Capítulo 3

Distribuição de Vídeo sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios

Igor M. Moraes[1], Miguel Elias M. Campista[1], Marcelo D. D. Moreira[1], Marcelo G. Rubinstein[2], Luís Henrique M. K. Costa[1] e Otto Carlos M. B. Duarte[1]

[1]Grupo de Teleinformática e Automação
COPPE/Poli - Universidade Federal do Rio de Janeiro

[2]PEL/DETEL - FEN - Universidade do Estado do Rio de Janeiro

Abstract

The main characteristic of peer-to-peer networks is scalability. In such networks, participating nodes work in a collaborative fashion and share their resources. Thus, the larger the number of participating nodes, the higher the network capacity. In addition, changes in network infrastructure are not required, because nodes themselves construct the communication structure. These characteristics are fundamental to the development of large-scale video streaming and thus peer-to-peer networks become a promising solution for this application in the Internet. This chapter presents the most used techniques, architectures and mechanisms for video streaming over peer-to-peer networks. Examples of peer-to-peer video systems are characterized and their advantages and problems are discussed. Finally, the open challenges are presented to motivate the research in this area.

Resumo

As redes par-a-par têm como principal característica a escalabilidade. Nessas redes, os nós participantes trabalham de forma colaborativa e compartilham seus recursos. Com isso, quanto maior o número de participantes, maior é a capacidade da rede. Além disso, a infra-estrutura de rede não precisa ser modificada, pois os próprios nós constroem a estrutura de comunicação. Essas características são fundamentais para o desenvolvimento da distribuição de fluxo contínuo de vídeo em larga escala e, por isso, fazem das redes par-a-par uma solução promissora para o desenvolvimento dessa aplicação na Internet. Este capítulo apresenta as técnicas, as arquiteturas e os mecanismos mais usados para a distribuição de vídeo sobre redes par-a-par. Exemplos de sistemas de vídeo par-a-par são caracterizados e suas virtudes e problemas são discutidos. Por fim, são apresentados os desafios em aberto com o intuito de motivar o estudo na área.

3.1. Introdução

A distribuição de vídeo é uma das aplicações de maior sucesso atualmente na Internet. Sítios que oferecem serviços de compartilhamento e distribuição de vídeos sob demanda são acessados por milhões de usuários diariamente. Somente o YouTube, o sítio mais famoso desse gênero, é acessado por cerca de 20 milhões de usuários a cada dia, o que resulta em um tempo de reprodução de aproximadamente 10 mil anos, se a duração de cada vídeo reproduzido durante os acessos diários for somada [Huang et al. 2007a]. Além disso, emissoras de TV de todo o mundo já possuem serviços de difusão de TV na Internet. Nos Estados Unidos, a CBS já atingiu a marca de 260 mil usuários simultâneos durante a transmissão das finais da liga universitária de basquete em março de 2006 [Liu et al. 2008]. No Brasil, o portal Globo.com atingiu a marca de 72 mil usuários simultâneos durante a transmissão da semifinal entre Portugal e França na Copa do Mundo de 2006 [IDG Now! 2006].

Nesse capítulo, a expressão reduzida "distribuição de vídeo" é usada para se referir à distribuição de fluxo contínuo de vídeo (*video streaming*). Na distribuição de fluxo contínuo, um usuário não precisa receber e armazenar todo o conteúdo de vídeo para exibi-lo. O vídeo é enviado em um fluxo contínuo e o seu conteúdo é exibido ao ser recebido por um usuário. As aplicações de distribuição de vídeo na Internet podem ser classificadas em aplicações de vídeo sob demanda e aplicações de difusão. No vídeo sob demanda, um usuário seleciona o conteúdo que deseja receber e pode controlar a sua reprodução, como em um videocassete. O controle das ações de reprodução pode ser feito localmente pelo usuário, uma vez que o conteúdo recebido também é armazenado, ou usando um protocolo, por exemplo, o RTSP (*Real-Time Streaming Protocol*), para enviar mensagens à fonte. Na difusão [1], uma fonte envia o mesmo conteúdo de vídeo para um grupo de usuários, como ocorre em uma transmissão de TV. Um usuário recebe e reproduz o vídeo a partir do instante de sua requisição e não tem controle sobre as ações de reprodução. A difusão de vídeo é o foco desse capítulo.

Tanto no vídeo sob demanda quanto na difusão , os maiores desafios são a escalabilidade e a garantia dos requisitos de qualidade de serviço das aplicações. Atualmente, as aplicações disponibilizam vídeos da ordem de centenas de kilobits por segundo e o número de usuários simultâneos é da ordem de centenas de milhares. Com isso, os recursos de banda passante exigidos de um provedor de serviço são da ordem de centenas de gigabits por segundo, considerando o uso do modelo cliente-servidor e da comunicação ponto-a-ponto (*unicast*). Estima-se que os custos mensais do YouTube com banda passante sejam superiores a um milhão de dólares [Huang et al. 2007a]. Portanto, no modelo cliente-servidor, quanto mais usuários e quanto maior for a qualidade do vídeo, maiores serão os custos dos provedores, o que torna esse modelo inadequado para a distribuição de vídeo na Internet.

Diversas alternativas ao modelo cliente-servidor foram propostas. O IP Multicast implementa a comunicação de grupo na camada de rede. Porém, a implantação desse

[1] Nesse caso, o termo difusão (*broadcast*) é usado como sinônimo de comunicação multidestinatária (*multicast*) para fazer a associação com a difusão convencional de rádio e TV. Formalmente, na difusão um mesmo conteúdo é enviado para todos os destinatários, enquanto que na comunicação multidestinatária um mesmo conteúdo é enviado apenas para um grupo de destinatários.

serviço na Internet é lenta, em virtude da complexidade para configurar e gerenciar o conjunto de protocolos de rede necessários à sua implementação. Outra alternativa são as redes de distribuição de conteúdo (*Content Distribution Networks* - CDNs) que possuem servidores espalhados geograficamente em diferentes *backbones*. A idéia básica das CDNs é mover o conteúdo requisitado por um dado cliente do servidor de origem para servidores mais próximos desse cliente e, com isso, entregar o conteúdo com maior taxa de transferência e menor atraso. Entretanto, a eficiência das CDNs depende do número de servidores disponíveis. Quanto mais servidores, maior a eficiência e maiores são os custos.

Uma das soluções mais promissoras atualmente é a distribuição de vídeo sobre redes par-a-par (*peer-to-peer* - P2P) [Liu et al. 2008, Sentinelli et al. 2007, Rejaie 2006]. Essa afirmação pode ser comprovada pelo número de sistemas de vídeo par-a-par encontrados comercialmente e propostos na literatura, bem como pelo número de usuários desses sistemas. Para avaliar o potencial desse tipo de sistema, a TV estatal chinesa adotou o GridMedia [Zhang et al. 2005b], um sistema de vídeo par-a-par comercial, para transmitir ao vivo a edição de 2006 do Festival da Primavera Chinês. Estima-se que aproximadamente dois milhões de usuários assistiram a essa transmissão e que o sistema suportou mais de 200 mil usuários simultâneos [Tang et al. 2007].

Define-se um sistema de distribuição de vídeo par-a-par como um conjunto de aplicativos e mecanismos responsável pela construção da arquitetura de distribuição e pelo encaminhamento do vídeo. Tais sistemas não exigem modificações no núcleo da rede, são escaláveis e têm custo reduzido. Nos sistemas de distribuição de vídeo par-a-par, os próprios nós participantes constroem a arquitetura de distribuição na camada de aplicação e, por isso, não exigem modificações nos roteadores da rede. Tal característica facilita a implantação desses sistemas. A escalabilidade e o baixo custo são garantidos pela cooperação entre os nós. Cada nó ao receber um vídeo pode encaminhar se solicitado esse mesmo vídeo para os demais nós pertencentes à arquitetura de distribuição e, conseqüentemente, compartilha seus recursos, como banda passante, espaço de armazenamento e processamento. Dessa forma, quanto mais nós pertencerem ao sistema de distribuição, maior será a capacidade total do sistema.

Apesar dos sistemas par-a-par para compartilhamento de arquivo serem um enorme sucesso há anos, os sistemas de distribuição de vídeo introduzem novos desafios. Os sistemas de compartilhamento de arquivos trabalham com longas transferências de dados sem restrições de tempo. Por sua vez, os sistemas de vídeo possuem requisitos estritos de banda passante e tempo para entregar o conteúdo que está sendo distribuído. Tais requisitos não são completamente atendidos pelas redes par-a-par, em virtude da sua natureza não-estruturada. Por isso, é necessário adaptar as técnicas e algoritmos de codificação, roteamento e escalonamento para esse ambiente altamente dinâmico e, geralmente, sem um ponto central de coordenação.

O objetivo principal deste capítulo é apresentar as diferentes características e os desafios da distribuição de vídeo em difusão sobre redes par-a-par na Internet. Primeiramente, na Seção 3.2, caracteriza-se o problema da distribuição de vídeo na Internet, apresentando-se as limitações da atual arquitetura da rede. Nessa mesma seção, algumas das técnicas empregadas e soluções propostas para a distribuição de vídeo, como

a comunicação multidestinatária e as redes de distribuição de conteúdo, são brevemente descritas. Em seguida, na Seção 3.3, as principais características da distribuição de vídeo sobre redes par-a-par são apresentadas. Descrevem-se as arquiteturas de distribuição, ressaltando suas vantagens e desvantagens. Ainda nessa seção, cada uma das arquiteturas é exemplificada através de sistemas de vídeo par-a-par propostos na literatura. Por sua vez, na Seção 3.4, são apresentados outros sistemas de distribuição par-a-par encontrados na literatura e disponíveis comercialmente. É feita uma classificação de acordo com a arquitetura de distribuição implementada por cada sistema. Por fim, na Seção 3.5, discute-se alguns dos principais problemas ainda em aberto na distribuição de vídeo sobre redes par-a-par, como a segurança, o incentivo à cooperação entre os pares e os aspectos práticos relacionados ao desenvolvimento desses sistemas na Internet.

3.2. Distribuição de Vídeo na Internet

A distribuição de vídeo na Internet traz novos desafios para o projeto de protocolos e aplicações. O primeiro desafio é garantir os requisitos de qualidade de serviço da aplicação de vídeo. O vídeo é uma mídia contínua que requer uma grande largura de banda passante, se comparada a outros tipos de mídia como dados e voz, e é sensível ao atraso e à variação do atraso (*jitter*). Apesar de ser uma mídia que apresenta alta redundância espacial e temporal, o vídeo também é sensível à taxa de perda de pacotes [Moraes et al. 2003]. Quando codificado, as informações redundantes contidas no vídeo são reduzidas para diminuir o volume de informação e, conseqüentemente, o espaço ocupado em memória e o tempo de transmissão. O segundo desafio é garantir a escalabilidade da aplicação. Espera-se que o número de usuários de sistemas de distribuição de vídeo seja da ordem de milhões e, portanto, é necessário encaminhar o conteúdo de vídeo de forma eficiente para os receptores. O terceiro desafio é atender de forma satisfatória aos receptores do vídeo, visto que cada um pode possuir capacidade de processamento diferente e estar conectado à Internet por meio de redes de acesso de capacidades diferentes. Nessa seção, são apresentadas algumas soluções propostas para lidar com tais desafios. Primeiramente, descreve-se a atual arquitetura da Internet e discutem-se suas características que dificultam o desenvolvimento da distribuição de vídeo. Em seguida, apresentam-se uma evolução dos serviços de comunicação de grupo, as principais técnicas de codificação e transmissão adaptativas e, por fim, as redes de distribuição de conteúdo.

3.2.1. A Arquitetura da Internet e suas Limitações

A Internet é uma rede baseada na técnica de comutação de pacotes. Ao contrário da técnica de comutação de circuitos, que pode ser utilizada nas redes de acesso à Internet, na comutação de pacotes não há reserva de recursos para os usuários. Isso permite o uso mais eficiente da rede uma vez que os recursos são compartilhados pelos usuários. Para a transmissão de dados, a comutação de pacotes é ideal, porém para a transmissão de mídias com restrições de tempo e perda de pacotes, como é o caso da distribuição de vídeo digital codificado, ela apresenta alguns problemas. Como não existe conexão, é necessária a adição de um cabeçalho em cada pacote. Isso geralmente aumenta a complexidade dos nós intermediários da rede, que precisam analisar o conteúdo do cabeçalho de cada pacote para encaminhá-lo corretamente. Além do atraso de propagação no meio, têm-se os atrasos de encapsulamento, de desencapsulamento e de processamento dos pacotes e o atraso

de espera dos pacotes nas filas dos roteadores. Na Internet, o IP (*Internet Protocol*) é o protocolo responsável pelo encaminhamento dos pacotes. O IP oferece um serviço não confiável, sem estabelecimento prévio de conexão, em que os pacotes são encaminhados de acordo com o modelo de melhor esforço. Como não há conexão e a banda passante para cada usuário é alocada dinamicamente, de acordo com a disponibilidade da rede, não há garantia de vazão, atraso ou variação do atraso. Dessa forma, por não haver reserva de recursos para cada usuário da rede, nem diferenciação no tratamento dos pacotes encaminhados pelos roteadores, a qualidade de serviço exigida pela distribuição de vídeo não é atendida pela atual arquitetura da Internet. Portanto, é necessário o desenvolvimento de mecanismos para garantir tais requisitos.

Grande parte das aplicações da Internet se baseia no modelo cliente-servidor. Nesse modelo, é estabelecida uma comunicação ponto-a-ponto entre um cliente e o servidor, que ao receber requisições de um mesmo conteúdo de diferentes clientes, envia uma cópia do conteúdo requisitado para cada cliente. Para a distribuição de vídeo, esse modelo não é adequado em grande escala. Umas das características da distribuição de vídeo é o envio de uma mesma informação para múltiplos receptores. Como o número de receptores potenciais é da ordem de milhões, o envio de diferentes cópias do mesmo vídeo através de múltiplos canais ponto-a-ponto entre o servidor e os receptores é inviável em termos de banda passante. Sendo assim, é necessário adotar técnicas de encaminhamento que garantam a escalabilidade de distribuição de vídeo.

Existe ainda o problema da heterogeneidade dos nós conectados a Internet e que são potenciais receptores do vídeo distribuído. Tal problema é ilustrado na Figura 3.1. A Internet é formada por redes com características distintas de banda passante. Os usuários podem estar conectados à rede através de uma linha convencional de telefonia, uma linha ADSL (*Asymmetric Digital Subscriber Line*), uma rede sem fio, etc. Deve-se considerar ainda que a banda passante da Internet é compartilhada e que os computadores de cada usuário possuem capacidades de processamento diferentes. Sendo assim, a diversidade na capacidade dos nós impede que uma única taxa de transmissão atenda satisfatoriamente a todos os receptores da distribuição de vídeo na Internet. Nesse caso, uma solução do tipo "melhor denominador comum" não é suficiente. Se a taxa de transmissão do vídeo for adaptada ao receptor de maior capacidade, haverá congestionamento nos enlaces receptores de menor capacidade. Por outro lado, se a taxa de transmissão do vídeo for adaptada ao receptor de menor capacidade, os receptores de maior capacidade receberão um vídeo, cuja qualidade será menor do que a que eles podem receber. Portanto, é necessário que receptores com capacidades diferentes recebam vídeos com níveis de qualidade diferentes.

3.2.2. Comunicação Multidestinatária

A comunicação multidestinatária (*multicast*), também chamada de comunicação de grupo, tem como principal objetivo garantir a escalabilidade das aplicações, cuja característica é enviar uma mesma informação para múltiplos receptores. Esse é o caso da distribuição de vídeo. Busca-se aumentar a eficiência do encaminhamento de uma mensagem para evitar o consumo desnecessário de recursos da rede. Na comunicação multidestinatária, a fonte deve enviar apenas uma cópia da informação e a rede deve ser capaz de replicar essa informação somente quando necessário.

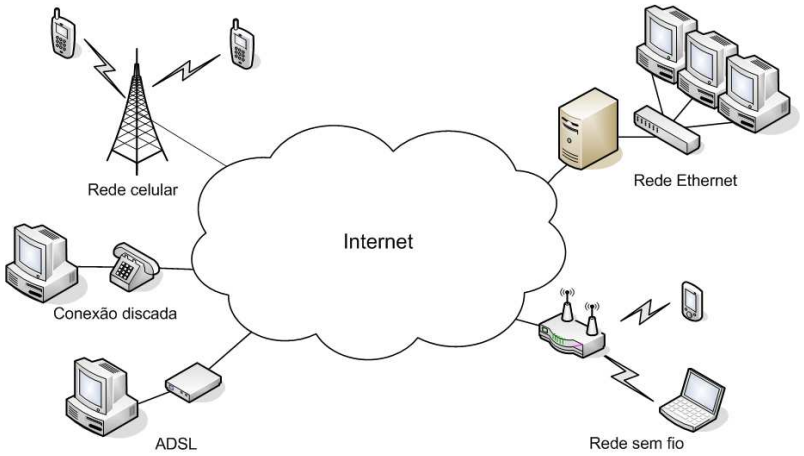


Figura 3.1. A heterogeneidade dos nós da Internet.

O serviço IP Multicast [Deering 1989] implementa a comunicação multidestinatária na camada de rede. A idéia básica desse serviço é reunir um conjunto de estações em um grupo. Define-se um grupo como um conjunto de zero ou mais estações identificadas por um único endereço IP de destino. Dessa forma, ao enviar um datagrama para o endereço IP do grupo, todos os membros do grupo recebem o datagrama enviado. É importante ressaltar que um datagrama multidestinatário é entregue com a mesma confiabilidade de um datagrama ponto-a-ponto (*unicast*), segundo o modelo de serviço de melhor esforço oferecido pelo protocolo IP. Proposto há quase 20 anos, o IP Multicast vem sendo implantado lentamente como serviço nativo na Internet, em virtude da complexidade para configurar e gerenciar o conjunto de protocolos de rede necessários à sua implementação [Costa e Duarte 2003]. A razão dessa complexidade deve-se ao fato do modelo de serviço IP Multicast definir um grupo como uma conversação muitos-para-muitos, isto é, *n* fontes para *m* receptores, da qual qualquer estação pode participar sem exigência de autorização. O modelo também define que uma estação pode pertencer a vários grupos, podendo entrar ou sair de um grupo a qualquer instante, e que uma fonte pode enviar dados, pertencendo ou não ao grupo. Não existe nenhum mecanismo de controle de acesso ao grupo e nem um sistema global de alocação de endereços. Sem isso não é possível evitar a interferência e a colisão entre as aplicações.

Em virtude da complexidade e, conseqüentemente, da lenta implantação do IP Multicast na Internet, novas alternativas foram propostas para simplificar o serviço. Uma dessas alternativas é a comunicação multidestinatária com fonte específica (*Source-Specific Multicast* - SSM) [Holbrook 2006,Bhattacharyya 2003]. O serviço SSM usa o conceito de canal, proposto pelo protocolo EXPRESS (*EXPlicitly REquested Single Source multicast*) [Holbrook e Cheriton 1999]. Um canal possui apenas uma fonte por grupo e é identificado por um par de endereços: o endereço da fonte e o endereço do grupo. Somente a fonte pode enviar datagramas para as estações pertencentes ao grupo. Conseqüentemente, a conversação é reduzida para um-para-muitos, o que simplifica a arquitetura do serviço. Um assinante solicita a recepção dos dados enviados a um canal, especificando explicitamente os endereços da fonte e do grupo. A fonte, por sua vez, envia os dados para o canal especificando apenas o endereço do grupo associado ao canal. A camada de rede garante

que todos os datagramas enviados pela a fonte para o grupo são entregues a todos os assinantes do canal, com a mesma confiabilidade de um datagrama ponto-a-ponto enviado pela fonte. A proposta da comunicação multidestinatária com fonte específica se justifica, pois a maioria das aplicações de grupo conhecidas atualmente possui uma única fonte, caso da difusão de TV. Além disso, nas aplicações com múltiplas fontes, o número de fontes é pequeno, caso da videoconferência, o que torna viável o uso de múltiplos canais.

Tanto o serviço IP Multicast quanto o SSM implementam a comunicação multidestinatária na camada de rede. Por isso, a implantação desses serviços requer modificações na infra-estrutura de rede. Os roteadores têm de ser capazes de lidar com os protocolos de roteamento *multicast* e também devem manter estados por grupo, o que aumenta a complexidade e compromete a escalabilidade do serviço. Uma alternativa para contornar tais problemas é implementar a comunicação multidestinatária na camada de aplicação, também conhecida por *multicast* aplicativo (*Application-Level Multicast* - ALM) [El-Sayed et al. 2003, Lao et al. 2005, Banerjee e Bhattacharjee 2005]. A idéia é que o roteamento, o gerenciamento dos participantes e a replicação das informações sejam de responsabilidade das próprias estações que participam da comunicação. Todas essas funções são implementadas assumindo a existência apenas do serviço ponto-a-ponto. Para tanto, uma rede sobreposta (*overlay*) deve ser construída na camada de aplicação. Um exemplo de rede sobreposta é apresentado na Figura 3.2. Essa rede é composta de túneis ponto-a-ponto entre duas estações. Esses túneis são os enlaces da rede sobreposta. As próprias estações são responsáveis pelo encaminhamento dos pacotes. Dessa forma, como todos os pacotes são transmitidos de acordo com o serviço ponto-a-ponto, não são necessárias modificações na infra-estrutura de rede nem o armazenamento de estados nos roteadores. Com isso, é possível acelerar a implantação de serviços de comunicação multidestinatária na Internet e torná-los mais escaláveis.

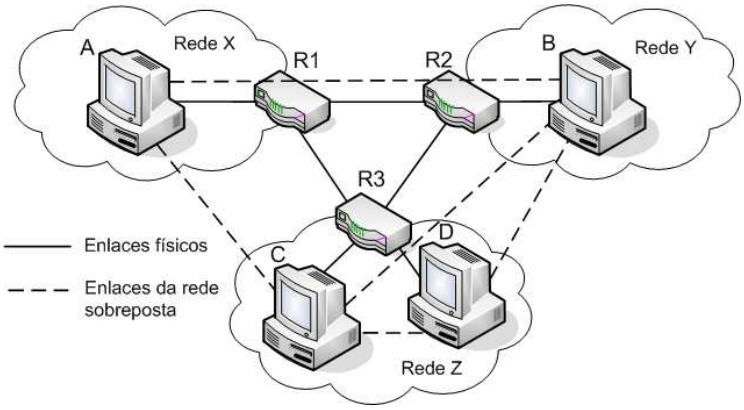


Figura 3.2. Um exemplo de uma rede sobreposta (*overlay*).

