Um Protocolo para a Disponibilização de Vídeo Sob-Demanda para Dispositivos Móveis

Danilo Freire de Souza Santos

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Gradução em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Engenharia da Computação

Angelo Perkusich, Dr.
Orientador

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S237p Santos, Danilo Freire de Souza

Um protocolo para a disponibilização de vídeo sob-demanda para dispositivos móveis / Danilo Freire de Souza Santos.─ Campina Grande, 2008.

84 f..: il. color

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Referências.

Orientador: Prof. Dr. Angelo Perkusich

1. Protocolo 2. Vídeo Sob-demanda 3. Dispositivos Móveis I. Título.

CDU 004.72(043)

Resumo

Atualmente, o número de dispositivos móveis produzidos, como celulares e PDAs, vem crescendo constantemente. Estes dispositivos estão sendo equipados com novas tecnologias de rede sem fio que disponibilizam taxas de transmissão elevadas e novos modos de comunicação com a Internet. Além disto, as próprias redes de telefonia móvel disponibilizam taxas de transmissão de dados cada vez mais elevadas, como é o exemplo da tecnologia 3G. Em paralelo, os fabricantes cada vez mais agregam novas características a estes dispositivos como, por exemplo, câmeras digitais de alta resolução. Neste contexto, os usuários destes dispositivos têm ao seu dispor um aparelho que pode produzir conteúdo digital através de câmeras digitais, por exemplo, e compartilhar este conteúdo com outras pessoas através da Internet, utilizando as várias formas de conectividade que estes aparelhos disponibilizam.

Portanto, este novo cenário faz com que os usuários destes dispositivos demandem novos tipos de serviços para o seu uso. Um destes novos serviços é o oferecimento de Vídeo sob-Demanda para dispositivos móveis. Neste tipo de serviço, o usuário pode acessar um determinado conteúdo de vídeo a qualquer momento.

Neste trabalho, portanto, é apresentado um protocolo de Vídeo sob-Demanda para usuários de dispositivos móveis, onde estes podem ser tanto consumidores quanto provedores do conteúdo de vídeo. Devido a diferentes interfaces de comunicação disponíveis nestes dispositivos, considera-se que estes possam estar em redes com diferentes capacidades de transmissão e, além disto, possam estar distribuídos pela Internet. Para isto, o protocolo leva em consideração dois aspectos relevantes a estes dispositivos: sua localização relativa na rede, e sua capacidade de transmissão. No decorrer do trabalho, o protocolo e seus algoritmos são apresentados. Além disto, são apresentados resultados de simulações que foram utilizados para avaliar o desempenho do protocolo em diferentes cenários de uso.

Abstract

In the last years, the number of mobile devices produced, such as PDAs and smart phones, is growing constantly. These devices are being shipped with new communication technologies that offer new ways to access the Internet. Also, mobile telephony operators are offering new networks technologies that make available high data transmission rates, such as 3G networks. In parallel, embedded capture media devices such as video cameras and audio recorders, are been shipped with these mobile devices. These new capabilities allow users to create their own multimedia content, and with new wireless technologies, users can share content using the Internet.

Following this trend, it is necessary to make available for the users new services and applications, thus enabling them to share their content. Video on-Demand (VoD) services have become one of these services. The main objective of VoD is to provide users with the possibility to watch videos from the beginning at any time.

In this work, therefore, it is presented a video on-demand protocol for mobile devices. This protocol makes possible to users access and share video content. Then, users act as both, consumers as well as providers of multimedia content. Due to different available communication interfaces in mobile devices, it is considerered that they can be in networks with different bandwidth characteristics, and also, they can be distributed through the Internet. Two major aspects related to mobility are taken into account: the bandwidth available that the mobile device can share with the network as well as the relative position of the mobile device in the network. Then, in this work it is presented the protocol and its algorithms. Also, it is presented simulation results used to evaluate the protocol in different scenarios.

Agradecimentos

Agradeço, antes de tudo, a minha mãe, meu pai e minha irmã (guria) pelo apoio, compreensão, e suporte para realizar todas as tarefas e superar todos os desafios da minha vida.

Agradeço também a todos os meus amigos, que durante algum momento deste mestrado me ajudaram (ou até atrapalharam). Dentre eles, agradeço aos amigos de longa data que conviveram comigo (e me agüentaram) como Ricardo, André, Natassia, Greyce, Carol, Fabrício, Bisso, Cristhiano, Nielison, Marquinhos, Saulo, Marília, e muitos outros. Em especial, agradeço a Estela, por realmente me suportar durante estes últimos meses.

Agradeço aos professores, amigos e orientadores Angelo Perkusich, Hyggo Almeida, e Marcos Morais pelas dicas, conversas, orientações, puxões de orelha, entre outras coisas.

Agradeço também, a todos os amigos do laboratório Embedded, como Kyller, Glauber, Leandro cabelo, Felipe Vigia, Mateus, Gutemberg, Walter, Saulo, Taciana, Raul, Leandro Sales, Thiago Santos, Luiz Paulo, Lorena (e Larissa), enfim, todo mundo. Em especial, da melhor sala do laboratório, sala 104, agradeço ao Olympio, Zé Luís, Ádrian, André, Mário, Diego, Yuri, e Marcos Fábio quando ele se decidir em que sala vai ficar.

Enfim, agradeço a todos que conviveram comigo durante estes últimos anos. Aos amigos os quais me esqueci de citar o nome, me desculpem, a idade atrapalha.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nas últimas décadas o mundo vem sendo testemunha de um contínuo crescimento no desenvolvimento e produção de dispositivos portáteis com conectividade de rede sem fio, ou simplesmente dispositivos móveis. A produção e venda de dispositivos como *smartphones* e PDAs (*Personal Digital Assistant*) vêm crescendo constantemente como pode ser constatado com base em estudos feitos pela Canalys[1]. Em um cenário local, no Brasil, por exemplo, a Anatel[2] apresentou resultados que mostram que ao final do mês de Julho de 2008 o número de celulares no Brasil chegou aos 135 milhões de unidades, o que indica uma densidade próxima de 70,5 celulares a cada 100 habitantes.

Além disto, a evolução das tecnologias de redes sem fio oferece novas possibilidades de serviços para estes dispositivos. Novas tecnologias de rede sem fio de curto e médio alcance, como *Bluetooth*[3] e *Wi-Fi*[4], disponibilizam novas alternativas de comunicação entre estes dispositivos, além de possibilitarem taxas de transmissão elevadas [11]. Em paralelo, as redes de telefonia móvel, ou simplesmente redes celulares, estão evoluindo e oferecendo serviços de transmissão de dados com taxas de transmissão elevadas, ao ponto de concorrerem com as conexões convencionais de banda larga com fio como ADSL (*Asymmetric Digital Subscriber Line*) e Internet a cabo, por exemplo. Todos estes fatores possibilitam que estes dispositivos móveis possuam várias interfaces de comunicação com a Internet, por exemplo, mantendo-se, então, sempre *online* [11].

Além destas características de conectividade, tais dispositivos agregam vários recursos de interfaces com usuário, como teclados, sensores embutidos, visores, microfones e câmeras. A utilização destes recursos cria novas expectativas para os usuários, os quais demandam novos recursos e qualidade para seus dispositivos. Pesquisas de fabricantes, como a *Schneider Kreuznach*[5], mostram que a maioria dos usuários espera utilizar exclusivamente o celular como câmera.

Deste modo, pode-se considerar um dispositivo móvel, ou mais especificamente um celular, como

[1] http://www.canalys.com
[2] http://www.anatel.gov.br
[3] http://www.bluetooth.com
[4] http://www.wi-fi.org
[5] http://www.schneiderkreuznach.com

uma coleção de recursos que podem ser re-arranjados sob demanda para suprir um determinado propósito, por exemplo, criar um celular de música com grande capacidade de armazenamento e com um player de alta qualidade. Citando as palavras de Bill Joy, co-fundador da Sun Microsystems[6]:

> *"A cell phone has a microphone. It has a numeric keypad. It has a display. It has a transmitter and receiver. That is what it is. It has a battery with a certain lifetime. It has a processor and some memory. If it is a European model it has a stored value card. It is a collection of things. It is some collection. The fact that it is all in one bundle, well, that is pretty artificial."* - **Bill Joy**.

O oferecimento de tantos recursos faz com que os usuários destes dispositivos, além de consumirem, sejam capazes de produzir conteúdo, especialmente conteúdo multimídia levando em consideração recursos como câmeras e microfones de alta qualidade.

Somando-se a capacidade de produzir conteúdo multimídia e a conectividade de rede disponível, não é uma surpresa que os usuários de dispositivos móveis queiram compartilhar seu conteúdo pela Internet. Fabricantes prevêem que em cinco anos, um quarto de todo o conteúdo de entretenimento de Internet e telefonia móvel virá do material criado pelos usuários de dispositivos móveis, como aponta estudo feito em dezembro de 2007 pela Nokia[7]. Portanto, seguindo o crescimento de serviços *Peer-to-Peer* (P2P) na Internet, esses usuários desejam compartilhar conteúdo de maneira P2P, sem a necessidade de uma infra-estrutura central, e principalmente de maneira simples e fácil [50].

Estes fatos levam a novas tendências e possibilidades, e com isto surge a necessidade novos tipos de serviços e aplicações para os consumidores finais. Uma nova categoria de serviço é o oferecimento de Vídeo Sob-Demanda (VoD) para dispositivos móveis. Neste tipo de serviço, o conteúdo é fornecido em tempo real ao usuário quando este o deseja. Tratando-se de dispositivos móveis, novos desafios e problemas surgem, pois estes têm recursos restritos de bateria, processamento, transmissão de dados, entre outros. Este trabalho está incluído neste contexto, da oferta de serviços de VoD para dispositivos móveis.

