uma rede para o mínimo necessário. Esta abordagem, embora eficiente, não é escalável pois requer a manutenção de estado por grupo nos roteadores e em alguns protocolos inunda periodicamente a rede para a construção das árvores de distribuição. Outros assuntos associados com o IP *multicast* como confiabilidade fim-a-fim, gerência do grupo, segurança, privacidade, controle de fluxo e congestionamento e modelo de cobrança oferecem desafios significantes para os quais ainda não surgiram soluções claras [38].

Uma das alternativas encontrada foi implementar a funcionalidade de comunicação multiponto na camada de aplicação. Uma das vantagens deste tipo de solução é que a complexidade que estava antes nos roteadores é movida para os nós finais da rede, de

forma que a solução fique totalmente independente da camada de rede.

Recentemente, as redes *peer-to-peer* (P2P) têm sido utilizadas para compartilhamento de arquivos, processamento distribuído, *multicast* na camada de aplicação entre outras aplicações [3, 52]. Nessas redes, os nós, também conhecidos como *peers*, formam um rede sobreposta (*overlay*) para rotear e distribuir as mensagens usando somente serviços de rede *unicast*. A comunicação multiponto é alcançada através da duplicação e retransmissão dos pacotes entre os membros do grupo. Embora aparentemente isto só eleve a funcionalidade de *multicast* à camada de aplicação, esta proposta atualmente revoluciona a forma como as aplicações de rede podem ser construídas. No IP *multicast*, exceto pelos nós de borda, a rede é composta por roteadores que não fazem nada mais que encaminhar pacotes. Em contrapartida, cada nó na rede sobreposta contribui com vários tipos de recursos computacionais como ciclos de processador, armazenamento, entre outros. Além disso, este tipo de solução oferece outros benefícios como o balanceamento de carga do sistema entre os participantes, o que diminui a carga colocada no servidor e contribui para o aumento de escalabilidade do sistema. É possível perceber que este tipo de solução oferece grandes vantagens em relação ao IP *multicast*, principalmente com relação à distribuição e implementação do serviço na Internet atual.

Na literatura são encontrados diversos trabalhos que abordam arquiteturas para realizar *multicast* na camada de aplicação [7, 10, 11, 12, 19, 24, 29, 32, 37, 38, 40, 53, 56, 62, 64]. Alguns dos desafios encontrados nesses trabalhos incluem escalabilidade da arquitetura, tempo de entrada no sistema, descontinuidades causadas por entradas e saídas de clientes e eficiência em relação ao número de pacotes duplicados nos enlaces. Uma das preocupações dos pesquisadores que está intimamente relacionada com estes desafios diz respeito a como a rede sobreposta será construída na arquitetura. Uma boa rede sobreposta deve refletir o máximo possível a rede física. Diversos trabalhos realizam esta tarefa partindo de uma rede inicial não otimizada, com nós muitas vezes ligados entre si de forma aleatória, realizando um refinamento incremental na rede para que o objetivo de se assemelhar à rede física seja alcançado.

Este trabalho propõe uma arquitetura escalável para transmissão ao vivo de fluxos de vídeo em um cenário de Internet TV, utilizando redes *peer-to-peer* para comunicação multiponto. O foco se manteve no processo de inclusão dos nós na árvore de distribuição de vídeo para construção da rede sobreposta. Destacam-se as seguintes contribuições:

- **Algoritmo de entrada de nós na rede:** A maior contribuição deste trabalho está na construção da rede sobreposta, pois introduz um novo algoritmo para construção

da rede inicial de forma eficiente, sem utilizar medidas ativas de rede e fornecendo um tempo de entrada constante aos clientes do sistema.

- **Comparação com o estado da arte:** Através de resultados de simulações será possível observar que a rede sobreposta inicial proposta neste trabalho fornece os menores valores de atraso médio e de distribuição de carga entre os usuários do grupo quando comparada a arquiteturas semelhantes [12, 37, 62].

- ***Framework* para simulação P2P:** Uma das dificuldades encontradas neste trabalho foi o fato do simulador de redes utilizado, o ns-2 (*network simulator 2*) [33], não possuir suporte nativo para comunicação P2P, um dos requisitos necessários para implementação do protocolo da nossa arquitetura. Sistemas P2P necessitam que um nó se conecte a vários outros nós da rede de maneira dinâmica. Por padrão, o ns-2 só permite conexões definidas *a priori* no roteiro de simulação. Diante desta limitação, implementamos um *framework* para simulação P2P no ns-2. Para acomodar uma grande variedade de estudos na área e modelar diversos protocolos P2P, projetamos nosso *framework* para ser genérico e fácil de estender e usar.

- **Simulador:** Para avaliar e analisar a viabilidade de nossa proposta, implementamos um aplicativo, chamado *netsim*, para realizar o cálculo dos custos da árvore de distribuição de nossa arquitetura e de árvores de outros trabalhos que implementamos para fins de comparação [12, 37, 62].

Os demais capítulos deste trabalho estão organizados da seguinte forma. O Capítulo 2 apresenta o estado da arte para arquiteturas que realizam *multicast* na camada de aplicação e traz uma classificação das propostas de acordo com o tipo de topologia usada. O Capítulo 3 propõe uma arquitetura para distribuição ao vivo de fluxos de vídeo utilizando redes P2P para comunicação multiponto. O Capítulo 4 expõe alguns aspectos de implementação da arquitetura no simulador de redes ns-2 e apresenta o simulador *netsim*. O Capítulo 5 conduz uma análise experimental comparando o algoritmo de construção da rede sobreposta inicial da arquitetura com alguns dos trabalhos mais importantes da literatura. O Capítulo 6 apresenta a conclusão deste trabalho e descreve alguns trabalhos futuros a esta dissertação.

Capítulo 2

Multicast na Camada de Aplicação

2.1 Introdução

A comunicação multidestinatária (*multicast*) é uma tecnologia de rede que permite que um nó envie dados de maneira eficiente para um grupo de outros nós. A implementação do *multicast* na camada de rede, conhecida como IP *multicast*, foi proposta por Deering [14] no início da década de 90. No IP *multicast*, os dados são enviados uma única vez pelo emissor e alcançam cada um dos receptores sem a existência de duplicação desnecessária de pacotes em um mesmo enlace da rede, sendo portanto a forma mais eficiente para realizar comunicação em grupo [6].

Os mecanimos de replicação e encaminhamento dos dados são todos implementados nos roteadores da rede. Os roteadores entre si definem uma árvore de distribuição por emissor e mantêm informação de estado por grupo. Uma vez que a implementação do IP *multicast* impõe dependência nos roteadores, sua implantação na Internet é restrita. Além disso, existe a dificuldade de implementação através de múltiplos sistemas autônomos (*Autonomous Systems* - AS), uma vez que os administradores de rede freqüentemente bloqueiam tráfego *multicast* nos roteadores de borda de seus AS, devido a questões de desempenho, comerciais e administrativas. Outros assuntos associados com o IP *multicast* como confiabilidade fim-a-fim, controle de fluxo e congestionamento e modelo de cobrança oferecem desafios significantes para os quais ainda não surgiram soluções claras [38].

O *multicast* na camada de aplicação procura resolver algumas das dificuldades encontradas na implantação do IP *multicast* na Internet. A vantagem deste tipo de solução é que a complexidade que estava antes nos roteadores é movida para os nós finais da rede, de forma que a solução seja totalmente independente da camada de rede. Embora aparentemente isto só eleve a funcionalidade de *multicast* à camada de aplicação, esta pro-

posta atualmente revoluciona a forma como as aplicações de rede podem ser construídas. O *multicast* na camada de aplicação possibilita a implantação de novas funcionalidades no modelo de comunicação multiponto, tais como gerência do grupo, segurança, privacidade, controle de fluxo e congestionamento, entre outras. É fácil perceber que este tipo de solução oferece grandes vantagens em relação ao IP *multicast*, principalmente porque um nó pode contribuir com vários outros recursos além da funcionalidade de roteamento. Além disso, o *multicast* na camada de aplicação utiliza os protocolos *unicast* já existentes na Internet, o que facilita sua implantação sem nenhuma modificação na estrutura atual de rede da Internet.

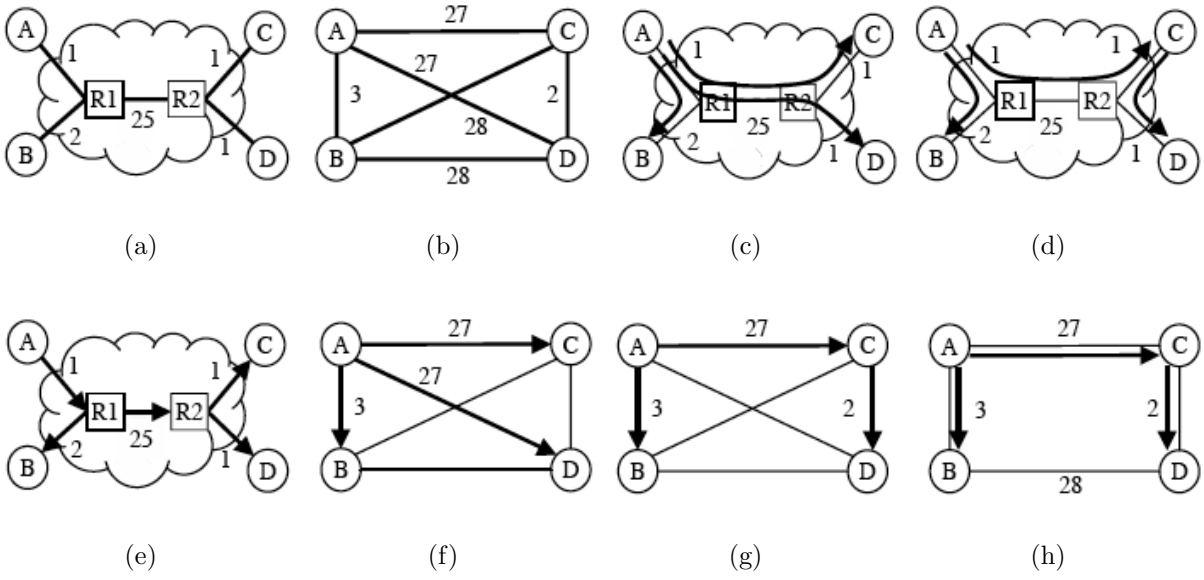


Figura 2.1: Exemplos ilustrando IP *multicast*, *multicast* na camada de aplicação e *unicast*. (Fonte: [12])

A idéia básica do *multicast* na camada de aplicação consiste em conectar os nós entre si formando uma rede com enlaces virtuais. Tal rede é chamada de rede sobreposta (*overlay*). Dessa forma, o objetivo de um protocolo de *multicast* na camada de aplicação é construir e manter uma rede sobreposta eficiente para a transmissão de dados. Uma vez que os protocolos de *multicast* na camada de aplicação podem enviar dados várias vezes sobre os mesmos enlaces físicos, eles são menos eficientes que o IP *multicast*. Para avaliar a qualidade e o desempenho dos protocolos de *multicast* na camada de aplicação, algumas métricas de desempenho foram propostas na literatura [6, 7, 12, 16]. Abaixo iremos descrever as mais comuns com alguns exemplos e delinear algumas preocupações fundamentais no projeto de um protocolo para *multicast* na camada de aplicação. Iremos seguir o trabalho de Chu *et. al.* [12] por apresentar uma descrição didática.

Considere a Figura 2.1(a) representando uma topologia física de rede. Os círculos A, B, C e D são nós finais enquanto que os quadrados $R1$ e $R2$ são roteadores. Os atrasos correspondentes a cada enlace estão indicados nas arestas. Observe que o enlace $R1 - R2$ possui o maior atraso de propagação, ilustrando um enlace típico para interconexão transcontinental, enquanto que os outros enlaces com um atraso mais baixo representam enlaces de redes locais. Assuma que o nó A deseja enviar dados para todos os outros nós.

A Figura 2.1(e) ilustra uma árvore *multicast* construída pelo protocolo DVMRP (*Distance Vector Multicast Routing Protocol*) [14]. O DVMRP é um dos protocolos clássicos de IP *multicast*, no qual os dados são entregues do emissor para os receptores usando uma árvore composta de caminhos curtos reversos (*Reverse Path Forwarding* - RPF). Observe que os roteadores $R1$ e $R2$ recebem uma única cópia do pacote de dados e o encaminham por várias interfaces. No máximo uma cópia do pacote é enviada para qualquer um dos enlaces físicos. Cada um dos receptores recebe os dados com um atraso igual ao que ocorreria se o nó A estivesse enviando diretamente os dados por uma transmissão *unicast* (considerando roteamento simétrico).

O *multicast* na camada de aplicação abstrai a topologia física de rede como um grafo completo, conforme pode ser observado na Figura 2.1(b). A partir deste grafo, o *multicast* na camada de aplicação constrói uma árvore geradora considerando como único emissor o nó A. Um exemplo particular desta árvore pode ser visto na Figura 2.1(f). Observe que este esquema pode ser encarado como transmissões *unicast*. A Figura 2.1(c) ilustra como esta transmissão *unicast* é mapeada na rede física. É fácil perceber que os enlaces $R1 - R2$ e $A - R1$ carregam respectivamente 2 e 3 cópias de cada pacote transmitido pelo nó A. Definimos o número de cópias idênticas que um enlace físico carrega de um pacote como **saturação do enlace**. As transmissões *unicast* possuem maior probabilidade de saturação nos enlaces próximos ao emissor.

O exemplo anterior da Figura 2.1(c) apresentou uma árvore geradora na qual o nó A transmite dados para os outros nós como uma simples transmissão *unicast*. No entanto, podemos construir árvores geradoras melhores. A Figura 2.1(g) ilustra um outro exemplo de árvore geradora, e a Figura 2.1(d) mostra como esta árvore é mapeada na rede física. É fácil perceber que a saturação máxima neste exemplo é 2. Esta árvore não reduz somente o pior caso de saturação nos enlaces, como também reduz a saturação no enlace mais caro $R1 - R2$ para 1.

Uma outra métrica de desempenho é o **uso de recursos de rede**. Esta métrica mede o uso de recursos da rede consumidos no processo de entrega dos dados para todos

os receptores. Pode ser definida como $\sum_{i=1}^{L} d_i * s_i$, onde L representa o número de enlaces ativos na transmissão de dados, d_i o atraso do enlace i e s_i a saturação do enlace i. Observe pela expressão que os enlaces com os maiores atrasos tendem a ser associados com os maiores custos. De acordo com os nossos exemplos, o uso de recursos pode ser calculado como 30 no caso de uma transmissão utilizando o DVMRP, 57 no caso de uma transmissão *unicast* e 32 para nossa nova árvore geradora, como é ilustrado respectivamente nas Figuras 2.1(e), 2.1(f) e 2.1(g).

Até agora, observamos que a nova árvore geradora (Figura 2.1(g)) melhora a saturação e o uso de recursos quando comparada à transmissão *unicast*. No entanto, ela aumenta o atraso do emissor para alguns dos receptores. Para ilustrar, considere o atraso do nó A para o nó D. Existe um aumento de atraso de 27 para 29. Definimos a razão do atraso entre dois membros sobre a rede sobreposta e o atraso *unicast* entre eles como **penalidade relativa ao atraso** (PRA). Assim, $< A, D >$ tem uma PRA de $\frac{29}{27}$, enquanto $< A, B >$ e $< A, C >$ têm uma PRA de 1.

O grau de saída de um nó na rede sobreposta é o número de filhos que este nó tem na árvore geradora. De acordo com esta consideração, o grau de saída do nó A na nova árvore geradora (2.1(g)) é 2, enquanto que na transmissão *unicast* é 3. Observe que o grau de saída tem um impacto direto na saturação dos enlaces próximos ao nó.

Cada nó na rede sobreposta realiza troca de mensagens com outros nós na árvore. Estas mensagens constituem as mensagens de controle do protocolo. Para um uso eficiente dos recursos da rede, o *overhead* destas mensagens deve ser o mínimo possível. O *overhead* das mensagens de controle é uma métrica muito importante sob o ponto de vista da escalabilidade, pois um alto *overhead* em um grupo com um grande número de usuários pode causar congestionamento em vários roteadores da rede e inviabilizar todo o sistema.

Existem vários trabalhos encontrados na literatura [7, 10, 11, 12, 19, 24, 29, 32, 37, 38, 40, 53, 56, 62, 64] propondo protocolos para *multicast* na camada de aplicação. Descrever todos eles em detalhes nesta dissertação é uma tarefa muito dispendiosa e nada simples. Em vez disso, optamos por classificar estes trabalhos em duas categorias diferentes e somente apresentar uma descrição detalhada dos protocolos mais citados e referenciados que representam cada uma destas categorias. Os demais serão descritos brevemente.

2.2 Propostas de *Multicast* na Camada de Aplicação

Na literatura, encontramos vários trabalhos para a construção eficiente de redes sobrepostas para realizar *multicast* na camada de aplicação. Normalmente estes trabalhos organizam os membros do grupo em dois tipos de topologias: uma topologia para controle da rede sobreposta e uma topologia para transmissão dos dados. A topologia de controle mantém informação sobres os membros do grupo, onde eles periodicamente trocam mensagens entre si para identificar e reparar particionamentos e também otimizar a rede sobreposta. A topologia de dados é um subconjunto da topologia de controle e identifica os caminhos no qual os dados irão percorrer na rede sobreposta. Dependendo da seqüência em que estas duas topologias são construídas, podemos classificar as diferentes propostas de *multicast* na camada de aplicação em duas categorias: *mesh-first* e *tree-first*.

Nas propostas pertencentes à categoria *mesh-first*, os membros do grupo *multicast* se organizam de maneira distribuída em uma rede sobreposta em malha. As redes em malha se diferenciam das redes em árvore por possuírem mais de um caminho entre qualquer par de nós. Dessa forma, espera-se neste tipo de rede a utilização de algum algoritmo de roteamento que forneça apenas um caminho entre um par de nós. Assim, as propostas da categoria *mesh-first* realizam a construção de suas árvores de distribuição em duas etapas. Na primeira etapa é construída a rede em malha, que é nada mais que um grafo com alta conectividade entre os membros. Na segunda etapa, são construídas árvores geradoras enraizadas nos emissores, com a inerente característica de possuírem os menores caminhos reversos para todos os outros membros. Essas árvores são criadas utilizando algoritmos RPF bem conhecidos, como ocorre por exemplo com o protocolo DVMRP [55].

Nas propostas pertencentes à categoria *tree-first*, os membros constroem diretamente uma árvore compartilhada pelo grupo para a distribuição dos dados. Os nós explicitamente selecionam seus nós pai a partir de um subconjunto de membros que eles conhecem. Pelo fato da árvore ser um grafo acíclico, se um nó falha ou abandona o grupo sem informar seus vizinhos, a árvore é particionada e os membros em uma partição não são capazes de se comunicar com membros localizados na outra partição. Assim, as propostas *tree-first* requerem mecanismos para detecção e recuperação de particionamentos. Uma rede em malha, de modo oposto, por possuir conexões redundantes entre os membros do grupo, fornece uma possibilidade menor de ocorrência de particionamentos. No entanto, como visto anteriormente, a existência de conexões redundantes requer que os membros executem um algoritmo de roteamento para contruir caminhos sem laços entre os membros, o que pode aumentar a complexidade do sistema e o *overhead* das mensagens do

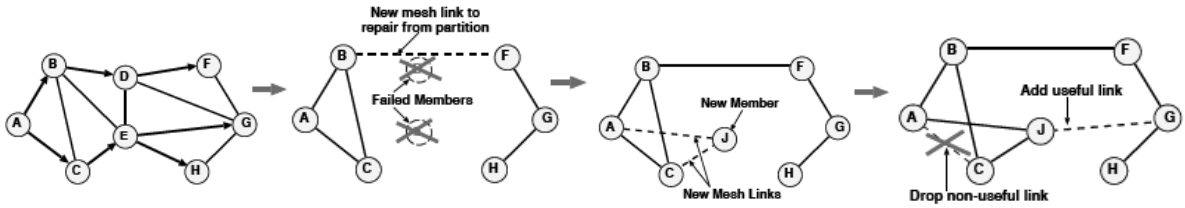
protocolo de controle.