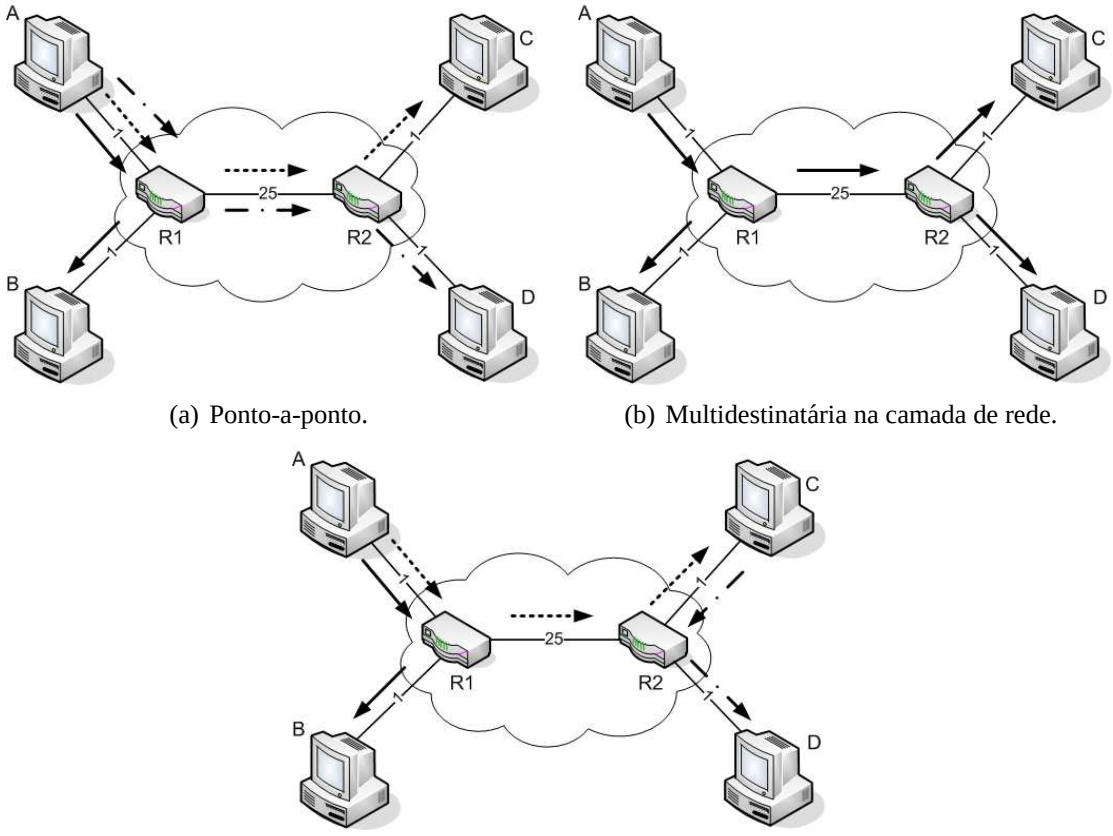
Existem duas possíveis formas de implementação da rede sobreposta na camada de aplicação. Na primeira, um conjunto de servidores ou *proxies*, geralmente chamados de nós âncora, são posicionados em pontos específicos da rede [Zhuang et al. 2001, Guo et al. 2004, Hefeeda et al. 2004]. A rede sobreposta é construída e mantida apenas pelos nós âncora. Toda estação que deseja participar da comunicação deve se conectar a um nó âncora específico. Em uma segunda abordagem, não existem nós dedicados

à construção da rede sobreposta. A rede é construída e mantida por todos os participantes da comunicação de forma autônoma e cooperativa [Chu et al. 2002, Tran et al. 2004, Zhang et al. 2005c, Banerjee et al. 2002b]. Em troca da utilização do serviço, os nós compartilham seus recursos, o que caracteriza um sistema par-a-par. Portanto, os sistemas par-a-par podem sem usados para implementar a comunicação multidestinatária na camada de aplicação. Esse é o caso da distribuição de vídeo sobre redes par-a-par que é detalhada na Seção 3.3 e é o foco deste capítulo.

Embora eficiente, a comunicação multidestinatária na rede sobreposta não apresenta o mesmo desempenho da comunicação multidestinatária implementada na camada de rede. É impossível impedir que os túneis que formam os enlaces da rede sobreposta atravessem os mesmos enlaces físicos e, assim, não há como evitar a redundância de tráfego nesses enlaces. Além disso, o tráfego entre estações é encaminhado por outras estações que não são dedicadas a essa tarefa e que, portanto, podem acarretar em aumento da latência. A Figura 3.3 mostra as diferenças entre a comunicação ponto-a-ponto e a comunicação multidestinatária na camada de rede e na camada de aplicação. A topologia da rede do exemplo é composta por dois roteadores (R_1 e R_2) e por quatro estações (*A*, *B*, *C*, e *D*). O custo do enlace entre os roteadores é maior do que o dos demais enlaces. No exemplo, a estação *A* deseja enviar uma mesma informação para as outras três estações. A Figura 3.3(a) mostra como essa informação é enviada por *A* usando o serviço ponto-a-ponto. Nota-se que existe uma grande redundância de tráfego nos enlaces próximos à fonte, bem como cópias da mesma informação em enlaces de maior custo. A Figura 3.3(b) mostra a árvore de distribuição construída por protocolo de roteamento multidestinatário. Nota-se que não existem transmissões redundantes. Apenas uma cópia da informação é transportada em cada enlace da rede. Além disso, nessa configuração, o atraso da informação recebida pelas estações é o mesmo, caso a informação tivesse sido enviada por *A* usando o serviço ponto-a-ponto. A Figura 3.3(c) mostra a árvore de distribuição construída por um sistema par-a-par na camada de aplicação. Observa-se que o número de cópias redundantes nos enlaces próximos à fonte é menor se comparado à comunicação ponto-a-ponto e que apenas uma cópia da informação foi transportada pelo enlace de maior custo. Notam-se ainda as diferenças de desempenho entre a comunicação multidestinatária na camada de aplicação e na de rede. Na camada de aplicação, ainda existe uma cópia redundante nos enlaces *A*-R_1 e *C*-R_2 e o custo do caminho entre *A* e *D* aumentou. O custo desse caminho que era igual a 27 na comunicação multidestinatária na camada de rede passou a ser 29 na camada de aplicação. Constata-se assim que existe um compromisso entre desempenho e facilidade de implantação, quando se compara as formas de implementação da comunicação multidestinatária.

3.2.3. Codificação e Transmissão Adaptativas

As aplicações de distribuição de vídeo devem se adaptar às condições da rede para atender satisfatoriamente aos seus diferentes receptores. Para isso, as aplicações devem responder aos congestionamentos nos enlaces da rede e também enviar aos receptores, que possuem diferentes capacidades de banda e processamento, vídeos de qualidades diferentes. Tais funcionalidades podem ser obtidas com a adoção de técnicas de codificação e transmissão de vídeo adaptativas. A idéia básica dessas técnicas é adaptar a taxa de transmissão do vídeo às condições dos receptores. Essas técnicas podem ser classificadas



(a) Ponto-a-ponto.

(b) Multidestinatária na camada de rede.

(c) Multidestinatária na camada de aplicação.

Figura 3.3. As diferenças entre a comunicação ponto-a-ponto e multidestinatária [Chu et al. 2002].

de acordo com a taxa que enviam o vídeo para os receptores (taxa única ou multitaxa) e também de acordo com o lugar no qual a adaptação é feita (nas estações ou no núcleo da rede) [Liu et al. 2003]. Vale ressaltar que a maiorias dessas técnicas assume a existência da comunicação multidestinatária.

Uma primeira abordagem é enviar uma única taxa de vídeo que se adapta ao longo do tempo (*Single-rate adaptation*) [Bolot et al. 1994, Liu et al. 2003]. Nessa abordagem, um fluxo de vídeo codificado é enviado pela fonte, que periodicamente solicita e recebe informações dos receptores sobre as condições da rede. Por exemplo, Bolot *et al.* definem três possíveis condições da rede: sem carga (*unloaded*), com carga (*loaded*) e congestionada (*congested*) [Bolot et al. 1994]. Um receptor, então, determina em qual dos três estados se encontra de acordo com a taxa de perda de pacotes e envia essa informação para fonte. Essa realimentação serve para que a fonte reduza ou aumente a taxa de envio do vídeo. Um fluxo de vídeo pode ser enviado para cada receptor ou para um grupo de receptores, assumindo a existência da comunicação multidestinatária. Em ambos os casos, o problema da abordagem de uma única taxa adaptativa é o aumento da carga nos enlaces próximos à fonte, em virtude do envio das informações de realimentação pelos receptores. Na comunicação multidestinatária, ainda existe o problema da heterogeneidade dos receptores que dificulta a definição de uma métrica global para adaptar a taxa de envio.

Para solucionar o problema da heterogeneidade dos receptores, foi proposta a técnica de replicação adaptativa, também chamada de *simulcast* [Cheung et al. 1996]. Nessa técnica, a fonte codifica simultaneamente múltiplos fluxos de vídeo com qualidades distintas[2] em um único arquivo. De acordo com suas restrições de banda passante, cada receptor escolhe qual dos fluxos deseja receber. Existem duas formas possíveis para transmitir o vídeo. Na primeira, usando a comunicação ponto-a-ponto, os receptores enviam mensagens para a fonte determinando qual fluxo desejam receber. Esse é o caso da tecnologia SureStream disponível em tocadores de vídeo da RealNetworks [Liu et al. 2003]. A outra possibilidade é enviar cada um dos fluxos para um endereço *multicast* diferente. Dessa forma, os receptores se inscrevem no grupo usado para transmitir o fluxo de qualidade correspondente à sua capacidade. Essa é a idéia do protocolo DSG (*Destination Set Grouping*) [Cheung et al. 1996, Li et al. 1999]. Apesar de lidar com o problema da heterogeneidade dos receptores, na técnica de replicação a banda passante da rede é usada de forma ineficiente, uma vez que são transmitidos vários fluxos com o mesmo conteúdo de vídeo, apenas de qualidades diferentes.

Uma proposta para atender os receptores heterogêneos e evitar a redundância da replicação é a codificação em camadas [McCanne et al. 1996, Vickers et al. 2000]. Nessa técnica, ilustrada na Figura 3.4, um codificador comprime o vídeo em uma ou mais camadas hierárquicas com diferentes prioridades. A camada que contém as informações essenciais do fluxo de vídeo é chamada de camada base. Essa é a camada de mais alta prioridade. Todo receptor deve ser capaz de receber pelo menos essa camada. Camadas adicionais com prioridade decrescente podem ser codificadas para refinar a qualidade da camada base. Essas camadas de refinamento são somadas à camada base, ou a outras camadas de refinamento com maior prioridade, aumentando progressivamente a qualidade da informação de vídeo reconstruída. No exemplo, consideram duas camadas de refinamento além da camada base. O refinamento pode estar relacionado à taxa de transmissão, à resolução, à taxa de quadros ou à relação sinal-ruído com que o fluxo de vídeo chega ao receptor. No exemplo da Figura 3.4, a taxa do vídeo enviado é igual a 1 Mbps. Assume-se que o vídeo está dividido em três camadas: a camada base com taxa de 128 kbps e mais duas camadas de refinamento 1 e 2 com taxas de 384 kbps e 512 kbps, respectivamente. Dessa forma, os receptores *A*, *B* e *C* recebem fluxos de vídeo com qualidades diferentes, uma vez que recebem um número diferente de camadas. Cada uma das camadas pode ser transmitida para um endereço multidestinatário. Dessa forma, cada receptor pode associar-se a um ou mais grupos de acordo com as suas limitações de banda passante. Cada uma das camadas é transmitida sobre uma árvore de distribuição diferente. A hierarquia entre as camadas é o principal obstáculo para a adoção da codificação em camadas na prática. Para garantir que a camada base seja entregue sem erros, a rede tem que tratar diferenciadamente os pacotes de cada camada e também retransmitir os pacotes perdidos da camada base. Dependendo da configuração da rede e do atraso tolerado pela aplicação, não é possível realizar essas tarefas.

A codificação em múltiplos descritores (*Multiple Description Coding* - MDC) busca simplificar a codificação em camadas. Na MDC, um fluxo de vídeo é codificado em diferentes subfluxos independentes, chamados de descritores. Cada descritor possui

[2]Cada fluxo é codificado com diferentes parâmetros de compressão.

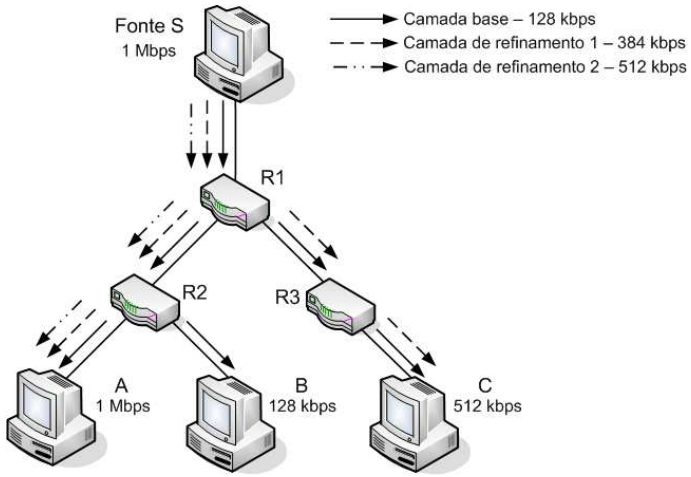


Figura 3.4. Um exemplo da codificação em camadas.

a mesma taxa e a mesma importância. Como não há hierarquia entre os descritores, todos os pacotes são tratados sem diferenciação e não é necessário retransmitir os pacotes perdidos. Os pacotes de um descritor podem ser enviados em um mesmo caminho ou em caminhos disjuntos e, dessa forma, aumenta-se a robustez da aplicação. A qualidade do vídeo recebido depende do número de descritores recebidos em paralelo. Assim como na codificação em camadas, quanto mais descritores recebidos, melhor é a qualidade do vídeo. Porém, cada descritor pode ser decodificado independentemente dos demais e é suficiente para gerar uma aproximação do conteúdo original. Isso facilita a manutenção da continuidade do vídeo. Atualmente, existem protótipos de codificadores MDC que conseguem gerar dois ou três descritores [Rejaie 2006].

Nas técnicas apresentadas anteriormente, a fonte e, em alguns casos, os receptores são os responsáveis por adaptar o fluxo de vídeo às condições da rede. Uma outra possibilidade é atribuir a tarefa de adaptação aos nós do núcleo da rede. Essa é a idéia da transcodificação [Amir et al. 1995]. Nessa técnica, a fonte transmite a uma taxa elevada um único fluxo de vídeo codificado. Cabe aos nós intermediários transcodificar - decodificar e recodificar - o fluxo de vídeo nos pontos de gargalo da rede. O custo computacional exigido pela técnica de transcodificação é elevado. Além disso, há desperdício de banda passante entre a fonte e os pontos de gargalo da rede [Liu et al. 2003].

3.2.4. Redes de Distribuição de Conteúdo

O objetivo das redes de distribuição de conteúdo (*Content Distribution Networks* - CDNs) é aumentar a eficiência e a escalabilidade do modelo cliente-servidor para a distribuição de vídeo. Uma CDN é formada por um conjunto de nós interconectados através da Internet que cooperam de forma transparente para distribuir um conteúdo específico aos usuários [de Albuquerque et al. 2006]. Esses nós são servidores ou nós auxiliares que, geralmente, estão espalhados geograficamente e pertencem a *backbones* diferentes, como mostra a Figura 3.5. A idéia básica das CDNs é replicar o conteúdo do servidor de origem para servidores ou nós auxiliares mais próximos dos clientes. Espera-se que, dessa forma, o servidor entregue o conteúdo aos clientes com uma maior taxa

de transferência e uma menor latência. Geralmente, quanto mais próximo de um cliente está o servidor, mais eficiente é a distribuição. Por isso, quanto mais servidores e nós auxiliares, melhor o desempenho de uma CDN. Entretanto, quanto maior o número e a capacidade dos nós de uma CDN, maior é o seu custo de implantação.

Uma rede de distribuição de conteúdo possui dois tipos de servidores: o servidor de origem e o servidor de réplica. O servidor de origem é o responsável pelo armazenamento, atribuição de identificadores e divulgação do conteúdo. O servidor de réplica, por sua vez, é responsável por encaminhar o conteúdo para um dado cliente. O funcionamento simplificado de uma rede de distribuição de conteúdo é ilustrado na Figura 3.5. Um cliente interessado em um dado conteúdo, no exemplo o nó *A*, envia uma requisição para o servidor de origem *S* (passo 1). Uma vez recebida a requisição de um conteúdo, o servidor de origem deve ser capaz de encaminhar essa requisição para o servidor de réplica mais próximo do cliente. No exemplo, o servidor de réplica *X* é escolhido por estar a um salto do nó *A* (passo 2). Nesse exemplo, assume-se que o conteúdo solicitado já está disponível no servidor de réplica *X*. Uma vez disponível, o conteúdo é entregue ao nó *A* pelo servidor de réplica *X* (passo 3).

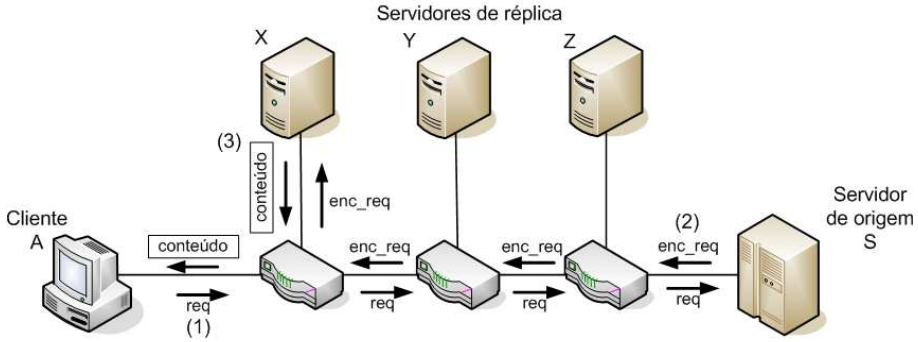


Figura 3.5. O funcionamento de uma rede de distribuição de conteúdo.

O encaminhamento da requisição, a escolha do servidor de réplica e a forma como o conteúdo do servidor de origem é replicado para os servidores de réplica são os principais desafios em uma CDN [de Albuquerque et al. 2006]. Existem diversas técnicas empregadas para redirecionar as requisições. A mais simples é redirecionar as requisições usando o HTTP (*Hypertext Transfer Protocol*). Nessa técnica, todas as requisições são feitas ao servidor de origem através de um navegador de Internet. Ao receber a requisição, o servidor de origem redireciona o navegador para o endereço de servidor de réplica indicado. Embora seja simples, essa técnica pode levar à sobrecarga do servidor de origem e cria um ponto único de falha. Uma outra técnica consiste em reconfigurar o sistema de nome de domínios (*Domain Name System* - DNS). Ao consultar o DNS sobre o nome associado ao servidor de origem, um cliente receberá o endereço IP de um servidor de réplica e não do servidor de origem. Essa técnica é transparente para o cliente. Geralmente, a escolha do servidor de réplica é feita de acordo com a distância entre o cliente e o servidor e também com base na carga do servidor. A distância pode ser dada pelo número de saltos ou pelo tempo de ida-e-volta (*Round Trip Time* - RTT) entre o cliente e o servidor de réplica. A carga do servidor pode ser informada pelo próprio para o servidor de origem ou pode ser obtida através do envio periódico de sondas. A forma como

o conteúdo do servidor de origem é replicado para os servidores de réplica depende do tipo de distribuição de vídeo disponibilizada pelo servidor de origem [Qiu et al. 2001, Banerjee et al. 2003]. No caso do vídeo sob demanda, o conteúdo do servidor de origem pode ser replicado integralmente ou parcialmente em intervalos de tempo predefinidos para os servidores de réplica. O conteúdo também pode ser replicado em virtude de novas requisições. Nessa situação, o conteúdo é enviado para o servidor de réplica escolhido que o armazena e reencaminha para o cliente. Assim, requisições posteriores já encontraram o conteúdo disponível nesse servidor de réplica. Na difusão de vídeo, o conteúdo dever ser continuamente enviado pelo servidor de origem para os servidores de réplica durante a realização da transmissão. Nesse caso, a vantagem dos servidores de réplica é reduzir o número de cópias de um mesmo conteúdo no caminho entre o servidor de origem e os clientes. As cópias são feitas e encaminhadas pelos servidores de réplica. Forma-se assim uma árvore de distribuição na qual o servidor de origem é a raiz, os servidores de réplica são os nós internos e os clientes são as folhas.

Existem diversos exemplos de redes de distribuição de conteúdo acadêmicas [Freedman et al. 2008, Pai et al. 2008] e comerciais [Limelight Networks 2008, Severn Stream Media Networks 2007, CDNetworks Co., Ltd. 2004]. A rede comercial mais popular é a Akamai [Akamai Technologies 2008b]. Estima-se que essa rede possui mais de 19 mil servidores espalhados pela Internet. A Akamai foi usada para transmitir ao vivo o concerto Live Earth realizado simultaneamente em diferentes cidades do mundo e que teve a duração de 24 horas. Durante esse concerto, foram distribuídos mais de 15 milhões de fluxos de vídeo com um pico de 237 mil usuários simultâneos [Akamai Technologies 2008a].

3.3. Distribuição de Vídeo sobre Redes Par-a-Par

Os sistemas par-a-par têm como principal característica a escalabilidade. Nesses sistemas, os nós participantes compartilham seus recursos e colaboram para o funcionamento e manutenção do sistema. Com isso, quanto maior o número de nós participantes, maior é a capacidade do sistema, visto que mais nós compartilham seus recursos. Além disso, os próprios nós constroem a estrutura de comunicação entre os pares, sem que haja necessidade de modificações na infra-estrutura de rede. Essas características são fundamentais para o desenvolvimento da distribuição de vídeo em larga escala e, por isso, fazem dos sistemas par-a-par uma alternativa promissora para a implantação desse serviço na Internet.

O enorme sucesso dos sistemas par-a-par para o compartilhamento de arquivos também é um indicativo do potencial do uso de sistemas similares para distribuir vídeo na Internet. Porém, se comparada ao compartilhamento de arquivos, a distribuição de vídeo possui diferentes requisitos e introduz novos desafios. A maior diferença está relacionada ao tipo de mídia distribuída. Nos sistemas de compartilhamento de arquivos, trabalha-se com longas transferências de dados sem restrições de tempo. Nos sistemas de vídeo, ao contrário, a distribuição deve atender a requisitos estritos de banda passante e tempo para garantir a cadência e a qualidade do vídeo nos receptores. Além disso, os sistemas de compartilhamento de arquivos têm como objetivo construir arquiteturas de distribuição que tornem a indexação e a busca de arquivos mais eficientes. Na distribuição de vídeo, por outro lado, o objetivo é tornar a comunicação entre os pares mais eficiente. Outra

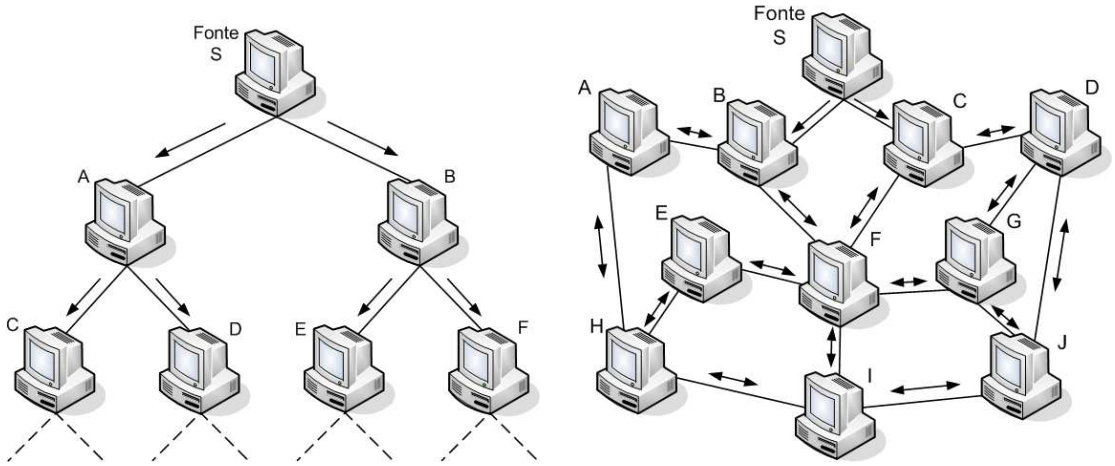
diferença é o número de usuários simultâneos em cada tipo de sistema. Um programa de TV distribuído por um sistema de vídeo atrai mais usuários simultâneos do que um arquivo específico disponibilizado por um sistema de compartilhamento de arquivos. Com mais usuários simultâneos e, conseqüentemente, mais recursos compartilhados, é possível atender os requisitos de qualidade de serviço do vídeo.