1.1 Problemática

Ao se tratar de dispositivos móveis, algumas considerações devem ser feitas. Estes têm recursos restritos de processamento, memória, bateria, e comunicação. Além disto, considerando a aplicação objetivo deste trabalho, Vídeo Sob-Demanda, e as características do cenário de implantação, redes sem fio e a Internet, pode-se observar alguns desafios a serem superados.

Como discutido anteriormente, os usuários de dispositivos móveis produzem e, portanto, desejam compartilhar seu conteúdo multimídia. Para tanto, é necessário que estes enviem de alguma forma

[6] http://www.sun.com

[7] http://www.nokia.com

o conteúdo produzido para todos os seus amigos, ou interessados, que façam parte de uma mesma comunidade na Internet, por exemplo. A solução trivial para este problema seria um dispositivo móvel enviar o conteúdo para todos os outros interessados. Entretanto, como descrito no Capítulo 2, tratando-se de dispositivos com taxas de transmissão restritas, esta solução se torna inviável para um número elevado de dispositivos interessados. Portanto, é necessário oferecer um mecanismo que aperfeiçoe esta distribuição de conteúdo, principalmente levando em consideração que dispositivos móveis podem ser os provedores de conteúdo.

Como uma das características dos dispositivos móveis é sua mobilidade, estes, portanto, podem mudar seu ponto de conexão várias vezes. Um dispositivo pode estar conectado em uma rede local que se comunica com a Internet através de uma conexão aDSL, ou pode estar conectado a uma rede de telefonia móvel que oferece acesso à Internet através de outros mecanismos. Portanto, a localização relativa destes dispositivos na rede é um aspecto de mobilidade importante, pois, conhecendo-se a posição relativa entre dois dispositivos, por exemplo, pode-se distribuir um determinado conteúdo de maneira eficaz.

1.2 Relevância e Trabalhos Relacionados

Poucos trabalhos abordam a disponibilização de Vídeo Sob-Demanda para dispositivos móveis considerando aspectos de mobilidade. Para as abordagens existentes, entretanto, não se considera que estes dispositivos possam tanto ser os provedores como consumidores de conteúdo. Assim, o conteúdo ainda reside em uma entidade centralizadora, e apenas no transporte do conteúdo leva-se em consideração aspectos relacionados a esses dispositivos. Além desta limitação, a maioria dos trabalhos relativos a disponibilização de VoD para dispositivos móveis não consideram uma implantação na Internet.

1.2.1 Trabalhos Relacionados

Tran *et al.* apresentou em [48] o MobiVoD, que é um sistema de VoD voltado para redes *Ad Hoc* móveis. Apesar de ser voltado para redes Ad Hoc, o sistema é dividido em três componentes: servidor de vídeo, clientes e encaminhadores locais (*local forwarders*). Portanto, o conteúdo reside em um servidor central. A principal característica deste sistema é o uso de um esquema de armazenamento seletivo, chamado de *Dominating-Set Cache e Random-Cache*, o qual permite aos clientes executar Vídeos Sob-Demanda utilizando *caches* vizinhos.

Sato *et al.* apresentou em [41] [42] um sistema P2P com ênfase em dispositivos portáteis, o P2MVOD. Neste, o conteúdo é dividido em segmentos de mesmos tamanhos, e então é transmitido de maneira *broadcast*. Para diminuir o tempo para o início da execução para os clientes, os segmentos são divididos em segmentos compartilhados e segmentos de *patch*. Os segmentos de *patch* são trans-

mitidos por nós vizinhos que já tenham recebido o segmento. Neste sistema, entretanto, o conteúdo multimídia ainda reside em um servidor central.

Fouliras *et al.* apresentou em [13] um protocolo de VoD chamado LEMP, o qual os clientes atuam tanto como receptores de conteúdo, quanto como servidores parciais. Entretanto, características como localização relativa, e capacidade de transmissão não são considerados neste protocolo.

Em outros trabalhos, Yi Qi *et al.* em [15] e Regant *et al.* em [23] abordam apenas melhorias em protocolos de difusão para sistemas de VoD móveis. Em [15] é sugerido um esquema de difusão chamado *Regular/Sub-channel Staggered Broadcasting* (RSB). Este é uma variação do protocolo *Staggered Broadcasting*, no qual é feito um particionamento dos canais da rede em canais regulares e sub-canais, de modo a reduzir a latência e ao mesmo minimizar a banda de transmissão da rede. Maiores detalhes serão abordados no Capítulo 2.

1.3 Objetivos

O objetivo deste trabalho é propor um protocolo de Vídeo Sob-Demanda para dispositivos móveis, considerando aspectos como a localização relativa destes dispositivos na Internet, e suas capacidades de transmissão. O protocolo deve levar em consideração que estes dispositivos são tanto provedores como consumidores de conteúdo. Como um dos possíveis cenários de implantação é a Internet, este protocolo deve funcionar de maneira P2P.

Por fim, este protocolo deve ser validado através de simulações de vários cenários e da implementação de um protótipo que integra todas as ferramentas utilizadas.

1.4 Estrutura da Dissertação

O restante deste trabalho está organizado da seguinte maneira:

- **No Capítulo 2** são apresentados os principais temas relacionados a este trabalho, assim como são apresentados detalhes das tecnologias utilizadas no desenvolvimento do mesmo.
- **No Capítulo 3** é apresentado o protocolo Granola, desenvolvido para atender os objetivos descritos anteriormente.
- **No Capítulo 4** são apresentados os resultados de várias simulações feitas com o protocolo Granola, assim como a descrição de um protótipo desenvolvido para validar as ferramentas utilizadas.
- **No Capítulo 5**, por fim, são apresentadas as considerações finais sobre o trabalho, assim como planos e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos necessários para um melhor entendimento do cenário e do processo de desenvolvimento deste trabalho. Primeiramente é apresentado um resumo sobre o que é Vídeo sob-Demanda e quais são suas principais soluções e arquiteturas existentes. Em seguida o conceito de redes P2P é apresentado. Também é discutido quais características das redes P2P devem ser levadas em consideração no desenvolvimento de um sistema de *streaming* multimídia. Na seção seguinte são discutidos os principais métodos para estimar a localização relativa de um dispositivo na Internet. Por final, na última seção é feita uma discussão sobre as ferramentas para estimar a capacidade de transmissão de dispositivos em uma rede.

2.1 Vídeo Sob-Demanda

Sistemas de Vídeo Sob-Demanda têm como objetivo oferecer aos usuários a possibilidade de "sintonizar" um vídeo a qualquer momento sem perder o início da execução do mesmo.

A solução mais simples para este serviço seria oferecer um fluxo unidirecional ponto-a-ponto entre o servidor (provedor de conteúdo) e o cliente (usuário final e consumidor de conteúdo). Entretanto, com o aumento do número de clientes a banda de transmissão necessária aumentaria proporcionalmente, pois para cada cliente seria necessária uma conexão individual. Na Figura 2.1 ilustra-se este cenário.

Em uma rede sem fio a banda de transmissão é mais restrita do que em um sistema de banda larga com fio e, portanto, é necessário um protocolo de transmissão mais eficaz. Neste contexto, vários protocolos de transmissão por difusão/difusão limitada (*broadcasting/multicast*) foram propostos [2] [21] [30] [22] [49] [27]. Estes protocolos disponibilizam soluções para compartilhar fluxos multimídia entre o maior número de usuários possíveis. Os protocolos baseados em difusão resolvem estes problemas agindo de uma maneira proativa, garantindo assim a provisão do serviço, porém com um certo atraso [19]. Enquanto que os protocolos baseados em difusão limitada atuam também de maneira reativa na tentativa de diminuir o atraso para a provisão do serviço [30].

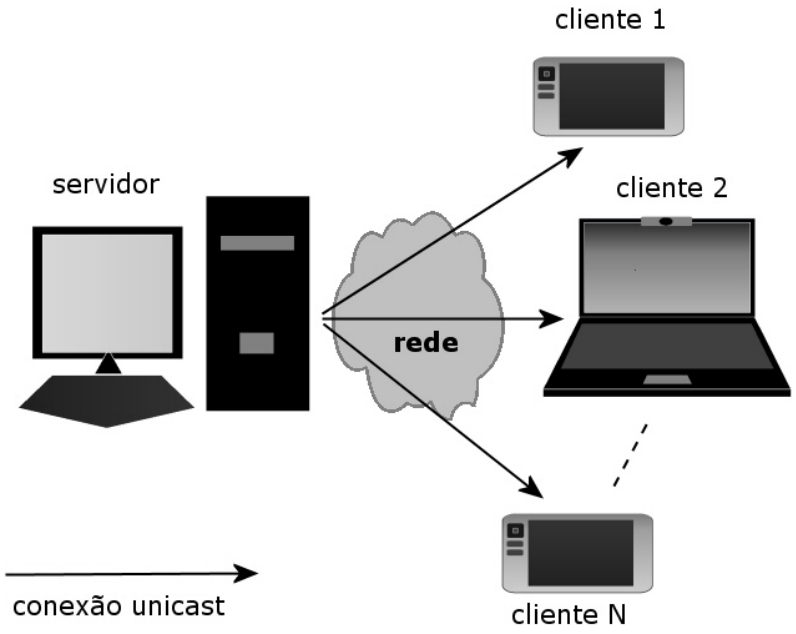


Figura 2.1: Esquema para um sistema de VoD comum.