Nas próximas seções iremos descrever alguns trabalhos de cada uma destas categorias.

2.2.1 *Mesh-First*

Nesta seção iremos descrever o protocolo Narada [12] como um exemplo dos protocolos que representam a categoria *mesh-first*.

2.2.1.1 Narada



(a) (b) (c) (d)

Figura 2.2: Topologia de controle e de dados no Narada. (Fonte: [6])

O protocolo Narada, proposto por Chu *et. al.*, foi o primeiro dos protocolos que demonstrou a viabilidade de implementar a funcionalidade de *multicast* na camada de aplicação. O Narada é um protocolo de rede distribuído que contrói suas árvores de distribuição em dois passos. No primeiro passo constrói uma rede em malha e no segundo passo constrói árvores geradoras baseadas nesta rede. As árvores geradoras são construídas usando um algoritmo de vetor de distância no qual são considerados os menores caminhos de cada um dos destinos para o emissor, de forma análoga ao DVMRP [55].

O Narada assume a existência de um nó especial para auxiliar o processo de entrada dos nós no grupo *multicast*. Este nó de inicialização, conhecido como *Rendezvous Point* (RP), fornece ao nó que deseja entrar no grupo uma lista com alguns membros que já fazem parte do grupo. De fato, todos os protocolos de *multicast* na camada de aplicação usam uma entidade equivalente ao RP para auxiliar no processo de entrada. Assim, iremos utilizar a sigla RP para denotar o nó de inicialização em todos os protocolos na camada de aplicação que iremos descrever neste capítulo.

O processo de construção da rede em malha no Narada pode ser descrito da seguinte maneira. Quando um novo nó deseja entrar no grupo, ele primeiro contacta o RP para

receber uma lista de membros já presentes na rede. O RP mantém informação sobre o estado de todos os membros que entraram no grupo *multicast*. O novo nó então seleciona aleatoriamente um subconjunto de membros da lista e envia várias requisições para que possa entrar na rede como vizinho de alguns destes membros. O processo de entrada é bem sucedido quando no mínimo um destes membros aceita o novo nó como seu vizinho.

No artigo, os autores não se preocuparam com o modo no qual um nó obtém a localização do RP. Eles argumentam que este procedimento é específico de cada aplicação e que o Narada é capaz de suportar diferentes modos de obter esta informação.

Após entrar na rede em malha, o novo nó inicia uma troca periódica de mensagens com seus vizinhos. Através destas mensagens, cada membro mantém uma lista sobre todos os outros membros pertencentes ao grupo. As trocas de mensagens são úteis pois possibilitam o conhecimento de mudança no estado da rede quando ocorrem novas entradas ou saídas de nós do grupo. A distribuição destas mensagens sobre cada um dos membros para todos os outros lida com um grande *overhead*, no caso $O(N^2)$, onde N representa o número de nós no grupo. Portanto, o protocolo Narada é somente efetivo quando o tamanho do grupo *multicast* é pequeno. Esta observação foi uma decisão explícita no projeto do Narada, no qual os autores optaram pelo grande *overhead* de controle em troca de uma melhora na robutez do processo de tratamento de falhas. Isto pode ser verificado de acordo com o seguinte exemplo.

Na Figura 2.2(b), quando os nós D e E falham simultaneamente, a rede em malha é particionada em duas partes. Como conseqüência, os nós A, B e C ficam impossibilitados de receber as mensagens de atualização de estado dos nós F, G e H, e vice-versa. Cada nó então sonda os respectivos nós dos quais pararam de receber as mensagens de atualização para tentar estabelecer novos enlaces e reparar a partição (Figura 2.2(b)). Caso um nó não responda, é assumido que este nó não está mais presente no sistema. Assim, esta informação é propagada pela rede para todos os nós do grupo. Note que a recuperação de particionamentos não requer a intervenção do RP e, portanto, funciona mesmo que o RP falhe.

Para criar os caminhos de entrega dos dados, os nós pertencentes ao grupo executam um protocolo de roteamento para calcular os caminhos do emissor para todos os outros nós da rede em malha. Estes caminhos, com qualquer um dos nós atuando como emissor, são calculados utilizando um algoritmo de vetor de distância no qual são considerados os menores caminhos de cada um dos destinos para o emissor, de forma análoga ao DVMRP empregado no IP *multicast*. Assim, é construída uma árvore de distribuição por emissor.

Um exemplo de caminhos de dados na rede em malha pode ser observado na Figura 2.2(a). Os caminhos dos dados são representados por arestas direcionadas partindo do nó A, que atua neste caso como emissor.

Como os caminhos de entrega dos dados no Narada são árvores geradoras criadas a partir da rede em malha, sua qualidade dos caminhos dos dados depende exclusivamente da qualidade dos enlaces que fazem parte dessa rede. Diante disto, o Narada realiza um processo de refinamento periódico na rede em malha para que exista um aumento de qualidade nos caminhos de entrega dos dados. No processo de refinamento, os nós sondam uns aos outros aleatoriamente e adicionam ou removem enlaces da rede em malha de acordo com uma função de utilidade. Por exemplo, um nó i verifica se um enlace deve ser adicionado ao nó j baseado nos seguintes critérios:

1. número de membros no qual o enlace até j reduz o atraso no roteamento a partir de i;
2. no quão significante é esta melhoria de atraso.

Este processo de refinamento pode ser ilustrado através da Figura 2.2(d). Adicionar o enlace $J-G$ é considerado útil porque um grande número de caminhos mais curtos podem ser criados utilizando esse novo enlace, como por exemplo entre os conjuntos de membros $\{A, C, J\}$ e $\{G, H\}$, e entre $\{C, J\}$ e $\{F\}$. O enlace $A - C$ é removido da rede em malha desde que exista um caminho com um atraso menor entre A e C. Assim, no Narada, as decisões de adicionar ou remover enlaces da rede em malha são feitas localmente pelos próprios nós pertencentes aos dois extremos do enlace.

2.2.2 *Tree-First*

Nesta seção iremos descrever a arquitetura CoopNet [37] e o protocolo HMTP [62] como exemplos de trabalhos que representam a categoria *tree-first*.

2.2.2.1 CoopNet

O sistema CoopNet (*Cooperative Networking*) propõe uma arquitetura híbrida para distribuição de vídeo, que mescla aspectos da arquitetura cliente-servidor com aspectos de rede P2P. Como na arquitetura cliente-servidor, existe uma fonte de vídeo que fornece os fluxos de vídeo para os clientes que desejam conectar-se à rede. No entanto, em momentos de sobrecarga, os clientes utilizam os recursos ociosos que possuem para auxiliar

a fonte de vídeo na distribuição dos fluxos, fornecendo dessa forma grande escalabilidade ao sistema.

Nos sistemas P2P de distribuição de vídeo, alguns clientes permanecem na rede por um pequeno período de tempo, cerca de alguns minutos, inferior ao que se observa em sistemas P2P de compartilhamento de arquivo. Isto acaba acarretando em rupturas na árvore de distribuição, o que ocasiona desconexão de outros clientes que estão recebendo os fluxos de vídeo. Embora os clientes desconectados possam continuar a utilizar o serviço entrando novamente na árvore de distribuição, este processo demanda tempo e pode resultar em interrupção da recepção de vídeo. Descontinuidades podem gerar um efeito cascata, onde os clientes não satisfeitos por terem tido o serviço interrompido podem deixar o grupo, ocasionando mais saídas de clientes, o que pode resultar em um serviço de transmissão de vídeo inaceitável. O sistema CoopNet minimiza este tipo de problema utilizando múltiplas árvores de distribuição e codificando o vídeo utilizando múltiplos descritores (*Multiple Description Coding* - MDC)[4, 37]. Neste método o sinal de áudio e vídeo é codificado em vários fluxos distintos, chamados de descritores, onde cada descritor pode ser individualmente decodificado em um sinal com ruído. Dessa forma, se cada descritor for transportado por uma árvore de distribuição diferente, tem-se que o fluxo não é totalmente interrompido quando ocorre uma desconexão. Ao invés disso, apenas um subconjunto dos descritores deixa temporariamente de ser entregue, ocasionando uma queda momentânea na qualidade de fluxo recebida, obtendo dessa forma um vídeo de menor qualidade.

O CoopNet é uma arquitetura que constrói uma árvore compartilhada por grupo na qual o controle da topologia é centralizado em um só nó. Este nó é o RP, que também é a raiz da árvore e fonte de vídeo. É esperado que neste modelo de distribuição o RP possua recursos suficientes para efetuar o processamento de entrada e saída dos nós na árvore de maneira eficiente. Em relação à largura de banda, o RP não é sobrecarregado visto que a carga de distribuição do conteúdo de vídeo é compartilhada por todos os nós. A centralização simplifica o protocolo mas introduz um único ponto de falha no sistema. Entretanto, no contexto da proposta, o ponto de centralização é o próprio RP que também é a origem do conteúdo. Se a origem do conteúdo falha, não faz a mínima diferença se o gerenciamento da árvore também falha. Além disso, vale salientar que o objetivo do CoopNet é complementar, não substituir, a arquitetura cliente-servidor.

O RP tem total conhecimento da topologia de todas as árvores de distribuição. Quando um novo nó i deseja entrar no sistema, ele primeiro contacta o RP. Este nó i

informa o nome do conteúdo desejado e sua respectiva largura de banda ao RP. Através da taxa de transmissão de vídeo e da largura de banda informada por este nó, o RP estima o número de clientes que o nó i poderá servir no futuro. Após, o RP envia uma resposta para o nó i com uma lista de nós pai, sendo um nó j por árvore de distribuição. Esta escolha é feita da seguinte maneira: a partir do servidor, o procedimento desce a árvore até encontrar um nível onde exista um conjunto de nós, que representaremos por V, com capacidade suficiente para servir o nó i. O RP então escolhe um destes nós $j \in V$ aleatoriamente para servir como pai do nó i. Observe que este procedimento *top-down* assegura uma árvore pequena e balanceada, o que tende a minimizar o atraso fim-a-fim entre o RP e os nós clientes. Isto também contribui para a diminuição da probabilidade de disrupção dos ramos da árvore no caso em que ocorra a saída de um nó interno. Após receber a mensagem do RP, o nó i envia mensagens para os nós pai designados em cada árvore de distribuição para entrar na árvore como nó filho.

O processo de saída de nós do sistema é dividido em duas classes: saída informada e falhas de nós. Em uma situação ideal, o nó p que deseja sair informa ao RP sua intenção de deixar o sistema. Neste caso, para cada árvore de distribuição, o RP identifica os filhos do nó p e executa uma nova operação de entrada destes nós no sistema, seguindo o procedimento *top-down* descrito anteriormente. O número de mensagens para o RP deve ser no máximo $\sum_i d_i$ envios e $\sum_i d_i$ recebimentos, onde d_i expressa o número de filhos do nó p na i-ésima árvore de distribuição. Cada nó filho q_i envia e recebe $M + 1$ mensagens, onde M representa o número de árvores de distribuição usadas. Para reduzir este volume de mensagens, o RP poderia determinar um novo pai k_i para cada um dos nós filhos q_i em cada árvore de distribuição e então designar um outro nó x para comunicar esta informação para cada nó filho q_i (preferencialmente o nó x deve ser p caso este ainda esteja disponível). Neste caso, o RP enviaria e receberia somente uma mensagem.

Uma falha de nó no sistema corresponde ao evento em que o nó que deseja partir não informa o RP sobre sua intenção de deixar o sistema. Para tratar este tipo de problema é utilizada uma técnica baseada na perda de pacotes. Nesta técnica, as falhas de nós são tratadas como um caso especial onde a perda de pacotes percebida pelos descendentes do nó com falha é de 100%. Cada nó i monitora sua respectiva taxa de perda de pacotes em cada árvore de distribuição. Quando a taxa de perda de pacotes ultrapassa determinado limiar t, o nó i contacta seu pai j para verificar se ele está sofrendo o mesmo problema. Caso esteja, a origem do problema é a banda de entrada do nó pai j e neste caso, o nó i deixa seu pai lidar com o problema. Caso o nó j não esteja sofrendo do mesmo problema ou o mesmo não responda, o nó i entra em contato com o RP para entrar novamente no

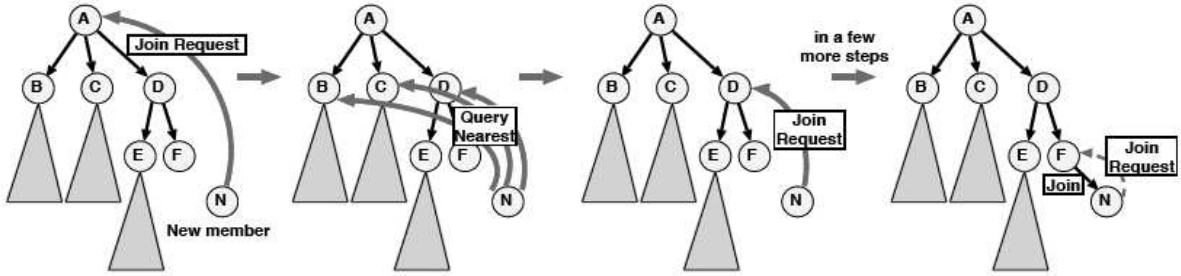


Figura 2.3: Processo de entrada no HMTP. (Fonte: [62])

sistema.

2.2.2.2 HMTP

O HMTP (*Host Multicast Tree Protocol*) é um protocolo distribuído de *multicast* na camada de aplicação que constrói uma árvore compartilhada por grupo para distribuição do conteúdo. Diferentemente do CoopNet, no HMTP o RP só tem conhecimento do nó que se localiza na raiz da árvore de distribuição. Logo, cada nó que deseja entrar no grupo é responsável por encontrar seu próprio pai na árvore compartilhada. Assim, cada nó executa um algoritmo recursivo de busca em profundidade para encontrar um nó pai potencialmente próximo de si.

O procedimento de construção da árvore de distribuição pode ser descrito da seguinte maneira. O nó i que deseja entrar no grupo contacta o RP para obter a raiz da árvore e assume esta raiz como um pai potencial. Em seguida calcula a distância até o pai potencial e para cada um dos filhos deste pai potencial. A métrica de distância utilizada é o tempo de ida e volta (*Round Trip Time* - RTT). Se o nó mais próximo é um dos filhos do pai potencial, o nó i continua a sua busca descendo a árvore usando este nó como novo pai potencial. Uma pilha é usada pra armazenar os pais potenciais encontrados durante o processo de busca. O processo continua até que o nó mais próximo encontrado seja o atual pai potencial do nó i. Quando isto acontece, o nó i envia uma requisição de entrada para este pai potencial para tornar-se filho. O pai potencial decide se aceita ou não a requisição baseado em seu grau de saída. Se o pai potencial rejeita a requisição, o nó i volta um nível através da pilha e reinicia o processo de busca. Este processo continua até que o nó i encontre um nó folha, ou um nó interno que esteja mais próximo do nó i do que seus filhos.

Na Figura 2.3, é ilustrado o procedimento de entrada para um novo nó N. Este nó

N calcula a distância até o nó A para cada um dos filhos do nó A (B, C, D). Após, o nó Nverifica que o nó D é o nó mais próximo. Então, o nó N prossegue para o próximo nível e encontra o nó F, o filho do nó D mais próximo. Como o nó F é folha, o nó N envia uma requisição de entrada para este nó. Como o nó F possui um grau de saída suficiente para servir o novo nó N, o nó F aceita a requisição. Assim, o nó N entra no grupo como filho do nó F.

Uma vez dentro do grupo, cada nó executa periodicamente o procedimento de entrada para que seja possível melhorar a qualidade da árvore de distribuição. No entanto, ao invés de iniciar o procedimento utilizando a raiz da árvore como pai potencial, cada nó escolhe aleatoriamente um nó que esteja em seu caminho até a raiz da árvore. Isto diminui o *overhead* do protocolo e possibilita que sejam explorados outros ramos da árvore de distribuição.

2.2.3 Outros Protocolos

Nas seções anteriores, descrevemos alguns protocolos de *multicast* na camada de aplicação e classificamos estes protocolos de acordo com seus respectivos processos de criação da árvore de distribuição. Nesta seção iremos descrever brevemente outros trabalhos, e apresentar sua classificação.

A arquitetura ALMI [38] utiliza uma abordagem centralizada para lidar com o problema de construção da árvore de distribuição. Um nó fixo, chamado de controlador, gerencia o processo de criação e manutenção da árvore. O controlador assegura conectividade entre os membros, recuperação de falhas e periodicamente recalcula e reconstrói a árvore geradora mínima (*Minimum Spanning Tree* - MST). Como este protocolo requer que cada membro estime a distância para um grande subconjunto de membros, ele não é adequado para aplicações de larga escala [48].

O Scattercast [11] é um outro exemplo de arquitetura que constrói uma rede em malha. O Scattercast requer o uso de servidores dedicados estrategicamente colocados na Internet, os quais em conjunto fornecem o serviço de *multicast* para os usuários. O Scattercast possui algumas similaridades com o protocolo Narada. Cada membro nesta arquitetura também mantém uma lista sobre todos os outros membros presentes no grupo e também é usada uma função para avaliar a utilidade de um enlace. O fato de cada membro manter uma lista com todos os outros membros limita esta proposta para aplicações com um grande número de usuários.

O Yoid [19] é uma arquitetura de propósito geral para redes sobrepostas. O Yoid possui três tipos de protocolos: um protocolo de identificação, um protocolo de transporte e um protocolo de construção da árvore. Estes protocolos têm como objetivo endereçar vários dos aspectos de transmissões P2P, tais como conectividade, controle de fluxo, confiabilidade e outros. O Yoid é particularmente interessante porque constrói primeiro a árvore de distribuição para depois construir uma rede em malha. Os enlaces adicionais da rede em malha são usados para a recuperação de particionamentos. Funcionalidade semelhante é encontrada no TMesh [56].

O Overcast [26] é uma arquitetura que constrói a árvore de distribuição de forma semelhante ao HMTP. No entanto, ao invés de utilizar o RTT como critério de seleção dos nós, um nó no Overcast escolhe como nó pai o membro que lhe fornece a máxima vazão. Outros exemplos de protocolos distribuídos baseados em árvore são o HostCast [29] o qual cria árvores com caminhos com baixo atraso, TBCP [32] e BTP [24] que definem técnicas genéricas para minimizar árvores de acordo com alguma métrica de desempenho, como por exemplo, o atraso.

As arquiteturas NICE [7] e ZIGZAG [53] usam agrupamentos (*clusters*) hierárquicos para construir as árvores de distribuição. Ambos protocolos utilizam um parâmetro que define o tamanho do agrupamento, com o intuito de limitar o grau de saída da rede. Assim não é considerada a capacidade real de cada um dos nós. Para uma rede sobreposta com n nós, o grau máximo de saída é $O(klogn)$ e $O(k^2)$ respectivamente para o NICE e ZIGZAG, onde k é o limite mínimo do tamanho do agrupamento. Não fica claro na literatura como construir redes sobrepostas com o NICE e o ZIGZAG para um ambiente com nós com diferentes graus de saída, mantendo ainda as propriedades e integridades originais destas arquiteturas.