Um sistema de distribuição de vídeo par-a-par é composto por aplicativos e mecanismos responsáveis pela construção e gerenciamento da estrutura de distribuição e pelo encaminhamento do vídeo. Tais tarefas são realizadas de forma distribuída pelos próprios nós participantes, geralmente, sem o auxílio de uma entidade central. Um sistema de vídeo par-a-par funciona da seguinte forma. Os nós se auto-organizam e constroem uma rede sobreposta na camada de aplicação. Uma vez construída a rede, os nós interessados em um mesmo conteúdo de vídeo se organizam em uma arquitetura de distribuição. Os nós que compõem a arquitetura de distribuição recebem o fluxo de vídeo desejado e também encaminham esse fluxo para outros nós da arquitetura. Cria-se assim um ambiente colaborativo, no qual os nós contribuem para o encaminhamento do vídeo com banda passante, processamento e espaço de armazenamento.

O desempenho de um sistema de distribuição de vídeo par-a-par está relacionado com a arquitetura de distribuição que esse sistema implementa. Primeiramente, o encaminhamento deve ser eficiente em termos de banda passante e atraso para garantir os requisitos de qualidade de serviço do vídeo distribuído. Para também garantir a escalabilidade do sistema, a arquitetura deve suportar um número grande de nós participantes. Ainda, a sobrecarga de controle para a sua construção e manutenção não deve afetar o desempenho do sistema, à medida que o número de pares aumenta. O sistema de distribuição também tem de lidar com a dinâmica de entrada e saída de pares do sistema e com a heterogeneidade desses pares. Durante a entrada e saída de participantes, a coesão da estrutura de encaminhamento de dados deve ser mantida e perdas de informação causadas pela saída de nós do sistema devem ser minimizadas. Além disso, como os nós possuem capacidades diferentes de processamento e armazenamento e estão conectados à Internet através de redes de acesso de diferentes capacidades, deve-se garantir que os recursos exigidos de um nó participante não excedam a capacidade desse nó. Também é desejável que o sistema implemente mecanismos para evitar que um nó de menor capacidade afete o desempenho de todo o sistema, bem como mecanismos que distribuam vídeos de diferentes qualidades para pares com diferentes capacidades. Por fim, os sistemas de distribuição também são afetados por aspectos práticos da Internet, como a escolha do protocolo de transporte utilizado no envio das mensagens de controle e do fluxo de vídeo e a presença de nós localizados em redes protegidas por *firewalls* e/ou que usam endereços IP privados. Tais aspectos são apresentados na Seção 3.5.6.

Basicamente, existem duas formas de arquiteturas de distribuição em sistemas de vídeo par-a-par: a arquitetura em árvore e a arquitetura em malha. As duas diferem na forma como os nós se organizam para encaminhar o fluxo de vídeo, como mostra a Figura 3.6. Na arquitetura em árvore, os nós se organizam em uma única ou em múltiplas árvores nas quais a fonte de vídeo é a raiz da árvore. São formadas relações de pai e filho entre os nós, nas quais somente o pai encaminha o conteúdo de vídeo para os seus filhos. Sendo assim, cada nó que deseja participar da distribuição deve se inscrever na árvore e a partir daí, sem que haja novas requisições, recebe o conteúdo ou da própria fonte ou de

outro nó já inscrito na árvore. Por exemplo, na Figura 3.6(a), o nó *A* recebe o fluxo de vídeo diretamente da fonte *S* e encaminha para seus filhos *C* e *D*. Nenhum outro nó envia pacotes de vídeo para o nó *A*. Na arquitetura em malha, os nós formam uma malha de distribuição e compartilham os "pedaços" (*chunks*) de um vídeo que estão espalhados pela rede. Os nós não possuem funções específicas e podem tanto receber quanto encaminhar os pedaços de um vídeo para quaisquer outros nós. Na Figura 3.6(b), por exemplo, o nó *F* recebe os pedaços de vídeo da fonte *S* de todos os seus parceiros – os nós *B*, *C*, *E*, *G* e *I* – e também encaminha esses pedaços para os parceiros. A fonte *S* é o único nó que só encaminha o fluxo de vídeo. Portanto, semelhante à arquitetura em árvore há apenas uma fonte de vídeo. Porém, na arquitetura em malha, cada nó pode receber vídeo de mais de um nó do sistema sem a organização hierárquica das árvores de distribuição. Vale ressaltar que os nós pertencentes à malha de distribuição têm de saber quais pedaços cada um dos seus pares possui e explicitamente requisitar os pedaços que desejam aos pares mais indicados. A seleção de pares geralmente é feita de acordo com a disponibilidade dos pedaços e a banda passante dos pares, como é descrito nas Seções 3.3.2.1 e 3.5.4.



(a) A arquitetura em árvore. (b) A arquitetura em malha.

Figura 3.6. As arquiteturas de distribuição para sistemas de vídeo par-a-par.

Nessa seção, as arquiteturas em árvore e em malha são caracterizadas e suas vantagens e desvantagens são analisadas. Para exemplificar cada uma das arquiteturas, são detalhados três sistemas de vídeo par-a-par. Para cada um dos sistemas, descrevem-se os mecanismos de construção e gerenciamento da arquitetura de distribuição, assim como as técnicas de encaminhamento utilizadas. Também são apresentados os mecanismos utilizados em cada sistema para lidar com problemas característicos da distribuição de vídeo par-a-par, como a dinâmica de entrada e saída de participantes e a heterogeneidade desses participantes. Por fim, apresentam-se algumas propostas de arquiteturas híbridas que reúnem características das arquiteturas em árvore e em malha.

3.3.1. Arquitetura de Distribuição em Árvore

A idéia básica da arquitetura de distribuição em árvore é semelhante à idéia do serviço IP Multicast. Os nós se organizam para formar uma árvore de distribuição, por

onde é encaminhado um dado fluxo de vídeo. A raiz da árvore é a fonte de vídeo e, geralmente, é construída uma árvore para cada fluxo de vídeo. Os nós pertencentes à árvore possuem relações de pai e filho, ou seja, quando um nó recebe um pacote de seu pai, ele encaminha uma cópia desse pacote para cada um de seus filhos, sem que haja uma requisição explícita por parte dos filhos. Dessa forma, o fluxo de vídeo é "empurrado" para os receptores.

As vantagens da arquitetura em árvore são a baixa latência e a baixa sobrecarga de controle quando a entrada e saída de pares não é freqüente, pois uma vez construída a árvore somente os pacotes de vídeo são encaminhados sem a necessidade do envio de mensagens de controle. Entretanto, se a saída de nós é freqüente, a sobrecarga de controle tende a ser alta, uma vez que a árvore deve ser reconstruída a cada uma dessas saídas. Quando um nó deixa a árvore, todos os seus nós descendentes deixam de receber o fluxo de vídeo, o que pode acarretar em descontinuidades de recepção. Sendo assim, o tempo de reparo da árvore deve ser pequeno a fim de reduzir a perda de pacotes causada pela saída de um nó. Uma alternativa para amenizar tal problema é, durante a reconstrução da árvore, um nó pai armazenar os pacotes de vídeo que deveriam ser encaminhados para o seu filho que deixou a árvore. Outra desvantagem da arquitetura em árvore é que os pacotes de um fluxo de vídeo seguem o mesmo caminho até um determinado receptor. Dessa forma, o algoritmo de construção da árvore deve balancear o número de filhos atribuídos a cada pai para evitar possíveis congestionamentos na rede. Outra característica da arquitetura em árvore que prejudica o seu desempenho é que a maioria dos nós participantes são folhas da árvore e, com isso, não contribuem com seus recursos no encaminhamento do fluxo de vídeo para outros nós, uma vez que não possuem filhos. A heterogeneidade dos receptores também pode degradar o desempenho do encaminhamento na arquitetura em árvore. Se um nó pai tem menor capacidade de banda passante do que seus filhos, conseqüentemente, esses filhos receberão um vídeo com uma qualidade mais baixa do que poderiam receber.

Uma proposta para tornar a arquitetura em árvore mais robusta à dinâmica de entrada e saída de nós e menos sensível à heterogeneidade dos receptores é o uso de múltiplas árvores [Castro et al. 2003, Dan et al. 2007, Magharei et al. 2007]. A idéia básica é fazer com que receptores de diferentes capacidades recebam vídeos de diferentes qualidades. Para tanto, a fonte deve utilizar um codificador em camadas ou MDC, descritos na Seção 3.2.3. Ao utilizar tais codificadores, a fonte divide um determinado fluxo de vídeo em subfluxos. Cada subfluxo, então, é encaminhado através de uma árvore diferente. Cada nó, por sua vez, determina em quantas árvores deve se inscrever de acordo com a sua banda passante. Quanto maior o número de árvores em que um nó está inscrito, maior será a qualidade do vídeo recebido por esse nó. Um subfluxo é encaminhado em cada árvore da mesma forma que um fluxo. Um nó pai ao receber um pacote encaminha uma cópia desse pacote para os seus filhos. Durante o processo de construção da estrutura multiárvore, um dado nó é obrigatoriamente interno em uma e somente uma das árvores. Nas demais árvores, ele deve ser obrigatoriamente uma folha. O objetivo desse procedimento é minimizar os efeitos da saída de um nó e maximizar a utilização da banda passante compartilhada pelos nós. Caso um nó esteja inscrito em mais de uma árvore, a saída de um antecessor desse nó não provoca descontinuidades no vídeo recebido. Somente a qualidade do vídeo recebido será mais baixa. Além disso, todos os nós contribuem com banda passante para o encaminhamento do vídeo, visto que não podem

ser folhas em todas as árvores.

Nas Seções 3.3.1.1 e 3.3.1.2 são apresentados dois exemplos de sistemas par-a-par que constroem, respectivamente, uma árvore para cada fonte e múltiplas árvores para cada fonte: o ESM/Narada e o SplitStream.

3.3.1.1. End System Multicast/Narada

O ESM (*End System Multicast*) [Chu et al. 2002] foi proposto com o objetivo de implementar as funcionalidades do serviço IP Multicast na camada de aplicação. O gerenciamento de grupo, o roteamento e a replicação de pacotes consideram que apenas o serviço IP unicast é implementado pelos roteadores da rede. O ESM indica duas formas possíveis para implementar essas funcionalidades. Em ambas, constrói-se uma rede sobreposta interconectando os nós participantes. A primeira solução utiliza *proxies* que implementam as funcionalidades do IP Multicast e estão localizados em diferentes pontos da Internet. Esses *proxies* formam uma árvore de distribuição estática e predefinida. Assim, um usuário interessado em receber o conteúdo de um dado grupo se conecta a um dos *proxies*. A segunda solução se baseia na construção de redes par-a-par, nas quais todos os nós interessados em receber o conteúdo de um dado grupo devem construir a arquitetura de distribuição e implementar o gerenciamento dos participantes e a replicação de pacotes. Essa solução, ao longo dos anos, se mostrou mais fácil de ser implantada. Por isso, atualmente, os próprios autores e desenvolvedores definem o ESM como um sistema de distribuição de vídeo par-a-par [ESM Group 2007, Liu et al. 2008]. Neste capítulo, somente a implementação considerando redes par-a-par é abordada.

O Narada é o mecanismo responsável pela construção da arquitetura de distribuição no ESM. Tal procedimento é constituído de duas etapas. Na primeira, o Narada constrói um grafo conectado[3], chamado de malha. A malha é, geralmente, um subgrafo do grafo completo dos nós participantes. Na segunda etapa, o Narada constrói as árvores de distribuição a partir da malha construída. Nas duas etapas, o Narada utiliza protocolos distintos.

Construção e Gerenciamento da Malha de Participantes

Para construir a malha que interconecta os nós participantes, o Narada usa um protocolo baseado em algoritmos baseados em fofocas (*gossip*). Nesses algoritmos, de forma geral, um nó gera uma mensagem e a envia para um conjunto aleatório de nós da rede sobreposta. Ao receberem essa mensagem, os nós escolhidos também sorteiam outros nós e encaminham a mensagem recebida para os nós sorteados e, dessa forma, a mensagem se espalha pela rede.

No Narada, um novo nó que deseja participar da malha deve primeiro ser capaz de adquirir uma lista de nós participantes. A forma como o nó adquire essa lista não é definida pelo Narada. Uma das formas de adquirir a lista de participantes é contatar diretamente a fonte do vídeo de interesse. Para isso, o endereço da fonte deve ser divulgado,

[3]Um grafo conectado é aquele em que é possível a partir de um nó atingir todos os outros nós do grafo.

por exemplo, em um sítio da Internet. A entrada de um nó na malha é exemplificada na Figura 3.7. Os números entre parênteses na figura representam a ordem das ações executadas durante o procedimento de entrada de um nó. Um novo nó *N*, que deseja entrar na malha, contata a fonte *S* (passo 1) e recebe uma lista de participantes (passo 2). Nesse exemplo, a lista contém os nós *A*, *B*, *D* e *E*. A lista adquirida não precisa ser nem completa nem acurada, basta que ele contenha ao menos um nó ativo na malha. Ao receber a lista, o nó escolhe alguns dos participantes presentes na lista. Em seguida, envia uma requisição para um dos nós escolhidos informando que deseja ser adicionado à malha como um de seus vizinhos. Esse processo é repetido até que um dos nós escolhidos responda à requisição. No exemplo, o nó *N* escolhe o nó *D* para ser seu vizinho na malha e, então, envia uma requisição para esse nó (passo 3). Ao receber a requisição, o nó *D* envia uma resposta para o nó *N* (passo 4). Ao receber a resposta, o nó *N* é considerado como participante da malha e um enlace entre *N* e *D* é criado (passo 5). Dessa forma, a malha de participantes vai sendo construída. Vale ressaltar que inicialmente um nó recém-chegado à malha possui apenas um vizinho, que foi o nó escolhido no momento da sua entrada. Entretanto, à medida que novos nós entram na malha, a vizinhança de um nó aumenta, pois novos nós o escolhem como vizinho. Além disso, a malha interconecta todos os participantes do sistema e não apenas os nós interessados em uma determinada fonte de vídeo.

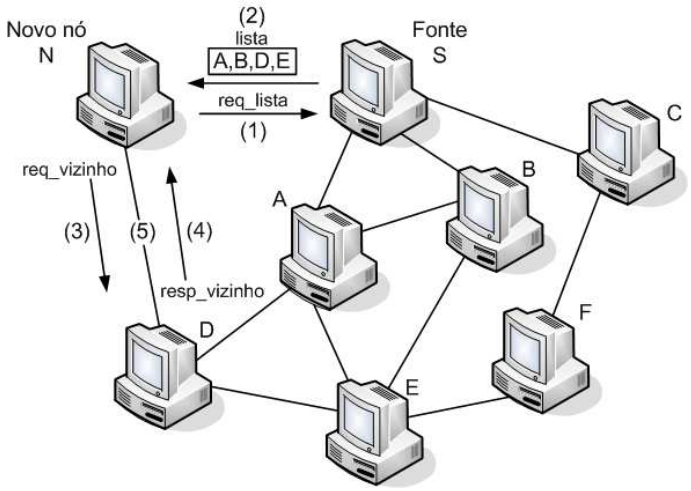


Figura 3.7. A entrada de um novo nó na malha de participantes com o Narada.

O Narada também é responsável pelo gerenciamento dos nós ativos da malha. Como não há uma entidade central, o gerenciamento é realizado em conjunto pelos próprios nós participantes. No Narada, todos os nós da malha mantêm uma lista dos demais nós participantes. Tanto na entrada de um novo nó quanto na saída de um nó, a lista de participantes armazenada por todos os nós deve ser atualizada. Sendo assim, nessas duas situações, mensagens de atualização devem ser geradas e propagadas pela malha. No entanto, como a partida de um nó sem anúncio prévio, pode provocar o particionamento da malha, as mensagens de atualização são enviadas periodicamente e não quando ocorre a entrada ou a saída de um nó. Por isso, um nó somente envia mensagens de atualização aos seus vizinhos na malha, a fim de evitar alta sobrecarga de controle. As mensagens

de atualização contêm a lista de participantes conhecidos por um dado nó. Cada entrada dessa mensagem é formada pelo endereço de um participante e o último número de seqüência usado por esse participante e que é conhecido pelo nó emissor da mensagem. O número de seqüência, crescente e monotônico, é usado para evitar laços (*loops*) no processo de manutenção da malha. Um nó, ao receber as mensagens de atualização dos seus vizinhos, atualiza a sua lista de participantes em duas situações: quando uma das atualizações recebidas contém um participante que ele não conhece ou quando a informação sobre um participante conhecido possui um número de seqüência maior do que o armazenado em sua lista. Por outro lado, um nó também retira de sua lista de participantes conhecidos um dado nó sobre o qual não receba atualizações durante um período de tempo definido.

Quando um nó deixa a malha, ele notifica os seus vizinhos sobre a sua saída e, através da troca periódica de mensagens de atualização, essa informação é difundida pela malha. No caso de uma falha de um nó participante o procedimento é diferente. Os próprios vizinhos têm de detectar a falha de um nó. Para tanto um nó, ao deixar de receber mensagens de atualização de um vizinho, envia várias sondas idênticas para esse vizinho. A redundância das sondas reduz a probabilidade de uma sonda ou da resposta de uma sonda ser perdida. Caso o vizinho suspeito de ter falhado não responda a nenhuma das sondas, ele é considerado como morto e essa informação é, então, difundida pela malha. A saída de um nó ainda pode provocar o particionamento da malha, criando dois subconjuntos. Nessa situação, os nós participantes devem detectar o particionamento e, em seguida, criar um novo enlace para reconectar a malha. A detecção do particionamento é possível, pois os nós em cada um dos subconjuntos da malha deixam de receber atualizações do número de seqüência dos nós que, agora, estão no outro subconjunto da malha. Um tempo máximo (T_{max}) é definido para o não recebimento de atualizações, que ao terminar indica o particionamento da malha. Cada nó mantém uma fila de nós dos quais deixou de receber atualizações durante um tempo maior ou igual a T_{max}. Um nó periodicamente executa um algoritmo de escalonamento para retirar um membro do início da fila. Para esse membro é enviada uma sonda, pelo caminho definido na camada de rede e não na rede sobreposta. Caso não receba resposta, o nó é considerado como morto. Do contrário, se o nó responde ao envio da sonda, um enlace é criado entre o emissor da sonda e o receptor. Dessa forma, a malha volta a estar conectada. Esse procedimento é exemplificado na Figura 3.8. Nesse exemplo, a malha original é composta pelos enlaces $S-A$, $S-D$, $A-E$, $D-E$, $B-E$, $E-F$, $B-C$ e $F-C$. Em determinado instante, nó E falha e a malha é particionada. Formam-se então dois subconjuntos, um formado pelos nós S, A e D e outro pelos nós B, C e F. Com isso, os nós A e D iniciam o processo de reconexão da malha. Ambos enviam uma sonda para o nó E, que é o nó que está no início da fila (passo 1). Como E não está ativo, nenhuma resposta é recebida por A e D. Esses nós, então, enviam uma nova sonda para os nós B e F (passo 2), que são os nós que estão no início da fila, respectivamente, de A e D. Dessa vez, B e F estão ativos e respondem às sondas. Criam-se assim dois enlaces entre os nós A e B e os nós D e F e, dessa forma, a malha é reconectada (passo 3). A nova malha é formada pelos enlaces $S-A$, $S-D$, $A-B$, $D-F$, $B-C$ e $F-C$. Como visto no exemplo, vários nós podem tentar reconectar a malha ao mesmo tempo, o que acarreta na criação de enlaces desnecessários. Para evitar esse problema, a cada nó é atribuída uma probabilidade de execução do escalonador. Dessa

forma, a probabilidade de nós simultaneamente tentarem reconectar a malha é reduzida.

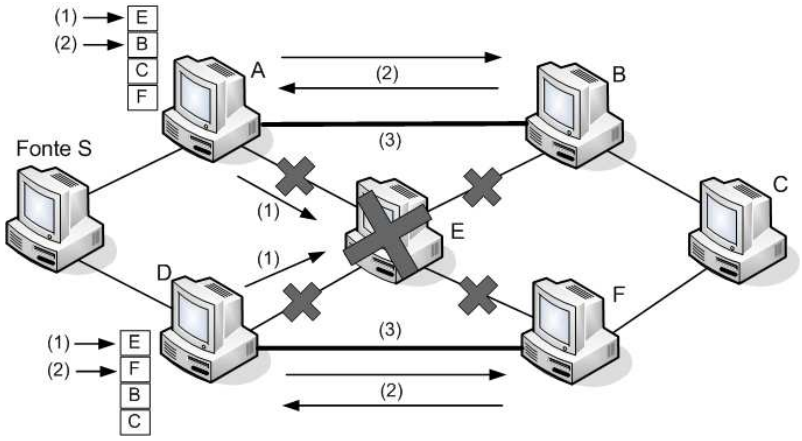


Figura 3.8. A reconexão da malha com o Narada.

As árvores de distribuição são construídas exclusivamente a partir dos enlaces da malha. Sendo assim, a construção da malha tem um papel fundamental na eficiência da distribuição do vídeo. Duas propriedades são importantes na construção da malha. A qualidade do caminho entre dois nós na malha deve ser próxima da qualidade do caminho ponto-a-ponto entre esses nós, ou seja, idealmente, o caminho entre um par de nós construído na camada de aplicação deve ser o mesmo construído na camada de rede pelos protocolos de roteamento. Na distribuição de vídeo, a qualidade se refere à disponibilidade de banda passante e ao atraso. Um nó também deve ter um número limitado de vizinhos na malha. Dessa forma, reduz-se a concentração de mensagens de controle na malha durante a execução do algoritmo de roteamento. Para garantir essas duas propriedades, o Narada implementa algoritmos para aumentar a eficiência da malha construída. Durante o processo de inicialização, a malha construída não é ótima, pois a seleção de vizinhos realizada por novos nós não é a ideal, dado que existe pouca informação sobre a topologia da malha nesse período inicial e a escolha dos vizinhos foi aleatória. Além disso, durante o processo de recuperação de um particionamento, um grande número de enlaces pode ser criado com o objetivo de reconectar a malha o mais rápido possível. Tais enlaces são essenciais durante o processo de recuperação, mas durante a operação normal da malha podem ser desnecessários. A dinâmica de entrada e saída de nós e as variações de carga também provocam modificações na malha e podem torná-la menos eficiente. Para tornar a malha mais eficiente, o Narada remove enlaces existentes e cria novos enlaces de acordo com o monitoramento da malha. Essa característica difere o Narada de protocolos de roteamento tradicionais. O monitoramento é feito pelos próprios nós da malha, que enviam sondas aleatoriamente aos demais nós. De acordo com as respostas das sondas, calcula-se a utilidade de um enlace existente e a utilidade de um possível novo enlace entre o emissor e o receptor da sonda. A função utilidade é definida de acordo com a métrica (ou métricas) para qual a malha deve ser otimizada.