Estes protocolos dividem o fluxo multimídia em uma série de segmentos os quais são transmitidos de maneira periódica em canais dedicados e distintos. Ao cliente é garantido que ao se executar um certo segmento do fluxo de vídeo o segmento posterior já tenha sido recebido, desta maneira o vídeo pode ser executado continuamente. O cliente, entretanto, terá que esperar para a ocorrência do primeiro segmento para então começar a execução da mídia. Portanto, protocolos de difusão de VoD oferecem apenas serviços quase VoD (*near VoD*)[1], enquanto que os protocolos de difusão limitada podem oferecer serviços verdadeiramente VoD (*true VoD*)[2] porém com um aumento do consumo da banda de transmissão.

2.1.1 Arquiteturas VoD

Várias arquiteturas para sistemas de VoD foram propostas, porém, as mais comuns podem ser divididas em 4 grupos [47]. Na arquitetura centralizada, o servidor central distribui o conteúdo para todos os clientes, como ilustrado na Figura 2.2(a). Esta é a arquitetura mais simples de implementação, entretanto ela é suscetível a vários problemas, destacando-se o grande número de requisições que o servidor deve suportar. Para tratar estes problemas, arquiteturas distribuídas foram propostas com o uso de servidores *proxy* instalados em regiões estratégicas. Estes servidores armazenam réplicas do conteúdo para reduzir a carga no servidor central como ilustrado na Figura 2.2(b). Uma extensão desta abordagem é o uso de redes de distribuição de conteúdo (*Content Distribution Networks - CDNs*). Nestas redes, requisições podem ser enviadas para outras CDNs que geralmente estão locali-

[1]Serviço onde o usuário tem que esperar um certo tempo até o início da execução do vídeo
[2]Serviço onde o vídeo é executado imediatamente

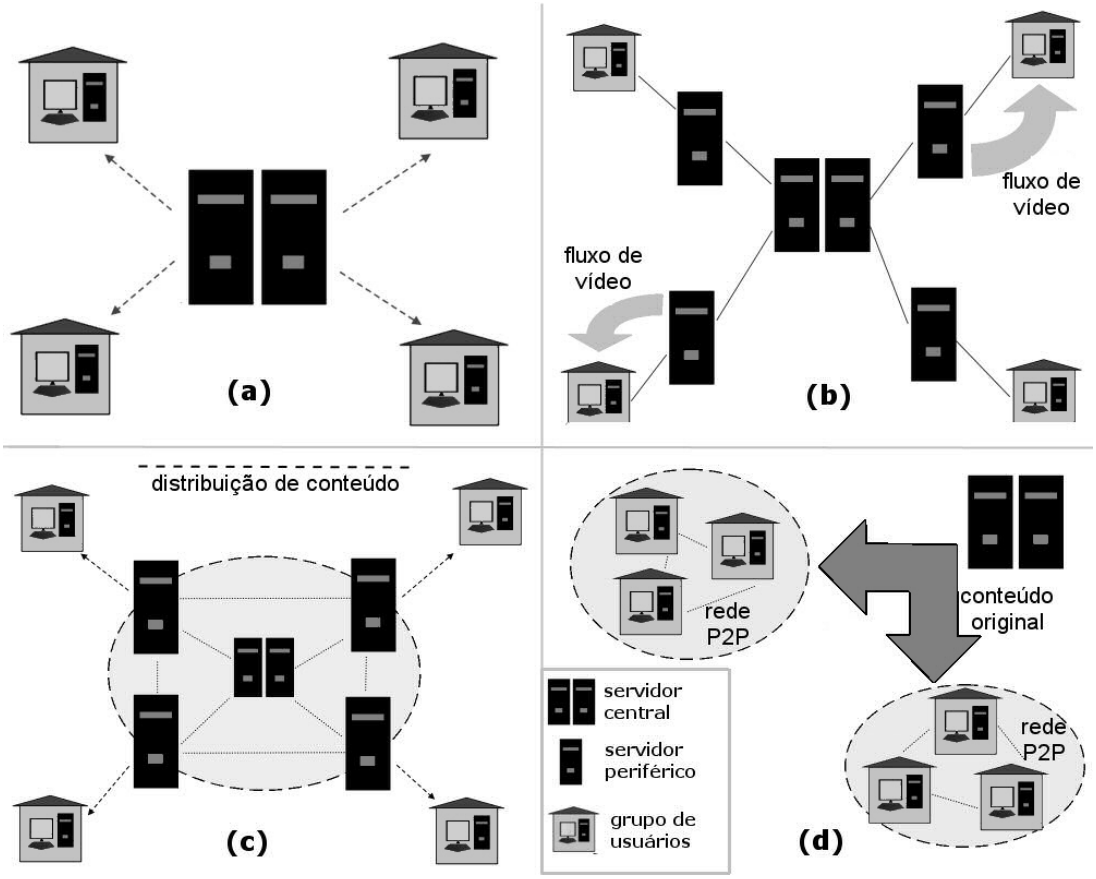


Figura 2.2: Topologias de sistemas VoD.

zadas na borda da rede [47], como ilustrado na Figura 2.2(c).

Recentemente foram propostas arquiteturas híbridas baseadas em redes P2P, nas quais o conteúdo é distribuído entre os usuários, como ilustrado na Figura 2.2(d). Arquiteturas híbridas são paradigmas promissores, pois a capacidade da rede cresce com o número de usuários [47].

Analisando estas arquiteturas comuns, o funcionamento de um sistema VoD pode ser descrito através de três procedimentos [47]:

- *Alocação e distribuição de conteúdo*: define qual a maneira mais eficaz de distribuir e armazenar o conteúdo entre os usuários em uma arquitetura híbrida, ou entre os servidores *proxy* em uma arquitetura baseada em *proxies*. Para fazer esta distribuição vários fatores devem ser considerados, dentre eles a popularidade de um certo conteúdo durante um certo tempo, e o quão dinâmico um determinado conteúdo se torna "frio" (pouco popular), por exemplo.

- *Requisição de Rotas*: processo através do qual as requisições dos clientes são direcionadas para o melhor servidor (mais próximo ou menos congestionado) para evitar o congestionamento ou queda de desempenho nos servidores, deste modo distribuindo melhor a carga pela rede. Este procedimento é apenas aplicável em sistemas VoD onde são utilizados múltiplos servidores, como uma arquitetura *proxy*.

- *Entrega de Conteúdo*: consiste da transmissão da mídia do servidor (provedor de conteúdo) para o cliente. Esta entrega pode ser feita de maneira *unicast* (mais comum) ou de maneira *multicast/broadcast*, como descrito anteriormente. Além destes mecanismos de entrega, a distrbuição de conteúdo, apesar de menos usual, pode ser feita utilizando redes P2P.

Nas próximas duas subseções serão apresentados os principais protocolos relativos aos mecanismos mais comuns para entrega de conteúdo de VoD.

2.1.2 Protocolos de Difusão

O protocolo de difusão mais simples e o mais antigo, é o *Staggered broadcasting*. Neste, vários canais com banda suficiente para transmitir o vídeo por inteiro são alocados. O início de cada vídeo é escalonado igualmente entre os canais. Portanto, o cliente sintoniza o canal que inicia a próxima transmissão, a qual é o início do vídeo.

Outros protocolos mais eficazes foram propostos, estes podem ser subdivididos em três grupos [19]. Para os protocolos do primeiro grupo o vídeo é dividido em segmentos de tamanho crescente, os quais são transmitidos em canais com mesma banda de transmissão. Estes são baseados no protocolo *Pyramid Broadcasting* (PB) [49]. O tamanho dos segmentos segue uma série geométrica, e para prover a entrega dos vídeos em tempo, cada canal transmite os segmentos em uma alta taxa e, portanto, os clientes devem ter uma alta capacidade de E/S e armazenamento de dados. Para tratar esses requisitos, variações deste protocolo foram propostas, destacando-se o *Permutation-based Pyramid Broadcasting* (PPB) [1], *Skyscraper Broadcasting* (SB) [22], e o *Fast Broadcasting* (FB) [28].

O segundo grupo de protocolos de difusão é baseado no protocolo *Harmonic Broadcasting* (HB) [27]. Neste caso o vídeo é dividido em segmentos de tamanhos iguais que são transmitidos em canais lógicos de banda de transmissão decrescente. Cada segmento é transmitido repetidamente em cada canal, portanto, neste caso requer-se muito menos banda de transmissão no servidor do que para os protocolos PB. Em HB os clientes começam a receber cada fluxo de segmentos logo após receber o primeiro segmento. Quando o cliente está pronto para consumir um segmento *i*, este terá recebido *i-1* pedaços daquele segmento e finalizará a recepção do último pedaço daquele segmento durante a execução do mesmo. Portanto, no caso de HB requer-se que a largura de banda do cliente seja a mesma que a taxa de transmissão do servidor, e que esse suporte o armazenamento de aproximadamente 37% do vídeo inteiro. Um dos principais problemas do HB é que ele não garante a entrega dos dados em tempo, portanto, outros protocolos foram propostos baseados nele, como o *Cautious Harmonic* (CHB), o *Quasi-Harmonic* (QHB) [36] e o *Polyharmonic* (PHB) [35].

O último grupo de protocolos, *Pagoda Broadcasting* [24] e o *New Pagoda* [34], é um híbrido dos protocolos PB e HB. O vídeo é dividido em segmentos de tamanho igual, assim como o HB, e são mapeados em fluxos de mesma banda como o PB. Esses protocolos utilizam multiplexação por divisão no tempo (TDM) para garantir que segmentos sucessivos sejam transmitidos com uma

freqüência decrescente (como o HB). O resultado disto é que não é necessário mais banda e, ao mesmo tempo, não são utilizados mais fluxos ou menos segmentos do que os protocolos baseados no HB.