As técnicas DHT (*Distributed Hash Table*) são normalmente utilizadas para roteamento e localização de conteúdo em redes de compartilhamento de arquivos P2P. No entanto, alguns protocolos constroem árvores de distribuição a partir de redes sobrepostas criadas através destas técnicas. Por exemplo, o Scribe [10] e o CAN-multicast [41] são baseados respectivamente no Pastry [44] e no CAN [40]. O Bayeux [64] é construído sobre o Tapestry [63]. Embora o funcionamento do Tapestry seja similar ao do Pastry, o Bayeux e o Scribe se diferenciam na forma em que os caminhos de dados são criados. As redes sobrepostas baseadas em DHT empregam esquemas de endereçamento que fornecem escalabilidade e robustez na distribuição dos dados sem a necessidade de um protocolo de roteamento adicional.

A Figura 2.4 ilustra a classificação de todos as arquiteturas de *multicast* a nível de aplicação descritas neste capítulo.

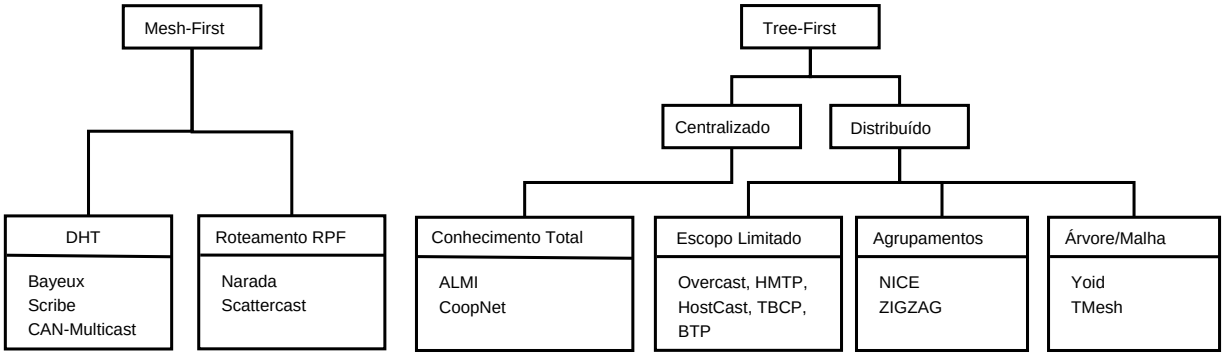


Figura 2.4: Classificação das arquiteturas de *multicast* a nível de aplicação.

Capítulo 3

TVoIP: Televisão sobre IP

Neste capítulo será apresentada em detalhes uma arquitetura de transmissão de vídeo ao vivo na Internet utilizando redes *peer-to-peer* para comunicação multiponto, denominada TVoIP. O objetivo é fornecer um sistema escalável que seja aplicável à transmissão de conteúdo televisivo na Internet. Em particular, este trabalho de dissertação visa fornecer uma arquitetura distribuída baseada em uma árvore de distribuição, descrevendo mecanismos de entrada e saída de nós do sistema em face do ambiente altamente dinâmico, considerando mecanismos para otimização da rede sobreposta e alguns aspectos sobre implantação da arquitetura na Internet atual.

3.1 Componentes da Arquitetura

Aplicações de Internet TV normalmente são caracterizadas por possuírem somente um emissor transmitindo conteúdo para um grande número de receptores. Partindo deste princípio, consideramos três componentes para a arquitetura do nosso sistema: um conjunto de clientes, um nó de inicialização e uma fonte de vídeo. Sob a perspectiva da arquitetura, os clientes são um conjunto de nós heterogêneos que formam a rede sobreposta P2P, cuja função é receber e repassar o conteúdo de vídeo para outros nós. Em relação à heterogeneidade, consideramos os nós heterogêneos em relação à largura de banda no enlace de acesso de suas redes locais ao *backbone*. O nó de inicialização, também conhecido na literatura como *bootstrap node* ou *rendezvous point*, gerencia o processo de entrada e saída dos nós do sistema através de uma árvore de distribuição. A fonte de vídeo é a raiz da árvore de distribuição e fornece o fluxo de vídeo ao vivo aos clientes do sistema.

As próximas sub-seções irão descrever com maiores detalhes esses três componentes da arquitetura.

3.1.1 O Cliente

Nesta arquitetura, o cliente além de receber os fluxos de vídeo tem como função retransmitir estes fluxos para outros clientes presentes na árvore de distribuição. Esta característica fornece ao sistema o benefício de escalabilidade, uma vez que a fonte de vídeo não é sobrecarregada por um grupo com um grande número de usuários. A localização exata de onde o cliente consegue receber os fluxos de vídeo é enviada pelo nó de inicialização quando o cliente entra no sistema. A interação entre cliente, nó de inicialização e fonte de vídeo durante o processo de entrada na árvore de distribuição pode ser descrita da seguinte maneira:

1. Quando um cliente deseja receber o conteúdo de vídeo, ele envia uma mensagem ao nó de inicialização para se juntar à árvore de distribuição. Além do nome do conteúdo desejado, o cliente informa a sua capacidade de saída para servir futuras requisições (um valor que representa o número de clientes que o nó pode servir baseado em largura de banda disponível).

2. O cliente recebe uma mensagem do nó de inicialização informando a localização na qual pode receber o fluxo de vídeo. A localização pode ser o endereço da fonte de vídeo ao vivo ou de um outro cliente presente no sistema. Assim, o cliente entra em contato com o nó que lhe foi informado para que este comece a lhe enviar o fluxo de vídeo.

Uma das dificuldades encontradas é determinar a capacidade de saída do cliente, uma vez que é função da largura de banda disponível no nó. Então, em vez de medir a largura de banda disponível, tarefa que normalmente envolve um grande *overhead* e complexidade [25, 28], cada cliente confia em um valor especificado pelo usuário que representa o número de clientes que ele pode e está disposto a servir. Alguns mecanismos de incentivo para que nós suportem outros clientes serão descritos na conclusão.

No caso específico das mensagens de repasse de vídeo, o cliente só repassa o fluxo de vídeo para um número de clientes que seja menor ou igual a sua capacidade, ou seja, sem ultrapassar o limite definido no momento de conexão. Espera-se dessa forma que o cliente forneça um serviço de boa qualidade para os outros nós sem esgotar seus recursos.

3.1.2 A Fonte de Vídeo

A fonte de vídeo nesta arquitetura tem como função realizar a distribuição sincronizada dos fluxos de vídeo ao vivo para os clientes do sistema. Em relação à largura de banda, a fonte de vídeo não é sobrecarregada uma vez que a carga de distribuição do conteúdo de vídeo é compartilhada por todos os nós. Como a fonte de vídeo se localiza na raiz da árvore, os únicos clientes que recebem diretamente o fluxo de vídeo são os que se encontram no primeiro nível da árvore de distribuição. Os demais clientes recebem o fluxo de vídeo através da retransmissão dos pacotes pelos membros que se encontram um nível acima na árvore, ou seja, pelos seus respectivos nós pai. Formalizando, podemos dizer que o cliente c_i só recebe fluxos de clientes c_{i-1}, onde $1 \leqslant i \leqslant h$ considerando i como o nível da árvore, c_0 como a fonte de vídeo e h como a altura da árvore.

3.1.3 O Nó de Inicialização

Nesta arquitetura assumimos a presença de um nó de inicialização bem conhecido, cujo propósito é auxiliar um novo cliente no processo de entrada no grupo *multicast*. O endereço do nó de inicialização é parte da informação do grupo e é obtido *off-line* pela aplicação. O nó de inicialização funciona como uma espécie de servidor de diretório, no qual reside uma lista com todos os membros pertencentes ao grupo. Nesta lista, são considerados endereços dos clientes, capacidade de banda disponível, número de filhos e informação a respeito da hierarquia do nó na árvore de distribuição.

O nó de inicialização fornece para o novo cliente a localização onde ele poderá obter o fluxo de vídeo. Isto é possível porque o nó de inicialização é quem gerencia a árvore de distribuição *multicast*. O nó de inicialização não participa no encaminhamento dos dados, o que faz com que sua localização não tenha impacto significativo no desempenho da disseminação dos fluxos de vídeo. É possível que um nó de inicialização sirva vários grupos *multicast*, o que é desejável em sistema de distribuição de TV, uma vez que cada grupo corresponde à transmissão de um programa televisivo diferente. O nó de inicialização pode ser uma simples estação de trabalho, um servidor dedicado, um *cluster* de servidores ou ainda estar localizado junto à fonte de vídeo.

Considerando o nó de inicialização, podemos observar que a centralização simplifica o protocolo mas introduz um único ponto de falha ao sistema. Entretanto, no contexto da proposta, a falha do nó de inicialização irá impossibilitar novos clientes de entrarem no grupo, mas não afetará a disseminação dos fluxos de vídeo entre os clientes que já

participam do grupo. Além disso, funcionalidades como políticas de acesso, criptografia, segurança e várias outras podem ser implementadas no nó de inicialização.

3.2 Gerenciamento da Árvore

Nesta seção iremos discutir o problema de gerenciamento da árvore de distribuição em face do ambiente altamente dinâmico, onde os nós estão freqüentemente saindo e entrando no sistema. Existem três objetivos para o algoritmo de gerenciamento da árvore, que podem ser enumerados a seguir:

1. **Árvore baixa**: A estrutura da árvore deve possuir a menor altura possível para minimizar o atraso fim-a-fim entre a fonte de vídeo e os nós clientes. Isto também contribui para a diminuição da probabilidade de disrupção dos ramos da árvore no caso em que ocorra a saída de um nó interno. Com a árvore sendo baixa, espera-se que seja larga e balanceada da melhor maneira possível para que se possa servir um grande número de nós. Também é desejável que a capacidade de banda dos nós internos da árvore seja a mais abundante possível.

2. **Entrada e saída rápida**: O processo de entrada e saída dos nós no sistema deve ser o mais rápido possível. Logo, os nós que estão na árvore devem receber o conteúdo de vídeo o mais rápido possível (no caso de uma entrada) e com o mínimo de interrupção (no caso de uma saída).

3. **Escalabilidade**: O gerenciamento da árvore deve fornecer ao sistema a possibilidade de escalar a um grande número de nós, mesmo em um ambiente altamente dinâmico. Arquiteturas P2P tendem a ser escaláveis uma vez que a entrada de um nó na rede sobreposta corresponde à adição de novos recursos ao sistema, tornando possível a entrega dos fluxos de vídeo para um número maior de nós. Além disso, o gerenciamento da árvore deve assegurar um baixo *overhead* das mensagens de controle para que não ocorra congestionamento nas filas dos roteadores da rede.

Tendo em vista estes objetivos, podemos iniciar a descrição do processo de gerenciamento da árvore.

3.2.1 Entrada de Clientes

Com o objetivo de reduzir a saturação nos enlaces e o uso de recursos da rede, o TVoIP agrupa na rede sobreposta nós próximos na rede física. A maneira mais simples de realizar esta tarefa é deixar que um novo cliente escolha para nó pai um outro participante que esteja próximo de si. Esta estratégia envolve o conhecimento prévio de um subconjunto de membros do grupo e a definição de uma métrica de distância. O subconjunto de membros do grupo pode ser obtido através do nó de inicialização e a métrica de distância normalmente utilizada é o RTT (*Round Trip Time*). Assim, o cliente sonda cada um dos nós do subconjunto de membros e define como nó pai o nó com menor RTT. Esta estratégia, embora funcional, não fornece nenhuma garantia em relação à altura da árvore de distribuição e apresenta um tempo de entrada que cresce com o tamanho do subconjunto de membros ($O(n)$). Essas observações vão contra dois de nossos objetivos para o gerenciamento da árvore: construir uma árvore baixa e larga e fornecer um procedimento de entrada e saída rápida para os clientes do sistema.

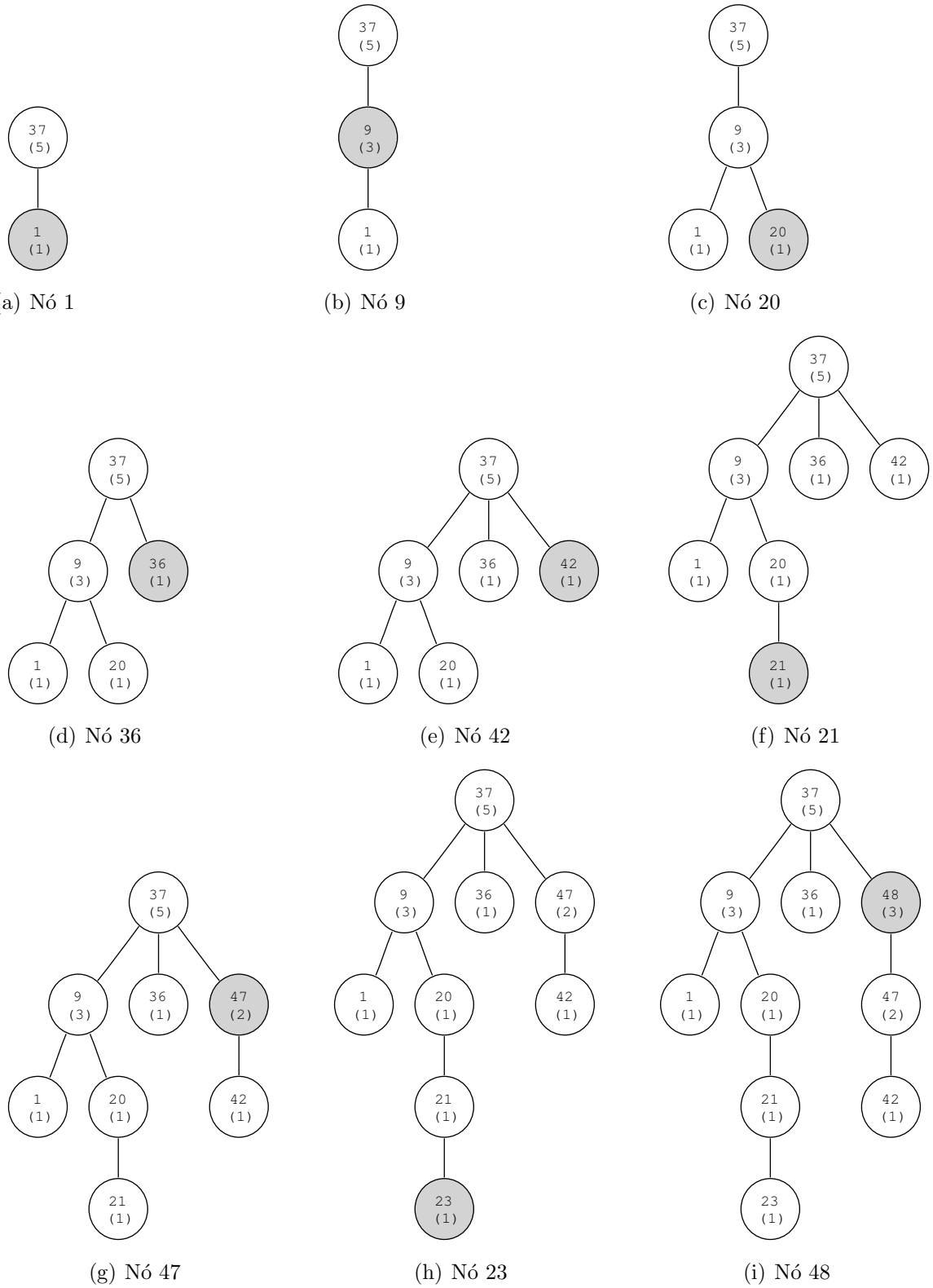
Neste trabalho, procuramos resolver o problema de uma maneira diferente. Para atingir o objetivo de manter uma árvore baixa e larga, consideramos manter os nós com maior grau de saída logicamente conectados próximos à raiz da árvore (fonte de vídeo). Com o intuito de fornecer um curto tempo de entrada aos clientes e não utilizar medidas ativas de rede para não aumentar o *overhead* das mensagens de controle, utilizamos como métrica de distância o valor absoluto da diferença entre o endereço de rede dos nós.

Para o realizar o cálculo de proximidade, considere D_{A-B} como a distância entre os nós A e B. Assuma E_A como o endereço de rede do nó A e E_B como o endereço de rede do nó B, onde E representa um endereço de rede de 4 *bytes* (IPv4). Assim, o cálculo pode ser realizado através da seguinte expressão:

$$D_{A-B} = \parallel E_A - E_B \parallel \tag{3.1}$$

Esta métrica, embora não realística, apresenta bons resultados (Seção 5.3), pois usualmente nós que estão na mesma rede se localizam geograficamente próximos.

Para descrever o procedimento de entrada dos nós na árvore de distribuição, iremos usar como referência a Figura 3.1. Cada imagem representa o estado da rede sobreposta após a entrada de cada um dos nós marcados em cinza. O endereço de cada nó é ilustrado na parte superior do mesmo e sua capacidade de saída na parte inferior entre parênteses. Por questões de simplicidade, iremos adotar um endereçamento não hierárquico, no qual



(a) Nó 1 (b) Nó 9 (c) Nó 20

(d) Nó 36 (e) Nó 42 (f) Nó 21

(g) Nó 47 (h) Nó 23 (i) Nó 48

Figura 3.1: Estado da rede sobreposta após a entrada de cada um dos nós marcados em cinza.

o endereço de rede é representado por um número natural. Assim, na Figura 3.1(b), o nó possui endereço 9 e capacidade 3.

A seguir iremos ilustrar a entrada dos seguintes nós na árvore de distribuição: 21, 20, 47 e 23. O procedimento também pode ser acompanhado através do Algoritmo 1.

Na Figura 3.1(f), o nó 21 é um novo cliente que deseja juntar-se ao grupo. Ele contacta o nó de inicialização para receber a localização do nó que irá ser seu pai na árvore de distribuição. O nó de inicialização tem total conhecimento da topologia da rede sobreposta. Assim, após receber a requisição do nó 21 e sua capacidade de saída, o nó de inicialização inicia o procedimento de entrada do nó 21 na árvore. O nó de inicialização assume a raiz 37 como um pai potencial. Em seguida calcula a distância entre o nó 21 e o nó 37 e entre o nó 21 e todos os filhos do nó 37 (nó 9, 36 e 42). Após, considera o nó mais próximo como sendo o novo pai potencial do nó 21, neste caso o nó 9. O processo é repetido e encontra-se o novo pai potencial, o nó 20. Neste caso, como o procedimento chegou em um nó que é folha da árvore, o nó de inicialização envia a localização do nó 20 para o nó 21 para que este possa entrar em contato com o nó 20 para receber o fluxo de vídeo.

Um outro exemplo considerado é ilustrado na Figura 3.1(c). Observe que o nó 20 possui como pai um nó interno da árvore. Após calcular a distância entre o nó 20 e o nó 9 e entre o nó 20 e todos os filhos do nó 9, o nó de inicialização verifica que o nó 9 está mais próximo do nó 20. De acordo com as informações sobre o nó 9 contidas em sua lista, o nó de inicialização sabe se o nó 9 tem capacidade suficiente para servir o novo nó 20. Neste caso, como o nó 9 ainda pode servir mais 2 nós, o nó de inicialização envia a localização do nó 9 ao nó 20. Se o nó 9 não pudesse servir o nó 20, o procedimento continua descendo a árvore utilizando como novo pai potencial o filho do nó 9 mais próximo do nó 20, de modo análogo ao descrito anteriormente. Procedimento semelhante ocorre na entrada do nó 42, conforme ilustrado na Figura 3.1(e).

Uma das características do algoritmo do procedimento de entrada é manter os nós com maior grau de saída próximos à raiz da árvore. A Figura 3.1(g) ilustra um exemplo em que podemos observar este comportamento. Observe que o nó 42 pode servir apenas 1 nó. Assumindo que o nó 47 pode servir 2 nós, quando o procedimento chegar ao nó 42, será verificado que o nó 42 possui um grau de saída menor que o do nó 47. Então o procedimento troca o nó 47 de lugar com o nó 42 e insere o nó 42 novamente na árvore a partir de seu antigo nó pai, que neste caso é o nó 37. Ao final do procedimento, observamos que o nó 42 é inserido como um filho do nó 47.