Para adicionar um enlace, cada nó participante periodicamente sorteia alguns nós que não sejam seus vizinhos e envia sondas para esses nós. A resposta de uma sonda contém uma cópia da tabela de roteamento do receptor da sonda. Na próxima seção,

o roteamento nas árvores de distribuição será descrito. A partir da tabela recebida, é possível determinar quais nós contidos na tabela de roteamento do emissor podem ser alcançados tendo como próximo salto o receptor da sonda. Dessa forma, calcula-se a utilidade da criação de um enlace entre o emissor e o receptor da sonda. Se ao usar o receptor como próximo salto ao invés do próximo salto atual, o custo para se alcançar um dado nó é menor, a utilidade do enlace é incrementada. Se a utilidade é superior a um limiar previamente definido, o enlace entre o emissor e o receptor da sonda é criado. O limiar é calculado de acordo com o número estimado de participantes na malha e o número de vizinhos do nó. Considerando por exemplo que a métrica para a qual se deseja otimizar a malha é a latência, o cálculo da utilidade é feito da seguinte forma, para o cenário da Figura 3.9. Assume-se nesse exemplo que o limiar é igual a 0,1. O nó *A* sorteia o nó *C* que não é seu vizinho e envia a esse nó uma sonda (passo 1). O nó *C*, então, responde ao nó *A* com uma cópia de sua tabela de roteamento (passo 2). Dessa forma, é possível calcular o tempo entre o envio e a resposta da sonda e, assim, determinar o custo de um novo enlace entre *A* e *D*. Nesse exemplo, assume-se que o custo desse novo enlace é 1. Ao receber a tabela de roteamento de *C*, o nó *A* verifica que *C* pode ser usado como próximo salto para alcançar a fonte *S*. O custo atual do caminho usado para atingir *S*, tendo *B* como próximo salto, é 3. Caso *C* passe a ser o próximo salto para *S*, esse custo será reduzido para 2 e, conseqüentemente, a utilidade é incrementada. A utilidade é dada pela razão:

$$utilidade = \frac{custo\ atual - novo\ custo}{custo\ atual}. \tag{1}$$

Logo, no exemplo a utilidade é igual 0,33. Como esse valor é superior ao limiar definido, o enlace entre *A* e *C* é adicionado (passo 3). A utilidade pode ser interpretada como uma tendência do comportamento do custo de um enlace, por isso ela é usada como métrica ao invés do custo do enlace diretamente. As variações do custo de um enlace podem indicar comportamentos passageiros e com isso novos enlaces seriam criados desnecessariamente.

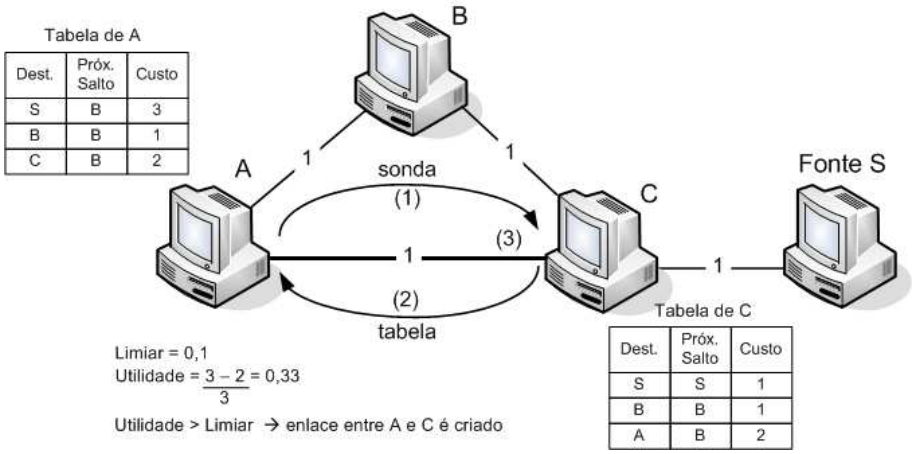


Figura 3.9. A adição de um enlace na malha com o Narada.

À primeira vista, o processo para remover um enlace é semelhante ao processo para adicionar um enlace, ou seja, estima-se a degradação do desempenho de um nó caso o

enlace entre o emissor e o receptor de uma sonda seja removido. Entretanto, é difícil de se obter essa estimativa, uma vez que seria necessário executar um procedimento paralelo de cálculo de rotas e troca tabelas de roteamento considerando que o enlace entre o emissor e o receptor da sonda não existe. Sendo assim, a remoção de um enlace é feita a partir de uma variável, chamada de utilização, calculada por um nó para todos os seus vizinhos. A utilização de um enlace entre dois nós *i* e *j* é dado pelo número de nós que são alcançáveis por *i* tendo *j* como próximo salto. Como periodicamente recebe atualizações das listas de participantes de seus vizinhos, um nó pode determinar a utilização consensual de um enlace. A utilização consensual é definida como o valor máximo entre as utilizações do enlace *i*-*j* calculados independentemente pelos nós *i* e *j*. Uma vez calculada a utilização consensual para todos os vizinhos, o enlace de menor utilização consensual é removido, se o valor dessa variável estiver abaixo de um limiar. O cálculo desse limiar também leva em conta o número estimado de participantes na malha e o número de vizinhos do nó.

O protocolo de transporte usado pelo Narada nos enlaces da rede sobreposta é o TFRC (*TCP Friendly Rate Control*) [Handley et al. 2003]. Basicamente, o protocolo TFRC funciona como o UDP (*User Datagram Protocol*), porém ele controla a taxa de envio de pacotes. O TFRC foi proposto para atender os requisitos das aplicações multimídias, que geralmente utilizam pacotes de tamanho fixo e variam a taxa de envio de pacotes para responder a um congestionamento. Com esse protocolo é possível atingir vazões próximas, mas geralmente inferiores às do TCP e, além disso, o TFRC não sofre com os atrasos de retransmissão e enfileiramento no *buffer* do nó emissor como o TCP.

Construção das Árvores de Distribuição e Roteamento

Duas características da distribuição de vídeo são fundamentais para se determinar qual tipo de árvore deve ser construída. A primeira é que um sistema de distribuição de vídeo deve suportar múltiplas fontes. A segunda é que o encaminhamento deve ser eficiente em relação às restrições de banda passante e tempo do vídeo. Dois tipos de árvores existentes são as árvores por fonte e as árvores compartilhadas. Em uma árvore por fonte, o nó raiz é a própria fonte de dados e, conseqüentemente, deve-se construir uma árvore para cada fonte. A vantagem desse tipo de árvore é construir árvores otimizadas, por exemplo, em termos de latência para a distribuição de vídeo. Entretanto, o emprego deste tipo de árvore em sistemas com múltiplas fontes aumenta a sobrecarga de controle, pois para cada fonte é necessário construir e gerenciar uma árvore diferente. Em uma árvore compartilhada, um nó localizado no "centro" da rede é a raiz e a mesma árvore é usada por várias fontes. Dessa forma, as árvores compartilhadas reduzem a sobrecarga de controle, porém apresentam um ponto central de falha e não são otimizadas para uma dada fonte, em termos de latência e/ou banda passante. O Narada constrói árvores por fonte. Esse é o principal motivo para se construir uma malha antes das árvores de distribuição. Como o gerenciamento dos nós participantes é feito na malha, tal tarefa não precisa ser replicada para todas as árvores construídas. Dessa forma, reduz-se a sobrecarga de controle para se manter uma árvore para cada fonte.

Apesar de conhecer os participantes, um nó não conhece a topologia da malha. Por isso, é necessária a utilização de um protocolo de roteamento para calcular as rotas entre

os nós da malha e difundir as tabelas de roteamento. O Narada usa um protocolo de vetor de distância para tal finalidade. Cada nó da árvore mantém em sua tabela de roteamento o custo para alcançar outros nós e o caminho usado para alcançar esses nós. Periodicamente, os nós enviam cópias das suas tabelas de roteamento para os seus vizinhos. A métrica de roteamento usada pelo protocolo depende das métricas para as quais a malha está sendo otimizada.

Para construir as árvores por fonte, usadas no encaminhamento do vídeo, o Narada adota o algoritmo de caminho reverso mais curto (*Reverse Shortest Path* - RSP). Ao receber um pacote de uma fonte de dados f, um nó *n* verifica se o pacote foi encaminhado por um vizinho *v* que ele usa como próximo salto para enviar dados à fonte f, ou seja, se o pacote chegou pelo caminho reverso mais curto entre a fonte f e o nó *n*. Em caso afirmativo, o nó *n* reenvia o pacote para todos os seus vizinhos, exceto para o vizinho *v* por onde o pacote chegou. Do contrário, o pacote é descartado.

Mudanças de topologia na malha geralmente são provocadas pela saída de nós ou pela remoção de enlaces de baixa utilidade. Quando mudanças ocorrem é necessário que as tabelas de roteamento de todos os nós sejam atualizadas. Como tal processo não ocorre de forma instantânea, é possível que alguns pacotes de vídeo sejam perdidos durante o período de convergência do roteamento. Sendo assim, para evitar a perda de pacotes durante a convergência das tabelas, o Narada introduz um novo custo para os caminhos, chamado de encaminhamento transiente (*Transient Forward* - TF). O valor atribuído por um nó a TF tem que ser maior do que o custo de qualquer rota válida da tabela desse nó e menor do que infinito. Dessa forma, um nó participante que pretende deixar a malha envia uma atualização com custo igual a TF para todos os nós com entradas válidas em sua tabela de roteamento. Assim, os demais nós ao receberem essa atualização selecionam rotas alternativas que não incluam o nó que está de partida. Para evitar a perda de pacotes, o nó que está deixando a malha continua a encaminhar os pacotes recebidos até que não seja mais usado por nenhum outro nó como próximo salto para alcançar um destino ou espera um determinado período de tempo para realmente deixar a malha.

O ESM disponibiliza dois aplicativos, um cliente e um servidor de vídeo, e também o motor do sistema (*system engine*), que é o responsável por implementar a maioria das funcionalidades do sistema. Os três possuem versões para os sistemas operacionais Windows e Mac OS e tem código aberto [ESM Group 2007].

3.3.1.2. SplitStream

Na estrutura em árvore, somente os nós internos da árvore encaminham os pacotes de vídeo. Como a maioria dos nós das árvores são folhas[4], poucos nós contribuem para a distribuição do vídeo, o que vai de encontro à principal característica das redes par-a-par. Além disso, quando usada pelos protocolos de roteamento multidestinatário, a estrutura em árvore assume que os nós internos são roteadores e, por isso, são dedicados e possuem alta disponibilidade. Tal fato não se aplica às redes par-a-par, já que os nós internos da

[4]A fração de folhas cresce de acordo com o grau da árvore. Por exemplo, em uma árvore binária mais de 50% dos nós são folhas. Em uma árvore de grau 16, o número de folhas é superior a 90% [Castro et al. 2003].

árvore são usuários do sistema que podem deixar de cooperar com a distribuição de vídeo a qualquer instante e sem uma notificação prévia. O SplitStream [Castro et al. 2003] busca solucionar esses problemas.

Assim como o ESM, o SplitStream assume a existência de uma rede sobreposta construída na camada de aplicação. Porém, ao contrário do ESM, o SplitStream não define um protocolo como o Narada para construir e gerenciar a malha de participantes. Ele apenas indica quais protocolos poderiam ser usados na construção da malha e das árvores de distribuição. Uma possível solução é usar a combinação entre o Pastry [Rowstron e Druschel 2001] e o Scribe [Castro et al. 2002b], que são descritos adiante.

A idéia básica do SplitStream é o dividir o conteúdo de um vídeo em subfluxos (*stripes*) e enviar cada subfluxo usando uma árvore de distribuição diferente. Um nó participante do sistema, por sua vez, se inscreve em uma ou mais árvores de acordo com o número de subfluxos que deseja receber. Quanto maior o número de subfluxos, maior a qualidade do vídeo recebido. O desafio então é construir uma floresta de árvores na qual um nó interno de uma árvore seja folha nas demais e lidar com as restrições de banda de cada nó. Dessa forma, os pacotes encaminhados são distribuídos por todos os nós participantes e na média aumenta-se a proporção de nós cooperativos.

A Figura 3.10 ilustra como o SplitStream balanceia a carga oferecida entre os nós participantes. O conteúdo da fonte é dividido em dois subfluxos e encaminhado em duas árvores independentes. As linhas cheias correspondem aos ramos da árvore que encaminha a subfluxo 0 e as tracejadas aos ramos da árvore usada para encaminhar o subfluxo 1. Assume-se que a fonte requer uma banda b para enviar o vídeo e que cada subfluxo requer $b/2$. Cada nó participante, exceto a fonte, está inscrito nas duas árvores e, conseqüentemente, deve receber os dois subfluxos. Com isso necessita de uma banda de entrada (*inbound*) b para receber o vídeo. Por outro lado, cada nó é interno em apenas uma árvore e encaminha um dos subfluxos para dois filhos. Logo, a banda de saída (*outbound*) também deve ser igual a b.

Generalizando o procedimento ilustrado no exemplo, se um fluxo de vídeo que requer uma banda b é divido em k subfluxos, as bandas de entrada e saída podem ser controladas com uma granularidade igual a b/k. O controle da banda de entrada é dado pelo número de árvores em que um nó se inscreve e o da banda de saída é dado pelo número de filhos desse nó. Ou seja, se um nó se inscreve em n árvores ele irá receber um fluxo de banda igual a $n \times b/k$. Por outro lado, se ele possui m vizinhos sua banda de saída será igual a $m \times b/k$. Dessa forma, o SplitStream lida com nós com diferentes capacidades de banda passante e também com nós que possuem capacidades de entrada e saída assimétricas.

O encaminhamento dos subfluxos usando múltiplas árvores também torna a estrutura de distribuição mais robusta em relação à saída e à falha de nós. Para tanto, busca-se explorar a diversidade de caminhos existentes na estrutura de distribuição construída na camada de aplicação. Ou seja, se um nó interno de uma das árvores deixa de encaminhar os pacotes de um dos subfluxos, um nó continuará a receber o vídeo, porém com uma qualidade menor até que a árvore responsável por encaminhar esse subfluxo seja reparada.

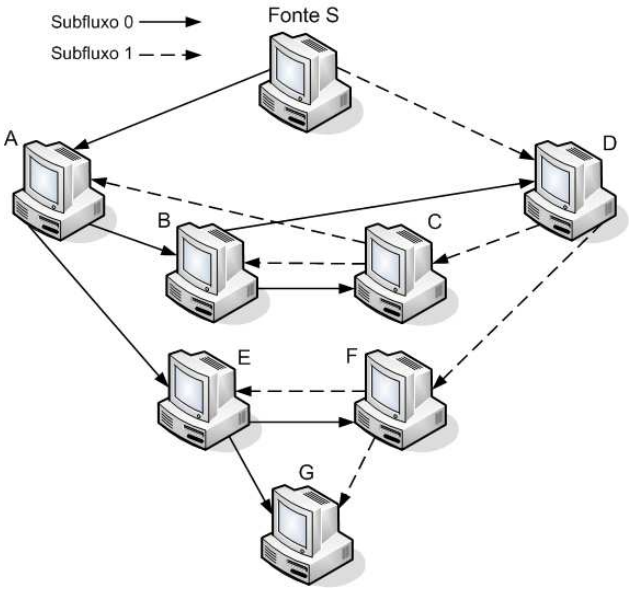


Figura 3.10. O encaminhamento dos subfluxos no SplitStream [Castro et al. 2003]

O SplitStream não define um codificador específico para as fontes de vídeo. Porém, o codificador deve ser capaz de dividir o conteúdo de vídeo em subfluxos. Esses subfluxos devem ter aproximadamente os mesmos requisitos de banda e deve ser possível reconstruir o conteúdo a partir de qualquer subconjunto de subfluxos. Tais características são atendidas pelo MDC, como visto na Seção 3.2.3.

Construção da Rede Sobreposta

O Pastry é um dos protocolos sugeridos para a construção da rede sobreposta no SplitStream. O Pastry estima a proximidade entre dois nós para construir um caminho entre esses nós na rede sobreposta. Para tanto, o Pastry assume que um nó é capaz de medir sua distância para qualquer outro nó com um endereço IP conhecido. A distância pode ser dada pelo número de saltos, pelo tempo de ida-e-volta (medido através de séries de *pings*) ou pela vazão. O número de saltos pode ser medido com a ferramenta `traceroute`), o tempo de ida-e-volta pode ser medido através de séries de *pings* e a vazão pode ser medida através de técnicas de par de pacotes [Castro et al. 2002a]. Considerando que a distância é dada em número de saltos, mostra-se que o atraso médio sofrido pelos pacotes nos caminhos construídos pelo Pastry é menor do que o dobro do atraso dos caminhos construídos pelos protocolos de roteamento na camada de rede [Castro et al. 2002a]. Como visto, essa é uma característica importante para a construção de árvores de distribuição eficientes, justificando o uso do Pastry pelo SplitStream.

Para cada nó, o Pastry atribui um identificador único de 128 bits. O identificador de um nó, também chamado de *nodeId*, é obtido através da aplicação de uma função *hash* ao endereço IP ou à chave pública do nó. Dessa forma, constrói-se um conjunto de identificadores de nó uniformemente distribuído. Atribui-se também, para cada objeto, um identificador de 128 bits chamado de chave (*key*). Dada uma mensagem e uma chave,

o Pastry encaminha essa mensagem para o nó com o identificador numericamente mais próximo da chave. Esse nó é chamado de raiz da chave é o responsável pelo objeto associado a essa chave.

Uma mensagem é encaminhada salto a salto de acordo com a seqüência de dígitos iniciais dos identificadores de nó e das chaves. Essa seqüência inicial é chamada de prefixo. Busca-se sempre encaminhar a mensagem para um nó que possua um maior prefixo em comum com a chave associada a essa mensagem. A Figura 3.11 ilustra esse procedimento. Dada a mensagem *m* e a chave 0xD46A1C, o nó 0x65A1FC deve ser capaz de rotear a mensagem na rede sobreposta. O primeiro passo dado pelo nó é consultar sua tabela de roteamento [Castro et al. 2002b] para verificar os nós que compartilham um prefixo com a chave. No caso, a mensagem é encaminhada para o nó 0xD13DA3, cujo identificador começa com o dígito D, assim como a chave. Ao receber a mensagem, o nó 0xD13DA3 consulta sua tabela para verificar se há um nó que possua um prefixo pelo menos um dígito maior do que o prefixo que ele compartilha com a chave. No exemplo, o nó 0xD4213F é escolhido por possuir os dois primeiros dígitos (D4) semelhantes aos da chave. O nó 0xD4213F, por sua vez, encaminha a mensagem para o nó 0xD462BA. Caso um nó não conheça outro nó que possua um maior prefixo em comum com a chave, ele encaminha a mensagem para um nó que compartilhe com a chave um prefixo do mesmo tamanho do que o prefixo que ele compartilha. Porém, esse prefixo tem que ser numericamente mais próximo da chave. É o que ocorre quando a mensagem é encaminha do nó 0xD462BA para o nó 0xD467C4, que nesse exemplo é a raiz da chave.

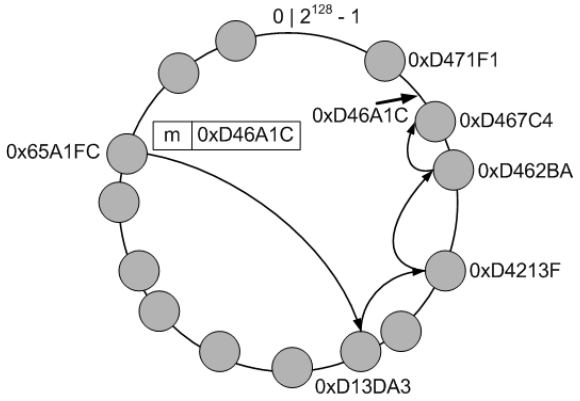


Figura 3.11. O roteamento com o Pastry [Castro et al. 2002b].

Construção das Múltiplas Árvores de Distribuição

A comunicação de grupo na rede sobreposta construída pelo Pastry é implementada pelo Scribe. Para tanto, a cada grupo é associado um identificador, chamado de *groupId*. Esse identificador é a raiz da árvore de distribuição e, portanto, deve ser atribuído a cada um dos subfluxos. Para se inscrever em um grupo, um nó deve enviar uma mensagem para o identificador desse grupo. Essa mensagem será roteada na rede sobreposta até atingir um nó pertencente à árvore de distribuição. Quando isso ocorre, a rota atravessada pela mensagem é adicionada à árvore de distribuição e o nó que a enviou passa a fazer

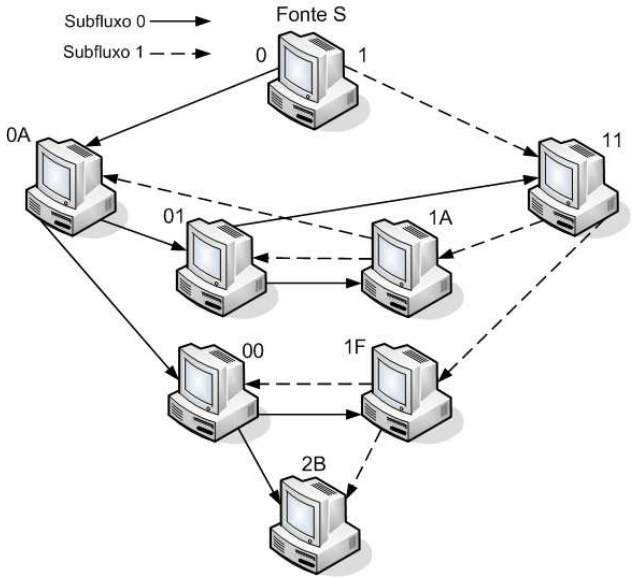
parte do grupo. Sendo assim, a árvore de distribuição para cada grupo é formada pela união das rotas, definidas pelo Pastry, entre cada membro do grupo e a raiz da árvore. As mensagens são enviadas pela raiz para os membros usando o encaminhamento pelo caminho reverso (*Reverse Path Forwarding* - RPF) [Costa e Duarte 2003].

O SplitStream usa o Scribe para construir as árvores de distribuição para cada um dos subfluxos. O objetivo é construir um conjunto de árvores disjuntas em relação aos nós internos. Um conjunto de árvores tem essa propriedade se cada nó é interno em apenas uma das árvores e folha nas demais. Para tanto, o SplitStream se baseia nas propriedades do roteamento implementado pelo Pastry. Como visto, o Pastry encaminha mensagens na direção dos nós cujos identificadores progressivamente compartilham prefixos de maior comprimento com a chave da mensagem. Uma vez que as árvores construídas pelo Scribe são formadas pelas rotas de todos os nós participantes para o identificador do grupo, os identificadores de todos os nós internos possuem em comum um dado número de dígitos iniciais com o identificador de grupo da árvore. Dessa forma, é possível garantir que um conjunto de árvores é disjunto em relação ao nó interno escolhendo simplesmente identificadores de grupo para cada árvore que diferem no dígito mais significativo. A Figura 3.12 exemplifica esse procedimento. O conteúdo da fonte é dividido em dois subfluxos e é encaminhado em duas árvores independentes. O subfluxo 0 recebe o identificador de grupo 0 e o subfluxo 1 recebe o identificador 1. As linhas cheias correspondem aos ramos da árvore que encaminha o subfluxo 0 e as tracejadas aos ramos da árvore usada para encaminhar o subfluxo 1. As árvores construídas de acordo com os identificadores dos nós são mostradas simultaneamente na Figura 3.12(a). Já as Figuras 3.12(b) e 3.12(c) mostram cada uma das árvores separadamente. Pode-se notar que nas árvores construídas para cada subfluxo, somente os nós com o identificador iniciado pelo identificador do subfluxo são nós internos. Os demais sempre são folhas.

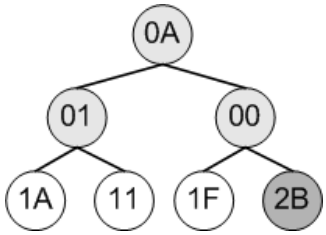
3.3.2. Arquitetura de Distribuição em Malha

A idéia básica da arquitetura de distribuição de vídeo em malha é a mesma de sistemas de compartilhamento de arquivos como o BitTorrent [Hales e Patarin 2005]. Nesses sistemas, um arquivo é dividido em pedaços (*chunks*) que são espalhados pelos nós participantes. Na distribuição de vídeo, ao invés de um arquivo, divide-se um fluxo de vídeo em pedaços. Como os nós se organizam e interagem para receber e encaminhar um determinado conteúdo, assim como fazem as abelhas em uma colméia, o conjunto de nós participantes é geralmente chamado de enxame (*swarm*).