2.1.3 Protocolos de Difusão Limitada

Ao contrário dos protocolos de difusão, os protocolos de difusão limitada oferecem acesso ao serviço de maneira imediata [30] mantendo a banda de transmissão do servidor baixa. Portanto, de maneira semelhante aos protocolos de difusão, vários canais são compartilhados entre múltiplos clientes. Com isso, as requisições têm que esperar até que um certo número de clientes seja alcançado, para que então estes clientes sejam servidos por um único canal. Esta técnica é conhecida como *batching*.

Para eliminar esta espera algumas técnicas dinâmicas foram propostas [21] [2]. *Patching* [21] é a técnica dinâmica mais comum, esta tem como objetivo aumentar substancialmente o número de requisições que cada canal pode servir ao mesmo tempo. Na técnica de *patching* alguns canais são usados para entregar a parte restante (geralmente a parte inicial de um fluxo) em relação a um canal *multicast*. Desta maneira, o cliente ao requisitar um fluxo multimídia receberá o início deste via um *patch* e ao mesmo tempo armazenará o fluxo *multicast* para que este seja executado ao final do *patch*. Considerando vários fatores, como o tamanho da mídia, taxa de requisições e a capacidade de armazenamento dos clientes, algumas técnicas específicas de *patching* foram propostas [17] [41] [37].

2.2 Redes P2P

O princípio das redes P2P é que aplicações são providas por um certo número de entidades, chamadas de *peers* ou nós, que trabalham conjuntamente para executar tarefas. Todos os nós são responsáveis por contribuir com a rede compartilhando seus recursos de armazenamento, processamento, ou comunicação. Para estas redes, um grande número de clientes se comunica apenas com um pequeno número de servidores centrais, os quais são responsáveis por executar as tarefas de maneira contrária à tradicional arquitetura Cliente-Servidor

Em uma rede P2P cada nó provê funcionalidades e serviços de um servidor e ao mesmo tempo atuam como um cliente. Desta maneira, os serviços e recursos que seriam oferecidos por uma entidade central são disponibilizados de uma maneira distribuída entre os nós do sistema. Os nós formam um grupo lógico (*cluster*) chamado de rede sobreposta (*overlay network*). Este tipo de rede é construída sobre outra rede de computadores, comumente uma rede IP. Na Figura 2.3 ilustra-se um exemplo deste tipo de rede

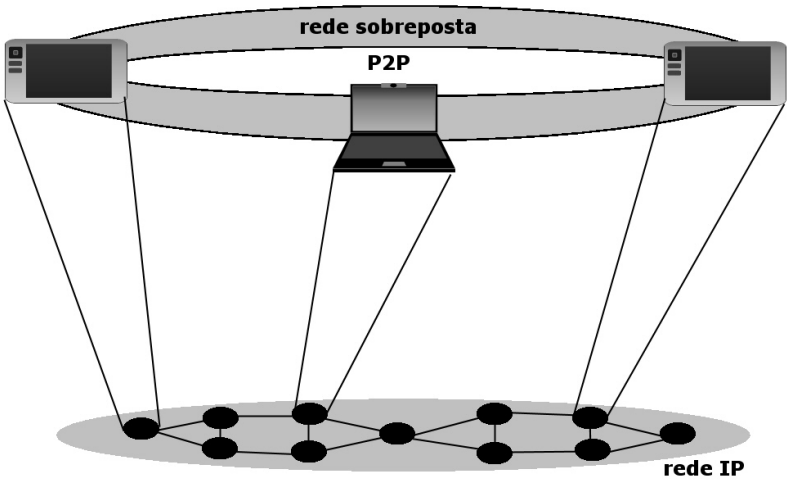


Figura 2.3: Rede P2P.

Na maioria dos sistemas P2P consideram-se nós efêmeros por natureza, ou seja, estes podem sair ou entrar na rede a qualquer momento. Esta característica faz com que as redes P2P se adaptem rapidamente a transição desses nós.

Como redes P2P são altamente distribuídas, estas devem ser capazes de armazenar e localizar informações de maneira eficaz. Estas podem ser divididas em dois grupos, redes estruturadas e não estruturadas.

Redes P2P não estruturadas não oferecem nenhuma estrutura específica para a topologia da rede. Nestas, os nós acessam a rede com nenhuma informação inicial da topologia. Com isso, os nós são organizados em grafos aleatórios de maneira hierárquica ou não, e utilizam mecanismos de busca do tipo inundação ou busca aleatória. Então, cada nó visitado analisa a busca em seu próprio diretório de conteúdo. Isto não é eficaz, pois buscas por conteúdo que não estão amplamente distribuídos pela rede, devem ser enviadas a uma grande quantidade de nós até que este conteúdo seja encontrado.

As redes estruturadas, por sua vez, oferecem alguma topologia específica para manter a rede, para isto, algumas fazem uso de DHTs (*Distributed Hash Tables*) para organizar os nós. Nestes sistemas, cada recurso tem um identificador, a qual é obtida através de uma função matemática conhecida como *hash* que é usada para transformar uma cadeia de caracteres de tamanho variável em um número de tamanho fixo. Exemplos de funções *hash* incluem o MD5 (*Message Digest 5*) e o SHA-1 (*Secure Hash Algorithm 1*). Nós na rede P2P podem ser encontrados usando uma chave que é similar ao um endereço IP na Internet, exceto que estas chaves atuam como um identificador para o nó. A chave pode ser o *hash* para a descrição do conteúdo em uma rede de compartilhamento de arquivos, ou simplesmente o *hash* do endereço IP em uma rede de comunicação P2P. Os nós mantém uma tabela com um número limitado de vizinhos na forma de uma tabela *hash* para uso de roteamento a nível de aplicação. Quando um usuário deseja fazer uma busca, este consulta sua lista de nós e entra em contato com o nó com a ID mais próxima da ID desejada. Se o nó encontrado não tem conhecimento de como encontrar o recurso desejado, este retorna informação sobre outro nó próximo, ou encaminha

a requisição para outro nó vizinho. Desta maneira, a requisição em algum momento alcança o nó responsável pelo recurso procurado, e então este responde ao nó procurador.

2.2.1 *Streaming* Multimídia em Redes P2P

Em um sistema de *streaming* multimídia P2P, um ou vários nós provedores de conteúdo encaminham dados multimídia para nós requisitantes. Estes nós requisitantes podem se tornar nós provedores de outros posteriores nós requisitantes. Como todo nó contribui para a rede sobreposta (compartilhando capacidade de armazenamento e banda de transmissão), a capacidade do sistema é ampliada em comparação a uma arquitetura cliente-servidor.

Neste tipo de sistemas, dois problemas encontrados em redes P2P são de especial atenção:

- **Falha dos nós:** como em redes P2P os nós nem sempre permanecem ativos, os nós requisitantes devem encontrar novos nós provedores na substituição de nós provedores antigos.
- **Banda de transmissão limitada e dinâmica:** os nós na rede P2P têm características diferentes, principalmente em relação a banda de transmissão que eles podem compartilhar. Isto faz que alguns nós provedores apenas possam compartilhar seu fluxo multimídia com apenas outro nó, por exemplo. Portanto, o sistema deve ser capaz de se adaptar as mudanças da banda de transmissão de cada nó, de modo que mantenha a qualidade do fluxo enviado.

Estes problemas são ainda mais críticos em redes P2P que têm dispositivos móveis em sua formação. Estes dispositivos têm em média uma banda de transmissão menor em relação a maioria dos computadores pessoais (os quais em sua maioria estão conectados a redes de banda larga). Além disto, por uma característica do seu meio de transmissão, as tecnologias de redes sem fio são mais propícias a sofrerem variações da banda de transmissão, seja porque o nó está se distanciando da estação base, ou porque o meio está sofrendo alguma interferência, por exemplo.

Para sobrepor estes problemas são necessários localizar os nós provedores mais próximos (para se obter uma menor latência na transmissão, por exemplo) e com suficiente banda de transmissão para compartilhar. Algumas técnicas para localização de nós para estes sistemas incluem:

- **Diretório centralizado:** Este é o sistema mais simples e mais comumente utilizada para a localização de nós. Nesta técnica, as informações de todos os nós são mantidas em um servidor centralizado. Neste sistema, um nó requisitante primeiro envia sua requisição ao servidor. O servidor, então, localiza o melhor nó para atender a requisição, e responde ao nó requisitante informando-o qual será seu respectivo nó provedor. Umas das vantagens deste sistema é sua fácil implementação e simples implantação. Além disto, com um servidor centralizado os procedimentos de entrada e saída da rede se tornam rápido, pois, é apenas necessário contactar o servidor central para informar o procedimento de entrada ou de saída. Entretanto, com um

número *N* de nós, o servidor deve manter *O(N)* estados, o que pode sobrecarregar o servidor caso *N* seja grande. Além disto, o servidor se torna um ponto de falha do sistema. Este tipo de abordagem foi utilizada por sistemas P2P famosos, como por exemplo o Napster[3].

- **Abordagem baseada em DHT:** Como descrito anteriormente, DHTs são usadas para a localização de recursos em uma rede P2P distribuída e estruturada. Nesta abordagem, o estado dos nós e a carga de roteamento está distribuída entre os próprios nós. Várias estruturas DHT foram propostas e estão disponíveis, como o Chord [44], Pastry [40], Tapestry [52] e CAN [38].

- **Espalhamento controlado de mensagens:** Esta uma técnica utilizada por redes P2P não estruturadas. De modo simplificado, neste mecanismo, uma mensagem de busca é enviada a todos os nós vizinhos do nó requisitante na rede sobreposta. Esta mensagem é re-enviada pelos nós até que o item procurado seja encontrado. O nó com o resultado da busca, então, responde ao nó requisitante. Entretanto, o uso desta técnica pode resultar em muito tráfego na rede, e não garante que a busca seja executada com sucesso. Este tipo de mecanismo é utilizado por sistemas, como por exemplo o Gnutella [26].