Todas as vezes que o procedimento move um nó de lugar dentro da topologia da árvore, o nó de inicialização adiciona este nó em uma lista de movidos. Após o término do procedimento, além do nó 47 receber uma mensagem contendo a localização de seu nó pai (nó 37), o nó 42 recebe uma mensagem de mudança de pai do nó 37 para o nó 47. Para que o nó 42 não experimente perda de pacotes durante a troca de pai, somente após receber pacotes de vídeo do nó 47 é que o nó 42 envia uma mensagem ao nó 37 notificando que mudou de pai. Assim o nó 37 atualiza sua lista de filhos, o que suspende o repasse dos fluxos de vídeo para o nó 42.

Se observarmos atentamente as figuras, podemos notar que na Figura 3.1(h) o nó 23 não está no lugar correto. O procedimento deveria ter caminhado em direção ao nó 36 e não em direção ao nó 9, pois de acordo com o cálculo da nossa métrica de distância (Equação 3.1), $D_{23-36} = 13 < D_{23-9} = 14$. Assim, esperávamos encontrar como pai do nó 23 o nó 36 e não o nó 21 como está ilustrado na figura. No entanto, observe que o nó 21 fornece um pai mais próximo do nó 23 que o nó 36. Baseado nesta observação incluímos uma modificação no algoritmo do procedimento de entrada para que este comportamento fosse levado em conta. É por isto que o nó 23 aparece como filho do nó 21 na Figura 3.1(h). Assim, se no procedimento de entrada o novo nó A estiver mais próximo de um dos filhos do pai potencial (nó B), o algoritmo também irá considerar para o cálculo um outro filho próximo (nó C). O algoritmo verifica se o valor absoluto da diferença entre as distâncias destes dois filhos está dentro de determinado limiar, isto é, $\parallel D_{A-B} - D_{A-C} \parallel < L$. Em caso positivo, o procedimento desce a árvore de distribuição utilizando as duas sub-árvores destes dois filhos e ao final o procedimento verifica qual dos caminhos fornece o melhor pai para o novo nó.

Com relação ao valor de L, se definirmos L com um valor muito elevado, aumentamos a probabilidade do algoritmo calcular dois caminhos, o que aumenta o *overhead* de processamento no nó de inicialização. De modo oposto, se definirmos L com um valor muito pequeno, o algoritmo não sofrerá benefício desta modificação. Assim, verificamos em nossas observações que $L = 3$ nos fornece um bom compromisso entre a qualidade da solução e o tempo de execução do algoritmo. O procedimento de entrada completo, ilustrado através dos exemplos de entrada dos nós 21, 20, 47 e 23, é apresentado no Algoritmo 1.

Resumindo a idéia, o procedimento de entrada tenta encontrar um pai para o novo nó executando uma busca em uma pequena parte da árvore de distribuição. O procedimento encerra quando encontra um nó folha ou quando encontra um nó interior que esteja mais

próximo do novo nó do que todos os seus filhos. Este algoritmo escala para um grande número de nós no grupo, mas não garante que o pai escolhido é o nó mais próximo entre todos os nós presentes na árvore. No entanto, uma vez que a estrutura da árvore é construída considerando a informação de distância, é provável que o procedimento de entrada tome as mesmas decisões a cada passo e agrupe nós com endereços próximos na árvore de distribuição.

Algoritmo 1 Entrada na árvore de distribuição

1: Assuma a raiz como pai potencial.
2: Se a capacidade do novo nó for maior que a capacidade do pai potencial, coloque o novo nó no lugar do pai potencial. Adicione o pai potencial na lista de movidos. Assuma como novo nó o pai potencial e como pai potencial o pai do pai potencial.
3: Calcule a distância para o pai potencial e para cada um de seus filhos.
4: Escolha o nó mais próximo de acordo com a distância.
5: Se o nó mais próximo não é o pai potencial, verifique se o módulo da diferença das distâncias entre os dois nós mais próximos é menor que o limiar L. Em caso afirmativo assuma cada um dos filhos como um pai potencial e retorne para o passo 2. Ao final do procedimento iremos ter dois pais candidatos. Verifique qual deles é o mais próximo e o assuma como pai e vá para o passo 8. Em caso negativo, assuma o nó mais próximo como pai potencial e retorne para o passo 2.
6: Se o nó mais próximo é o pai potencial e se este possui capacidade para servir o novo nó, assuma-o como pai. (Se aplicável, retorne o pai potencial como pai candidato ao passo 5).
7: Caso contrário, assuma o nó mais próximo como pai potencial e retorne para o passo 2.
8: Envie a localização do pai para o novo nó.
9: Envie uma mensagem de mudança de pai para cada um dos nós na lista de movidos.

O processo de agrupamento de nós com endereços próximos possibilita que a estrutura da árvore seja mais congruente com a topologia de rede física. Através de observações, notamos que o algoritmo do procedimento de entrada produz diferentes árvores de distribuição, considerando o mesmo conjunto de nós. Isto ocorre devido à ordem de entrada dos nós na rede. Entretanto, com o uso de um algoritmo de refinamento da árvore (Seção 3.2.5), as árvores eventualmente convergem para estruturas estáveis que possuem qualidades similares.

3.2.2 Saída de Clientes

O processo de saída de nós do grupo *multicast* pode ser divido em duas classes: saída informada e falhas de nós (Seção 3.2.3). Em uma situação ideal, o nó i que deseja sair informa ao nó de inicialização sua intenção de deixar o sistema. Recebendo esta informação, o nó de inicialização identifica o nó pai e os nós filhos deste nó i, de acordo

com a lista de membros que possui. Assim o nó de inicialização envia uma mensagem para o pai do nó i, para que este atualize imediatamente sua lista de filhos. Com a lista de filhos atualizada, o repasse dos fluxos de vídeo para o nó i é suspenso. Para os filhos do nó i, o nó de inicialização executa uma nova operação de entrada destes nós no sistema, seguindo o procedimento descrito na Seção 3.2.1. Note que implicitamente também é movida a sub-árvore que se encontra enraizada em cada um destes filhos.

Uma outra abordagem consiste em realizar uma procura na sub-árvore do nó que deseja deixar o sistema, visando encontrar um outro nó que possua capacidade suficiente para substituí-lo. Observe que o custo de tal procedimento é $O(log_k n)$, o que é bastante razoável para uma busca em um ramo da árvore, onde n e k representam respectivamente o número de nós da árvore e o valor médio do grau de saída dos nós. Esperamos medir o tempo de reparo[1] desta possibilidade e avaliar sua viabilidade em um trabalho futuro.

3.2.3 Recuperação de Particionamentos

Uma falha de nó no sistema corresponde ao evento em que o nó que deseja partir não informa ao nó de inicialização sobre sua intenção de deixar o sistema. Isto ocorre em situações em que o nó é desligado, em que ocorre uma falha de *hardware* ou quando o nó é desconectado da rede. Quando acontece uma falha em um nó interno da árvore, ocorre o que chamamos de partição na árvore de distribuição. Em casos de particionamento da árvore, os descendentes do nó devem ser capazes de detectar a falha e reparar a árvore de distribuição. O reparo da árvore é similar ao procedimento usado no caso da saída informada de um nó. A diferença entre estes eventos é que os filhos do nó não recebem uma notificação sobre a falha.

Para detectar uma partição na árvore, podem ser usadas mensagens *heartbeat* [49] para verificar se um nó está ativo no sistema. Assim, quando um nó não recebe a mensagem de seu pai por algum período, o nó assume que seu pai não está mais presente no sistema. No entanto, com o intuito de manter o *overhead* do protocolo de nossa arquitetura baixo, utilizamos uma outra abordagem pra detectar partições.

Quando ocorre um particionamento de uma árvore de distribuição, os descendentes do nó não têm certeza se um pacote foi perdido ou se está meramente atrasado. O método *Retransmission Timeout (RTO)* proposto por Begen em [8] pode ser usado para estimar o tempo esperado de chegada de pacotes. Assim, se um pacote não é recebido dentro do

[1]Tempo decorrido entre o momento em que um nó é desconectado da árvore de distribuição e sua reentrada na rede sobreposta.

período de tempo esperado, é assumido que este pacote foi perdido. Quando a taxa de perdas ultrapassa determinado limiar, é considerado pelo nó filho que seu nó pai não está mais presente na rede. Assim, este nó entra em contato com o nó de inicialização para entrar novamente no sistema. O pai do nó que apresentou a falha é informado pelo nó de inicialização e simplesmente atualiza sua lista de filhos.

3.2.4 Detecção e Resolução de Laços

Quando ocorre um particionamento na árvore de distribuição, o processo de reparo deve garantir que os filhos entrem no sistema novamente utilizando um novo pai na árvore. No entanto, é muito importante que estes nós não escolham seus descendentes como novos nós pai. Na literatura são encontrados diversos trabalhos que abordam mecanismos para detecção e resolução de laços [62] e também para impedir a construção destes [19].

Neste trabalho, optamos por não realizar um procedimento de reparo da árvore distribuído. O nó de inicialização é quem informa a cada nó onde ele deve entrar na topologia. Desse modo, o procedimento de reparo proposto não apresenta problemas de laços na árvore de distribuição.

3.2.5 Refinamento da Árvore

Devido ao ambiente altamente dinâmico, as condições da rede e de permanência no grupo mudam a todo momento. Uma re-estruturação da rede sobreposta gera uma nova árvore de distribuição e possibilita maior eficiência na distribuição dos fluxos de vídeo no sistema. Isto pode ser realizado executando periodicamente o algoritmo de entrada na árvore para os clientes que estão presentes no grupo. Um intervalo da ordem de alguns minutos pode ser suficiente. Em cada iteração, os clientes são escolhidos pelo nó de inicialização de forma aleatória e são excluídos de uma lista de refinamento. Isto ocorre para que todos os nós tenham a mesma oportunidade de executar o algoritmo. Após o último nó ter executado o algoritmo, todos os nós são colocados novamente na lista de refinamento para que o procedimento continue.

Um membro muda para um novo pai na topologia se esse pai é mais próximo que o pai corrente. Para evitar oscilações, é necessário assumir um limiar para verificar o ganho de proximidade na troca. Se o ganho estiver abaixo deste limiar, não é realizada a mudança de pai. Caso contrário, o nó de inicialização envia uma mensagem de mudança de pai para o nó em questão para que a troca de nós pai seja feita, de modo análogo ao descrito

na Seção 3.2.1.

3.2.6 Considerações sobre a Implantação do Sistema Proposto

Por motivos de simplicidade, a construção de redes sobrepostas usualmente assume comunicação de redes em duas vias. Logo, espera-se que cada nó possa iniciar e receber conexões de outros nós. No entanto, na Internet atual, isto não é sempre verdade devido ao uso de NAT (*Network Address Translation*) e *firewalls*. Um estudo realizado em [57] nos mostra que cerca de 34% dos nós em redes P2P possuem conectividade restrita, ou seja, estes nós não podem aceitar conexões de outros nós na Internet. A carência da possibilidade de uma conexão em duas vias revela um desafio, pois somente um sub-grupo de nós pode atuar como roteadores na rede sobreposta.

Uma solução simples para o problema de NAT consiste em assumirmos que os nós atrás de um *gateway* NAT serão somente nós folhas na árvore de distribuição. Dessa forma, somente os nós que são globalmente endereçáveis serão nós pai. No entanto, se a maioria de clientes estiverem atrás de NATs, esta solução pode não ser a mais eficiente. O impacto deste problema pode ser reduzido se permitirmos que nós atrás do mesmo *gateway* NAT possam criar relações pai-filho. A estratégia consiste em associar cada membro do grupo com dois pares de endereços e portas: um endereço IP global com sua respectiva porta como vistos pelo nó de inicialização e o IP local e sua respectiva porta, como vistos pelo próprio nó. Dessa forma, quando o nó de inicialização executar o procedimento de entrada e ocorrer que $D_{A-B} = 0$, o nó de inicialização sabe que existem dois nós atrás do mesmo NAT. Assim o nó de inicialização envia para o nó que deseja se juntar ao grupo o endereço de rede local do nó que está presente na árvore. Observe que o uso desta estratégia também possibilita que um nó, mesmo atrás de um NAT, possa receber conexões, uma vez que o nó de inicialização conhece o endereço e porta global que serão traduzidos para o endereço e porta local pelo *gateway* NAT.

Um outro problema relacionado ao NAT é o *time-out* das informações que são adicionadas dinamicamente na tabela de tradução de endereços. Este problema pode ser resolvido facilmente utilizando o envio periódico de mensagens *keepalive* por cada um dos nós que possuem endereço privado de rede no sistema. Assim conseguimos manter todas as conexões via NAT ativas.

Capítulo 4

Implementação da Arquitetura Proposta

Recentemente, as redes *peer-to-peer* (P2P) têm sido utilizadas para compartilhamento de arquivos, mensagens instantâneas, *multicast* em nível de aplicação, entre outros tipos de aplicações. Esta tecnologia muda o paradigma cliente-servidor atual, de forma que não depende de uma organização central ou hierárquica, e ainda fornece aos seus integrantes as mesmas capacidades e responsabilidades. Assim, um integrante pode acessar diretamente os recursos de qualquer outro, sem a necessidade de um controle centralizado.

As redes P2P são redes sobrepostas que funcionam na Internet com o objetivo de compartilhar recursos entre os usuários, sendo que por princípio não há diferenciação entre os participantes. Existem diversas outras definições na literatura, porém em geral é aceito pela comunidade que sistemas P2P devem suportar os seguintes requisitos [43]:

- A rede deve ser escalável;
- Os nós devem estar localizados nas bordas da rede;
- A rede deve possuir nós com conectividade variável ou temporária e endereços também variáveis;
- Os nós devem possuir a capacidade de se comunicarem diretamente uns com os outros;
- A rede deve possuir a capacidade de lidar com diferentes taxas de transmissão entre nós;
- A rede deve possuir nós com autonomia parcial ou total em relação a um servidor centralizado;
- A rede deve assegurar que os nós possuam capacidades iguais de fornecer e consumir recursos de outros nós.

O crescimento de popularidade das aplicações P2P tem inspirado diversas atividades de pesquisa nesta área, muitas envolvendo modelagem e projetos de arquiteturas e protocolos [23]. Realizar a validação desses trabalhos utilizando uma implementação real em cenários com um grande número de nós representa um grande desafio. Desse modo, o uso de simuladores nos fornece uma alternativa viável para fins de avaliação e validação das propostas.

Estudos realizados por outros trabalhos [42, 45, 46] sugerem que o desempenho de sistemas P2P é altamente dependente das características físicas da rede. Como o tráfego P2P cada vez mais corresponde a uma grande proporção do tráfego total da Internet [1], se torna crucial que as pesquisas nesta área incluam considerações sobre o impacto gerado por estes sistemas na rede física.

O ns-2 [33] é um simulador de eventos discretos que objetiva a pesquisa de redes de computadores, principalmente redes IP. O ns-2 fornece suporte para a simulação de protocolos de roteamento, *unicast* e *multicast* tanto sobre redes convencionais como sobre redes sem fio, sejam elas locais ou via satélite. Este simulador oferece condições suficientes para avaliação do impacto dos sistemas P2P na rede física, uma vez que o ns-2 é um simulador de redes com detalhamento a nível de pacotes.

O ns-2 começou a ser difundido em 1989 como uma variação do simulador de redes REAL e tem evoluído substancialmente nos últimos anos. Em 1995, o desenvolvimento do ns-2 foi patrocinado pelo DARPA (*Defense Advanced Research Projects Agency*), através do projeto VINT (*Virtual InterNetwork Testbed*) no LBNL (*Lawrence Berkeley National Laboratory*), Xerox PARC (*Palo Alto Research Center*), UCB (*University of California at Berkeley*) e USC/ISI (*University of Southern California / Information Sciences Institute*). Atualmente o desenvolvimento do ns-2 é patrocinado pelo DARPA / SAMAN (*Simulation Augmented by Measurement and Analysis for Networks*) e através do NSF (*National Science Foundation*) com a CONSER (*Collaborative Simulation for Education and Research*), e em colaboração com outros pesquisadores [17].

Apesar de existirem outros simuladores, o ns-2 tem sido consistentemente adotado pelas comunidades acadêmicas em todo o mundo pelos seguintes motivos:

- **Documentação:** o ns-2 tem um manual bastante detalhado que é mantido pelos desenvolvedores e atualizado regularmente, além de vários tutoriais disponíveis na Internet;
- **Grande número de usuários:** uma vez que o ns-2 é muito utilizado em várias uni-

versidades e centros de pesquisa no mundo, conta com a colaboração de um grande número de pessoas que estão constantemente disponibilizando novas funcionalidades para extender o aplicativo;

- **Lista de discussão:** o ns-2 possui listas de discussão oficiais para desenvolvedores e usuários, fornecendo um ambiente cooperativo para troca de idéias, esclarecimento de dúvidas e solução de problemas.

Assim optamos pela utilização do ns-2 como simulador de redes para avaliação da nossa proposta. No entanto, uma das dificuldades encontradas foi o fato do ns-2 não possuir suporte nativo para comunicação P2P, um dos requisitos necessários para implementação do protocolo de comunicação usado em nossa arquitetura. Sistemas P2P necessitam que um nó se conecte a vários outros nós da rede de maneira dinâmica. Por padrão, o ns-2 só permite conexões definidas *a priori* no roteiro de simulação.

O trabalho proposto em [23] apresenta um *framework* para simulações P2P do protocolo Gnutella [21]. Este *framework* não atendia nossas necessidades por ser voltado a redes P2P de compartilhamento de arquivo, e por este motivo resolvemos implementar nosso próprio *framework* para simulação P2P no ns-2. Para acomodar uma grande variedade de estudos na área e modelar diversos protocolos P2P, nosso *framework* conta com as seguintes características:

- **Genérico:** Sistemas P2P podem se diferenciar em vários aspectos, tais como inicialização, manutenção do relacionamento entre nós, roteamento das mensagens do protocolo e comportamento dos usuários. O *framework* deve capturar os aspectos comuns enquanto acomoda um grande grau de diversidade em cada aspecto. Para resolver este requisito, uma caracterização em alto nível de sistemas P2P é requerida.

- **Fácil de estender e usar:** Grande parte dos usuários têm a necessidade de reprojetar ou adicionar novos componentes no sistema P2P. Para evitar esforços duplicados pelos usuários, o *framework* foi projetado para ser modular e facilitar sua extensão.

O restante do capítulo apresenta como foi realizada a implementação do *framework* para simulação P2P no ns-2, bem como a implementação do protocolo das arquiteturas TVoIP, CoopNet, Narada e HMTP utilizando este *framework*. Além disso, iremos apresentar um simulador próprio que utilizamos para realizar cálculos sobre o custo da árvore de distribuição seguindo algumas das métricas de desempenho que foram apresentadas no Capítulo 2.

4.1 *Framework* P2P

Em nosso *framework*, cada nó da rede P2P possui três camadas arquiteturais, conforme pode ser observado na Figura 4.1. A camada base representa o nível de pacotes da rede do simulador, a qual fornece a transferência de pacotes para a camada de transporte. A camada do meio é o agente P2P (PeerAgent), o qual lida diretamente com o protocolo específico de comunicação P2P. Suas responsabilidades incluem transmissão, recepção, processamento e encaminhamento das mensagens do protocolo P2P. A camada de aplicação P2P (PeerApp), no topo da arquitetura, executa ações relacionadas à aplicação tais como compartilhamento de recursos, processamento de requisições e relacionamento entre os nós. Esta separação de funcionalidades entre PeerAgent e PeerApp é baseada em duas considerações:

- Existe freqüentemente a necessidade de um mapeamento muitos-para-um entre aplicação e protocolo P2P ou vice-versa. Por exemplo, as aplicações LimeWire [30] e BearShare [36] utilizam o mesmo protocolo Gnutella para compartilhamento de arquivos e podem ser implementadas utilizando o mesmo agente do protocolo Gnutella. Em outros casos, pode ser interessante comparar diferentes protocolos executando a mesma aplicação no topo de cada agente.