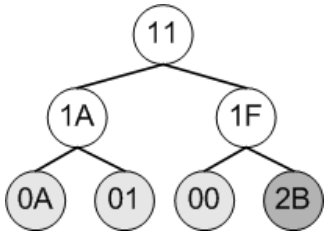
A principal diferença da arquitetura em malha para a arquitetura em árvore é que na primeira não se constrói e nem se mantém uma estrutura explícita para o encaminhamento do fluxo de vídeo. Argumenta-se que ao invés de reparar constantemente a estrutura de distribuição, em virtude da dinâmica de entrada e saída de participantes em um sistema par-a-par, é mais eficiente disseminar a disponibilidade dos pedaços de vídeo entre os nós para que cada nó solicite explicitamente os pedaços que deseja. Para tanto, cada nó participante mantém um conjunto de nós, geralmente chamados de parceiros, com os quais periodicamente troca informações sobre os pedaços do vídeo que possui. Assim, um nó sabe quais parceiros possuem quais pedaços do vídeo e, dessa forma, requisita o envio dos pedaços desejados aos respectivos parceiros. Por isso, diz-se que na arquitetura em malha os receptores “puxam” os pedaços do vídeo. Vale ressaltar que os pedaços são



(a) As múltiplas árvores.



(b) A árvore para o subfluxo 0.



(c) A árvore para o subfluxo 1.

Figura 3.12. A construção das árvores disjuntas com o Scribe.

enviados usando comunicações ponto-a-ponto na camada de aplicação entre um nó e seus parceiros.

A vantagem da arquitetura em malha é ser menos susceptível à entrada e à saída de nós do que a arquitetura em árvore. Primeiro, porque o conteúdo de vídeo é dividido em pedaços que estão disponíveis em diversos nós. Segundo, porque um nó recebe esses diferentes pedaços de diferentes parceiros e não somente de um nó pai, como ocorre em uma árvore de distribuição. Essas duas características reduzem a possibilidade de descontinuidades de recepção. Mesmo com a saída de um parceiro, um nó continua a receber os pedaços de vídeo, já que os pedaços que eram armazenados pelo nó que deixou o sistema podem ainda ser encontrados nos parceiros remanescentes. Por outro lado, a sobrecarga de controle, geralmente, é maior na arquitetura em malha do que na arquitetura em árvore, pois é necessário trocar as informações sobre a disponibilidade dos pedaços de vídeo entre os parceiros e enviar as requisições dos pedaços. Como não há uma estrutura explícita de distribuição na arquitetura em malha, os atrasos de inicialização e encaminhamento do vídeo tendem a ser maiores do que na arquitetura em árvore. Além disso, o desempenho dos sistemas baseados na arquitetura em malha depende do tamanho dos *buffers* nos nós, já que os pedaços de vídeo podem ser recebidos fora da ordem de reprodução e terão de ser armazenados. Também, quanto maiores os *buffers* dos nós, mais pedaços de vídeo

eles podem armazenar. Isso aumenta a disponibilidade do vídeo na rede, favorecendo a aplicação.

Na Seção 3.3.2.1, descreve-se o CoolStreaming/DONet, um exemplo de sistema par-a-par de distribuição de vídeo que utiliza a arquitetura em malha.

3.3.2.1. CoolStreaming/DONet

O CoolStreaming (*Cooperative Overlay Streaming*) é a implementação do sistema DONet (*Data-driven Overlay Network*) [Zhang et al. 2005c]. O funcionamento do CoolStreaming se baseia na divisão de um determinado vídeo em pedaços. Sendo assim, cada participante troca periodicamente informações sobre a disponibilidade dos pedaços com um conjunto de nós, chamados de parceiros. Dessa forma, um nó solicita aos seus parceiros os pedaços de vídeo que não possui e disponibiliza os pedaços que possui. Para cada pedaço de vídeo, um nó pode ser tanto receptor quanto emissor, ou até mesmo ambos. A exceção é a fonte do vídeo, que sempre é emissor, e por isso é chamada de nó origem. A fonte pode ser um servidor de vídeo dedicado ou simplesmente um nó participante que possui um conteúdo de vídeo para distribuir.

A principal característica do CoolStreaming é não manter uma estrutura de distribuição global para encaminhar um fluxo de vídeo. Um nó se conecta à rede sobreposta através de seus parceiros e dessa forma a malha de distribuição é formada. Por isso, o CoolStreaming é classificado como um sistema baseado na arquitetura em malha, apesar de não manter uma estrutura de distribuição global. Para cada fonte, forma-se uma malha de distribuição. Os nós também não possuem funções predefinidas, como os nós pais e nós filhos em uma árvore. Todo nó participante do sistema tem de encaminhar o conteúdo de vídeo para outros nós que desejam receber esse conteúdo. Diz-se que, por não haver uma estrutura que restrinja a direção do encaminhamento do vídeo, a disponibilidade é que direciona o encaminhamento de um fluxo de vídeo.

Um nó participante do CoolStreaming deve ser capaz de manter uma visão parcial dos outros participantes, estabelecer e manter parcerias com outros participantes e escalonar a transmissão dos pedaços de vídeo. Essas tarefas são, respectivamente, de responsabilidade do gerenciador de participantes, do gerenciador de parceiros e do escalonador, mostrados na Figura 3.13. Nas seções seguintes, descreve-se como um novo nó participa da distribuição e como seus parceiros são selecionados. Discute-se também como a informação de disponibilidade de pedaços é mapeada e difundida entre os parceiros, bem como as medidas de recuperação da malha de distribuição quando há uma falha ou a saída de um nó.

Construção da Malha de Distribuição

Cada nó participante do CoolStreaming possui um identificador único, tal como o endereço IP, e mantém um *cache* de participantes, chamado de (*membership cache* - mCache), que contém uma lista parcial de identificadores dos nós ativos do sistema. A razão pela qual essa lista é parcial é explicada adiante. Para entrar no sistema, um nó

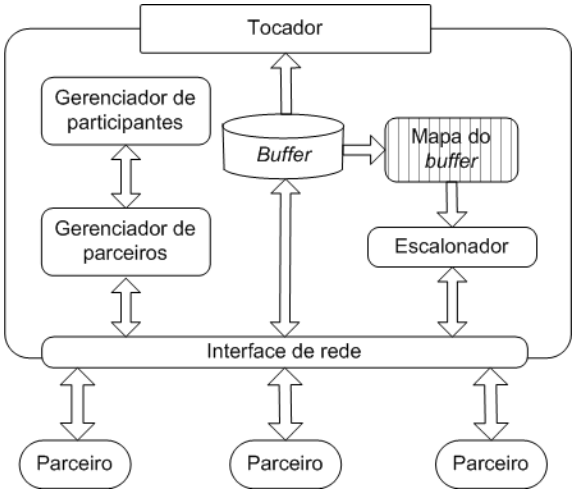


Figura 3.13. A arquitetura de um nó participante do sistema CoolStreaming [Zhang et al. 2005c].

deve primeiro contatar o nó origem. Ao ser contatada, a origem escolhe aleatoriamente um nó do seu mCache que será o nó adjunto. Em seguida, a origem redireciona o novo nó para o nó adjunto que, por sua vez, envia uma lista de possíveis parceiros ao novo nó. Uma vez recebida a lista, o novo nó contata os candidatos, estabelece as parcerias e passa a fazer parte da malha de distribuição. O uso de nós adjuntos para enviar a lista de possíveis candidatos reduz a carga do nó de origem e torna mais uniforme a seleção de parceiros [Zhang et al. 2005c]. Para que o procedimento descrito seja viável o nó origem tem de estar ativo durante a distribuição do vídeo e o seu endereço e/ou identificador tem de ser conhecido. A Figura 3.14 exemplifica a entrada de um nó no sistema. Nesse exemplo, o nó *N* deseja participar do sistema e, então, envia uma mensagem para a fonte *S* (passo 1). Assume-se que o identificador de *S* é conhecido pelo nó *N*. Ao receber a mensagem de *N*, a fonte *S* responde com o identificador do nó adjunto, nesse caso o nó *D* (passo 2). Com a resposta, o nó *N* envia uma outra mensagem para o nó adjunto *D* (passo 3), que responde a *N* com a lista de possíveis candidatos a parceiros, no caso *A* e *B* (passo 4). Em seguida, *N* envia uma mensagem para estabelecer parcerias com os nós *A* e *B* (passo 5). Como ambos respondem positivamente (passo 6), os enlaces entre os nós *N* e *A* e entre os nós *N* e *B* são criados (passo 7). A partir desse ponto, *N* se torna um participante do sistema.

Um ponto fundamental do funcionamento do CoolStreaming é a criação e a atualização do *cache* de participantes. Para acompanhar a dinâmica da malha, cada nó periodicamente envia uma mensagem para anunciar sua existência e, conseqüentemente, mostrar que é um participante ativo. Cada mensagem possui quatro campos: o número de seqüência da mensagem, o identificador do nó, o número atual de parceiros do nó e o tempo de vida, que indica o tempo de validade remanescente da mensagem. As mensagens são difundidas através do protocolo SCAM (*Scalable Gossip Membership*) [Ganesh et al. 2003]. Como na maioria dos protocolos baseados em fofocas, no SCAM um nó gera uma mensagem de existência e a envia para um conjunto aleatório de nós escolhidos do seu *cache* de participantes. Ao receberem essa mensagem, os nós escolhidos também sorteiam outros nós, armazenados nos seus *caches* de participantes, e encaminham a men-

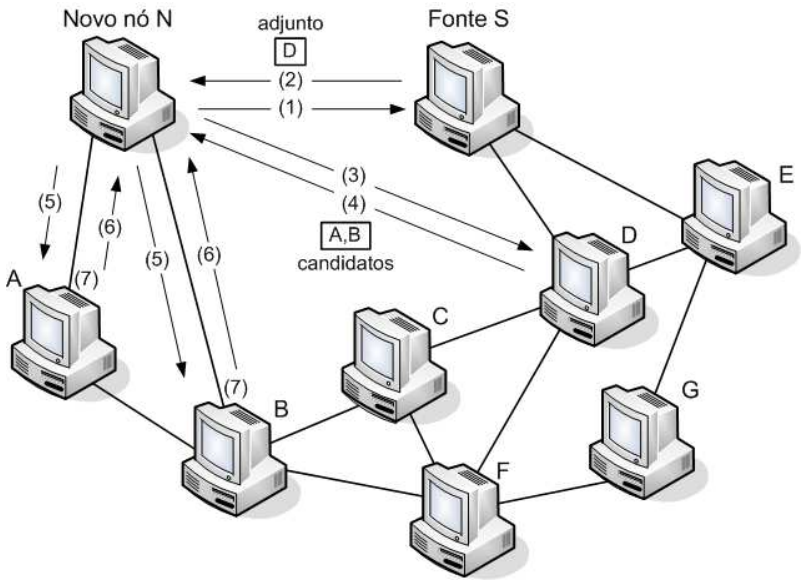


Figura 3.14. A entrada de um novo nó no sistema CoolStreaming.

sagem recebida para os nós sorteados. Dessa forma, a mensagem se espalha pela rede. Ao receber uma mensagem de existência com um número de seqüência não conhecido, o nó verifica o identificador de nó contido nessa mensagem. Caso o nó seja conhecido, a entrada do *cache* de participantes referente a esse nó é atualizada. Do contrário, se o nó não é conhecido, uma nova entrada é criada. Uma entrada do *cache* de participantes contém os quatro campos contidos na mensagem de existência e mais um campo no qual é armazenado o instante da última atualização da entrada. Diz-se que o mCache contém uma lista parcial dos participantes do sistema devido à aleatoriedade do algoritmo usado para difundir as mensagens de existência e por não haver troca do *cache* de participantes entre os nós do sistema.

Um nó periodicamente pode estabelecer novas parcerias com outros nós sorteados do seu *cache* de participantes. Esse procedimento tem duas finalidades. A primeira é auxiliar um nó a manter um número estável de parceiros, em virtude de saída ou falha de nós. A segunda é fazer com que um nó encontre parceiros de melhor qualidade. Para medir a qualidade de um parceiro, um nó define uma pontuação que varia de acordo com a quantidade de pedaços recebidos desse parceiro. Seja um nó i, parceiro de um nó j. A pontuação de j calculada por i é dada por $max\{\overline{s}_{i,j}, \overline{s}_{j,i}\}$, onde $\overline{s}_{i,j}$ representa o número médio de pedaços que o nó i recebeu de j por unidade de tempo e $\overline{s}_{j,i}$ o número médio de pedaços que i enviou para j por unidade de tempo. Dessa forma, como um nó pode ser receptor e emissor, considera-se o valor máximo entre as duas direções. Intuitivamente, um parceiro com mais pedaços disponíveis e maior capacidade para enviar os pedaços terá uma maior pontuação e, conseqüentemente, será um parceiro de maior qualidade. Sempre que um novo parceiro com maior pontuação é escolhido, o nó tem de retirar de sua lista o parceiro com a menor pontuação para manter o número de parceiros estável.

O número de parceiros influencia na sobrecarga de controle e na continuidade da recepção do vídeo. Quanto maior o número de parceiros, maior a sobrecarga de controle.

Entretanto, a parcela do tráfego de controle é muito pequena se comparada ao tráfego de vídeo. Em experimentos realizados com o CoolStreaming, o tráfego de controle é menor que 2% do tráfego total [Zhang et al. 2005c]. Quanto maior é o número de parceiros, maior também é a continuidade do vídeo, pois cada nó possui mais opções de emissores. Entretanto, a partir de um determinado valor o ganho na continuidade é marginal. Nos experimentos com o CoolStreaming para malhas de distribuição de diferentes tamanhos, esse valor é igual a quatro [Zhang et al. 2005c].

Algoritmos de Escalonamento de Pedaços

O algoritmo de escalonamento de pedaços é fundamental para garantir os requisitos de banda passante e tempo da distribuição de vídeo. Assim como nos sistemas de compartilhamento de arquivos, os pedaços podem ser recebidos fora de ordem, mas devem ser recebidos antes do seu tempo de reprodução. Além disso, na difusão de vídeo, o progresso da reprodução do vídeo é fortemente sincronizado entre os participantes, pois os nós têm interesse em um dado conteúdo durante um mesmo período de tempo e também porque não podem controlar as ações de reprodução do vídeo. Os resultados dos experimentos realizados com o CoolStreaming mostram que o intervalo entre os trechos de um mesmo vídeo reproduzidos por diferentes participantes não é superior a um minuto [Zhang et al. 2005c]. Esse problema não é observado no compartilhamento de arquivos, uma vez que o interesse em um arquivo, geralmente, não está associado a um dado período de tempo, como ocorre, por exemplo, em transmissões ao vivo. Portanto, cabe ao algoritmo de escalonamento determinar qual a maneira mais eficiente para um nó simultaneamente receber e encaminhar os pedaços de vídeo.

No CoolStreaming, um fluxo de vídeo é dividido em pedaços de tamanho uniforme. A disponibilidade desses pedaços no *buffer* de um nó é representada por um mapa de *buffer* (*Buffer Map* - BM). Como visto, os pedaços de vídeo só são úteis se forem recebidos antes do seu tempo de reprodução. Por isso, o mapa de *buffer* é representado por uma janela deslizante, como mostra a Figura 3.15. Nesse exemplo, o *buffer* pode armazenar até doze pedaços e o valor da janela deslizante é igual a seis pedaços. Os retângulos preenchidos indicam os pedaços disponíveis no *buffer*. Experimentos mostram que para pedaços contendo um segundo de vídeo, uma janela de 120 pedaços é suficiente para representar o mapa de *buffer*. Em outras palavras, nesse caso, um pedaço é armazenado por um nó durante dois minutos. Essa é a configuração padrão do protótipo do CoolStreaming. Sendo assim, utiliza-se uma palavra de 120 bits para codificar o mapa de *buffer*. Um bit assume o valor 1 se o pedaço correspondente está disponível. Do contrário, o bit assume o valor 0. No exemplo da Figura 3.15, o mapa de *buffer* codificado corresponde à palavra 101110. Cada pedaço é identificado por um número de seqüência, cujo comprimento é 2 bytes. Somente o número do primeiro pedaço da janela deslizante é armazenado. Por isso, são necessários mais 2 bytes. Como o número de seqüência é incremental e o tamanho da janela é fixo, é possível identificar os pedaços na janela em um dado período.

Cada nó envia periodicamente o seu mapa de *buffer* para seus parceiros. Com isso, um nó conhece quais pedaços do vídeo seus parceiros possuem e, dessa forma, pode

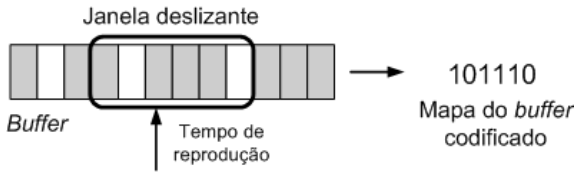


Figura 3.15. O mapa de *buffer* no sistema CoolStreaming.

definir de qual parceiro e quando vai requisitar um dado pedaço. Essa é a função do escalonador de pedaços que deve ainda lidar com duas restrições: o tempo de reprodução de cada pedaço e a heterogeneidade dos receptores. Esse problema é uma variação do escalonamento de máquinas paralelas, que é um problema NP-completo [Lenstra et al. 1990]. Além disso, o algoritmo de escalonamento deve se adaptar rapidamente à dinâmica da malha de distribuição. Por isso, o escalonador usado no CoolStreaming define uma heurística simples e de resposta rápida às variações da malha.

A heurística de escalonamento de pedaços funciona da seguinte forma. Inicialmente, um nó calcula o número de emissores potenciais para cada pedaço, isto é, os parceiros que possuem um dado pedaço armazenado em seus *buffers*. O algoritmo de escalonamento determina o emissor de cada pedaço de acordo com o número de emissores potenciais de cada pedaço e também com a capacidade de saída de cada possível emissor. Assume-se que um pedaço com menos emissores potenciais tem uma maior chance de ser recebido após o seu tempo de reprodução. Por isso, o algoritmo prioriza pedaços que possuam um menor número de emissores potenciais. Se houver mais de um possível emissor por pedaço, escolhe-se aquele com maior banda passante e que possua disponibilidade para enviar o pedaço antes do seu tempo de reprodução. O algoritmo de escalonamento é executado periodicamente por cada nó e, dessa forma, se define a escala de pedaços a serem requisitados a um mesmo emissor. Uma vez definida a escala para um emissor, utiliza-se uma seqüência de bits semelhante ao mapa de *buffer* para representá-la, na qual os bits em 1 representam os pedaços desejados. Cada escala é enviada para o seu respectivo emissor que, ao recebê-la, envia os pedaços requisitados ordenadamente.

Vale ressaltar que o nó origem é unicamente um emissor e sempre possui todos os pedaços do vídeo disponíveis. Para evitar a sobrecarga de requisições de seus parceiros, a origem implementa um algoritmo de escalonamento adaptativo. Se necessário, a origem difunde um mapa de *buffer* conservativo para reduzir a sua carga. A idéia é enviar para determinados parceiros um mapa que não contém todos os pedaços disponíveis, ou seja, com alguns bits zerados. Dessa forma, os parceiros que receberem esse mapa "incompleto" irão requisitar os pedaços "não-disponíveis" de outros parceiros e não mais da origem.

Recuperação de Falhas

Um nó pode deixar a malha de distribuição a qualquer instante, bem como pode falhar. Nas duas situações, a ausência de um nó pode ser facilmente detectada pelos demais nós depois de um período de inatividade e de trocas de mapas de *buffer*. Como

a probabilidade de falhas simultâneas de parceiros é pequena, o nó afetado pela saída de um parceiro reage rapidamente, sem que haja necessidade de executar mecanismos específicos de recuperação. Basta apenas que os pedaços que seriam enviados pelo nó que deixou a malha sejam reescalonados de acordo com os mapas do *buffer* dos parceiros remanescentes.

Apesar de não serem necessários mecanismos específicos, o CoolStreaming implementa dois procedimentos para aumentar a robustez do sistema em relação à saída de nós. Quando um nó deixa o sistema por vontade própria ele deve enviar uma mensagem de partida (*departure message*). Por outro lado, quando um nó falha, a mensagem de partida é enviada pelos parceiros que detectam a falha. A mensagem de partida tem o mesmo formato da mensagem de existência, porém o campo que contém o número de parceiros tem o valor -1. Também como a mensagem de existência, ela é difundida pela rede usando um algoritmo baseado em fofocas. Como diferentes parceiros podem detectar a falha de um nó, mensagens de partida duplicadas podem ser recebidas por um dado nó. Nesse caso, somente a primeira mensagem recebida é encaminhada e as demais são descartadas. Cada nó ao receber uma mensagem de partida apaga a entrada relativa ao nó que deixou a rede do seu *cache* de participantes.

O módulo cliente do sistema CoolStreaming está disponível para os sistemas operacionais Windows e também na forma de *plugin* para ser embutido em navegadores *web* [Coolstreaming 2008].

3.3.3. Arquitetura de Distribuição Híbrida

O objetivo da arquitetura em árvore é reduzir a latência no encaminhamento do vídeo. Para tanto, implementa-se a comunicação multidestinatária na camada de aplicação. Apenas uma cópia do fluxo de vídeo é enviada pela fonte. Os nós internos da árvore são responsáveis pela replicação do conteúdo e o encaminhamento para os nós descendentes. Por outro lado, a arquitetura em malha busca aumentar a disponibilidade do conteúdo para tornar o sistema mais robusto à dinâmica de entrada e saída de pares. No entanto, o encaminhamento dos pedaços é menos eficiente se comparado à replicação em uma árvore, pois são estabelecidas comunicações ponto-a-ponto entre um nó e seus parceiros. Portanto, nem a arquitetura em árvore nem a arquitetura em malha resolvem completamente os desafios introduzidos pela dinâmica dos sistemas par-a-par na distribuição de vídeo. A arquitetura em árvore sofre com a instabilidade provocada pela saída de participantes, com a sobrecarga de controle para manter a malha de participantes conectada e com a subutilização de banda em virtude do número de nós folhas e da possível escolha ineficiente de nós pai. Já na arquitetura em malha, apesar de esta ser mais simples, existe o compromisso entre a latência e a sobrecarga de controle. Se os participantes enviarem notificações para cada pedaço de vídeo recebido, a sobrecarga de controle aumenta. Aproveitar essas notificações para periodicamente enviar os mapas de *buffer* reduz a sobrecarga de controle, mas aumenta a latência. Portanto, uma questão que surge é como combinar as duas arquiteturas para construir uma estrutura de distribuição híbrida, que seja eficiente e robusta.

Existem diversas propostas de arquiteturas híbridas [Zhang et al. 2005a, Venkataraman et al. 2006, Castro et al. 2006]. Uma delas é chamada de arquitetura de *backbone*

em árvore (*treebone-based*) [Wang et al. 2007]. Essa arquitetura representa uma solução intermediária entre as arquiteturas em árvore e em malha. Não é necessário manter uma estrutura de comunicação prévia entre os participantes, como na arquitetura em árvore, mas define-se uma hierarquia entre os nós, o que não ocorre na arquitetura em malha. A arquitetura de *backbone* em árvore se baseia no fato de que a maioria dos pedaços de vídeo encaminhados na malha de distribuição segue uma árvore de distribuição específica ou um pequeno conjunto de árvores. A similaridade entre as árvores, definida como a fração de enlaces em comum entre as árvores, pode ser maior do que 70% [Wang et al. 2008]. Dessa forma, conclui-se que o desempenho da rede sobreposta depende do conjunto e da organização dos nós internos em comum. A idéia da arquitetura de *backbone* em árvore é, portanto, organizar esse conjunto de nós internos em comum, chamados de nós de núcleo, para tornar a distribuição de vídeo mais eficiente. Assume-se que esse conjunto é formado em sua maioria por nós estáveis e, dessa forma, tais nós serão os nós internos da árvore de distribuição. Cria-se assim um *backbone*, daí a denominação da arquitetura. Os demais nós se organizam em uma malha de distribuição. Espera-se que essa estrutura aumente a eficiência da distribuição de vídeo reduzindo a sobrecarga de controle e a latência, pois a maioria dos pedaços de vídeo será encaminhada pelo *backbone* ou por parte dos enlaces que o compõem. A Figura 3.16 ilustra um exemplo de estrutura de distribuição construída pela arquitetura de *backbone* em árvore. As linhas contínuas representam os enlaces da árvore de distribuição e as linhas tracejadas os enlaces da malha. A questão chave dessa arquitetura híbrida é identificar e posicionar os nós estáveis na árvore de distribuição. Nem sempre os nós mais estáveis são aqueles com mais capacidade de banda passante e, portanto, se forem colocados como nós internos da árvore prejudicarão seus descendentes de maior capacidade. Além disso, a estabilidade em sistemas par-a-par depende do comportamento humano, ou seja, de quando um usuário decide entrar no sistema.