Em um sistema de *streaming* multimídia P2P, fluxos de áudio e vídeo são divididos em pequenos segmentos de dados. Estes dados são numerados e devem ser entregues ao cliente antes do tempo de sua execução. Para tanto, é necessário construir e manter um sistema de entrega de conteúdo eficaz. Algumas técnicas para a distribuição de conteúdo em redes P2P incluem:

- **Árvore *multicast*:** como ilustrado na Figura 2.4, os nós participantes da árvore de distribuição recebem os dados de seus respectivos pais e os encaminham para seus nós filhos. Esta é uma das técnicas mais simples e intuitivas, entretanto, sofre de alguns problemas. Um nó na periferia da árvore, um nó folha, apenas recebe dados e não os encaminha. Uma árvore é mais suscetível a falhas, pois, caso um nó interior falhe, seus nós filhos e descendentes irão sofrer de uma interrupção no fluxo, portanto, é necessário mecanismos de recuperação eficientes e rápidos. Por causa destes problemas, esta abordagem é mais apropriada para pequenos grupos de distribuição. Sistemas como oStream [5] utilizam árvores *multicast* para oferecer um sistema P2P de *streaming* multimídia.

- **Múltiplas árvores *multicast*:** como alternativa as árvores *multicast* simples, foram propostos sistemas com múltiplas árvores. Nestes sistemas, o fluxo multimídia é distribuído através de múltiplas árvores. Portanto, um nó pode receber fragmentos do fluxo multimídia originados em diferentes nós. Esta abordagem é ilustrada na Figura 2.5. Uma das desvantagens deste sistema é sua dificuldade de implementação em relação à técnica anterior. Apesar de mais

[3] http://www.napster.com

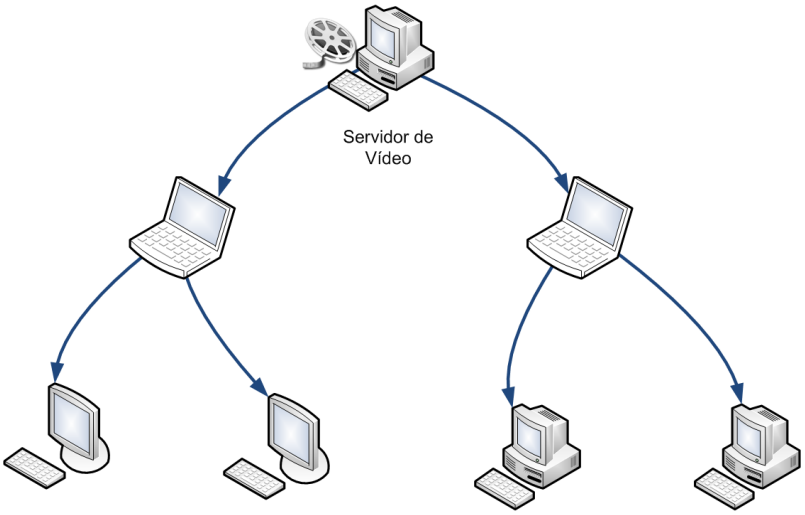


Figura 2.4: Ilustração de uma árvore *multicast* simples.

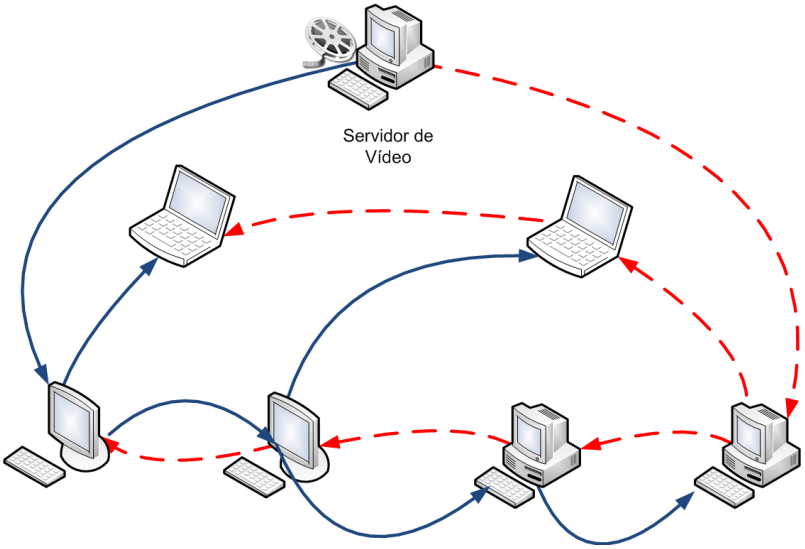


Figura 2.5: Ilustração de múltiplas árvores *multicast*.

complexo, sistemas como o SplitStream [4] e o CoopNet[4] utilizam múltiplas árvores *multicast* para a distribuição de conteúdo multimídia.

- ***Pull-based gossiping*:** Os protocolos *gossip*, também conhecidos como protocolos epidêmicos, não se baseiam em qualquer tipo de estrutura de rede para a entrega de dados. O modo mais comum destes protocolos é o modo *push*. Neste modo, nós enviam dados recebidos para um número de vizinhos escolhidos aleatoriamente. Entretanto, este modo de envio faz com que um número excessivo de dados duplicados seja enviado para a rede. Portanto, para sistemas de *streaming* multimídia são adotados protocolos epidêmicos do tipo *pull*. Neste modo, os nós requerem aos seus vizinhos os fragmentos de dados necessários, portanto, reduzindo a redundância na entrega de dados na rede. Sistemas como o CoolStreaming [51] utilizam esta técnica.

[4]http://research.microsoft.com/projects/CoopNet/

Outros sistemas, por exemplo, combinam estes protocolos com uma árvore de *multicast*, como HON (Hybrid Overlay Network) [53] para um sistema P2P de VoD.

2.3 Localizando Dispositivos na Internet

Vários sistemas foram propostos para estimar a localização de nós na Internet. Destes, a maioria foi proposta para estimar a distância entre os nós, ou seja, estimar o quão próximo estão dois determinados nós na rede, por exemplo. O IDMaps (*Internet Distance Maps*) [14] é um destes sistemas. Neste, localizadores (*tracers*) são colocados em locais estratégicos na Internet. Estes localizadores medem a latência entre eles próprios, e anunciam esta informação a seus clientes. Então, a distância entre dois clientes A e B é estimada como a soma da distância entre A e o localizador mais próximo de A, a distância entre B e o localizador mais próximo de B, e a distância entre os dois localizadores anteriores. Um problema relacionado ao IDMaps vem de quantos localizadores devem ser usados no sistema. O número de localizadores é essencial para se obter um sistema mais preciso. Porém, um cliente deve procurar todos os localizadores existentes, para então determinar qual está mais próximo. Portanto, um número muito grande de localizadores pode fazer desta busca uma tarefa muito demorada. Outros esquemas utilizam localizadores, porém com algumas diferenças no cálculo da distância, como o DDM [46] e o M-Coop [43]. Uma característica comum e limitante destes sistemas é que os endereços do nó destinatário e do nó fonte devem ser conhecidos a priori. Ou seja, estes sistemas apenas oferecem uma estimativa da distância entre dois determinados pontos.

Outros sistemas utilizados para a localização de nós faz uso de um método baseado em coordenadas de rede. Algoritmos de coordenadas de rede caracterizam a rede como um espaço geométrico. Cada nó na rede é caracterizado como um ponto no espaço geométrico. Com isso, a distância entre dois nós é determinado pela distância entre os dois respectivos pontos no espaço geométrico. Quando for necessário calcular a distância entre dois nós, portanto, não é mais necessário fazer novas medições na rede, como por exemplo medir latência entre os nós, calcula-se somente a distância entre as coordenadas.

Para o cálculo de coordenadas de rede, destacam-se alguns sistemas. O GNP (*Global Network Positioning*) [32] e o NPS (*Network Positioning Systems*) [33] fazem uso de nós marcos (*landmarks*), os quais medem RTTs (*Round-trip time*) entre si e usam estas informações para calcular coordenadas Cartesianas para cada nó marco. Estas coordenadas são então distribuídas entre os nós clientes, os quais medem RTTs para os nós marco e calculam as coordenadas para si. Após este cálculo de coordenadas, os nós guardam sua posição, coordenadas, as quais podem ser usadas para medir a distância entre os nós. Entretanto, estes sistemas ainda são dependentes de uma infra-estrutura previamente formada, neste caso, os nós marco.

Outro sistema para o cálculo de coordenadas de rede é o Vivaldi [6]. Este, entretanto, não faz uso de nós marco, e funciona de uma maneira distribuída. Portanto, em um sistema que utiliza o

Vivaldi, todos os nós são considerados iguais no cálculo de suas coordenadas. O princípio utilizado no Vivaldi é considerar toda a rede como um sistema de molas. Nesta consideração, Vivaldi coloca uma mola entre cada par de nós, e determina o comprimento de repouso como a latência entre os nós. O comprimento de cada mola é a distância entre as coordenadas dos nós. A energia potencial de uma mola é proporcional ao quadrado da diferença entre o comprimento de repouso e o comprimento atual da mola. Esta diferença, então, é igual ao erro de predição das coordenadas. Portanto, minimizar a energia potencial do sistema de molas corresponde a minimizar o erro de predição das coordenadas. Na próxima subseção são apresentados mais detalhes do algoritmo utilizado pelo Vivaldi.