- Pequisadores conduzindo trabalhos nestas duas camadas freqüentemente têm diferentes preocupações ou áreas de interesse. As pesquisas na camada do protocolo focam no comportamento do sistema de comunicação. Pesquisas na camada de aplicação podem manter foco em assuntos relacionados ao processamento de requisições, comportamento do usuário, relacionamento entre os nós e outros. Assim, espera-se que esta separação possibilite que eles possam independentemente considerar seus componentes de interesse.

A implementação do *framework* seguiu as linguagens de programação utilizadas no ns-2. Assim, foram implementadas um conjunto de classes base em C++ e algumas interfaces para acesso a estas classes em TCL. Adotamos a mesma nomenclatura das classes base encontradas no *framework* P2P do protocolo Gnutella [23]. A seguir iremos descrever os componentes arquiteturais, conforme apresentado na Figura 4.1.

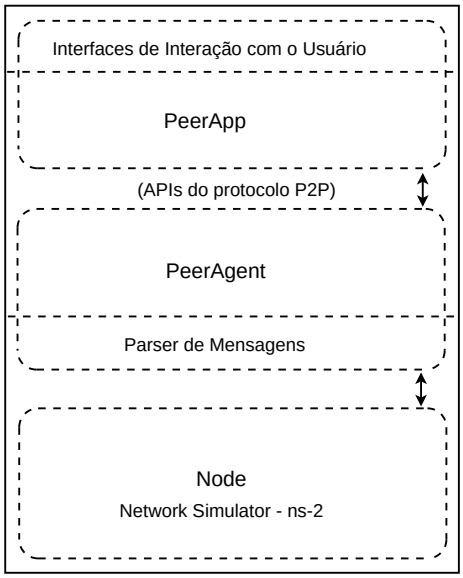


Figura 4.1: Arquitetura do *framework*

4.1.1 PeerApp

O componente PeerApp é a interface direta para os usuários da aplicação P2P. Ele encapsula o comportamento das mais variadas aplicações. Identificamos algumas das atividades mais comuns encontradas nas aplicações, que foram categorizadas como:

- **Tarefas básicas de um nó:** Durante a permanência do nó no sistema, o nó executa certos tipos de operações bem definidas como entrada e saída da rede e busca por conteúdo. Os usuários da aplicação acessam estas operações através de APIs (*Application Programming Interface*), que são chamadas de procedimento que podem disparar certas mensagens de protocolo como também mudanças no estado de um nó.

- **Manutenção do relacionamento entre nós:** A troca de mensagens de um nó em um sistema P2P normalmente envolve dois tipos de operações: inicialização e manutenção. Na inicialização o nó aprende, seleciona e se conecta a um número de nós no momento de entrada no sistema. Na manutenção o nó se adapta às mudanças do sistema, possivelmente removendo alguns nós e incluindo novos outros, para que seja possível melhorar a conectividade.

4.1.2 PeerAgent

O componente PeerAgent executa o processamento e transmissão das mensagens do protocolo P2P. As APIs entre PeerApp e PeerAgent correspondem aos tipos de mensagens definidos em um protocolo específico de comunicação P2P. Um pacote entregue a um PeerAgent pelo ns-2 é primeiro passado a um *parser* de mensagens. Baseado nos detalhes da mensagem, o PeerAgent pode descartar, entregar ao PeerApp ou encaminhar esta mensagem para outros nós. Além disso, realizamos a implementação de um mecanismo de entrega confiável das mensagens de protocolo, de forma similar à utilizada na primeira versão do protocolo TCP [39].

4.2 Especializando o *Framework*

Utilizando os componentes descritos na seção anterior, especializamos o *framework* para suportar os protocolos de *multicast* na camada de aplicação. Os componentes principais, RendezvousPoint e Peer são subclasses de PeerApp (Figura 4.2). A classe Peer representa um nó na rede sobreposta e a classe RendezvousPoint representa o nó de inicialização que os nós contactam para a entrada na rede.

Implementamos a arquitetura TVoIP seguindo a descrição realizada no capítulo anterior (Capítulo 3). Os nós entram na rede contactando o nó de inicialização, o qual tem conhecimento total da rede sobreposta e gerencia a árvore de distribuição. Cada nó, ao enviar uma requisição de entrada ao nó de inicialização, recebe a localização de seu respectivo pai na árvore de distribuição. Eventualmente, no processo de entrada de um novo nó, ocorrem mudanças de posição nos nós internos da árvore. Estes nós recebem do nó de inicialização uma mensagem informando a mudança de pai na rede sobreposta. Os nós internos se conectam ao novo pai e realizam a desconexão do pai anterior somente após receberem pacotes de dados do novo pai.

Para avaliar a eficiência do nosso protocolo em relação a outros trabalhos, consideramos realizar a implementação das arquiteturas CoopNet, Narada e HMTP. As arquiteturas foram implementadas de acordo com as descrições realizadas no Capítulo 2, sem realizar nenhum mecanismo de otimização na rede sobreposta. O nó de inicialização do protocolo Narada, ao receber uma mensagem de entrada na rede, envia uma lista de nós para o novo nó, o qual escolhe um nó aleatoriamente e envia uma requisição para este nó para tornar-se filho. No protocolo HMTP, o nó de inicialização sempre retorna a raiz da árvore, o que faz com que os nós que desejam entrar no grupo executem um algoritmo de busca

em profundidade para encontrar um nó próximo de acordo com o RTT. No CoopNet, a árvore de distribuição é preenchida por nível, e o nó de inicialização que também é a fonte de vídeo, escolhe um nó aleatório entre os nós que possuem disponibilidade no nível para tornar pai do nó que deseja juntar-se ao grupo. A Figura 4.2 ilustra o diagrama de hierarquia das classes principais utilizadas na implementação.

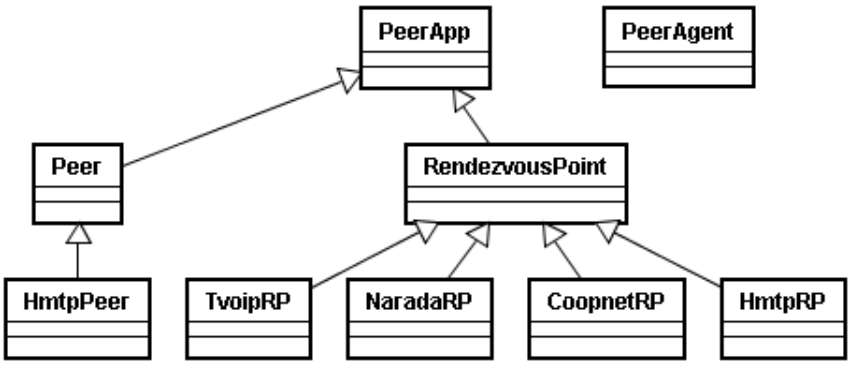


Figura 4.2: Diagrama de hierarquia das principais classes utilizadas na implementação das arquiteturas no ns-2.

Para realizarmos uma comparação justa entre os protocolos, assumimos que em todas as arquiteturas o nó de inicialização é também a fonte de vídeo. Além disso, não foram considerados o algoritmo de melhoramento da árvore de distribuição do protocolo HMTP e o algoritmo de remoção e inserção dinâmica de enlaces na rede em malha do protocolo Narada. Para o CoopNet foi considerada somente uma árvore de distribuição, ou seja, o envio de um único descritor de vídeo.

Exemplos de uso do *framework* e sua documentação encontram-se disponíveis na página do grupo de redes da Universidade Federal Fluminense (UFF) [54].

4.3 O Simulador Netsim

Para realizar o cálculo do custo da árvore de distribuição, optamos por implementar um novo aplicativo, o qual chamamos de *netsim*. O *netsim* também foi implementado em linguagem C++.

Este aplicativo recebe como entrada um arquivo de topologia da rede, gerado através da ferramenta geradora de topologias BRITE (*Boston University Representative Internet Topology Generator*)[34], e cria um grafo que representa a rede física. O *netsim* também recebe um arquivo com os tempos de entrada de cada um dos nós na rede sobreposta

exportado pelo ns-2. A partir da rede física é calculada a árvore de caminhos mínimos, utilizando o algoritmo de Dijkstra [27], para representar os caminhos *unicast* entre o emissor e cada um dos receptores.

Foram desenvolvidos três tipos de pacotes de classes (Figura 4.3):

- **Metrics:** Este pacote contém as classes para cálculo das métricas de desempenho.
- **Overlay:** Este pacote contém as implementações das arquiteturas HMTP, Narada, CoopNet e TVoIP.
- **Physical**: Este pacote contém as classes que compõem a rede física, bem como as rotinas para leitura da topologia e cálculo da árvore de caminhos mínimos.

Através do simulador *netsim*, realizamos a simulação e posterior cálculo da saturação dos enlaces, penalidade relativa ao atraso e uso de recursos para as árvores de distribuição das arquiteturas CoopNet, Narada, HMTP e TVoIP.

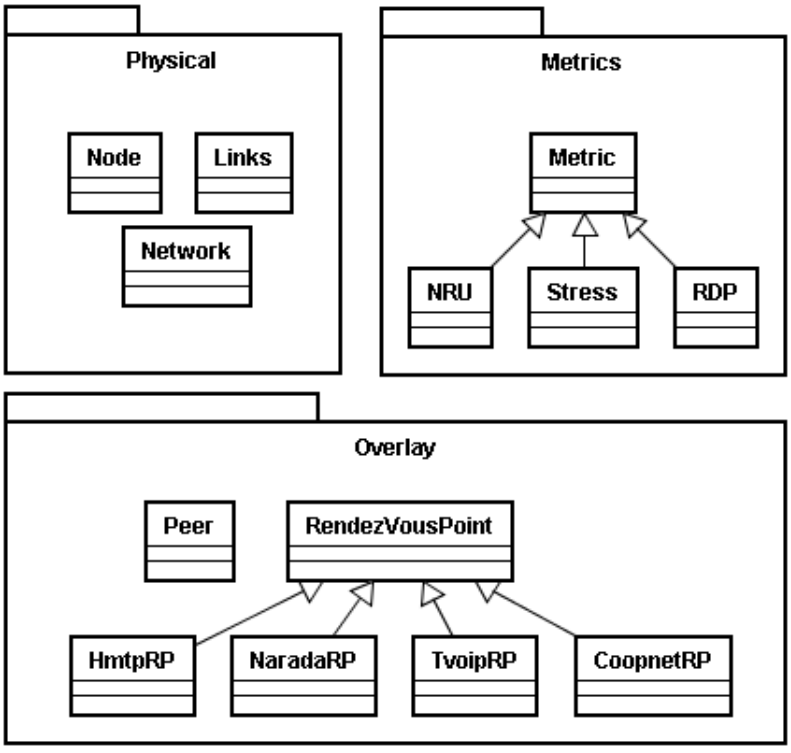


Figura 4.3: Diagrama de hierarquia das classes utilizadas na implementação do *netsim*.

Exemplos de uso do *netsim* e sua documentação encontram-se disponíveis na página do grupo de redes da UFF [54].

Capítulo 5

Análise Experimental

Neste capítulo são apresentados os resultados computacionais obtidos através de simulações para a arquitetura proposta. O objetivo principal dos experimentos realizados é comprovar a viabilidade e escalabilidade da arquitetura em um cenário de Internet TV quando comparado a outros trabalhos como *unicast*, IP *multicast*, Narada, HMTP e CoopNet.

5.1 Métricas de Desempenho

Uma rede sobreposta bem construída busca otimizar tanto a qualidade dos caminhos percorridos pelos dados como o desempenho observado nos nós finais. A qualidade dos caminhos dos dados é avaliada observando as características: número de pacotes duplicados nos enlaces, uso de recursos de rede e aumento de atraso em relação ao caminho *unicast*. O desempenho nos nós finais é avaliado de acordo com as características percebidas pelos usuários tais como atraso, *jitter*, perda de pacotes, tempo de entrada no sistema e carga em relação à largura de banda disponível.

Nos experimentos realizados para avaliação da arquitetura, a qualidade dos caminhos dos dados foi avaliada de acordo com os resultados obtidos através do simulador *netsim* (Seção 4.3), desenvolvido para este propósito. As métricas de desempenho utilizadas foram as seguintes:

- **Saturação do enlace:** Também conhecida como *link stress*, esta métrica mede o número de pacotes idênticos que trafegam nos enlaces. O objetivo é verificar a eficiência do TVoIP em distribuir a carga da rede nos enlaces físicos;
- **Penalidade relativa ao atraso (PRA):** Também conhecida como *stretch*, a penalidade relativa ao atraso entre emissor e receptor é definida como a razão do

atraso entre eles na rede sobreposta e o equivalente utilizando *unicast*. Isto pode ser expresso como $\frac{d'_i}{d_i}$, onde d'_i representa o atraso do nó emissor até o nó i na rede sobreposta e d_i o atraso *unicast* entre eles na rede física. O objetivo é medir o aumento de atraso que os nós finais percebem ao utilizarem o TVoIP e os demais protocolos de *multicast* a nível de aplicação;

- **Uso de recursos normalizado (URN):** Esta métrica é definida como a razão entre o uso de recursos de um protocolo *multicast* a nível de aplicação e o uso de recursos do IP *multicast*. O uso de recursos pode ser definido como $\sum_{i=1}^{L} d_i * s_i$, onde L expressa o número de enlaces ativos na transmissão de dados, d_i o atraso referente ao enlace i e s_i a saturação do enlace i. O objetivo é avaliar o uso de recursos adicionais de rede consumidos pelos protocolos *multicast* a nível de aplicação quando comparado ao IP *multicast*.

A utilização do IP *multicast* e do *unicast* nas métricas de desempenho nos fornece limites teóricos usados como referência de comparação para os protocolos de *multicast* a nível de aplicação [7]. As árvores de distribuição formadas pelo IP *multicast* possuem a menor saturação, uma vez que cada enlace encaminha apenas uma cópia de cada pacote. Os caminhos *unicast* fornecem o menor atraso e por isto foram considerados como unidade na métrica PRA. Diante disso, podemos afirmar que o IP *multicast* possui PRA de valor 1 (assumindo roteamento simétrico), saturação no pior caso igual a 1 e por definição URN de valor 1. Os caminhos *unicast* possuem PRA por definição de valor 1 e uma saturação no pior caso de n, onde n representa o número de receptores no grupo *multicast*.

Para avaliar o desempenho nos nós finais, foram utilizados os resultados obtidos pelo simulador de redes ns-2 com a implementação do módulo TVoIP discutida no capítulo anterior (Seção 4.2). As métricas de desempenho utilizadas foram as seguintes:

- **Atraso médio da árvore de distribuição:** Considere d_i como sendo o atraso médio fim-a-fim sofrido pelos pacotes no caminho do nó emissor ao nó i. O atraso médio da árvore de distribuição é calculado através da média aritmética destes atrasos, ou seja, pela expressão $\frac{1}{n}\sum_{i=1}^{n} d_i$, onde n representa o tamanho do grupo *multicast*. O objetivo é avaliar o atraso percebido pelos usuários do TVoIP frente a outros trabalhos;

- ***Jitter* médio da árvore de distribuição:** Considere j_i como sendo a variação média do atraso fim-a-fim sofrido pelos pacotes no caminho do nó emissor ao nó i. O *jitter* médio da árvore de distribuição é calculado através da média aritmética

destas variações, ou seja, pela expressão $\frac{1}{n}\sum_{i=1}^{n} j_i$, onde n representa o tamanho do grupo *multicast*. O objetivo é verificar a continuidade do vídeo recebido pelos usuários do TVoIP quando comparado a outros trabalhos;

- **Carga média da árvore de distribuição:** Considere l_i como sendo a carga sofrida pelo nó i, calculada através da seguinte expressão: $l_i = \frac{filhos_i}{capacidade_i}$, onde $filhos_i$ representa o número de nós que o nó i encaminha pacotes e $capacidade_i$ o número de clientes suportados pelo nó i de acordo com sua largura de banda anunciada. O cálculo da carga média da árvore de distribuição pode ser expresso como a média aritmética da carga de todos os nós internos, isto é, dos nós que estão servindo clientes. Isto pode ser calculado através da expressão $\frac{1}{q}\sum_{i=1}^{q} l_i$, onde q expressa o número de nós internos na árvore. O objetivo é verificar a eficiência dos protocolos de *multicast* a nível de aplicação em distribuir a carga entre os nós da rede;

- **Perda de pacotes média da árvore de distribuição:** A perda de pacotes média é calculada através da média aritmética do número de pacotes perdidos por cada nó na árvore de distribuição. Isto pode ser calculado através da expressão $\frac{1}{n}\sum_{i=1}^{n} p_i$ onde p_i expressa o número de pacotes perdidos pelo nó i e n o tamanho do grupo *multicast* . O objetivo é avaliar como o aumento da taxa de transferência de vídeo influencia o número de pacotes perdidos no TVoIP e nos outros trabalhos;

- **Tempo de entrada médio da árvore de distribuição:** O tempo de entrada médio da árvore de distribuição é calculado através da média aritmética do tempo de entrada de cada nó na árvore de distribuição. Isto pode ser calculado pela expressão $\frac{1}{n}\sum_{i=1}^{n} t_i$ onde t_i representa o tempo de entrada do nó i e n o tamanho do grupo *multicast* . Assim podemos comparar os diferentes tempo de entrada na rede de cada uma dos trabalhos com o do TVoIP.

5.2 Modelo de Simulação

Estudos anteriores [51] mostram que as topologias de rede de larga escala da Internet seguem as propriedades das redes *small-world* e das leis de potência (*power law*). As leis de potência descrevem o grau de um nó enquanto que as redes *small-world* descrevem características sobre o tamanho dos caminhos e coeficiente de agrupamento [9]. O estudo encontrado em [45] nos mostra que as topologias geradas usando sistemas autônomos

(AS) possuem as propriedades das redes *small-world* e das leis de potência. Assim estas topologias conseguem representar as redes de larga escala da Internet.

A ferramenta de geração de topologias BRITE [34] fornece a possibilidade de gerar topologias baseadas em AS, e por isto foi considerada neste trabalho. Usando o BRITE, geramos 5 topologias para a rede física seguindo o modelo de Waxman [61] com 100 AS com 10 roteadores cada, totalizando uma rede com 1000 nós. Os parâmetros para geração das topologias resultaram em atrasos de propagação nos enlaces que variaram de 0.15 ms até 42 ms para todas as topologias. As taxas de transmissão dos enlaces foram configuradas de forma que os enlaces do *backbone* tivessem capacidade de 100 Mbps. Os enlaces de acesso possuem largura de banda variando de 3 Mbps a 10 Mbps, modelando enlaces típicos para interconexão de redes ao *backbone* (rádio a 3 Mbps, xDSL até 8Mbps, cabo de até 10 Mbps). As redes locais possuem capacidade de 10 Mbps modelando redes corporativas e residenciais.

A localização dos clientes e do nó de inicialização (*bootstrap node*) na topologia foi escolhida de maneira uniformemente aleatória utilizando um subconjunto S de nós da topologia. Este subconjunto S representa os nós de borda da rede. O número de clientes do grupo *multicast* variou de 100 a 700, com incrementos de 100 clientes. Para cada cliente, o número de vizinhos na rede sobreposta (grau de saída) variou de 1 a 4. O processo de entrada na rede sobreposta foi modelado como uma distribuição de Poisson, pois consegue modelar satisfatoriamente eventos de entrada de usuários em uma rede P2P [23].