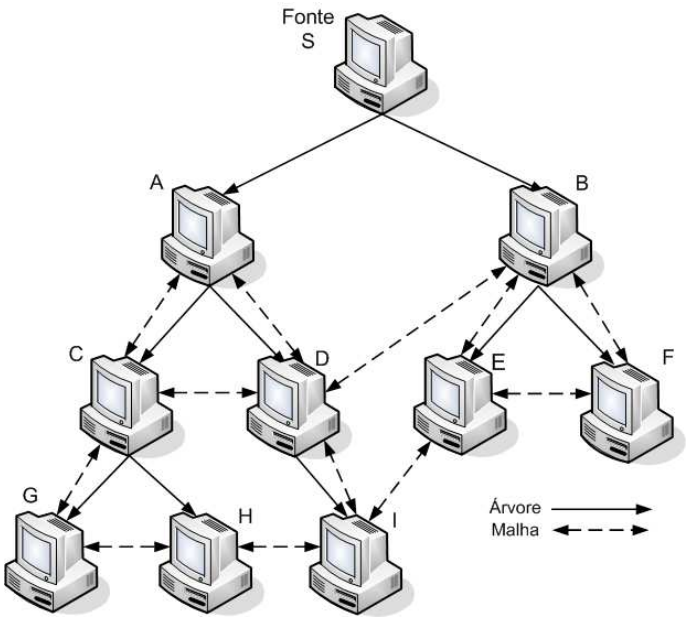


Figura 3.16. Um exemplo da arquitetura de *backbone* em árvore.

3.4. Sistemas de Distribuição de Vídeo Par-a-Par

Diversos sistemas de distribuição de vídeo sobre redes par-a-par estão sendo desenvolvidos, tanto pela comunidade acadêmica quanto pela indústria. O ESM e o CoolStreaming, descritos na seção anterior, são exemplos de sistemas de vídeo par-a-par desenvolvidos para fins de pesquisa. O PPLive, o SopCast e o GridMedia são exemplos de sistemas comercias de distribuição de vídeo par-a-par. Muitos desses sistemas já possuem milhares de participantes. O CoolStreaming, por exemplo, já registrou cerca de 50 mil usuários simultâneos no sistema e até 25 mil usuários assistindo ao mesmo canal [Liu et al. 2008]. Em janeiro de 2006, o PPLive atingiu a marca de mais de 200 mil usuários durante a distribuição de um programa popular da TV estatal chinesa. As taxas de envio variaram entre 400 e 800 kbps, resultando em uma taxa agregada de 100 Gbps [Sentinelli et al. 2007]. Também em 2006, o GridMedia foi usado para transmitir o Festival da Primavera na China. Estima-se que aproximadamente dois milhões de usuários assistiram a essa transmissão e que o sistema também suportou mais de 200 mil usuários simultâneos [Tang et al. 2007]. Esses dados comprovam o potencial dos sistemas par-a-par para distribuir vídeo de forma escalável.

A maioria dos sistemas de distribuição de vídeo é proprietária, sem especificação ou código fonte aberto. Por isso, muitas características relacionadas com a arquitetura interna do aplicativo não são conhecidas. Uma característica que pode ser estimada através de medições é o protocolo de transporte usado por cada sistema [Sentinelli et al. 2007, Silverston e Fourmaux 2007]. Outra característica detectada é que muitos dos sistemas de código fechado utilizam servidores para gerenciar informações dos pares da rede [Yiu et al. 2007]. Nessa seção, alguns dos sistemas de vídeo par-a-par são brevemente descritos. Os sistemas também são classificados de acordo com a arquitetura de distribuição que adotam: em árvore, malha ou híbrida.

3.4.1. Arquitetura em Árvore

NICE - O NICE [Banerjee et al. 2002a] foi desenvolvido inicialmente para distribuir vídeo a uma baixa taxa de dados para um grande número de receptores. Baseando-se no tempo de ida e volta entre os nós participantes, o NICE constrói uma hierarquia de nós. Dessa forma, cada nó deve manter informações mais precisas sobre pares mais próximos na hierarquia criada e informações menos precisas de pares mais distantes. Não é necessário que cada nó armazene informações sobre a topologia global da rede. Uma vez construída a hierarquia, é possível calcular as rotas sem a necessidade de novas trocas de mensagens entre os nós.

ZIGZAG - O ZIGZAG [Tran et al. 2003] foi desenvolvido tendo como objetivo o cumprimento de três requisitos básicos: atraso fim-a-fim limitado, robustez à dinâmica dos nós da rede e escalabilidade. Esses requisitos são alcançados lidando com um compromisso entre a sobrecarga de controle e o atraso fim-a-fim. O melhor atraso fim-a-fim possível seria alcançado caso a árvore de distribuição tivesse o grau igual a um. Entretanto, essa solução sobrecarregaria o nó fonte da árvore, pois ele teria que ser capaz de enviar vídeo a todos os nós descendentes. Para contornar esse problema, o ZIGZAG organiza os nós em grupos (*clusters*) de tamanhos limitados, definindo o grau máximo para cada árvore criada. Os grupos são organizados de forma hierárquica a fim de encontrar

uma solução intermediária para o compromisso entre atraso e sobrecarga. Para reduzir o tempo de recuperação da árvore em caso de falhas, o ZIGZAG utiliza um esquema de recuperação local.

3.4.2. Arquitetura em Malha

PPLive - Atualmente, o PPLive [Sentinelli et al. 2007] é o aplicativo par-a-par mais popular para a distribuição de vídeo. Esse aplicativo é de origem chinesa e é utilizado com bastante sucesso na exibição de programas populares da TV do mesmo país. O PPLive é similar o BitTorrent, sistema usado para o compartilhamento de arquivos. A diferença está nos requisitos de tempo que o PPLive deve garantir, já que o conteúdo transmitido é tipicamente ao vivo, como na TV aberta. Para lidar com os requisitos de atraso da aplicação, o PPLive utiliza dois *buffers* para armazenar os pedaços de vídeo recebidos. Esses *buffers* são empregados para dar tempo para a rede reagir a possíveis falhas e, assim, amenizar os efeitos da variação do atraso. Dos dois *buffers*, um é gerenciado pelo próprio sistema e o outro é gerenciado pelo aplicativo usado para exibir o vídeo. Ambos são carregados simultaneamente. Em versões recentes, além da difusão de canais de TV, o PPLive também possui um serviço de vídeo sob demanda.

SopCast - Criado por um estudante chinês, o SopCast [Sentinelli et al. 2007] se tornou um sucesso instantaneamente, atingido a marca de 100 mil usuários simultâneos somente após alguns meses do seu lançamento. O grande sucesso do SopCast se deve ao fato de que é possível produzir e distribuir o seu próprio conteúdo de vídeo, uma vez que um módulo servidor de vídeo é disponibilizado. Para tanto, o usuário deve se registrar e se autenticar no sistema. Uma característica do SopCast interessante é que o protocolo UDP é utilizado tanto para a distribuição do vídeo quanto para maioria das funções de controle. Essa conclusão foi obtida a partir de medidas realizadas com farejadores (*sniffers*) de rede [Silverston e Fourmaux 2007]. Assim como o PPLive, versões recentes do SopCast disponibilizam um serviço de vídeo sob demanda.

PPStream - O PPStream é outra aplicação chinesa de distribuição de vídeo. O PPStream somente utiliza o TCP tanto para transmissão de vídeo quanto para o controle. Esses resultados também foram obtidos com farejadores de redes [Silverston e Fourmaux 2007].

TVAnts - O TVAnts também é originário da China. O TVAnts utiliza os protocolos TCP e UDP para transmissão de vídeo e sinalização. Diferentes do SopCast e do PPStream, no TVAnts não há predominância de nenhum protocolo de transporte. Tanto o TCP quanto o UDP são utilizados em proporções semelhantes em todas as funções [Silverston e Fourmaux 2007]. O TVAnts também utiliza um esquema de armazenamento em dois níveis. Entretanto, ao contrário do PPLive, o *buffer* do tocador de vídeo só é carregado quando o *buffer* do sistema atinge um limiar próximo de sua capacidade total. O que aumenta o tempo de inicialização desse sistema.

3.4.3. Arquitetura Híbrida

GridMedia - O GridMedia [Tang et al. 2007, Zhang et al. 2005b] é mais um aplicativo chinês de grande sucesso. Nesse aplicativo, os nós se organizam em malha e trocam mapas de *buffer* para conhecer os pedaços de vídeo disponíveis em outros membros da

rede. Entretanto, diferente dos aplicativos comuns baseados na arquitetura em malha, o GridMedia pode enviar pedaços de vídeo sem receber requisições de pares. Essa forma de distribuição de vídeo é típica de aplicativos que adotam a arquitetura em árvore e, por isso, o GridMedia é considerado híbrido. Apesar da estratégia de envio de pedaços após requisições ainda ser a base empregada, o GridMedia pode adotar uma estratégia pró-ativa, ou seja, ele pode "empurrar" o pedaço de vídeo recém recebido aos vizinhos em sua lista. Essa estratégia somente é utilizada em momentos de estabilidade da rede, como por exemplo, quando não há um elevado número de entrada e saída de membros. Em momentos de instabilidade, a estratégia adotada é a de enviar pedaços de vídeo ("puxar pedaços de vídeo") apenas após requisições explícitas, já que essa estratégia é considerada mais robusta. A estratégia de "puxar" é também adotada pelos nós que entram na rede para começar a receber os primeiros pedaços de vídeo. O objetivo de se misturar as duas estratégias é reduzir a latência de transmissão, que pode ser considerável em decorrência de sobrecarga de requisições e de trocas de mapas de *buffer*.

CoolStreaming+ - O CoolStreaming+ [Li et al. 2007] é uma versão aprimorada e com características diferentes do seu predecessor, o CoolStreaming. A diferença principal é que, ao invés de sempre requisitar novos pedaços de vídeo aos outros pares da rede, o CoolStreaming+ somente realiza essa operação no início da transmissão de cada vídeo. O CoolStreaming+ parte da premissa que requisitar pedaços de vídeo de outros pares da rede aumenta a sobrecarga de controle e o atraso de transmissão. Assim, o CoolStreaming+ divide cada vídeo em subfluxos e apenas seleciona os pares que irão transmitir cada subfluxo. Essa seleção acontece como em aplicativos que adotam a estratégia de "puxar" o vídeo, que é o caso do CoolStreaming original. Após selecionar os pares, requisições para o início da transmissão dos subfluxos são enviadas. Entretanto, uma vez iniciada a recepção de cada subfluxo, o restante do conteúdo é enviado usando a estratégia de "empurrar", que é a mesma adotada pelos aplicativos baseados em árvores de distribuição. Nesse momento, a arquitetura da rede passa a ser semelhante a uma árvore de distribuição, na qual cada nó fonte de subfluxo é um pai e cada receptor, um descendente.

AnySee - No AnySee [Castro et al. 2006], os nós também se organizam em malha, porém, diferentemente de outras aplicações, os nós participantes não recebem os pedaços de vídeo simultaneamente de diferentes pares. Cada nó na malha mantém um caminho principal e um conjunto de caminhos secundários para uma determinada fonte de vídeo. Caso os pacotes recebidos pelo caminho principal não atendam os requisitos de tempo exigidos pelo vídeo, o nó passa a usar um dos caminhos alternativos. O objetivo desse mecanismo é reduzir a latência entre fonte e destino. Como conseqüência da escolha de caminhos principais que atendam requisitos de atraso e da manutenção de caminhos secundários, é possível utilizar *buffers* menores, já que a dinâmica dos nós da rede não afeta tanto o desempenho do Anysee. O emprego de *buffers* menores favorece a aplicação ao reduzir o atraso de início de exibição do vídeo, já que é mais rápido encher o *buffer*. Alguns experimentos [Sentinelli et al. 2007] mostram que o AnySee utiliza um *buffer* de aproximadamente 40 segundos enquanto que os principais sistemas em malha, como o PPLive e o SopCast, usam *buffers* entre 1 e 2 minutos. Mostra-se ainda que os atrasos de reprodução e inicialização no Anysee estão entre 20 e 30 segundos. Para os sistemas em malha, esses atrasos são superiores a um minuto.

3.5. Desafios

A pesquisa sobre sistemas de distribuição de vídeo sobre redes par-a-par apresentou grandes avanços nos últimos anos. No entanto, ainda existem problemas em aberto e diversos aspectos práticos que precisam ser investigados para que os sistemas par-a-par sejam adotados para a distribuição de vídeo em larga escala na Internet. Nessa seção, alguns desses desafios são discutidos.

3.5.1. Segurança

A segurança é um dos grandes desafios da distribuição de vídeo sobre redes par-a-par. De forma geral, as vulnerabilidades de sistemas par-a-par estão associadas à característica colaborativa desses sistemas, à comunicação multidestinatária e aos requisitos da distribuição de vídeo que dificultam a adoção de técnicas de segurança tradicionais propostas para sistemas par-a-par [Barcellos e Gaspary 2006]. Diferentemente do compartilhamento de arquivos, a distribuição de vídeo possui requisitos de tempo real, isto é, baixa latência e banda passante disponível. Esses requisitos proporcionam novas oportunidades de ataques e podem agravar os danos causados pelos ataques já existentes no compartilhamento de arquivos. Assim, é necessário adaptar as técnicas tradicionais de segurança para sistemas par-a-par e desenvolver mecanismos de segurança específicos para a distribuição de vídeo.

Os sistemas par-a-par assumem o correto funcionamento dos nós participantes do sistema. Essa característica faz desses sistemas vulneráveis a ataques, nos quais nós agem maliciosamente para prejudicar o funcionamento do sistema ou beneficiar-se de sua natureza colaborativa ao consumir mais recursos do que oferece ao sistema. As conseqüências das ações maliciosas são mais devastadoras ainda quando os ataques são realizados em conluio, devido à dificuldade de detecção e prevenção desse tipo de ação e ao número de nós legítimos que podem ser afetados. As motivações para essas ações são diversas, incluindo análise de tráfego de sistemas que tentam prover comunicação anônima, e censura contra sistemas que tentam prover alta disponibilidade de dados. Por exemplo, é bem sabido que os sistemas par-a-par de compartilhamento de arquivos sofrem intensos ataques da indústria fonográfica, cujo intuito é reduzir a troca ilegal de músicas protegidas por direitos autorais [Liang et al. 2005].

Uma alternativa para evitar ações maliciosas é a exclusão de nós não confiáveis usando autoridades certificadoras [Rowstron e Druschel 2001]. Nesse caso, o problema foi repassado para o mecanismo de reputação dos nós. Além disso, há muitas situações, como em canais abertos de distribuição de vídeo, nas quais não é desejável restringir o acesso ao sistema. Nessas situações, o sistema deve ser capaz de operar ainda que alguns nós participantes sejam maliciosos. Esse é o objetivo do roteamento seguro. O roteamento seguro [Wallach 2002] deve garantir que um nó legítimo consiga alcançar, com alta probabilidade, todos os nós do sistema. Isso deve ocorre mesmo que, durante o trajeto das mensagens, os nós intermediários possam corrompê-las, descartá-las ou encaminhá-las incorretamente; ou ainda tentem se passar por um nó legítimo. O roteamento seguro requer a solução de três problemas: atribuição segura de identificadores, manutenção segura das tabelas de roteamento e encaminhamento seguro de mensagens. A atribuição segura de identificadores garante que um atacante não possa escolher os identificadores

atribuídos aos nós que o atacante controla. Isso impede que um atacante controle, por exemplo, todos os nós responsáveis pelo armazenamento de pedaços de um determinado vídeo, ou ainda intermedeie todo o tráfego de um certo nó. A manutenção segura das tabelas de roteamento garante que a fração de nós maliciosos que consta nas tabelas de roteamento dos nós legítimos não excede, na média, a fração global de nós maliciosos. Na ausência desse mecanismo, um atacante poderia impedir a entrega de uma mensagem usando um número relativamente pequeno de nós em conluio. Finalmente, o encaminhamento seguro de mensagens garante que o nó destino recebe ao menos uma cópia da mensagem enviada pelo nó origem. Em suma, os três mecanismos que compõem o roteamento seguro garantem a disponibilidade do sistema no nível de roteamento. Porém, isso não é o bastante. O projeto de um sistema completo de segurança deve levar em conta as especificidades do ambiente a ser protegido. No caso dos sistemas de distribuição de vídeo sobre redes par-a-par, deve-se considerar também os requisitos de tempo real e alguns requisitos de segurança adicionais, como autenticação, integridade, autorização e controle do comportamento dos nós.

Em geral, os sistemas de distribuição de vídeo sobre redes par-a-par disponíveis comercialmente não consideram a presença de nós maliciosos na rede. A preocupação inicial era garantir de forma eficiente a distribuição contínua e em tempo real do conteúdo de vídeo. Sendo assim, a tolerância a comportamentos maliciosos é ainda um desafio. Ataques maliciosos a sistemas de distribuição de vídeo par-a-par exploram basicamente quatro vulnerabilidades, que são discutidas a seguir.

A primeira vulnerabilidade é a ausência de controle de acesso à rede sobreposta. Um atacante pode comprometer o protocolo de construção da rede sobreposta ou então o protocolo de gerenciamento de participantes, sobre o qual funciona a rede sobreposta. Um exemplo de dano causado por nós maliciosos é o particionamento da rede sobreposta, devido a respostas com rotas falsas. Outro exemplo ocorre em sistemas que constroem redes sobrepostas baseadas em anéis e que são vulneráveis a ataques do tipo “eclipse” [Singh et al. 2004]. Nesses ataques, o atacante controla um grande número de vizinhos dos nós legítimos, impedindo o funcionamento do protocolo de construção da rede sobreposta. Um nó legítimo é aquele que só faz ações corretas. Nós maliciosos podem ainda acusar os nós legítimos de estarem desligados. Um mecanismo proposto como solução para esse problema é o protocolo de gerenciamento de participantes *Fireflies* [Johansen et al. 2006]. Esse protocolo foi projetado para ser escalável e dar suporte a redes sobrepostas tolerantes a intrusão. O *Fireflies* provê aos nós participantes uma visão razoavelmente atualizada de quais nós estão ativos e garante que o conjunto de nós legítimos forma um subgrafo conectado. Assim, os nós legítimos não podem ser “eclipsados” pelos nós maliciosos. O *Fireflies* é usado como parte integrante do *SecureStream* [Haridasan e van Renesse 2006], um sistema de segurança para distribuição multidestinatária de vídeo capaz de prover tolerância a comportamentos maliciosos.

Outra vulnerabilidade é a ausência de autenticação e integridade fim-a-fim das mensagens. Isso permite a ocorrência do ataque de modificação, que corresponde à falsificação ou alteração do conteúdo original da mensagem. Um caso especial desse ataque é a injeção de pedaços poluídos na distribuição de vídeo. Como não é feita nenhuma verificação da autenticidade dos pedaços recebidos pelos pares, os pedaços poluídos são propagados e acabam por prejudicar a reprodução do vídeo no receptor. Dhungel *et al.*

apresentam um experimento real com o sistema PPLive [Vu et al. 2007] para mostrar o quão devastador pode ser o ataque da poluição [Dhungel et al. 2007]. Vale lembrar que o PPLive é um sistema baseado na arquitetura em malha. O número de participantes de um determinado canal de vídeo caiu rapidamente de 3300 para 500 após o início do ataque, indicando que a qualidade do vídeo reproduzido tornou-se inaceitável para a maioria desses participantes.

Os ataques de modificação poderiam ser evitados com o uso de uma infra-estrutura de chave pública (*Public Key Infrastructure* - PKI). Com a assinatura das mensagens pelo nó de origem os participantes poderiam verificar se houve modificação no conteúdo original da mensagem e também poderiam garantir que os dados recebidos são de fato oriundos do nó de origem. Portanto, os participantes poderiam concluir que os dados são autênticos e não prejudicam a qualidade do vídeo a ser reproduzido. No contexto de distribuição de conteúdo em tempo real, entretanto, o custo das assinaturas torna-se proibitivamente elevado, forçando a utilização de outras formas de autenticação. Dhungel *et al.* fazem uma análise comparativa de três diferentes técnicas de amortização de assinaturas: o encadeamento em estrela (*Star Chaining*), o encadeamento em árvore de Merkle (*Merkle-Tree Chaining*) e a abordagem "assinar e corrigir". Nessas técnicas os pedaços são agrupados em blocos, de tal forma que somente uma assinatura é necessária por bloco. Não obstante, cada pedaço pode ser verificado individualmente. O custo dessas vantagens é o uso de mais banda. Por exemplo, no encadeamento em estrela a cada pedaço é anexada a assinatura do bloco inteiro, a posição do pedaço no bloco e também os "*hashes*" de todos os outros pedaços do bloco. Os autores concluem que as duas últimas técnicas são as mais apropriadas para o contexto da distribuição de vídeo sobre redes par-a-par. O encadeamento em árvore de Merkle é mais eficiente em termos de sobrecarga computacional e de atraso no receptor, enquanto que a abordagem "assinar e corrigir", por utilizar códigos corretores de erro, é mais eficiente em termos de consumo de banda passante, ao custo de maior sobrecarga computacional e de um maior atraso no receptor.

A terceira vulnerabilidade é a ausência de controle ou limitação do comportamento dos nós. Com isso, ataques de negação de serviço podem comprometer a banda disponível para a transmissão de um fluxo de vídeo. Esses ataques podem prejudicar a disponibilidade do sistema ou fazer com que a qualidade do vídeo reproduzido chegue a níveis inaceitáveis pelos usuários. Uma forma de negação de serviço é sobrecarregar os participantes legítimos com pedidos de uma grande quantidade de "pedaços" de vídeo ou comprometer de alguma forma que os nós legítimos contribuam para a sessão de vídeo. Os protocolos *Ripple-stream* [Wang et al. 2006] e *Oversight* [Conner et al. 2006] são duas propostas para solucionar esse problema. Para identificar os atacantes e evitar a indisponibilidade do sistema o *Ripple-stream* utiliza um sistema de reputação baseado em créditos que permite avaliar o comportamento dos nós da rede. Além disso, é introduzido um mecanismo de seleção de pares que organiza a rede sobreposta de tal forma que os nós com alta credibilidade se localizem no núcleo da estrutura de distribuição. Como os nós maliciosos terão baixa credibilidade, eles serão empurrados para a borda da estrutura e, portanto, terão ação limitada. Já o *Oversight* utiliza uma abordagem de limitar a taxa de recepção do conteúdo definida pelos nós. Dessa maneira, o protocolo impede que nós egoístas ou maliciosos utilizem uma quantidade de dados que poderia esgotar os recursos do sistema.

Finalmente, a última vulnerabilidade é a ausência de incentivo à cooperação entre nós. Como mencionado anteriormente, os sistemas par-a-par pressupõem o correto funcionamento dos nós participantes do sistema. Assim, quando um nó deixa de colaborar com os outros nós, a eficiência do sistema fica comprometida. Dada a ênfase na entrega de dados com baixo atraso, a falta de colaboração dos nós torna-se ainda mais grave no contexto da distribuição de vídeo. Um exemplo de ataque que explora essa vulnerabilidade é o ataque da omissão. Ao não encaminhar todos ou parte das mensagens recebidas, nós maliciosos ou egoístas podem comprometer inteiramente a disponibilidade do sistema, visto que uma mensagem recebida após o prazo que ela poderia ser usada na reprodução do vídeo é inútil.