2.3.1 Vivaldi

Como descrito anteriormente, o Vivaldi atribui uma coordenada sintética a um nó em um espaço de coordenadas. Esta atribuição tem como objetivo prevê o RTT da transmissão de pacotes entre os nós. Entretanto, não é possível prever com precisão o RTT entre os nós usando um espaço de coordenadas com poucas dimensões, pois, latências na Internet violam a desigualdade triangular [6]. Portanto, o algoritmo tenta achar coordenadas que minimizem erros de predição.

Para esta minimização, suponha que L_{ij} seja o RTT atual entre os nós i e j, e que x_i seja as coordenadas atribuídas para o nó i. Considerando que a distância entre os dois nós seja a predição do RTT, o erro de predição pode ser definido usando a função de erro quadrática:

$$E = \sum_i \sum_j (L_{ij} - ||x_i - x_j||)^2 \tag{2.1}$$

onde, $||x_i - x_j||$ é a distância entre os nós i e j.

Portanto, um dos objetivos do Vivaldi é minimizar a Equação 2.1, para tanto, no trabalho onde o Vivaldi foi introduzido [6], foi apresentada uma alternativa centralizada, a qual não será apresentada nesta dissertação. Este algoritmo centralizado calcula as coordenadas para todos os nós dado todos os RTTs. A principal contribuição no contexto do Vivaldi é estender esse algoritmo de modo que cada nó calcula e ajusta continuamente suas coordenadas, tendo como base as medições de RTT feitas dele mesmo a um conjunto de outros nós. Além do RTT, para o cálculo é usado as coordenadas desses outros nós. Portanto, cada nó obtém suas coordenadas de maneira distribuída.

Para tanto, cada nó simula seu próprio movimento no sistema de molas. Cada nó mantém sua posição atual, começando da origem. Toda vez que um nó se comunica com outro, ele mede o RTT para este outro nó, e também adquire as coordenadas daquele nó. Com essas informações vindas de outros nós, cada nó permite ser empurrado por um curto período de tempo por uma das molas. Cada movimento deste reduz o erro de predição em relação a um dos nós do sistema. Com isto, o nó continuamente se comunica com os outros nós, trocando suas novas informações de coordenadas, de modo que eles convirjam para coordenadas que diminuam o erro de predição.

Esta atualização de coordenadas é feita através das seguintes equações:

$$F_{ij} = (L_{ij} - ||x_i - x_j||) \times u(x_i - x_j) \tag{2.2}$$

$$x_i = x_i + \delta \times F_{ij} \tag{2.3}$$

onde: δ é o intervalo de tempo de atualização.

Na equação 2.2, a coordenada x_i é atualizada na direção da força que a mola entre i e j exerce em i. Esta força é caracterizada pela lei de Hooke, e equivale a equação 2.3. O δ é utilizado para simular a evolução da rede de molas, ou seja, a cada intervalo de tempo o algoritmo movimenta o nó x_i na direção da força F_{ij}.

Portanto, para a convergência das coordenadas, a escolha de um δ é de fundamental importância. Uma escolha equivocada de δ, por exemplo um δ muito alto, pode fazer o sistema oscilar bastante antes de convergir. Outro problema, é lidar com nós que tenham coordenadas com um alto erro de predição, como é o caso de nós que estão entrando na rede. Isto faz, por exemplo, com que um nó i que se comunique com um nó j com um erro de predição alto, aumente seu erro de predição ao invés de diminuí-lo.

Para sobrepor estes problemas, utiliza-se no Vivaldi um intervalo de tempo adaptativo que utiliza no seu cálculo o erro de predição do próprio nó e do nó remoto. Este cálculo é feito através da equação:

$$\delta = c_c \times \frac{ErroLocal}{ErroLocal + ErroRemoto} \tag{2.4}$$

onde: c_c é um parâmetro de sintonização do sistema.

Com estas premissas, o algoritmo do Vivaldi pode ser descrito utilizando o pseudocódigo apresentado no Algoritmo 2.1. Em (1), o algoritmo calcula o peso da amostra recebida utilizando como base o erro local e o erro remoto. Em seguida, o algoritmo atualiza o erro local (3) sendo este uma soma ponderada (utilizando a constante c_e e o peso da amostra) do erro da amostra (e_s) (2) e o erro atual do nó (e_i). Com este novo valor do erro atual do nó, o algoritmo atualiza o intervalo de tempo, e em seguida atualiza seu valor de coordenada.

2.4 Estimando Capacidade de Transmissão

Determinar uma estimativa da capacidade de transmissão de um dispositivo é bastante importante para determinar vários parâmetros de um sistema de distribuição de fluxo multimídia, como por exemplo:

- **Quantos dispositivos filhos pode ter um dispositivo provedor de conteúdo:** determinar quantos fluxos de multimídia um determinado dispositivo pode suprir sem que exista uma queda na taxa de transmissão individual de cada fluxo.

Algoritmo 2.1 Algoritmo do Vivaldi com intervalo de tempo adaptativo

Entrada(s): rtt, x_j, e_i

//peso entre o erro local e remoto (1)

1: $w = e_i/(e_i + e_j)$

//cálculo do erro relativo desta amostra (2)

2: $es = \mid ||x_i - x_j|| - rtt \mid /rtt$

//atualiza o erro local (3)

3: $e_i = e_s \times c_e \times w + e_i \times (1 - c_e \times w)$

//atualiza a coordenada localizadas

4: $\delta = c_c \times w$

5: $x_i = x_i + \delta \times (rtt - ||x_i - x_j||) \times u(x_i - x_j)$

- **Taxa de codificação de uma determinada mídia:** codificar um determinado vídeo de acordo com sua capacidade de transmissão.

Vários meios podem ser utilizados para se obter uma estimativa da capacidade de transmissão, dentre eles se destacam:

- Através de uma avaliação do tipo de interface de comunicação, por exemplo, caso seja um dispositivo conectado a uma rede GPRS pode-se estimar que ele tenha uma capacidade de transmissão máxima de 171.2 Kbps [11].

- Através de ferramentas que estimam a capacidade de transmissão e a banda de transmissão disponível entre dois pontos.

O primeiro meio citado, apesar de simples e aparentemente ineficaz, é bastante útil para se obter uma estimativa no caso de um nó entrante em uma rede P2P, por exemplo. Deste modo, é possível fornecer uma informação inicial e a partir daí, por exemplo, utilizar outras ferramentas para atualizar essa estimativa inicial durante sua participação na rede.

Várias ferramentas de software podem ser utilizadas para calcular apenas a capacidade de transmissão fim-a-fim em uma rede, como o Pathrate [10]. Considerando que um caminho fim-a-fim P seja formado por um conjunto de N enlaces em seqüência, e que cada enlace i tenha uma capacidade máxima de transmissão C_i, portanto, a capacidade de um caminho pode ser definida como:

$$C = \min_{i=1....N} C_i \tag{2.5}$$

Portanto, a capacidade fim-a-fim é definida como a taxa de transmissão máxima que um caminho pode oferecer a um fluxo, quando não exista mais tráfego em P.

Outras ferramentas, entretanto, estimam a banda de transmissão disponível em uma transmissão fim-a-fim. Ou seja, *a taxa de transmissão máxima que um caminho pode oferecer a um fluxo sem*

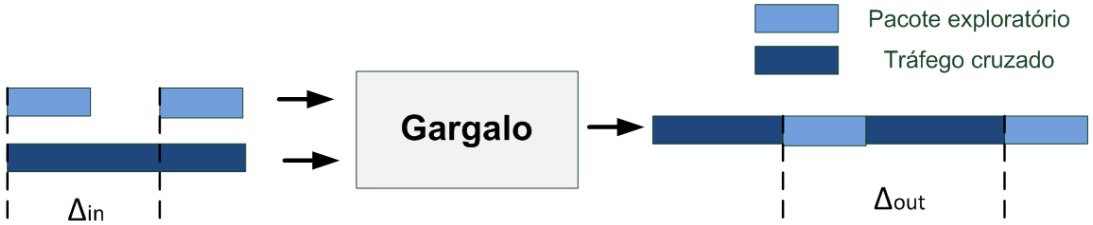


Figura 2.6: Modelo PGM para estimar banda de transmissão disponível.

reduzir a taxa de outros fluxos em P. No contexto deste trabalho, estas ferramentas são de interesse maior, pois, com estas, é possível ter uma estimativa da banda de transmissão que um determinado nó ainda é capaz de compartilhar, ou se um terminado fluxo está sendo enviado por um caminho que já esteja próximo da saturação (banda disponível próximo de zero, por exemplo).

Considere que $\lambda_i(t)$ é o tráfego em um enlace i em um instante t, e que um enlace ou está sem utilização ou transmitindo em sua capacidade máxima. Portanto, pode-se definir que a banda de transmissão disponível média (A_i) em um intervalo de tempo T no enlace i é:

$$A_i(t,T) = \frac{1}{T}\int_t^{t+T} (C_i - \lambda_i(t))dt \tag{2.6}$$

E que a banda de transmissão disponível em um caminho P, é:

$$A = \min_{i=1....N} A_i \tag{2.7}$$

Dentre as recentes ferramentas utilizadas para estimar a banda de transmissão disponível, pode-se dividi-las em dois grupos de acordo com a técnica utilizada [45]:

- **O Modelo de Análise de Lacunas** (*The probe gap model* - GPM) explora a informação da lacuna de tempo entre a chegada de dois pacotes de "exploração" no dispositivo recebedor. Os par de pacotes é enviado com uma diferença de tempo Δin e chega ao dispositivo recebedor com um intervalo de tempo Δout. Assumindo a existência de apenas um "gargalo" entre os enlaces do caminho, portanto, Δout é o tempo necessário para este gargalo transmitir o segundo pacote exploratório e o tráfego cruzado que está passando depois de Δin, como ilustrado Figura 2.6. Portanto, o tempo necessário para transmitir o tráfego cruzado é $\Delta out - \Delta in$, e taxa deste tráfego é $\frac{\Delta out - \Delta in}{\Delta in} \times C$, onde C é a capacidade de transmissão do enlace gargalo. A banda de transmissão disponível é então:

  $$A = C \times (1 - \frac{\Delta out - \Delta in}{\Delta in}) \tag{2.8}$$

  Ferramentas como Spuce [45] e IGI [20], utilizam este modelo para estimar a banda de transmissão disponível.