De forma a variar a qualidade do vídeo, neste trabalho foram utilizadas taxas de transmissão que variam de 150 kbps até 2 Mbps, modelando as taxas de transmissão de vídeo de codificadores de baixa qualidade até codificadores de alta qualidade. Com relação ao processo de perdas de pacotes, foram consideradas apenas perdas por congestionamento nas filas dos roteadores.

Para avaliar a eficiência da nossa proposta em relação a outros trabalhos, consideramos realizar a implementação dos protocolos Narada, HMTP e CoopNet. As arquiteturas foram implementadas de acordo com as especificações descritas no Capítulo 2, sem realizar nenhum mecanismo de otimização na rede sobreposta. Assim, conforme descrito na Seção 4.2, não foram considerados o algoritmo de melhoramento da árvore do protocolo HMTP e o algoritmo de remoção e inserção dinâmica de enlaces na rede em malha do protocolo Narada. Para o CoopNet foi considerado somente uma árvore de distribuição, ou seja, o envio de um único descritor de vídeo.

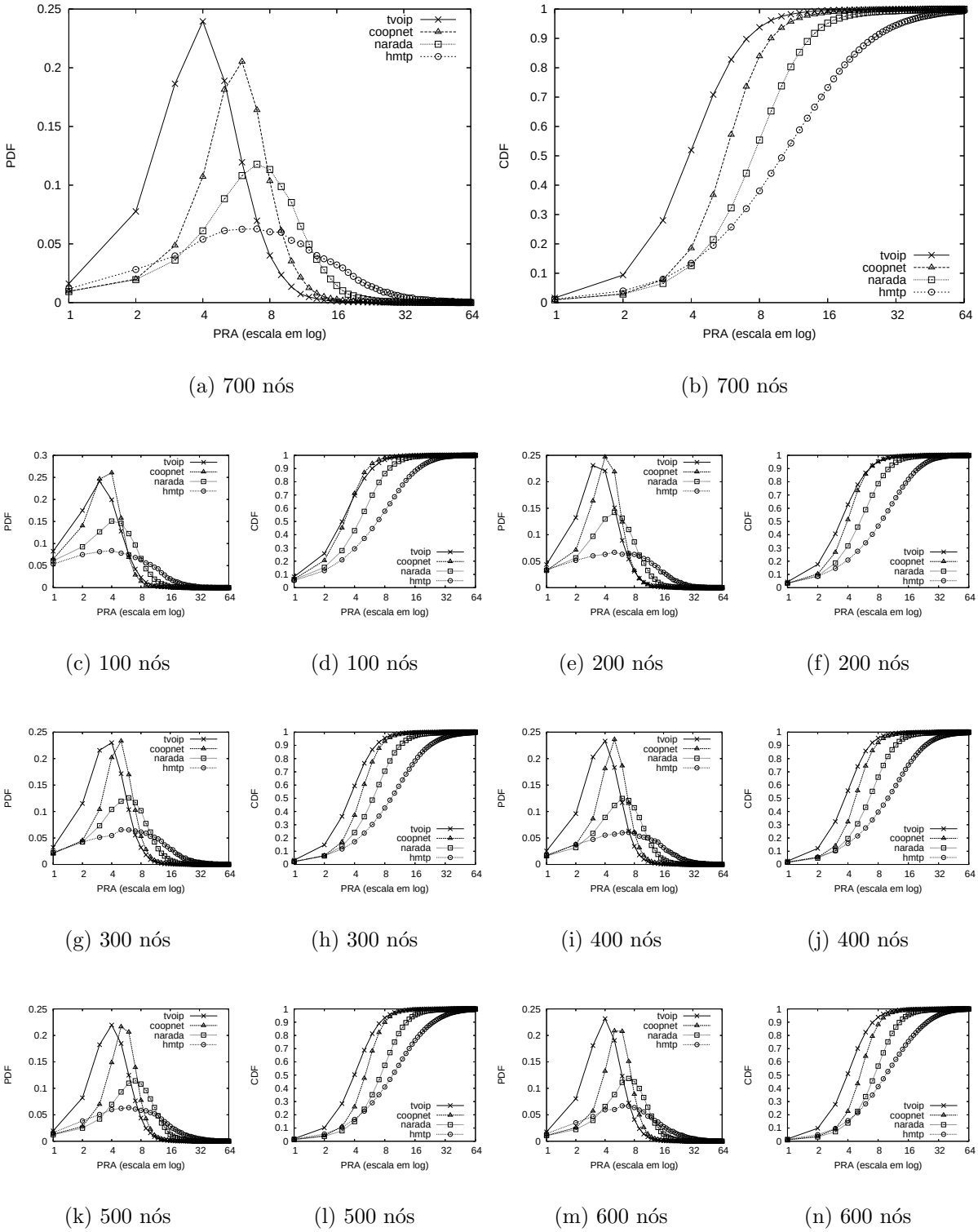
5.3 Resultados

Esta seção visa avaliar o desempenho da arquitetura TVoIP frente a outras arquiteturas de acordo com as métricas de desempenho descritas na Seção 5.1. Devido à característica aleatória de escolha dos clientes (Seção 5.2) e por utilizarmos 5 diferentes topologias de rede, foram executadas 100 iterações para cada experimento, sendo 20 iterações para cada topologia, e calculada a média aritmética com um intervalo de confiança de 95% para cada ponto encontrado nos gráficos. O simulador ns-2 foi configurado para executar durante um período de simulação de 900 segundos para cada iteração. Por padrão, foram considerados para os experimentos um tamanho de grupo de 100 usuários e uma taxa de transmissão de 150 kbps.

5.3.1 Qualidade dos Caminhos dos Dados

Esta seção tem como objetivo avaliar os caminhos percorridos pelos dados observando as características: aumento de atraso em relação ao caminho *unicast*, número de pacotes duplicados nos enlaces e uso de recursos de rede.

A Figura 5.1 ilustra pares de gráficos com as curvas de função densidade de probabilidade (*Probability Density Function* - PDF) e função distribuição cumulativa (*Cumulative Distribution Function* - CDF) da PRA para diferentes tamanhos de grupos *multicast*. O eixo horizontal representa os valores da PRA enquanto que o eixo vertical representa a porcentagem de pares emissor-receptor que possuem determinado valor de PRA para PDF e que são menores que este valor na CDF. Cada curva corresponde ao estado da rede após a entrada do último cliente no grupo *multicast*. Podemos observar claramente em todos os gráficos que o TVoIP possui o menor valor de PRA, seguido pelo CoopNet, Narada e HMTP; exceto nos casos onde o grupo é pequeno como por exemplo nas Figuras 5.1(c) onde o TVoIP e o CoopNet possuem valores similares. Note que este comportamento se repete para todos os tamanhos de grupo *multicast*. Isto indica que o TVoIP consegue construir redes sobrepostas com caminhos com um atraso menor que as outras propostas. Esta característica se deve ao fato do TVoIP construir árvores baixas, devido à sua propriedade de sempre manter os nós com maior grau de saída da árvore conectados logicamente próximos da raiz. Observe que o CoopNet tem a característica de preencher a árvore de distribuição por nível, o que garante que sua árvore também possua altura baixa. Os protocolos Narada e o HMTP possuem os piores resultados, pois constroem árvores altas.



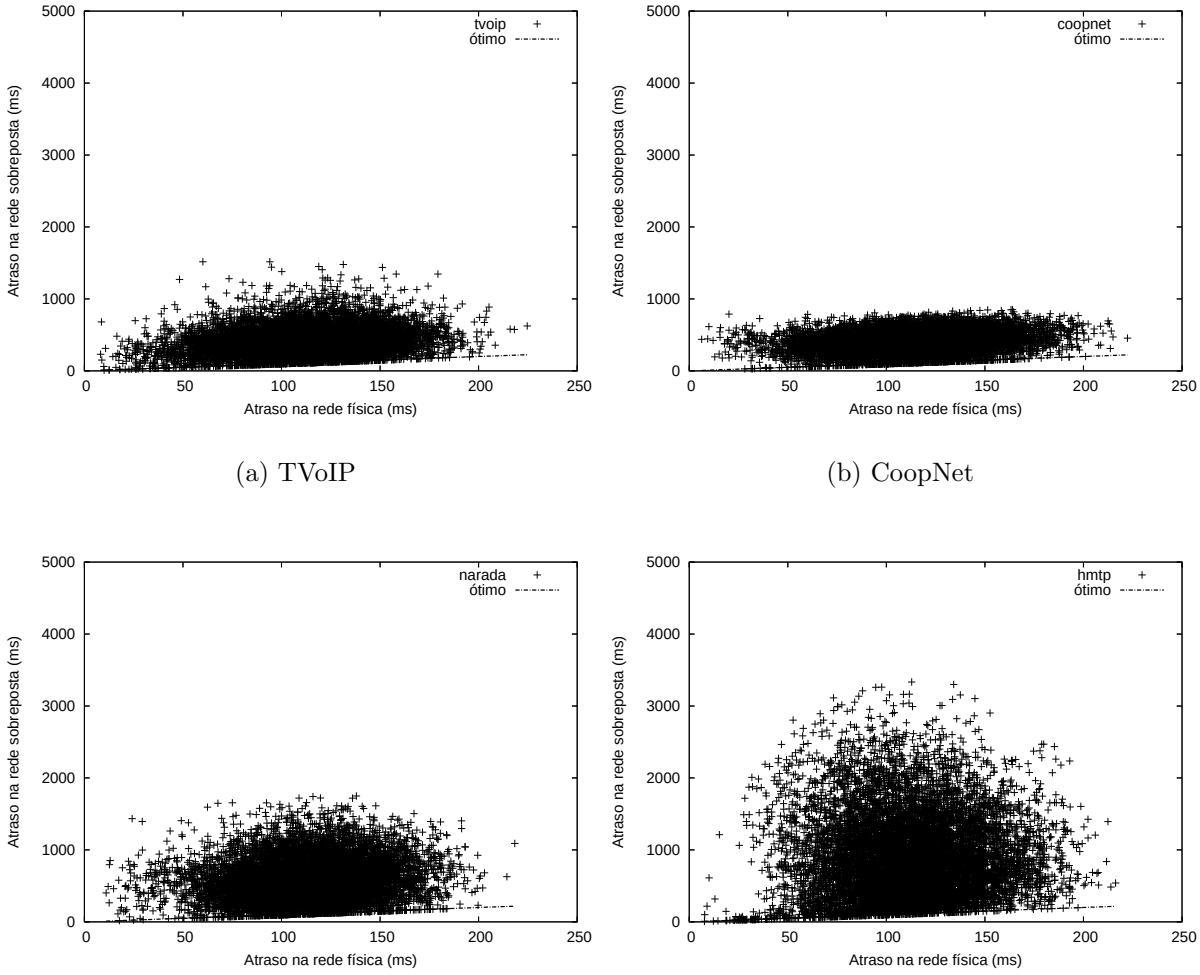
(a) 700 nós (b) 700 nós

(c) 100 nós (d) 100 nós (e) 200 nós (f) 200 nós

(g) 300 nós (h) 300 nós (i) 400 nós (j) 400 nós

(k) 500 nós (l) 500 nós (m) 600 nós (n) 600 nós

Figura 5.1: PDF e CDF da PRA para diferentes tamanhos de grupos *multicast*.



(a) TVoIP

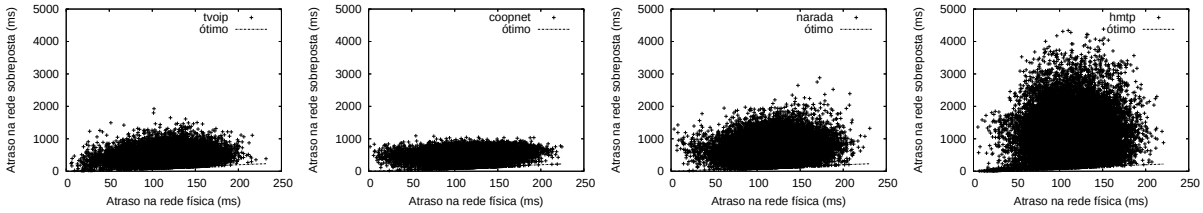
(b) CoopNet

(c) Narada

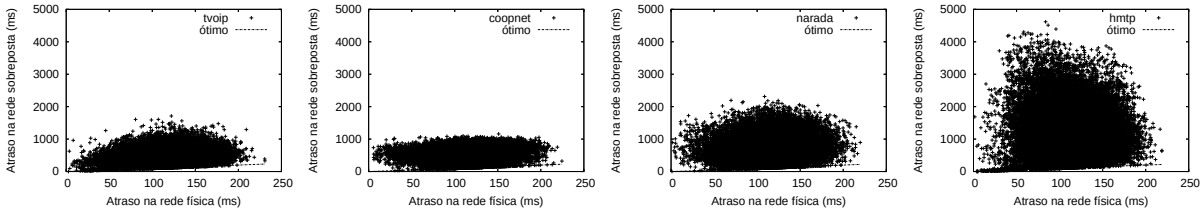
(d) HMTP

Figura 5.2: Atrasos da rede sobreposta e da rede física para pares emissor-receptor de grupos de 100 nós.

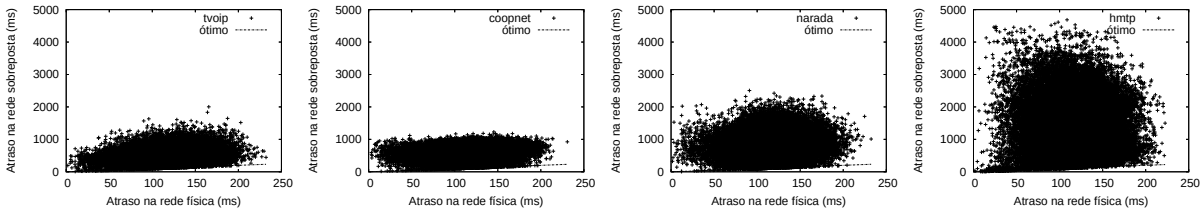
As Figuras 5.2 e 5.3 comparam o atraso da rede sobreposta com o atraso *unicast* da rede física e ilustram como estão distribuídos estes atrasos. Cada ponto no gráfico representa a existência de um par emissor-receptor para determinado valor de atraso na rede sobreposta e na rede física. Como um par emissor-receptor fornece um valor muito particular em cada umas das iterações, para este experimento não foi realizado o cálculo da média de cada uma das amostras, e sim plotado os valores dos pares encontrados em cada iteração. Observe que este grupo de gráficos apresenta uma outra visão da métrica PRA, pois conforme descrito na Seção 5.1, $PRA = \frac{d'_i}{d_i}$, onde d'_i representa o atraso do par i na rede sobreposta e d_i na rede física.



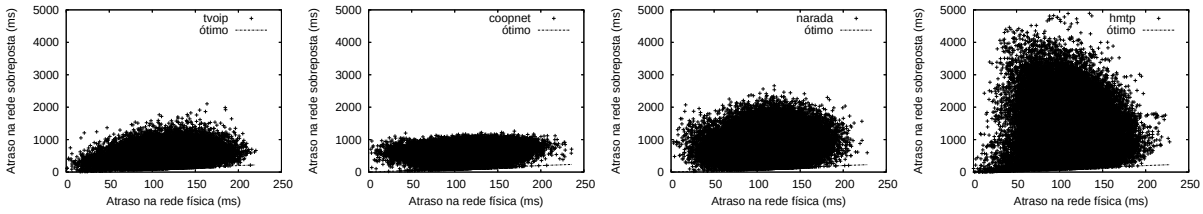
(a) TVoIP - 200 nós (b) CoopNet - 200 nós (c) Narada - 200 nós (d) HMTP - 200 nós



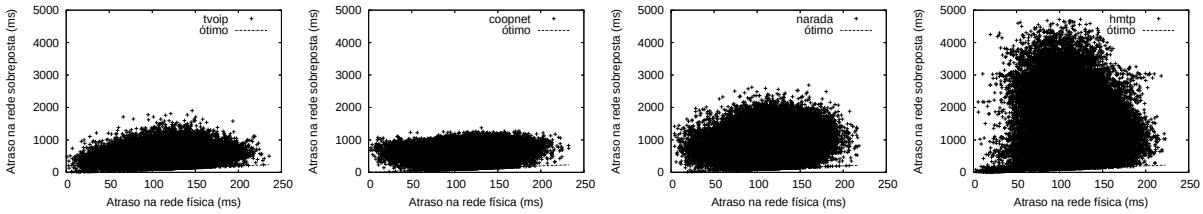
(e) TVoIP - 300 nós (f) CoopNet - 300 nós (g) Narada - 300 nós (h) HMTP - 300 nós



(i) TVoIP - 400 nós (j) CoopNet - 400 nós (k) Narada - 400 nós (l) HMTP - 400 nós



(m) TVoIP - 500 nós (n) CoopNet - 500 nós (o) Narada - 500 nós (p) HMTP - 500 nós



(q) TVoIP - 600 nós (r) CoopNet - 600 nós (s) Narada - 600 nós (t) HMTP - 600 nós

Figura 5.3: Atrasos da rede sobreposta e da rede física para pares emissor-receptor de grupos de 200 a 600 nós.

Nas Figuras 5.2 e 5.3 também podemos observar que os protocolos TVoIP e o CoopNet apresentam os melhores resultados em relação ao atraso. Fica claro nas figuras que o TVoIP apresenta uma variância maior que o CoopNet. Isto porque o CoopNet preenche sua árvore de distribuição por nível, o que faz com que o aumento do número de saltos no caminho emissor-receptor seja incremental, ou seja, começa no menor valor e vai crescendo conforme o aumento de níveis da árvore. Este comportamento garante ao CoopNet a menor variância em relação ao atraso. De qualquer forma, mesmo com uma variância maior, as barras de erro da Figura 5.4 nos mostram que na média esta variância possui um valor pequeno, e portanto na média o TVoIP possui os melhores resultados em relação à penalidade relativa ao atraso.

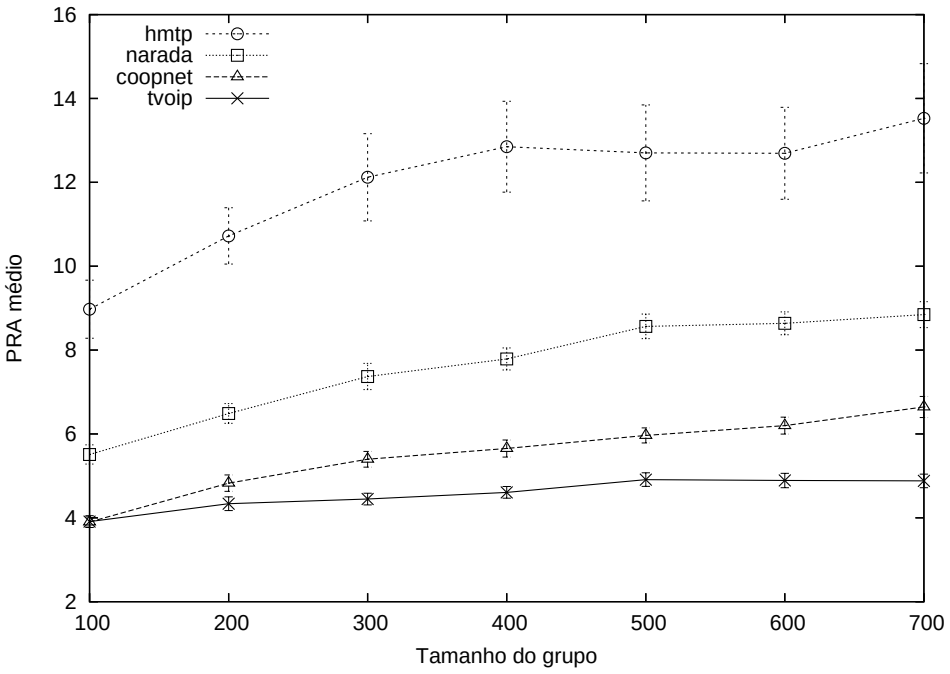
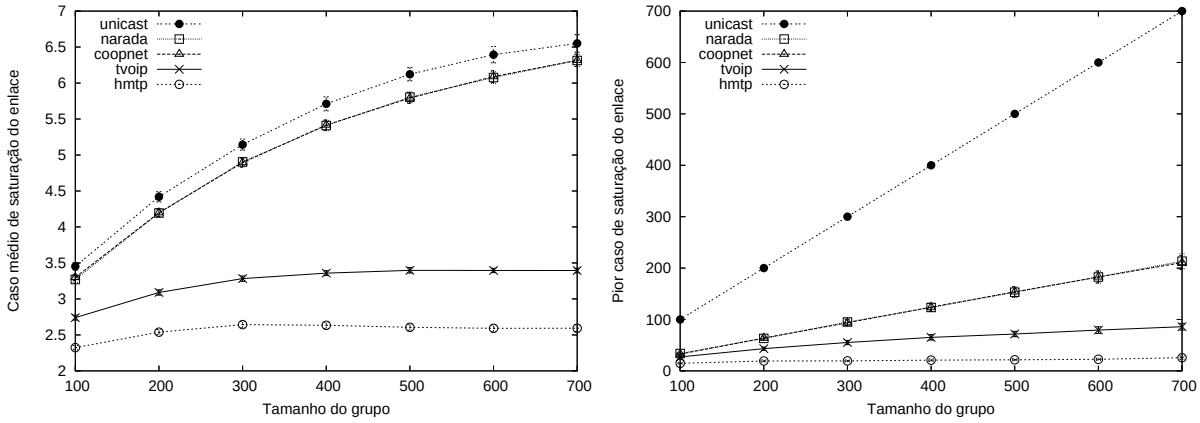


Figura 5.4: Valor médio da PRA para diferentes tamanhos de grupos *multicast*.