Nos sistemas baseados na arquitetura em árvore também existem vulnerabilidades associadas ao encaminhamento de dados típico da comunicação multidestinatária (*multicast*). Nesse tipo de comunicação, todos os nós participantes recebem os pacotes enviados pela fonte, que é a raiz da árvore de distribuição. Judge e Ammar [Judge e Ammar 2003] apontam os aspectos de segurança que são afetados pelas as propriedades fundamentais da comunicação multidestinatária e como essas propriedades causam vulnerabilidades. Destacam-se três propriedades da comunicação multidestinatária: a participação nos grupos é aberta, todos os membros recebem todos os pacotes enviados ao endereço de grupo e qualquer nó é livre para enviar pacotes para o grupo. Essas propriedades causam vulnerabilidades por criarem novas oportunidades de ataque ou porque as soluções existentes para a comunicação ponto-a-ponto (*unicast*) não se aplicam ao caso da comunicação multidestinatária. O problema da primeira propriedade é a impossibilidade de restringir a comunicação a um conjunto de nós autorizados. Assim, são possíveis ataques como a espionagem e a negação de serviço. A solução para esse problema é a encriptação dos dados de grupo, que depende de mecanismos de gerenciamento de chaves de grupo e de controle de acesso dos receptores. A segunda propriedade faz com que mecanismos de segurança usados na comunicação ponto-a-ponto baseados na individualização dos pacotes para impedir duplicação e propagação não autorizadas não funcionem na comunicação multidestinatária. Por fim, a terceira propriedade acarreta dois problemas. Em primeiro lugar, os membros do grupo precisam ser capazes de verificar que as mensagens recebidas são realmente oriundas do nó de origem alegado. Outro problema é a possibilidade de ocorrência de ataques de negação de serviço por parte de um nó não autorizado. A solução é o uso de mecanismos de controle de acesso da origem a fim de impedir que nós não autorizados enviem dados ao grupo.

3.5.2. Incentivos à Cooperação

Em sistemas par-a-par, observa-se na prática que muitos usuários contribuem com menos recursos do que consomem [Agarwal et al. 2007]. Esse comportamento caracteriza o problema dos nós oportunistas (*free-riders*). Conseqüentemente, ao falhar a cooperação, interesses individuais podem levar o sistema ao colapso. Para reduzir esse tipo de ação, os sistemas par-a-par devem prover mecanismos que incentivem o comportamento cooperativo. Alguns trabalhos recentes propõem o uso de mecanismos de reputação ou de dinheiro virtual como solução para evitar comportamentos egoístas [Zhang et al. 2007]. Nós com baixa reputação ou com pouco dinheiro virtual recebem menos recursos do sistema, enquanto que os nós colaborativos podem consumir mais recursos, usufruindo

do seu bom retrospecto. Esquemas semelhantes foram propostos para a distribuição de vídeo [Habib e Chuang 2004, Tan e Jarvis 2006]. Tais esquemas tipicamente requerem uma arquitetura centralizada para o gerenciamento do histórico dos nós, o que compromete a escalabilidade do sistema. Uma exceção é o sistema proposto por Qiu *et al.*, que utiliza um mecanismo de incentivo baseado em créditos que opera utilizando somente as informações locais de cada nó [Qiu et al. 2007].

Estudos do sistema de compartilhamento de arquivos BitTorrent [Bharambe e Padmanabhan 2006] mostram que a estratégia "olho-por-olho" (*tit-for-tat*) é eficaz na redução de nós oportunistas. Além disso, essa estratégia não sofre do problema de escalabilidade, visto que os nós avaliam a contribuição dos vizinhos de maneira distribuída. Entretanto, o projeto de um mecanismo de incentivos para a distribuição de vídeo em tempo real é mais desafiador do que o do compartilhamento de arquivos tradicional. No compartilhamento de arquivos, a medida primária de desempenho é o tempo para se receber completamente um arquivo. Já na distribuição de vídeo em tempo real, a qualidade do vídeo reproduzido é a medida primária de desempenho. Assim, o mecanismo natural para incentivar a cooperação dos nós é prover uma qualidade de vídeo proporcional aos recursos disponibilizados pelo nó. Nesse sentido, foi proposto o uso de codificação em múltiplos descritores, apresentada na Seção 3.2.3, a fim de prover uma estratégia similar a do "olho-por-olho" para sistemas de distribuição de vídeo de arquitetura em malha [Liu et al. 2007c]. A idéia é que os nós que contribuem com mais banda recebam mais descritores e, com isso, uma qualidade de vídeo melhor. Isso incentiva a contribuição dos nós e pune os nós oportunistas que, por não contribuírem com o sistema, são pouco servidos pelos vizinhos e podem reproduzir um vídeo com qualidade inaceitável. Uma outra proposta é o uso da codificação em camadas, que obtém uma melhor qualidade de vídeo devido à sua melhor eficiência de codificação [Liu et al. 2007b]. Porém, essa técnica introduz dois desafios: a importância desigual das diferentes camadas e a disponibilidade das camadas mais altas. Devido à dependência entre as camadas, as camadas inferiores devem ter prioridade de transmissão maior do que as camadas mais elevadas. Além disso, é possível que um nó altamente cooperativo não tenha vizinho que possua as camadas elevadas, que deveriam proporcionar-lhe melhor qualidade de vídeo. Assim, um desafio é localizar outros nós altamente cooperativos para obter as camadas mais elevadas.

3.5.3. Codificação de Canal

A codificação de canal ou de rede (*network coding*) é uma técnica da teoria da informação aplicada recentemente em comunicações. A codificação de rede rompe com o paradigma da Internet no qual os nós intermediários não devem alterar os pacotes de dados, mas apenas encaminhá-los em direção ao destinatário desejado. Com a codificação de rede, os pacotes de dados podem ser codificados em nós intermediários antes de serem encaminhados. Em redes cabeadas, essa técnica pode ser utilizada para aumentar a robustez da rede. A Figura 3.17 ilustra um exemplo típico. Em redes onde perdas são possíveis, o nó D é capaz de receber as mensagens de A (m_a) e de C (m_c) mesmo se perder uma das duas. Suponha que o nó A envie sua mensagem m_a para D (passo 1) seguido pelo nó C que envia m_c para D (passo 2). Suponha também que o nó B já tenha recebido m_a e m_c em oportunidades anteriores e que envia uma mensagem codificada $m_a.m_c$, utilizando um operador `XOR` bit-a-bit por exemplo, (passo 3) também para D. Assim, caso haja neces-

sidade, o nó *D* decodifica a mensagem recebida de *B* e recupera a mensagem m_a ou m_c perdida sem necessidade de retransmissões, como exemplificado na Figura 3.17.

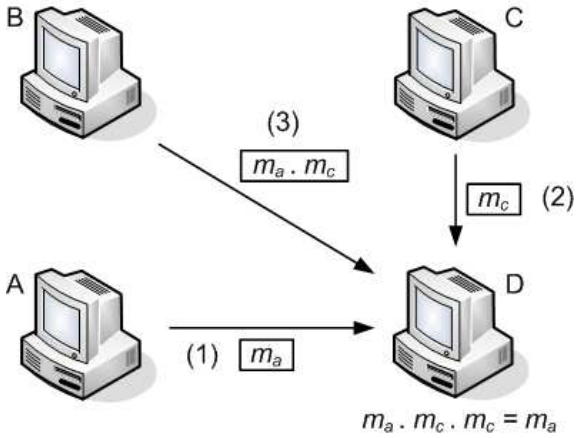


Figura 3.17. Um exemplo típico de codificação de rede.

Em sistemas de distribuição de vídeo par-a-par baseados na arquitetura em malha, técnicas de codificação de rede podem ser utilizadas para reduzir o número de requisições de pedaços de vídeo. A Figura3.18 ilustra um exemplo no qual o nó *A* possui os pedaços p_1, p_2, p_3 do vídeo. Por outro lado, o nó *B* possui apenas p_1 e p_2, o nó *C* p_1 e p_3, e o nó *D* p_2 e p_3. Para enviar a todos os nós os pedaços desejados sem requisições individuais, em casos especiais o nó *A* poderia codificar p_1, p_2 e p_3 e enviar para os nós *B*, *C* e *D*. Assim, todos os nós poderão obter o pedaço que desejam ao decodificar a mensagem enviada por *A*. Outro exemplo da utilização de técnicas de codificação de rede para sistemas de distribuição de vídeo par-a-par é a proposta de Wang e Li [Wang e Li 2007]. Nessa proposta, cada pedaço do vídeo é dividido em *n* blocos, nos quais cada bloco é codificado em função dos outros pedaços. As técnicas de codificação empregadas permitem que os blocos sejam decodificados antes mesmo que os *n* blocos referentes a um determinado pedaço sejam recebidos. A adaptação das técnicas de codificação de rede para sistemas de distribuição de vídeo par-a-par ainda é um tema pouco explorado. O emprego dessa técnica pode aumentar a escalabilidade dos sistemas ao reduzir o número de pedaços de vídeo transmitidos e, conseqüentemente, a sobrecarga de controle.

3.5.4. Algoritmos de Seleção de Pares

O processo de escolha de pares inicia-se após a descoberta de participantes que compõem o sistema de distribuição do vídeo desejado. Essa informação é normalmente solicitada à fonte do vídeo pelos nós que entram na rede. Os detalhes desse processo são vistos na Seção 3.3. Esse processo também é executado periodicamente para aumentar a eficiência do encaminhamento do conteúdo de vídeo na estrutura de distribuição. Se o sistema se baseia na arquitetura em árvore ou multiárvore, o processo de escolha de pares se torna um processo de escolha de nó pai na árvore de distribuição. Em arquiteturas multiárvore, esse problema é estendido para a escolha de um nó pai em cada árvore. Nos sistemas baseado na arquitetura em malha, os nós selecionam os pares de acordo com a disponibilidade dos pedaços do vídeo desejado.

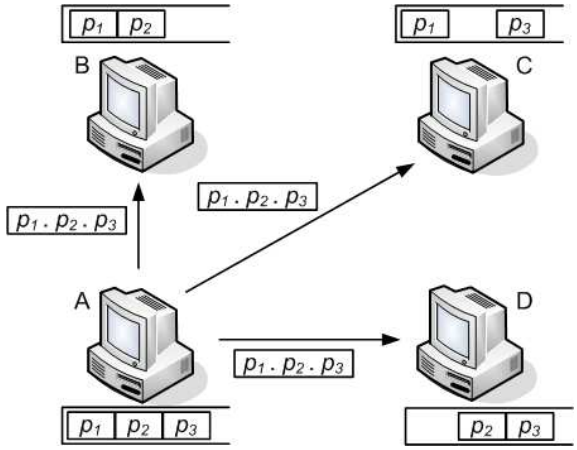


Figura 3.18. A codificação de rede em sistemas de distribuição de vídeo par-a-par.

Independentemente da arquitetura da rede, o processo de seleção de pares é baseado em métricas de desempenho relacionadas normalmente à capacidade de transmissão, à continuidade de reprodução e à manutenção da qualidade do vídeo [Cui et al. 2007]. Cada aplicação define os parâmetros de rede a medir para estimar as métricas de desempenho. Tipicamente, esses parâmetros estão relacionados ao atraso entre pares, muitas vezes estimado a partir do tempo de ida-e-volta de sondas, e a taxa de transmissão alcançável. Em arquiteturas em árvore, o número de descendentes também é considerado. Evita-se escolher como nós pai aqueles com um alto número de descendentes, já que a banda é dividida entre todos os descendentes. Por outro lado, em redes que possuem arquitetura em malha, o número de pedaços de vídeo disponíveis é usado como métrica. Quanto maior o número de pedaços disponível, maior é a métrica atribuída ao par.

A forma otimizada de escolher os melhores pares é através do conhecimento global dos membros juntamente com as suas respectivas métricas de desempenho. Baseado em conhecimento global, cada membro pode tornar sua escolha determinística. Logo, é possível escolher os pares que ofereçam as melhores condições de transmissão de vídeo conforme os requisitos desejados. A disseminação de informações globais apresenta, entretanto, um problema. Ela não é escalável [Jurca et al. 2007]. Em redes de distribuição de vídeo a escalabilidade é altamente desejável. Quanto maior for o número de pares no sistema, maior é a probabilidade do vídeo estar disponível e de ser distribuído com qualidade. Além disso, os sistemas par-a-par são dinâmicos e os membros podem entrar e sair a qualquer momento. Essas características implicam em sobrecarga de controle elevada para manter informações globais atualizadas. Protocolos baseados em fofocas são empregados para disseminar informações ao mesmo tempo em que mantêm a escalabilidade da rede. Esses protocolos são apropriados para redes descentralizadas e dinâmicas [Boyd et al. 2005]. O compromisso do emprego de protocolos baseados em fofocas é a redução da visão de cada membro com relação à rede. Trabalhos como os de Rejaie e Stafford [Rejaie e Stafford 2004] e Zhang *et al.* [Zhang et al. 2005c] alegam que a amostra da rede conhecida já é suficiente para a escolha de pares que atendam os requisitos desejados. Entretanto, Liang e Nahrstedt [Liang e Nahrstedt 2006] salientam que a forma aleatória de gerenciamento de grupo e a visão parcial da rede podem não ser apropriadas para a escolha de membros que atendam determinados requisitos de qualidade de serviço. Para

garantir esses requisitos, Liang e Nahrestedt propõem a separação dos membros da rede em grupos de acordo com as características de desempenho. Apenas dentro de cada grupo, protocolos baseados em fofocas são utilizados. Portanto, encontrar mecanismos eficientes de escolha de pares e escaláveis em redes altamente dinâmicas permanece um desafio em aberto.

A presença de pares em condições satisfatórias para a transmissão de vídeo está ligada à organização dos nós fisicamente na rede [Liu et al. 2007a]. A arquitetura da rede, árvore, multiárvore, ou malha, e o processo com o qual a rede é montada contribuem para um melhor desempenho global da aplicação. Nós fisicamente próximos têm maiores chances de se tornarem melhores fontes de vídeo. Entretanto, em redes sobrepostas, um nó vizinho pode não estar fisicamente próximo. Logo, organizar a rede sobreposta de forma a otimizar as distâncias físicas entre os nós vizinhos é uma área que precisa ser investigada. Além disso, o emprego de métricas que considerem todos os possíveis requisitos para otimizar de forma global o desempenho da aplicação ainda é um tema em aberto em sistemas de distribuição de vídeo par-a-par.

3.5.5. Vídeo sob Demanda

O foco deste capítulo é a difusão de vídeo usando sistemas par-a-par. No entanto, tais sistemas também podem ser usados para distribuir vídeo sob demanda. Basicamente, existem duas formas de utilização de sistemas par-a-par na distribuição de vídeo sob demanda. Na primeira, os participantes do sistema apenas auxiliam um servidor a entregar o conteúdo de vídeo a outros participantes. Essa técnica é chamada de vídeo sob demanda assistido por pares (*peer-assisted video-on-demand*) [Huang et al. 2007a, Janardhan e Schulzrinne 2007, Huang et al. 2007b] e tem como objetivo usar os sistemas par-a-par para substituir as redes de distribuição de conteúdo usadas atualmente, devido ao alto custo dessas redes. Na segunda abordagem, os participantes do sistema tratam as mensagens de controle de reprodução solicitadas pelos usuários, avançando ou retrocedendo o seu ponto de reprodução [Annapureddy et al. 2007, Vratonjic et al. 2007, Cheng et al. 2007]. O principal desafio dessa abordagem é que um novo participante do sistema pode começar a ver o vídeo do ponto que desejar, ao invés de assistir o vídeo a partir do instante de sua entrada, como ocorre na difusão.

Huang *et al.* apresentam um exemplo da primeira abordagem [Huang et al. 2007a]. O objetivo é usar os sistemas par-a-par para auxiliar os servidores na entrega do vídeo aos receptores e, conseqüentemente, reduzir os custos dos provedores de conteúdo. Nessa técnica, um servidor (ou um conjunto de servidores) armazena os vídeos e, à medida que receptores solicitam esses vídeos ao servidor, passam a contribuir também para a sua distribuição. Um receptor pode redistribuir apenas o vídeo que está assistindo ou pode ainda redistribuir outros vídeos que já tenha assistido, mas que ainda estão armazenados. Os pares não implementam nenhuma operação de controle de reprodução. Para avaliar a eficiência da técnica de vídeo sob demanda assistido por pares, os autores usaram arquivos de traços coletados durante nove meses do sítio MSN Video [Microsoft 2008]. Em um mês analisado, a banda de saída requerida pelo serviço de distribuição usando o modelo cliente-servidor foi de 1,23 Gbps. Com a cooperação entre os receptores a banda de saída exigida caiu para 36,9 Mbps. Uma economia de banda de 97%, comprovando que os sistemas par-a-par aumentam a escalabilidade de serviços de distribuição de vídeo.

Quando os sistemas par-a-par tem que tratar as solicitações de controle de reprodução o desafio é maior. Nos sistemas de difusão baseados na arquitetura em malha, diz-se que os participantes estão sincronizados, pois os nós têm interesse em um dado vídeo durante um mesmo período de tempo. Por exemplo, dados obtidos com o CoolStreaming mostram que o intervalo entre os trechos de um mesmo vídeo reproduzidos por diferentes participantes não é superior a um minuto [Zhang et al. 2005c]. A maioria das propostas de sistemas de distribuição de vídeo sob demanda em redes par-a-par se baseia na arquitetura em malha ou na arquitetura híbrida [Annapureddy et al. 2007, Vratonjic et al. 2007]. Entretanto, nesses sistemas a sincronização entre os participantes é reduzida, pois cada participante pode estar assistindo a uma determinada parte do vídeo de acordo com a sua vontade. Dessa forma, o número de participantes que simultaneamente se interessa por um mesmo trecho de um vídeo é menor e, conseqüentemente, o número de parceiros dos quais se podem solicitar pedaços desse trecho também é menor. Portanto, quando um participante muda o seu ponto de reprodução, ou seja, avança ou retrocede o vídeo, ele precisa rapidamente achar outros participantes que tenham os pedaços de vídeo referentes a esse novo ponto de reprodução. Caso não encontre os parceiros, o participante deverá solicitar os pedaços diretamente da fonte. Isso pode levar à sobrecarga da fonte e ao congestionamento dos enlaces próximos a fonte, reduzindo a eficiência da distribuição.

Para evitar a sobrecarga da fonte, Vratonjic *et al.* propõem o BulletMedia, um sistema par-a-par que implementa as funções de avanço e retrocesso do vídeo [Vratonjic et al. 2007]. A idéia básica do BulletMedia é que todo pedaço de um vídeo deve ser replicado pelos participantes mesmo que ele não seja necessário para os pontos de reprodução atuais dos participantes ativos do sistema. Sendo assim, os participantes devem solicitar e armazenar pedaços de vídeo que não sejam usados para a reprodução imediata. Ao aumentar a disponibilidade de pedaços nos participantes, reduz-se a dependência do sistema em relação à fonte e também o tempo de espera para retomar a reprodução após um avanço ou retrocesso. O BulletMedia constrói duas redes sobrepostas: uma rede em malha e uma rede estruturada baseada em tabelas de *hash* distribuídas. Por isso, é classificado como um sistema baseado em uma arquitetura híbrida. A rede sobreposta em malha é usada para encaminhar os pedaços de vídeo. A rede sobreposta estruturada é usada para controlar a replicação dos pedaços. Ela também é usada pelos participantes para descobrirem a localização dos pedaços disponíveis no sistema e, conseqüentemente, afetar menos a continuidade de recepção após um avanço ou retrocesso.

3.5.6. Aspectos Práticos

Existem diferentes aspectos práticos, provenientes de formas de utilização, configuração ou implementação de protocolos e aplicações, que influenciam a utilização de sistemas par-a-par na Internet. A escolha da forma de endereçamento e do protocolo de roteamento e o tempo de inicialização e troca de canais são os principais aspectos práticos identificados.

Endereçamento

Um aspecto prático importante diz respeito à forma de endereçamento utilizada. O uso de endereços válidos ou inválidos, também chamados de “públicos” ou “privados”,

influencia o desempenho de sistemas de distribuição de vídeo par-a-par.

De acordo com dados coletados em 2005, mais de 50% dos usuários de banda larga do Reino Unido e dos Estados Unidos estão conectados à Internet com endereços inválidos, através de dispositivos que fazem tradução de endereços (*Network Address Translation* - NAT) [Rodriguez et al. 2006]. Usando NAT, vários computadores localizados em um domicílio ou escritório podem compartilhar o acesso à Internet utilizando um único endereço IP válido. O dispositivo NAT, por exemplo, um roteador com uma conexão ADSL e portas Ethernet, "traduz" os endereços dos pacotes de saída, através da substituição do endereço de origem (um endereço IP inválido não-roteável) e do número de porta escritos pela estação pelos endereço IP e número de porta do dispositivo NAT (o roteador). Este procedimento é chamado de mapeamento de portas. A principal desvantagem da utilização do mecanismo NAT é que computadores conectados à Internet através de um dispositivo NAT podem se conectar a computadores com endereço IP válido, mas em princípio não podem servir como servidores de conexões de computadores na Internet, uma vez que o seu endereço, inválido não é conhecido por máquinas da Internet (por essa razão esse endereço, inválido e, portanto, não-roteável, é também chamado privado). Desta forma, o princípio de conectividade fim-a-fim provida pela camada IP é quebrado.

Nos sistemas par-a-par, os participantes precisam ser bidirecionalmente alcançáveis. Como freqüentemente o NAT impede participantes de estabelecerem conexões entre si, o desempenho desses sistemas é afetado. Pelo mesmo motivo, o uso de *firewalls* também prejudica o desempenho desses sistemas. Atualmente, a maioria dos sistemas par-a-par depende de configuração manual das portas para que seja possível a comunicação entre pares conectados através de dispositivos NAT. No entanto, algumas técnicas foram propostas para fazer com que sistemas par-a-par possam funcionar através de dispositivos NAT. A maioria das técnicas se baseia na utilização se baseiam na utilização de outro endereço acima do endereço IP. Por exemplo, o protocolo STUN (*Simple Traversal of UDP through NATs*) [Rosenberg et al. 2003], que também possui uma extensão para o TCP (STUNT) [Guha et al. 2004], primeiro utiliza um identificador URI (*Uniform Resource Identifier*) e o protocolo SIP (*Session Initiation Protocol*) para o par se comunicar com um servidor STUN/STUNT. Então, o servidor informa ao cliente STUN/STUNT o endereço IP público do dispositivo NAT, e qual número de porta está aberto neste dispositivo NAT para permitir o tráfego de entrada. Outra técnica que se baseia no mapeamento de portas automático é a UPnP (*Universal Plug and Play*) [UPnP Forum 2008]. Usando o UPnP, a aplicação primeiro detecta se está conectada através de um dispositivo NAT que implementa o UPnP. Então, a aplicação é informada do endereço público do dispositivo NAT e configura mapeamentos de portas TCP e UDP para encaminhar pacotes recebidos nas portas externas do dispositivo NAT para as portas internas utilizadas pelo cliente.

Protocolo de Transporte

A escolha do protocolo de transporte é fundamental para o desempenho dos sistemas de vídeo par-a-par. Dependendo da escolha, um sistema pode ser mais eficiente na entrega de conteúdo de vídeo, porém a comunicação e a cooperação entre os parti-

cipantes podem ser afetadas. O processamento requerido pelo UDP é menor do que o do TCP, uma vez que o UDP não estabelece conexão e nem implementa mecanismos de controle de erros e de fluxo. Por isso, o UDP é o protocolo de transporte mais indicado para aplicações multimídias. No entanto, os sistemas de vídeo par-a-par se dividem entre o uso do UDP e do TCP para o envio de mensagens de controle e comunicação entre os participantes [Silverston e Fourmaux 2007]. Por não possuir controle de fluxo, a maioria dos dispositivos de *firewall* e NAT bloqueia o tráfego UDP para evitar possíveis congestionamentos na rede e sobrecarga de processamento nos roteadores. Por isso, muitos sistemas de vídeo par-a-par usam o TCP como protocolo de transporte para atravessarem dispositivos de *firewall* e NAT. Ao usar o TCP para distribuição de vídeo, um sistema pode experimentar variações na taxa efetiva do vídeo recebido por seus participantes em virtude da dinâmica do mecanismo de controle de fluxo do TCP. Em função da arquitetura fechada da maioria dos sistemas, medições foram realizadas de modo a identificar os protocolos de transporte utilizados tanto para o tráfego de vídeo quanto o tráfego de controle [Sentinelli et al. 2007, Silverston e Fourmaux 2007].