- **O Modelo de Análise da Taxa** (*The probe rate model* - PRM) é baseado no conceito de auto-indução de congestão, ou seja, neste modelo o dispositivo remetente envia tráfego exploratório a altas taxas de transmissão. Se este tráfego chega ao recebedor com a mesma taxa então esta taxa está abaixo da banda de transmissão disponível. Entretanto, se este tráfego exploratório ultrapassa a banda disponível, a taxa de chegada no recebedor vai ser menor do que a enviada. Então, pode-se medir a banda de transmissão disponível procurando o ponto o qual a taxa de transmissão do tráfego exploratório diminui. Ferramentas como o Pathload [25] e Pathchirp [39], utilizam este modelo.

2.5 Sumário

Neste capítulo foram apresentados os conceitos necessários para o desenvolvimento do protocolo introduzido no próximo capítulo. Dentre os principais conceitos apresentados estão os principais tipos de sistemas P2P usados para *streaming* de multimídia. Em seguida, foi apresentado em detalhes as principais características do protocolo Vivaldi, usado para a determinação das coordenadas de rede de um nó. Por fim, foram apresentados os modelos e as principais ferramentas usadas para determinar a capacidade de transmissão disponível de um nó.

Capítulo 3

Protocolo de VoD para Dispositivos Móveis

Neste capítulo é apresentado o protocolo de distribuição de Vídeo sob-Demanda desenvolvido neste trabalho, chamado de Granola [8] [7]. Na primeira seção é apresentado o cenário de aplicação de um sistema de VoD onde este protocolo se aplicaria, dando uma maior atenção às limitações que este cenário apresenta. Em seguida é feita uma descrição resumida do protocolo de distribuição, tendo como exemplo um sistema VoD descrito na seção anterior. Por fim é feita uma descrição detalhada do protocolo, além disso, são apresentados e avaliados três algoritmos de construção de árvores de distribuição desenvolvidos no decorrer deste trabalho.

3.1 Cenário e Requisitos

No processo de desenvolvimento de um sistema de Vídeo sob-Demanda que tem como principal objetivo oferecer um serviço de VoD para dispositivos portáteis com conectividade de rede sem fio (ou simplesmente dispositivos móveis - DM), alguns requisitos foram observados de modo a especificar o cenário de aplicação. Dentre estes requisitos destacam-se:

- **Os DM estão distribuídos pela Internet:** os dispositivos podem estar conectados a redes diferentes, de modo que possam estar em locais distantes e sem conectividade a nível de enlace dados e, portanto, o único modo de comunicação seja através de uma rede compartilhada, neste caso a Internet.
- **OS DM estão conectados em interfaces de redes diferentes:** como dispositivos móveis, estes podem estar conectados em redes sem fio com características diferentes, como uma rede GPRS ou uma WLAN conectada a um gateway DSL.
- **Os DM são os provedores de conteúdo:** os próprios dispositivos móveis são os criadores de conteúdo multimídia. Deste modo, uma tendência comum é que eles mesmos sejam os provedores de conteúdo, pois, com isso, estes podem criar o conteúdo e disponibilizá-lo imediatamente.

Com estes requisitos definidos, o cenário de aplicação do sistema proposto pode ser ilustrado na Figura 3.1. Neste cenário, por exemplo, dispositvos móveis estão conectados a redes de telefonia móvel diferentes e com tecnologias de transmissão diferentes, como GPRS e EDGE. Outros dispositivos podem estar conectados em redes locais em seus domicílios, por exemplo, através de *gateways* entre Wi-Fi e aDSL.

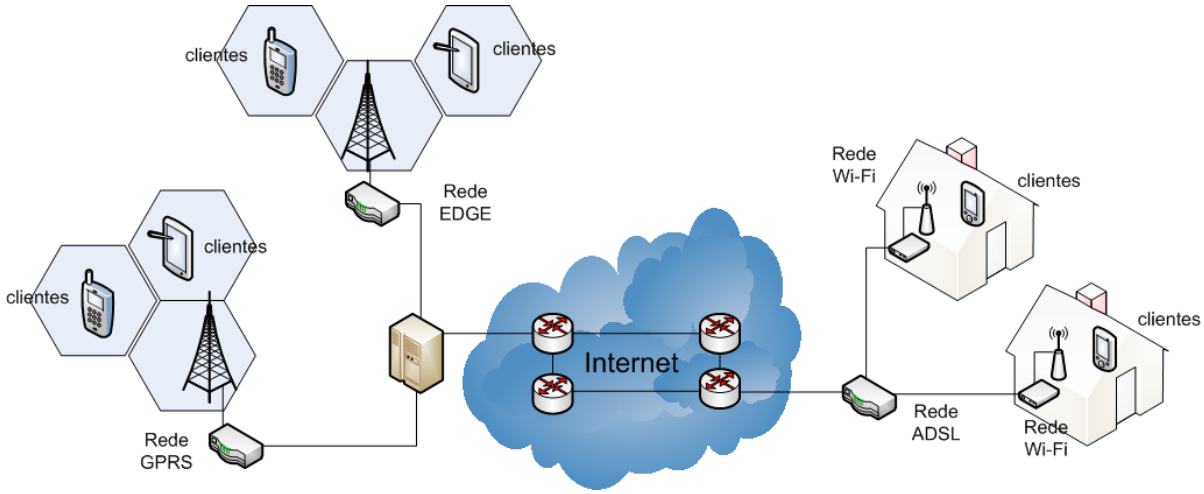


Figura 3.1: Cenário de aplicação para o sistema proposto.

3.2 Visão Geral do Sistema

Com os principais requisitos e cenário de aplicação definidos, foi proposto um sistema de Vídeo sob-Demanda para dispositivos móveis utilizando o paradigma de redes P2P. Neste sistema os mecanismos de localização de recursos (neste caso vídeo) e de distribuição de multimídia são separados. Deste modo, pode-se utilizar um dos mecanismos P2P introduzidos no Capítulo 2 para localizar os recursos na rede, como um diretório centralizado, um mecanismo de rede P2P estruturada, ou um mecanismo de rede P2P não estruturada. A escolha deste mecanismo de localização de recursos está fora do escopo deste trabalho.

Em relação ao meio de distribuição de conteúdo, neste trabalho é apresentado um protocolo o qual organiza os dispositivos em camadas de distribuição, e em cada camada é construída uma árvore *multicast* com os dispositivos. Na construção destas árvores são levados em consideração dois dos requisitos apresentados anteriormente: a localização relativa dos dispositivos na Internet, e sua capacidade de transmissão em relação a sua interface de rede. A Figura 3.2 ilustra o uso deste protocolo e sua interação com o mecanismo de localização.

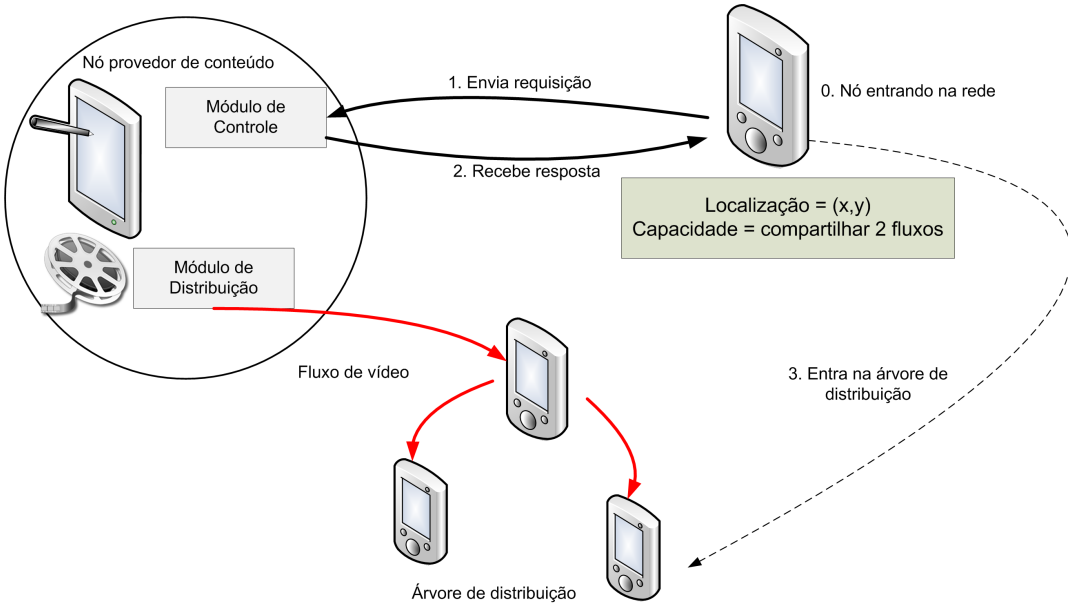


Figura 3.2: Visão geral do sistema.