A Figura 5.4 ilustra de forma mais clara como a penalidade relativa ao atraso (PRA) cresce com o aumento de clientes no grupo *multicast*. Cada ponto no gráfico representa o valor médio da PRA para um determinado tamanho de grupo com sua respectiva barra de erro. Observe que o coeficiente angular da curva do TVoIP possui um valor muito baixo (próximo de zero), o que sugere que o tamanho do grupo *multicast* não influencia na PRA para nossa arquitetura.

As Figuras 5.5(a) e 5.5(b) apresentam respectivamente os resultados do caso médio e do pior caso de saturação nos enlaces. Conforme esperado, o protocolo HMTP possui os melhores resultados, uma vez que procura conectar vizinhos próximos da rede física



(a) Caso médio

(b) Pior caso

Figura 5.5: Valores de saturação de enlaces para diferentes tamanhos de grupos *multicast*.

na rede sobreposta. O TVoIP também possui esta propriedade, visto que tenta agrupar nós de acordo com os endereços de rede. No entanto, o TVoIP prioriza a propriedade de construir uma árvore baixa antes de agrupar os endereços de rede, o que faz com que os nós com maior grau de saída da árvore sempre se localizem próximos da raiz, mesmo em situações onde poderiam estar melhor posicionados na árvore. Esta característica fornece ao TVoIP o segundo melhor resultado. As arquiteturas Narada e CoopNet possuem valores de saturação similares, visto que ambas utilizam um processo de escolha aleatória do nó que irá atuar como pai do novo nó que deseja juntar-se à rede. A curva *unicast* possui os piores resultados, uma vez que os enlaces próximos ao emissor possuem maior probabilidade de saturação. No entanto, o valor do pior caso não influencia no valor médio da curva, como pode ser observado nas barras de erro da Figura 5.5(a).

A Figura 5.6 ilustra a influência do tamanho do grupo *multicast* no uso de recursos da rede. Podemos observar que o protocolo HMTP consome menos recursos de rede que os outros protocolos, consumindo em torno de 2,3 vezes mais recursos que o IP *multicast* e cerca de 42% menos recursos que o TVoIP. Isto pode ser explicado pelo fato do URN considerar a saturação dos enlaces no cálculo (Seção 5.1). Assim, podemos observar as mesmas características encontradas nos gráficos de saturação: o protocolo HMTP possui os melhores resultados, seguido pelos protocolos TVoIP, CoopNet, Narada e *unicast* que possui os piores resultados.

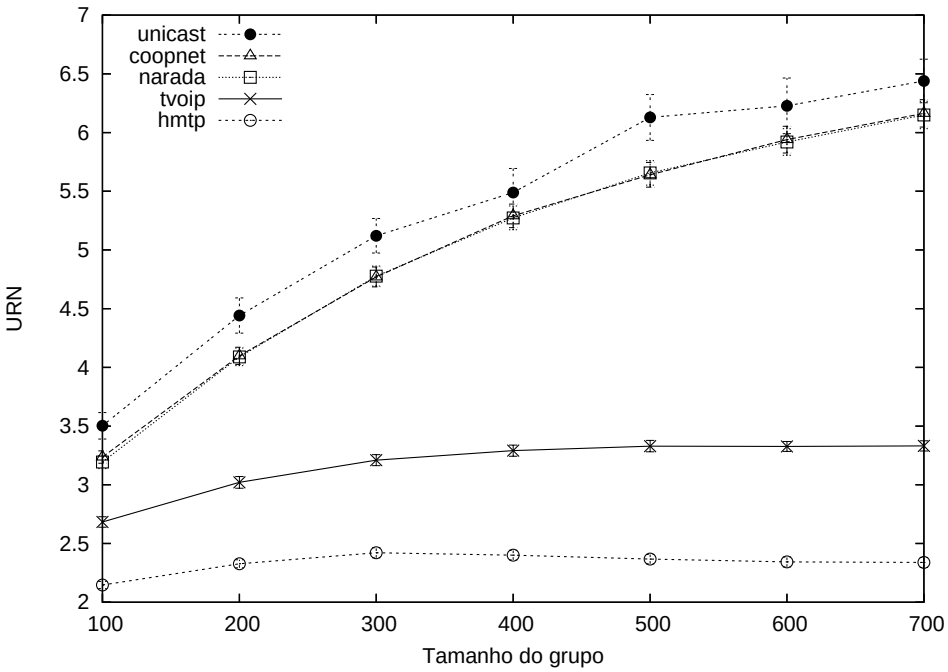


Figura 5.6: Valores de URN para diferentes tamanhos de grupos *multicast*.

5.3.2 Desempenho nos Nós Finais de Rede

Esta seção tem como objetivo avaliar o desempenho nos nós finais de acordo com as características percebidas pelos usuários tais como atraso, *jitter*, perda de pacotes, tempo de entrada no sistema e carga em relação à largura de banda disponível.

A Figura 5.7 ilustra a influência da taxa de transmissão de vídeo na perda de pacotes. Note que com exceção do *unicast*, a porcentagem de perdas para todos os protocolos começa a se tornar significativa quando a taxa de transmissão possui valores superiores a 512 kbps. Como esperado, os caminhos *unicast* possuem o maior valor de perda de pacotes. Este comportamento é justificado pela maior probabilidade de saturação nos enlaces próximos ao emissor, o que implica em maior probabilidade de congestionamento nestes enlaces. Independentemente da taxa de transmissão de vídeo, o protocolo HMTP tem um desempenho notavelmente melhor do que os outros algoritmos, chegando a ter 36% menos perdas com uma taxa de transmissão de vídeo de 2Mbps que o TVoIP. Devido à característica de não construir uma topologia muito dispersa, o protocolo HMTP consegue diminuir o tráfego entre domínios, minimizando desta forma o congestionamento nas filas dos roteadores dos enlaces de acesso. Mais uma vez, o TVoIP apresentou o segundo melhor resultado, visto que prioriza a propriedade de manter a árvore baixa para depois realizar o agrupamento do nós. A arquitetura CoopNet possui a característica de preencher a

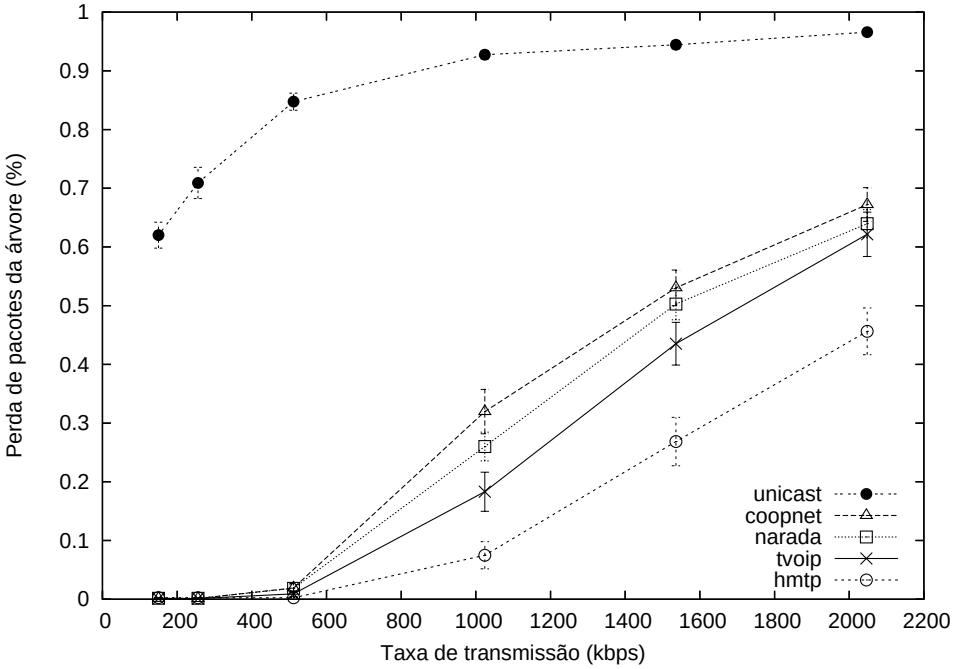


Figura 5.7: Relação entre a taxa de transmissão de vídeo e a perda de pacotes para um grupo de 100 usuários.

árvore de distribuição por nível, o que implica que um nível precisa estar completamente cheio para outro ser preenchido. Assim, conforme pode ser visto na Figura 5.8, os nós que utilizam o protocolo CoopNet experimentam a maior carga, o que implica em maior probabilidade de congestionamento nos enlaces e resulta no desempenho não satisfatório mostrado na Figura 5.7.

A Figura 5.8 ilustra a influência do tamanho do grupo *multicast* na carga média da árvore de distribuição. Observe que o TVoIP possui a menor carga na árvore de distribuição, com valores de aproximadamente 75%. Este comportamento já era esperado, devido a característica do protocolo de sempre priorizar a propriedade de manter a árvore baixa, colocando os nós com maior grau de saída próximos à raiz da árvore. A solução *unicast* apresenta os piores resultados, com valores constantes de 100%, pois o *unicast* sobrecarrega o emissor com um número de conexões igual ao tamanho do grupo *multicast*. A arquitetura CoopNet, conforme discutido anteriormente, apresenta um resultado próximo de 93% de carga, pois preenche a árvore de distribuição por nível. Além disso, podemos observar que para todas as curvas da figura o coeficiente angular é muito baixo, indicando que o tamanho do grupo *multicast* não influencia na carga média de distribuição.

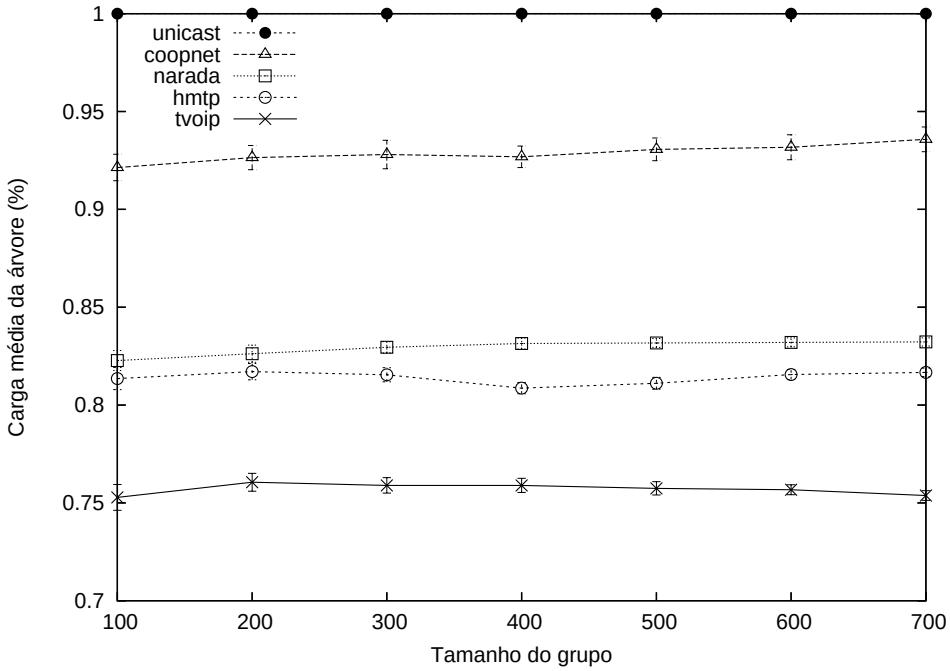


Figura 5.8: Relação entre o tamanho do grupo *multicast* e a carga média da árvore de distribuição para uma taxa de transmissão de vídeo de 150 kbps.

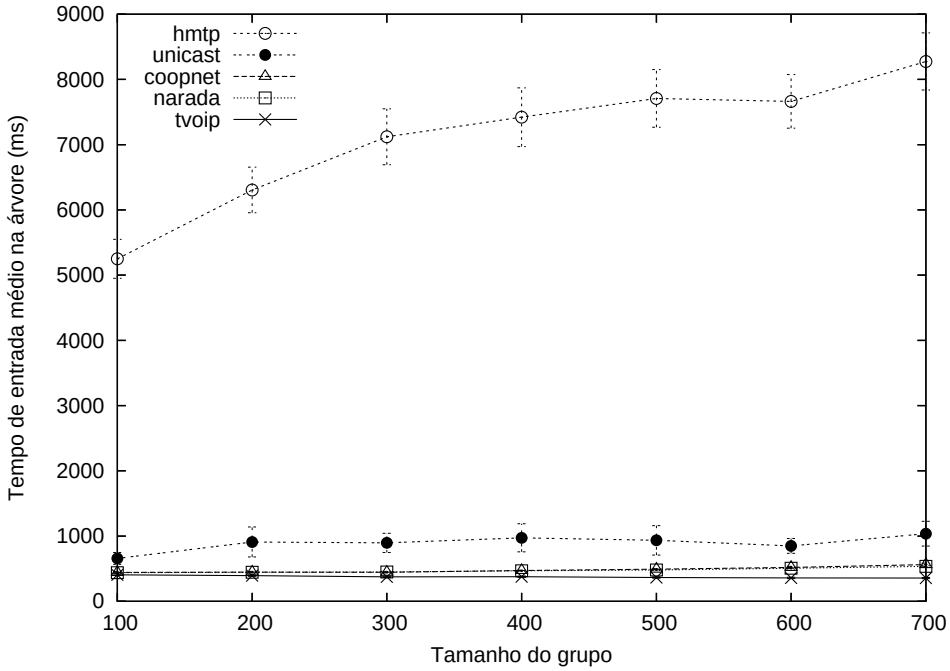


Figura 5.9: Relação entre o tamanho do grupo *multicast* e o tempo de entrada na árvore de distribuição para uma taxa de transmissão de vídeo de 150 kbps.

A Figura 5.9 ilustra a influência do tamanho do grupo no tempo de entrada na árvore de distribuição. Podemos observar que o TVoIP possui o menor tempo de entrada na árvore, com valores variando de 350 ms a 400 ms. Além disso possui um tempo 50% melhor que o protocolo Narada, 57% que o protocolo CoopNet e 2218% que o protocolo HMTP para um grupo com 700 clientes. O pior desempenho do protocolo HMTP pode ser explicado através do seu processo de entrada na árvore de distribuição *multicast* , pois cada nó desce a árvore realizando medições de atraso até encontrar um bom nó para se conectar. Isto faz com que o tempo de entrada na árvore seja muito demorado, e que cresça de acordo com o tamanho do grupo *multicast*. Para se ter uma idéia, com um grupo de 700 nós, o tempo de espera médio de um nó foi de 8,27 s, o que é considerado um valor extremamente alto. Para os outros protocolos, as curvas possuem coeficiente angular muito baixo, indicando que o tamanho do grupo *multicast* não influencia o tempo de entrada médio da árvore.

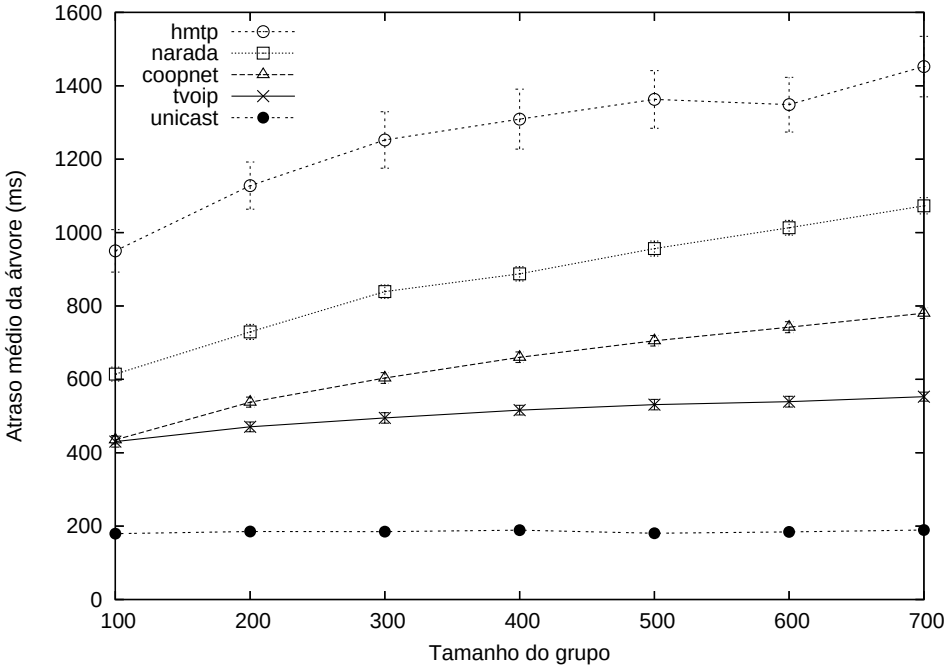


Figura 5.10: Relação entre o tamanho do grupo *multicast* e o atraso médio da árvore de distribuição para uma taxa de transmissão de vídeo de 150 kbps.

A Figura 5.10 apresenta a influência do tamanho do grupo *multicast* no atraso médio da árvore de distribuição. Podemos observar que os caminhos *unicast* possuem os melhores resultados para o atraso, gerando uma curva quase que constante para diferentes números de clientes no grupo *multicast*. Embora o *unicast* forneça o menor atraso entre um par de nós, esta curva foi privilegiada devido à configuração de tamanho de fila de 100 pacotes no *buffer* dos roteadores do ns-2. Isto explica os altos valores de perdas de pacotes

para o *unicast* na Figura 5.7, e os baixos valores para o atraso na Figura 5.10, pois são considerados somente os atrasos dos pacotes que conseguem chegar ao nó destino. Talvez os resultados pudessem ser diferentes com um tamanho de *buffer* maior, pois diminuiria o número de pacotes perdidos pelo *unicast* e conseqüentemente aumentaria o tempo de espera destes pacotes na fila, aumentando conseqüentemente o atraso. Entre os protocolos de *multicast* na camada de aplicação, o TVoIP apresentou os melhores resultados, devido à já mencionada propriedade de manter a árvore de distribuição baixa e ao fato de agrupar nós próximos de acordo com os endereços de rede. Observe que o coeficiente angular da curva do TVoIP possui um valor baixo, o que indica que o tamanho do grupo *multicast* influencia pouco no atraso para este protocolo. Isto mostra que a árvore do TVoIP cresce aumentando o número de ramificações, sugerindo uma árvore baixa e larga. Observe que este comportamento é desejado, visto que minimiza o atraso fim-a-fim entre emissor e receptor. O CoopNet por também possuir a característica de manter a árvore baixa, apresenta um bom resultado. O protocolo HMTP apresenta o pior resultado, pois não leva em consideração o grau de saída de um nó para construção da rede sobreposta, gerando árvores de distribuição altas. Com isto podemos concluir que árvores *multicast* baixas nos fornecem melhores resultados em relação ao atraso.