Tempo de Inicialização e de Mudança de Canais

O tempo de inicialização e mudança de canais também é um problema na difusão de vídeo par-a-par. Na maioria dos sistemas, um novo nó leva em média 10 segundos para se tornar um participante da rede sobreposta e outros 10 segundos, em média, para abrir o tocador de vídeo e armazenar uma determinada quantidade de pacotes de vídeo suficiente para iniciar a reprodução [Liu et al. 2008]. Os tempos de inicialização e de troca de canal podem ser ainda maiores quando os canais selecionados são menos populares. Esses tempos também podem aumentar em aplicações que usam codificação de canal e o TCP como protocolo de transporte. Esse último devido aos tempos de estabelecimento de conexão e retransmissão de segmentos perdidos. Para simplificar sua implementação e garantir compatibilidade e portabilidade, a maioria dos sistemas de vídeo par-a-par utiliza tocadores de vídeo disponíveis comercialmente, ao invés de implementarem os próprios aplicativos. Dessa forma, as tarefas de distribuição e reprodução são separadas e em cada uma existe uma etapa de armazenamento (*buffering*). Dependendo da estratégia adotada, em princípio, o armazenamento em duas etapas acarreta em maiores latências, mas, dependendo do tamanho e do algoritmo de escalonamento definidos para cada um dos *buffers*, a eficiência do sistema de distribuição pode aumentar [Liu et al. 2008].

3.6. Considerações Finais

O uso de sistemas par-a-par é uma das mais promissoras soluções para a implantação de um serviço de distribuição de vídeo na Internet, principalmente, devido à escalabilidade desses sistemas. Esse tipo de sistema já é um sucesso para o compartilhamento de arquivos e, a cada dia, é mais usado para a distribuição de vídeo. Embora alguns sistemas de vídeo par-a-par já suportem milhares de usuários simultâneos, como visto neste capítulo, tais sistemas apresentam uma série de desafios a serem suplantados para sua utilização na distribuição de vídeo.

Além dos desafios técnicos, existem os conflitos de interesse entre os provedo-

res de serviço de rede, os provedores de conteúdo e os usuários. Para os provedores de serviço, os sistemas de vídeo par-a-par podem abrir novas possibilidades de negócios. Entretanto, podem também exigir um aumento da capacidade do serviço, em virtude dos requisitos de qualidade de serviço do vídeo distribuído. Uma questão importante para os provedores de conteúdo, como as estações de TV, é assegurar o direito de propriedade intelectual sobre o conteúdo distribuído. Atualmente, a maioria dos sistemas de vídeo par-a-par é usado para distribuir o conteúdo de canais de TV, aberta ou a cabo, de diferentes regiões do planeta. Para muitos, essa é uma prática ilícita, mas difícil de ser combatida devido à dispersão geográfica dos usuários. Dessa forma, o desafio para os provedores de conteúdo é como garantir suas receitas com a oferta de programação através da distribuição de vídeo par-a-par. É possível que os usuários não paguem por um serviço que hoje é "gratuito" e que pode sofrer variações de atraso e até mesmo ser interrompido.

É este cenário complexo que faz da distribuição de vídeo sobre redes par-a-par um tema de pesquisa desafiador.

Agradecimentos

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ, FINEP e FUNTTEL.

Referências

[Agarwal et al. 2007] Agarwal, S., Singh, J. P. e Dube, S. (2007). Analysis and implementation of gossip-based P2P streaming with distributed incentive mechanisms for peer cooperation. *Advances in Multimedia*, 2007(2):1–12.

[Akamai Technologies 2008a] Akamai Technologies (2008a). Akamai helps MSN deliver more than 65 million video streams around the world for record-breaking Live Earth event. http://www.akamai.com/html/customers/case_study_live_earth.html. Acessado em 5 de março de 2008.

[Akamai Technologies 2008b] Akamai Technologies (2008b). Akamai: The leader in web application acceleration and performance management, streaming media services and content delivery. http://www.akamai.com. Acessado em 5 de março de 2008.

[Amir et al. 1995] Amir, E., McCanne, S. e Zhang, H. (1995). An application level video gateway. Em *ACM Multimedia*, páginas 255–265.

[Annapureddy et al. 2007] Annapureddy, S., Guha, S., Gkantsidis, C., Gunawardena, D. e Rodriguez, P. (2007). Exploring VoD in P2P swarming systems. Em *IEEE INFOCOM*, páginas 2571–2575.

[Banerjee e Bhattacharjee 2005] Banerjee, S. e Bhattacharjee, B. (2005). A comparative study of application layer multicast protocols. Relatório técnico, University of Wisconsin-Madison.

[Banerjee et al. 2002a] Banerjee, S., Bhattacharjee, B. e Kommareddy, C. (2002a). Scalable application layer multicast. Em *Conference on applications, technologies, architectures, and protocols for computer communications*, páginas 205–217.

[Banerjee et al. 2002b] Banerjee, S., Bhattacharjee, B. e Kommareddy, C. (2002b). Scalable application level multicast. Em *ACM SIGCOMM*, páginas 205–217.

[Banerjee et al. 2003] Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B. e Khuller, S. (2003). Construction of an efficient overlay multicast infrastructure for real-time applications. Em *IEEE INFOCOM*, páginas 1587–1596.

[Barcellos e Gaspary 2006] Barcellos, A. M. P. e Gaspary, L. P. (2006). Segurança em redes P2P: Princípios, tecnologias e desafios. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*.

[Bharambe e Padmanabhan 2006] Bharambe, A. R. e Padmanabhan, C. H. V. N. (2006). Analyzing and improving a BitTorrent networks performance mechanisms. Em *IEEE INFOCOM*, páginas 1–12.

[Bhattacharyya 2003] Bhattacharyya, S. (2003). An overview of source-specific multicast (SSM). IETF Network Working Group RFC 3569.

[Bolot et al. 1994] Bolot, J.-C., Turletti, T. e Wakeman, I. (1994). Scalable feedback control for multicast video distribution in the Internet. Em *ACM SIGCOMM*, páginas 58–67.

[Boyd et al. 2005] Boyd, S., Ghosh, A., Prabhakar, B. e Shah, D. (2005). Gossip algorithms: Design, analysis and applications. Em *IEEE INFOCOM*, páginas 1653–1664.

[Castro et al. 2002a] Castro, M., Druschel, P., Hu, Y. C. e Rowstron, A. (2002a). Exploiting network proximity in peer-to-peer networks. Relatório Técnico MSR-TR-2002-82, Microsoft Research.

[Castro et al. 2006] Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A. e Singh, A. (2006). AnySee: Peer-to-peer live streaming. Em *IEEE INFOCOM*, páginas 1–10.

[Castro et al. 2003] Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A. e Singh, A. (2003). SplitStream: High-bandwidth multicast in cooperative environments. Em *ACM Symposium on Operating Systems Principle - SOSP*, páginas 298–313.

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

[CDNetworks Co., Ltd. 2004] CDNetworks Co., Ltd. (2004). CDNetworks :: Global CDN service leader. http://www.us.cdnetworks.com. Acessado em 6 de março de 2008.

[Cheng et al. 2007] Cheng, B., Liu, X., Zhang, Z. e Jin, H. (2007). A measurement study of a peer-to-peer video-on-demand system. Em *International Workshop on Peer-to-Peer Systems - IPTPS*.

[Cheung et al. 1996] Cheung, S. Y., Ammar, M. H. e Li, X. (1996). On the use of destination set grouping to improve fairness in multicast video distribution. Em *IEEE INFOCOM*, páginas 553–560.

[Chu et al. 2002] Chu, Y.-H., Rao, S. G., Seshan, S. e Zhang, H. (2002). A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471.

[Conner et al. 2006] Conner, W., Nahrstedt, K. e Gupta, I. (2006). Preventing DoS attacks in peer-to-peer media streaming systems. Em *SPIE/ACM Conference on Multimedia Computing and Networking - MMCN*.

[Coolstreaming 2008] Coolstreaming (2008). Coolstreaming - broadcast your TV. http://www.coolstreaming.us. Acessado em 5 de março de 2008.

[Costa e Duarte 2003] Costa, L. H. M. K. e Duarte, O. C. M. B. (2003). Roteamento multicast na Internet. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*.

[Cui et al. 2007] Cui, Y., Dai, L. e Xue, Y. (2007). Optimizing P2P streaming throughput under peer churning. Em *IEEE GLOBECOM*, páginas 231–235.

[Dan et al. 2007] Dan, G., Fodor, V. e Chatzidrossos, I. (2007). On the performance of multiple-tree-based peer-to-peer live streaming. Em *IEEE INFOCOM*, páginas 2556–2560.

[de Albuquerque et al. 2006] de Albuquerque, C. V. N., Proença, T. e Oliveira, E. (2006). TVoIP: TV sobre IP arquiteturas para transmissão em larga escala. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*.

[Deering 1989] Deering, S. (1989). Host extensions for IP multicasting. IETF Network Working Group RFC 1112.

[Dhungel et al. 2007] Dhungel, P., Hei, X., Ross, K. e Saxena, N. (2007). The pollution attack in P2P live video streaming: Measurement results and defenses. Em *Peer-to-Peer Streaming and IP-TV Workshop (P2P-TV)*.

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

[ESM Group 2007] ESM Group (2007). End System Multicast. http://esm.cs.cmu.edu/. Acessado em 17 de março de 2008.

[Freedman et al. 2008] Freedman, M. J., Freudenthal, E., Mazières, D. e Shanahan, K. (2008). CoDeeN - a content distribution network for PlanetLab. http://codeen.cs.princeton.edu. Acessado em 6 de março de 2008.

[Ganesh et al. 2003] Ganesh, A. J., Kermarrec, A.-M. e Massouli, L. (2003). Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149.

[Guha et al. 2004] Guha, S., Takeda, Y. e Francis, P. (2004). NUTSS: A SIP-based approach to UDP and TCP network connectivity. Em *ACM SIGCOMM*.

[Guo et al. 2004] Guo, L., Chen, S., Ren, S., Chen, X. e Jiang, S. (2004). PROP: a scalable and reliable P2P assisted proxy streaming system. Em *International Conference on Distributed Computing Systems - ICDCS*, páginas 778–786.

[Habib e Chuang 2004] Habib, A. e Chuang, J. (2004). Incentive mechanism for peer-to-peer media streaming. Em *IEEE International Workshop on Quality of Service - IWQoS*, páginas 171–180.

[Hales e Patarin 2005] Hales, D. e Patarin, S. (2005). Computational sociology for systems "in the wild": the case of BitTorrent. *IEEE Distributed Systems Online*, 6(7).

[Handley et al. 2003] Handley, M., Floyd, S., Padhye, J. e Widmer, J. (2003). TCP friendly rate control (TFRC): Protocol specification. IETF Network Working Group RFC 3448.

[Haridasan e van Renesse 2006] Haridasan, M. e van Renesse, R. (2006). Defense against intrusion in a live streaming multicast system. Em *International Conference on Peer-to-Peer Computing*.

[Hefeeda et al. 2004] Hefeeda, M. M., Bhargava, B. K. e Yau, D. K. Y. (2004). A hybrid architecture for cost-effective on-demand media streaming. *Computer Networks*, 44(3):353–382.

[Holbrook 2006] Holbrook, H. (2006). Source-specific multicast for IP. IETF Network Working Group RFC 4607.

[Holbrook e Cheriton 1999] Holbrook, H. W. e Cheriton, D. R. (1999). IP multicast channels: EXPRESS support for large-scale single-source applications. Em *ACM SIGCOMM*, páginas 65–78.

[Huang et al. 2007a] Huang, C., Li, J. e Ross, K. W. (2007a). Can Internet video-on-demand be profitable? Em *ACM SIGCOMM*.

[Huang et al. 2007b] Huang, C., Li, J. e Ross, K. W. (2007b). Peer-assisted VoD: Making internet video distribution cheap. Em *International Workshop on Peer-to-Peer Systems - IPTPS*.

[IDG Now! 2006] IDG Now! (2006). Partida entre Portugal e França bate novo recorde de transmissão online. http://idgnow.uol.com.br/internet/2006/07/12/idgnoticia.2006-07-12.6926355736/. Acessado em 3 de março de 2008.

[Janardhan e Schulzrinne 2007] Janardhan, V. e Schulzrinne, H. (2007). Peer assisted VoD for set-top box based IP network. Em *Peer-to-Peer Streaming and IP-TV Workshop - P2P-TV*.

[Johansen et al. 2006] Johansen, H., Allavena, A. e van Renesse, R. (2006). Fireflies: Scalable support for intrusion-tolerant network overlays. Em *EuroSys*, páginas 3–13.

[Judge e Ammar 2003] Judge, P. e Ammar, M. (2003). Security issues and solutions in multicast content distribution: a survey. *IEEE Network*, 17(1):30–36.

[Jurca et al. 2007] Jurca, D., Chakareski, J., Wagner, J.-P. e Frossard, P. (2007). Enabling adaptive video streaming in P2P systems. *IEEE Communications Magazine*, 45(6):108–114.

[Lao et al. 2005] Lao, L., Cui, J.-H., Gerla, M. e Maggiorini, D. (2005). A comparative study of multicast protocols: Top, bottom, or in the middle? Em *IEEE INFOCOM*, páginas 13–17.

[Lenstra et al. 1990] Lenstra, J. K., Shmoys, D. B. e Tardos, É. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(3):259–271.

[Li et al. 2007] Li, B., Xie, S., Keung, G. Y., Liu, J., Stoica, I., Zhang, H. e Zhang, X. (2007). An empirical study of the Coolstreaming+ system. *IEEE Journal on Selected Areas in Communications*, 25(9):1627–1639.

[Li et al. 1999] Li, X., Ammar, M. e Paul, S. (1999). Video multicast over the internet. *IEEE Network*, 13(2):46–60.

[Liang et al. 2005] Liang, J., Kumar, R., Xi, Y. e Ross, K. (2005). Pollution in P2P file sharing systems. Em *IEEE INFOCOM*, páginas 1174–1185.

[Liang e Nahrstedt 2006] Liang, J. e Nahrstedt, K. (2006). RandPeer: Membership management for QoS sensitive peer-to-peer applications. Em *IEEE INFOCOM*, páginas 1–10.

[Limelight Networks 2008] Limelight Networks (2008). High performance content delivery network for digital media: Limelight networks. http://www.limelightnetworks.com. Acessado em 5 de março de 2008.

[Liu et al. 2003] Liu, J., Li, B. e Zhang, Y.-Q. (2003). Adaptive video multicast over the internet. *IEEE Multimedia*, 10(1):22–33.

[Liu et al. 2008] Liu, J., Rao, S. G., Li, B. e Zhang, H. (2008). Opportunities and challenges of peer-to-peer Internet video broadcast. *Proceedings of the IEEE*, 96(1):11–24.

[Liu et al. 2007a] Liu, Z., Shen, Y., Panwar, S., Ross, K. e Wang, Y. (2007a). On scalability of proximity-aware peer-to-peer streaming. Em *IEEE INFOCOM*, páginas 2561–2565.

[Liu et al. 2007b] Liu, Z., Shen, Y., Panwar, S., Ross, K. e Wang, Y. (2007b). Using layered video to provide incentives in p2p live streaming. Em *Peer-to-Peer Streaming and IP-TV Workshop - P2P-TV*.

[Liu et al. 2007c] Liu, Z., Shen, Y., Panwar, S. S., Ross, K. W. e Wang, Y. (2007c). P2P video live streaming with MDC: Providing incentives for redistribution. Em *IEEE International Conference on Multimedia and Expo - ICME*, páginas 48–51.

[Magharei et al. 2007] Magharei, N., Rejaie, R. e Guo, Y. (2007). Mesh or multiple-tree: A comparative study of live p2p streaming approaches. Em *IEEE INFOCOM*, páginas 1424–1432.

[McCanne et al. 1996] McCanne, S., Jacobson, V. e Vetterli, M. (1996). Receiver-driven layered multicast. Em *ACM SIGCOMM*, páginas 117–130.

[Microsoft 2008] Microsoft (2008). MSN video. http://video.msn.com. Acessado em 17 de março de 2008.

[Moraes et al. 2003] Moraes, I. M., Bicudo, M. D. D., Cardoso, K. V., Vasconcellos, S. V., de Rezende, J. F. e Duarte, O. C. M. B. (2003). Desenvolvimento de um ambiente de testes com suporte à qualidade de serviço para a transmissão de vídeo digital. Em *IV Workshop da RNP2*.

[Pai et al. 2008] Pai, V., Peterson, L. e Park, K. (2008). CoDeeN - a content distribution network for PlanetLab. http://codeen.cs.princeton.edu. Acessado em 6 de março de 2008.

[Qiu et al. 2001] Qiu, L., Padmanabhan, V. N. e Voelker, G. M. (2001). On the placement of web server replicas. Em *IEEE INFOCOM*, páginas 1587–1596.

[Qiu et al. 2007] Qiu, T., Nikolaidis, I. e Li, F. (2007). On the design of incentive-aware P2P streaming. *Journal of Internet Engineering*, 1(2):61–71.

[Rejaie 2006] Rejaie, R. (2006). Anyone can broadcast video over the Internet. *Communications of the ACM*, 49(11):55–58.

[Rejaie e Stafford 2004] Rejaie, R. e Stafford, S. (2004). A framework for architecting peer-to-peer receiver-driven overlays. Em *International Workshop on Network and Operating System Support for Digital Audio and Video - NOSSDAV*, páginas 42–47.

[Rodriguez et al. 2006] Rodriguez, P., Tan, S.-M. e Gkantsidis, C. (2006). On the feasibility of commercial, legal P2P content distribution. *ACM SIGCOMM Computer Communication Review*, 36(1):75–78.

[Rosenberg et al. 2003] Rosenberg, J., Weinberger, J., Huitema, C. e Mahy, R. (2003). STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs). IETF Network Working Group RFC 3489.

[Rowstron e Druschel 2001] Rowstron, A. e Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Em *IFIP/ACM Middleware*.

[Sentinelli et al. 2007] Sentinelli, A., Marfia, G., Gerla, M., Kleinrock, L. e Tewari, S. (2007). Will IPTV ride the peer-to-peer stream? *IEEE Communications Magazine*, 45(6):86–92.

[Severn Stream Media Networks 2007] Severn Stream Media Networks (2007). Severn stream media networks - the intelligent content delivery network for digital multimedia. http://www.severnstream.com. Acessado em 6 de março de 2008.

[Silverston e Fourmaux 2007] Silverston, T. e Fourmaux, O. (2007). Measuring P2P IPTV systems. Em *International Workshop on Network and Operating System Support for Digital Audio and Video - NOSSDAV*.

[Singh et al. 2004] Singh, A., Castro, M., Druschel, P. e Rowstron, A. (2004). Defending against eclipse attacks on overlay networks. Em *ACM SIGOPS European Workshop*.

[Tan e Jarvis 2006] Tan, G. e Jarvis, S. A. (2006). A payment-based incentive and service differentiation mechanism for peer-to-peer streaming broadcast. Em *IEEE International Workshop on Quality of Service - IWQoS*, páginas 41–50.

[Tang et al. 2007] Tang, Y., Luo, J.-G., Zhang, Q., Zhang, M. e Yang, S.-Q. (2007). Deploying P2P networks for large-scale live video-streaming service. *IEEE Communications Magazine*, 45(6):100–106.

[Tran et al. 2003] Tran, D. A., Hua, K. A. e Do, T. T. (2003). ZIGZAG: An efficient peer-to-peer scheme for media streaming. Em *IEEE INFOCOM*, páginas 1283–1292.

[Tran et al. 2004] Tran, D. A., Hua, K. A. e Do, T. T. (2004). A peer-to-peer architecture for media streaming. *IEEE Journal on Selected Areas in Communications*, 22(1):121–133.

[UPnP Forum 2008] UPnP Forum (2008). UPnP forum [online]. http://www.upnp.org. Acessado em 2 de março de 2008.

[Venkataraman et al. 2006] Venkataraman, P. V., Yoshida, K. e Francis, P. (2006). Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. Em *IEEE International Conference on Network Protocols - ICNP*, páginas 2–11.

[Vickers et al. 2000] Vickers, B., de Albuquerque, C. V. N. e Suda, T. (2000). Source-adaptive multi-layered multicast algorithms for real-time video distribution. *IEEE/ACM Transactions on Networking*, 8(6):720–733.

[Vratonjic et al. 2007] Vratonjic, N., Gupta, P., Knezevic, N., Kostic, D. e Rowstron, A. (2007). Enabling DVD-like features in P2P video-on-demand systems. Em *Peer-to-Peer Streaming and IP-TV Workshop - P2P-TV*.

[Vu et al. 2007] Vu, L., Gupta, I., Liang, J. e Nahrstedt, K. (2007). Measurement of a large-scale overlay for multimedia streaming. Em *International Symposium on High Performance Distributed Computing - HPDC*, páginas 241–242.

[Wallach 2002] Wallach, D. S. (2002). A survey of peer-to-peer security issues. Em *International Symposium on Software Security*.

[Wang et al. 2008] Wang, F., Liu, J. e Xiong, Y. (2008). Stable peers: Existence, importance, and application in peer-to-peer live video streaming. Em *IEEE INFOCOM*.

[Wang et al. 2007] Wang, F., Xiong, Y. e Liu, J. (2007). mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast. Em *International Conference on Distributed Computing Systems - ICDCS*, páginas 49–49.

[Wang e Li 2007] Wang, M. e Li, B. (2007). Lava: A reality check of network coding in peer-to-peer live streaming. Em *IEEE INFOCOM*, páginas 1082–1090.

[Wang et al. 2006] Wang, W., Xiong, Y., Zhang, Q. e Jamin, S. (2006). Ripple-stream: Safeguarding P2P streaming against DoS attacks. Em *IEEE International Conference on Multimedia and Expo - ICME*, páginas 1417–1420.

[Yiu et al. 2007] Yiu, W.-P. K., Jin, X. e Gary, S.-H. (2007). Challenges and approaches in large-spale P2P media streaming. *IEEE Multimedia*, 14(2):50–59.

[Zhang et al. 2005a] Zhang, M., Luo, J.-G., Zhao, L. e Yang, S.-Q. (2005a). A peer-to-peer network for live media streaming using a push-pull approach. Em *ACM Multimedia*, páginas 287–290.

[Zhang et al. 2005b] Zhang, M., Tang, Y., Zhao, L., Luo, J.-G. e Yang, S.-Q. (2005b). Gridmedia: A multi-sender based peer-to-peer multicast system for video streaming. Em *IEEE International Conference on Multimedia and Expo - ICME*, páginas 614–617.

[Zhang et al. 2005c] Zhang, X., Liu, J., Li, B. e Yum, T.-S. P. (2005c). CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. Em *IEEE INFOCOM*, páginas 2102–2111.

[Zhang et al. 2007] Zhang, Z., Chen, S. e Yoon, M. (2007). MARCH: A distributed incentive scheme for peer-to-peer networks. Em *IEEE INFOCOM*, páginas 1091–1099.

[Zhuang et al. 2001] Zhuang, S., Zhao, B., Joseph, A., Katz, R. e Kubiatowicz, J. (2001). Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. Em *International Workshop on Network and Operating System Support for Digital Audio and Video - NOSSDAV*, páginas 11–20.