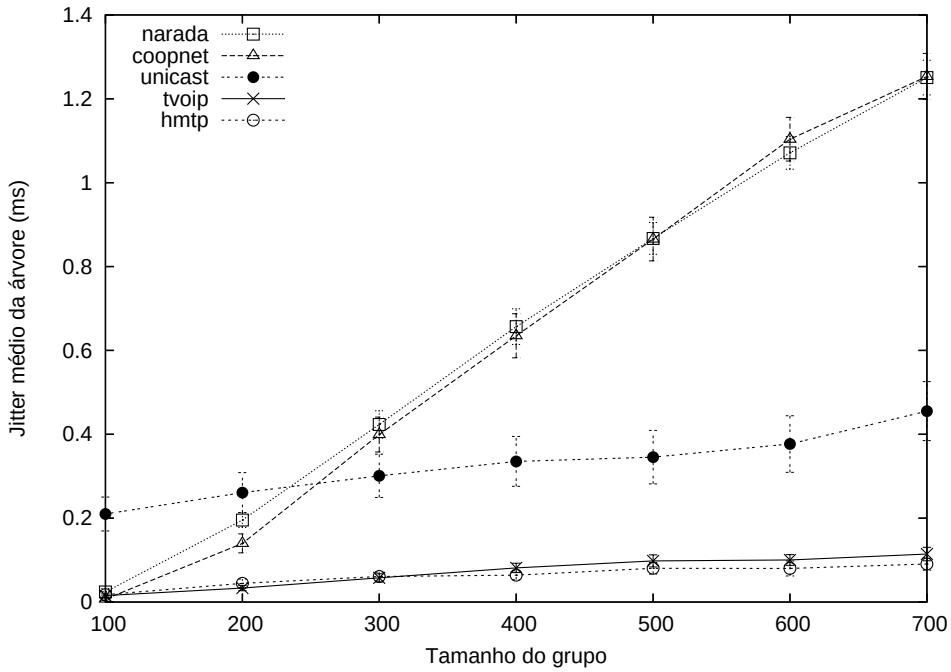


Figura 5.11: Relação entre o tamanho do grupo *multicast* e o *jitter* médio da árvore de distribuição para uma taxa de transmissão de vídeo de 150 kbps.

A Figura 5.11 apresenta a influência do tamanho do grupo *multicast* no *jitter* médio da árvore de distribuição. Para um grupo com até 300 clientes, o protocolo TVoIP apresentou

os melhores resultados, e para grupos com um número de clientes acima desse valor, o uso do protocolo HMTP é mais indicado. No entanto os valores de *jitter* para ambos protocolos são muito baixos, na ordem de 10^{-1} ms. As arquiteturas Narada e CoopNet possuem os piores valores, apresentando curvas com coeficientes angulares altos. Isto indica que o *jitter* nestes protocolos é afetado pelo número de clientes no grupo. Isto pode ser explicado pela característica que ambos protocolos possuem de realizar um processo de escolha aleatória do nó que irá atuar como pai do novo nó que deseja juntar-se à rede. Com isto, os nós não são agrupados próximos, o que faz com que os pacotes trafeguem por caminhos maiores, o que aumenta a chance dos pacotes passarem por enlaces congestionados e conseqüentemente sofrerem maior variação no atraso.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho apresentou uma arquitetura escalável para transmissão de vídeo ao vivo na Internet utilizando redes *peer-to-peer* para comunicação multiponto. Foi considerado como caso de uso um cenário de Internet TV no qual existe somente um emissor transmitindo conteúdo para um grande número de receptores. Foram descritos mecanismos de entrada e saída de nós do sistema em face do ambiente altamente dinâmico encontrado em aplicações P2P, mecanismos para otimização da rede sobreposta e alguns aspectos sobre a implantação da arquitetura proposta na Internet atual.

A principal contribuição deste trabalho foi o algoritmo de entrada de nós que constrói a rede sobreposta P2P. Este algoritmo utiliza o endereço de rede para cálculo da métrica de distância e sempre mantém os nós com maior grau de saída próximos à raiz da árvore. Além disso, evita sempre que possível a troca de mensagens entre os nós, para que o protocolo da nossa arquitetura tenha um baixo *overhead*. Esta característica contribui para a escalabilidade do sistema.

Através de simulações foi possível avaliar e comparar o desempenho do TVoIP com outras propostas bem conhecidas da literatura. Os resultados mostram que nossa arquitetura consegue construir redes sobrepostas com caminhos que fornecem o menor atraso em relação às outras propostas. Além disso, foi verificado que o tamanho do grupo *multicast* não influencia na penalidade relativa ao atraso para nossa arquitetura (Figura 5.4).

Um dos grandes diferenciais do sistema de TV digital é a possibilidade de interação com o usuário. Seria inviável considerar a adição deste tipo de funcionalidade interativa [2] em aplicações de Internet TV se o atraso possuir um valor muito elevado. O fato de nossa arquitetura fornecer o menor atraso fim-a-fim entre emissor e receptor, conforme apresentado nas Figuras 5.1 e 5.4, possibilita o suporte a aplicações interativas em nosso trabalho.

Outros experimentos mostram que nossa arquitetura consegue bons resultados para o uso de recursos da rede e saturação dos enlaces (Figuras 5.5 e 5.6). Sobre a carga média da árvore, foi verificado que o TVoIP fornece os melhores resultados (Figura 5.8), o que sugere uma boa distribuição da carga entre os nós do sistema. Isto contribui para um ambiente cooperativo mais justo.

A nossa arquitetura apresentou bons resultados para a perda de pacotes e para o *jitter*, conforme apresentado nas Figuras 5.7 e 5.11. Os baixos valores para o *jitter* sugerem que o TVoIP fornece um boa continuidade no vídeo recebido pelos usuários. Em relação a perdas de pacotes, nossa arquitetura apresentou resultados inferiores apenas para o protocolo HMTP. Isto porque o TVoIP prioriza a propriedade de manter a árvore baixa para depois realizar o agrupamento do nós, o que não ocorre com o HMTP. No entanto, aplicações multimídia são tolerantes a perdas ocasionais, que podem causar ruídos, muitas vezes imperceptíveis no processo de reprodução.

Uma característica muito importante no projeto de um sistema com requisitos de escalabilidade é o tempo de entrada de um nó no sistema. Se o tempo de entrada cresce com o tamanho do grupo *multicast*, a aplicação pode se tornar inviável para grupos com um grande número de usuários. Os experimentos nos mostram que este é um ponto forte de nossa arquitetura (Figura 5.9), pois a mesma apresenta os melhores resultados quando comparada às arquiteturas de outros trabalhos.

O foco deste trabalho se manteve no processo de inclusão dos nós na árvore de distribuição de vídeo para construção da rede sobreposta. No caso da saída de um nó, o nó de inicialização pode executar uma nova operação de entrada no sistema nos filhos deste nó ou pode procurar um nó na sub-árvore do nó com capacidade suficiente para substituí-lo. Será avaliado em um próximo trabalho o tempo de reparo destas duas possibilidades.

Uma re-estruturação periódica da árvore de distribuição possibilita maior eficiência na distribuição dos fluxos no sistema. Discutimos um processo de refinamento da árvore de distribuição no Capítulo 3. No entanto não realizamos nenhum experimento para verificar o número médio de iterações para a árvore convergir para uma estrutura estável e nem definimos um limiar para o ganho de proximidade na troca de posição. Esperamos realizar estes experimentos em um trabalho futuro.

O uso de NATs e *firewalls* na Internet atual impõe restrições fundamentais sobre a conectividade de nós na rede sobreposta. Realizamos algumas considerações no Capítulo 3 sobre a implantação da arquitetura considerando estas limitações.

A distribuição de vídeo em redes P2P apresenta outros desafios. Uma vez que assumimos que cada cliente confia em um valor de capacidade de saída especificado pelo usuário (Seção 3.1.1), podemos ter dificuldades se o comportamento do usuário for egoísta. Normalmente usuários egoístas procuram tirar o máximo de proveito dos recursos da rede, cooperando o mínimo na distribuição dos fluxos. Um outro problema é o curto tempo de permanência dos mesmos no sistema, o que demanda esforços para se contornar as interrupções causadas nas entregas dos fluxos quando as desconexões ocorrem. Uma possível solução para estes desafios é o uso de mecanismos de incentivo que estimulem os clientes a cooperarem e também a permanecerem conectados durante intervalos de tempo maiores, de forma a reduzir os problema de desconexão. Os trabalhos [31, 50] descrevem vários mecanismos de incentivo para uso em redes P2P, tais como padrões baseados em confiança e padrões baseados em comércio. Esperamos incluir um mecanismo de incentivo para resolver estas limitações em um trabalho futuro.

Outras contribuições do trabalho incluem o desenvolvimento de um simulador para realizar o cálculo dos custos da árvore de distribuição de nossa arquitetura e um *framework* para simulação de redes P2P no simulador de redes ns-2. As implementações foram conduzidas com o cuidado de prover códigos reusáveis e eficientes, baseados na linguagem de programação C++ e no projeto orientado a objetos. Dessa forma, é disponibilizado para o domínio público um código bastante flexível. Salienta-se que nenhum algoritmo foi implementado através de bibliotecas de código fechado.

Uma outra contribuição deste trabalho é a possiblidade de estender algumas das arquiteturas encontradas na literatura. O nosso nó de inicialização pode ser incorporado facilmente nas arquiteturas CoopNet, Narada e HMTP. Considere como exemplo o protocolo HMTP. Em vez de seu nó de inicialização retornar sempre a raiz da árvore, o nó de inicialização poderia retornar diretamente um bom pai na topologia. Isto não iria interferir no seu processo de otimização e nem no algoritmo de reparo da árvore, e iria fornecer um tempo de entrada constante ao nós que desejam entrar no sistema, como acontece em nossa arquitetura. O mesmo se aplica às arquiteturas Narada e CoopNet, uma vez que escolhem um nó aleatório na topologia para usá-lo como nó pai.

Assim, podemos concluir com este trabalho que o uso de um nó de inicialização que forneça uma boa localização de entrada para um novo nó que deseja juntar-se a um grupo *multicast* fornece uma melhora significativa de desempenho na rede sobreposta das arquiteturas de *multicast* em nível de aplicação, sejam elas de natureza centralizada ou distribuída.

Referências Bibliográficas

[1] Abilene network. Internet2 netflow: Weekly reports. `http://netflow.internet2.edu/weekly/`.

[2] C. Albuquerque, T. Proença e E. Oliveira. *TV sobre IP - Arquiteturas para Transmissão em Larga Escala*, capítulo 3. Minicurso SBRC 2006, Maio 2006.

[3] S. Androutsellis-Theotokis e D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

[4] J. Apostolopoulos, T. Wong, S. Wee e D. Tan. On multiple description streaming with content delivery networks. In *Proceedings of INFOCOM*, pp. 1736–1745, Junho 2002.

[5] Apple. Apple movie trailers. `http://www.apple.com/trailers/`.

[6] S. Banerjee e B. Bhattacharjee. A comparative study of application layer multicast protocols. Submetido para revisão, 2002.

[7] S. Banerjee, B. Bhattacharjee e C. Kommareddy. Scalable application layer multicast. In *Proceedings of the Special Interest Group on Data Communication (SIGCOMM)*, pp. 205–217, New York, NY, USA, 2002. ACM Press.

[8] A. Begen e Y. Altunbasak. Timely inference of late/lost packets in real-time streaming applications. In *Picture Coding Symposium (PCS)*, Dezembro 2004.

[9] T. Bu e D. Towsley. On distinguishing between internet power law topology generators. In *Proceedings of INFOCOM*, volume 2, pp. 638–647, Junho 2002.

[10] M. Castro, P. Druschel, A. Kermarrec e A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1489–1499, 2002.

[11] Y. Chawathe. Scattercast: an adaptable broadcast distribution framework. *Multimedia Systems*, 9(1):104–118, 2003.

[12] Y. Chu, S. Rao e H. Zhang. A case for end system multicast. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 1–12, New York, NY, USA, 2000. ACM Press.

[13] CNN. CNN. `http://www.cnn.com`.

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

[15] Democracy. Democracy. `http://www.getdemocracy.com/`.

[16] A. El-Sayed, V. Roca e L. Mathy. A survey of proposals for an alternative group communication service. *IEEE Network*, 17(1):46–51, Janeiro 2003.

[17] K. Fall e K. Varadhan. The ns manual (formerly ns notes and documentation), 2006. `http://www.isi.edu/nsnam/ns/doc/index.html`.

[18] ForexTV. ForexTV. `http://www.forextv.com`.

[19] P. Francis, Y. Pryadkin, P. Radoslavov, R. Govindan e B. Lindell. Yoid: Your own internet distribution, Abril 2000. `http://www.aciri.org/yoid/`.

[20] Globo. Globo.com. `http://www.globo.com`.

[21] Gnutella. `http://www.gnutella.com/`.

[22] Google. Google video. `http://www.google.com`.

[23] Q. He, M. Ammar, G. Riley, H. Raj e R. Fujimoto. Mapping peer behavior to packet-level details: A framework for packet-level simulation of peer-to-peer systems. In *Proceedings of MASCOTS*, pp. 71–78, 2003.

[24] D. Helder e S. Jamin. End-host multicast communication using switch-trees protocols. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, pp. 419, Washington, DC, USA, 2002. IEEE Computer Society.

[25] M. Jain e C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Transactions on Networking (TON)*, 11(4):537–549, 2003.

[26] J. Janotti, D. Gifford, K. Johnson, M. Kaashoek e J. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, pp. 197–212, Outubro 2000.

[27] J. Kurose e K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, 3 edição, 2004.

[28] K. Lai e M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *Proceedings of the Special Interest Group on Data Communication (SIGCOMM)*, pp. 283–294, 2000.

[29] Z. Li e P. Mohapatra. Hostcast: A new overlay multicasting protocol. In *Proceedings of IEEE International Communications Conference (ICC)*, volume 1, pp. 702–706, 2003.

[30] Lime Wire LLC. Limewire: The fastest P2P file sharing program on the planet. `http://www.limewire.com/`.

[31] D. Manzato. Mecanismos de incentivo à cooperação em redes peer-to-peer para distribuição de fluxos de vídeo. Dissertação de Mestrado, Universidade Estadual de Campinas (UNICAMP), Fevereiro 2006.

[32] L. Mathy, R. Canonico e D. Hutchison. An overlay tree building control protocol. *Lecture Notes in Computer Science*, 2233:76–87, 2001.

[33] S. McCanne e S. Floyd. The Network Simulator – ns-2. `http://www.isi.edu/nsnam/ns/`.

[34] A. Medina, A. Lakhina, I. Matta e J. Byers. BRITE: Universal topology generation from a user's perspective. Relatório Técnico 2001-003, Boston University, 1 2001.

[35] MTV. Music television. `http://www.mtv.com`.

[36] Musiclab LLC. Bearshare: The power to share. `http://www.bearshare.com/`.

[37] V. Padmanabhan, H. Wang, P. Chou e K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, pp. 177–186, New York, NY, USA, 2002. ACM Press.

[38] D. Pendarakis, S. Shi, D. Verma e M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, pp. 49–60, 2001.

[39] J. Postel. Transmission Control Protocol. RFC-793, IETF, 1981.

[40] S. Ratnasamy, P. Francis, M. Handley, R. Karp e S. Schenker. A scalable content-addressable network. In *Proceedings of the Special Interest Group on Data Communication (SIGCOMM)*, pp. 161–172, New York, NY, USA, 2001. ACM Press.

[41] S. Ratnasamy, M. Handley, R. Karp e S. Shenker. Application-level multicast using content-addressable networks. *Networked Group Communications*, pp. 14–29, 2001.

[42] M. Ripeanu, I. Foster e A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *Internet Computing Journal*, 6(1), Janeiro 2002.

[43] J. Rocha, M. Domingues, A. Callado, E. Souto, G. Silvestre, C. Kamienski e D. Sadok. *Peer-to-Peer: Computação Colaborativa na Internet*, capítulo 1, pp. 3–46. Minicurso SBRC 2004, Maio 2004.

[44] A. Rowstron e P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[45] S. Saroiu, P. Gummadi e S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN)*, Janeiro 2002.

[46] S. Sen e J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW)*, pp. 137–150, New York, NY, USA, 2002. ACM Press.

[47] A. Stavrou, D. Rubenstein e S. Sahu. A lightweight, robust P2P system to handle flash crowds. *Proceedings of the Special Interest Group on Data Communication (SIGCOMM)*, 32(3):17–17, 2002.

[48] S. Tan, A. Waters e J. Crawford. Meshtree: A Delay optimised Overlay Multicast Tree Building Protocol. Relatório Técnico 5-05, University of Kent, Abril 2005.

[49] A. Tanenbaum e M. Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[50] Y. Tang, H. Wang e W. Dou. Trust based incentive in P2P network. In *Procedings of the EEE International Conference on E-Commerce Technology for Dynamic E-Business*, pp. 302–305, 2004.

[51] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker e W. Willinger. Network topology generators: degree-based vs. structural. In *Proceedings of the Special Interest Group on Data Communication (SIGCOMM)*, pp. 147–159, New York, NY, USA, 2002. ACM Press.

[52] D. Tran, K. Hua e T. Do. Scalable media streaming in large peer-to-peer networks. In *Proceedings of the tenth ACM international conference on Multimedia (MULTIMEDIA)*, pp. 247–250, New York, NY, USA, 2002. ACM Press.

[53] D. Tran, K. Hua e T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *INFOCOM*, pp. 1283–1292, 2003.

[54] Universidade Federal Fluminense. Página do grupo de redes da UFF. `http://tvd.ic.uff.br/`.

[55] D. Waitzman, C. Partridge e S. Deering. Distance vector multicast routing protocol. RFC 1075, Novembro 1988.

[56] W. Wang, D. Helder, S. Jamin e L. Zhang. Overlay optimizations for end-host multicast. In *Proceedings of the Int'l Workshop on Networked Group Communication (NGC)*, Outubro 2002.

[57] W. Wang, C. Jin e S. Jamin. Network overlay construction under limited end-to-end addressability. Relatório técnico cse-tr-489-04, EECS Department, University of Michigan, 2004.

[58] Yahoo. Yahoo movie trailers. `http://movies.yahoo.com/trailers/`.

[59] Yahoo. Yahoo music videos. `http://music.yahoo.com/musicvideos/`.

[60] YouTube. YouTube. `http://www.youtube.com`.

[61] E. Zegura, K. Calvert e S. Bhattacharjee. How to model an internetwork. In *Proceedings of INFOCOM*, volume 2, pp. 594–602, San Francisco, CA, Maio 1996.

[62] B. Zhang, S. Jamin e L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proceedings of INFOCOM*, pp. 1366–1375, Junho 2002.

[63] B. Zhao, J. Kubiatowicz e A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Relatório Técnico UCB/CSD-01-1141, UC Berkeley, Abril 2001.

[64] S. Zhuang, B. Zhao, A. Joseph, R. Katz e J. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, pp. 11–20, New York, NY, USA, 2001. ACM Press.