3.3 Protocolo de distribuição

Como descrito anteriormente, a maioria dos protocolos e soluções para o oferecimento de serviços de VoD são baseados no uso de protocolos de *multicast*. Portanto, estas soluções não são apropriadas para implantação usando dispositivos distribuídos pela Internet [3]. Logo, o uso de um protocolo de difusão em nível de aplicação *(multicast* em nível de aplicação) é uma solução promissora para esse problema. Neste contexto, é introduzido o uso de uma técnica de VoD descrita no Capítulo 2, o *batching*. Com o *batching* os dispositivos são organizados em grupos, e cada grupo é atendido por um canal *multicast*. Para o protocolo introduzido neste trabalho, portanto, estes grupos, os quais serão chamados de camadas, são atendidos por um método de distribuição de difusão em nível de aplicação (*multicast* em nível de aplicação). Entretanto, o *batching* impõe um custo: os dispositivos devem esperar um intervalo de tempo T até que o um grupo seja formado e então um canal *multicast* seja fornecido. Portanto, este também é um dos custos do protocolo introduzido. A técnica de *batching* pode ser ilustrada na Figura 3.3.

Entretanto, considerando os requisitos definidos na seção anterior, um DM pode ser o provedor de conteúdo e, portanto, pode estar conectado a uma interface de rede com baixa capacidade de transmissão, por exemplo. Com isto, é possível que este DM não seja capaz de fornecer um fluxo multimídia para diferentes camadas. Como ilustrado na Figura 3.4, para sobrepor esta limitação, no protocolo aqui introduzido é feita uma comunicação hierárquica entre as camadas de comunicação. Deste modo, a camada superior oferece o fluxo multimídia para a camada inferior. Com esta regra imposta, cada camada terá no mínimo duas entidades, um nó fonte (src) e um nó sorvedouro ($sink$). O nó fonte é responsável por receber o fluxo do nó sorvedouro da camada superior, e distribuir este fluxo em sua camada.

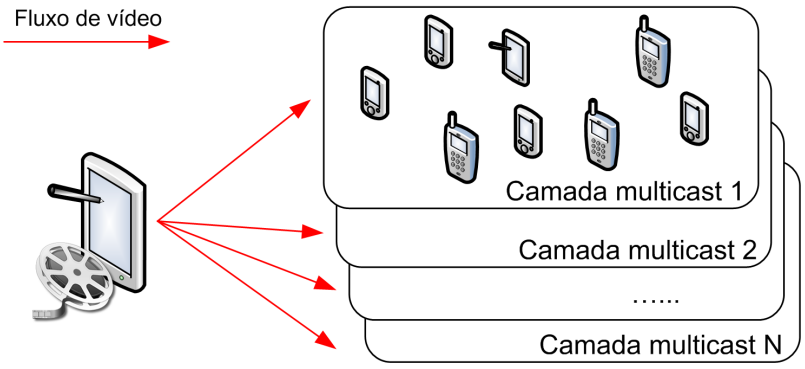


Figura 3.3: Técnica de *batching*.

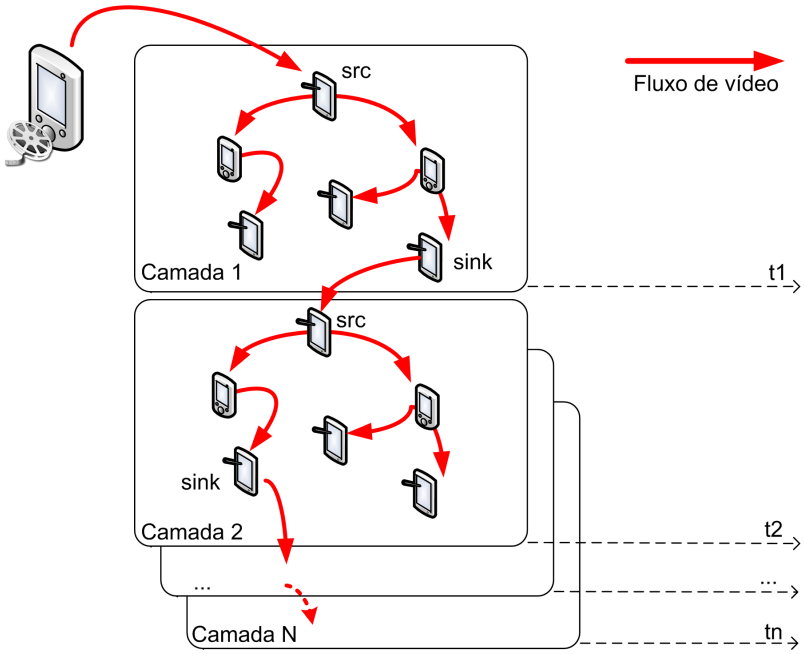


Figura 3.4: Técnica de distribuição do protocolo introduzido.

Como descrito anteriormente, em cada camada é usado um mecanismo de distribuição de conteúdo multimídia em nível de aplicação. No protocolo introduzido neste trabalho é utilizada uma árvore de distribuição *multicast*. Na formação desta árvore são consideradas:

- **a posição relativa do nó na rede:** utiliza-se as coordenadas de rede estimadas através de uma ferramenta como o Vivaldi [6].
- **capacidade de transmissão:** inicialmente, quando o nó entra na rede, é utilizada uma estimativa relacionada a capacidade de sua interface de comunicação. Posteriormente, no decorrer da transmissão, é utilizada uma ferramenta que estima a banda de transmissão disponível. Caso a banda disponível esteja saturada, o nó informa o nó controlador, o nó provedor de conteúdo, sobre esta mudança.

A seguir são apresentados os procedimentos de entrada, saída, e atualização dos nós na rede. Por fim, também são apresentados os algoritmos de construção das árvores de distribuição em cada camada.

3.3.1 Funcionamento do Protocolo

Para descrever o funcionamento do protocolo Granola as seguintes entidades são definidas [8] [7]:

- o nó provedor do conteúdo ($prov$), o qual é a fonte original da mídia;
- os nós fonte de cada camada (src), o qual fornece o fluxo multimídia para a sua camada;
- os nós sorvedouros de cada camada ($sink$), o qual fornece o fluxo multimídia para a camada inferior;
- a rede de comunicação de controle, a qual viabiliza a comunicação entre os nós requisitantes de conteúdo e o nó provedor;
- a rede de distribuição, a qual é a árvore de distribuição controlada pelo nó provedor através da comunicação feita na rede controle.

Além destas definições, também considera-se que todos os nós que participam da rede, no mínimo, sejam capazes de compartilhar um fluxo multimídia com outro nó. Além disto, também considera-se que o nó seja capaz de armazenar todo o conteúdo em seu meio de armazenamento. Estas considerações foram feitas para, inicialmente, evitar que nós "caronas" participem da rede, e para evitar que alguns nós não sejam capazes de armazenar todo o conteúdo o que, portanto, poderia resultar na incapacidade de compartilhar o fluxo com outros nós.

O funcionamento do protocolo começa a partir do momento que um nó localiza um conteúdo de seu interesse, neste caso vídeo, e requisita ao nó provedor este conteúdo. A partir de então são formadas duas estruturas de comunicação: uma cliente-servidor para o controle, e uma árvore de distribuição de conteúdo multimídia, como ilustrado na Figura 3.5.

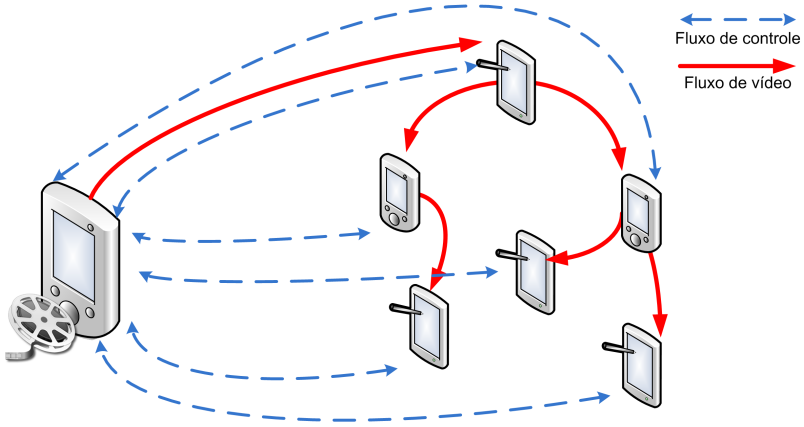


Figura 3.5: Redes de comunicação do sistema.

Por fim, considera-se que um vídeo V tem uma duração total Tt, e que este é dividido em N segmentos de $deltat$ de modo que $Tt = N \times deltat$.

Nó entrando na rede

Como ilustrado na Figura 3.6, após encontrar o nó que contém o vídeo de seu interesse, um nó requisitante inicialmente obtém as informações de contexto relevantes para o protocolo, as quais são:

- coordenadas atuais na rede $c(x, y)$;
- uma estimativa de sua capacidade de transmissão $shareM$.

Depois de obtidas essas informações, o nó envia ao nó provedor de conteúdo (ou nó controlador) uma mensagem de requisição (*JOIN*) com estas informações, e então espera uma resposta deste nó. O tempo de espera máximo tm é definido pelo gerenciador do sistema, por exemplo, 50% do tempo total do vídeo Tt que está sendo compartilhado, após este intervalo, o nó deve re-enviar a requisição.

Ao receber a primeira requisição o nó controlador inicia um relógio. A informação de tempo é usada para agrupar os nós requisitantes em camadas de acordo com o horário de chegada da requisição. O tempo de espera para a formação de cada camada, tc, pode ser definido individualmente para cada vídeo, desde que não ultrapasse 50% do tempo do vídeo. Ao final do período de tempo Ntc para a formação de uma camada Cn (N-ésima camada), a qual é formada pelas requisições entre o período de tempo $(N-1)tc$ e Ntc, os nós desta camada são organizados em uma árvore de distribuição de acordo com um dos algoritmos que serão apresentados ao final deste capítulo.

Após a construção da árvore de distribuição, para cada nó na camada são determinados os seguintes parâmetros:

- função do nó na camada, se $sink$, src, ou nó interno;
- os seus nós filhos;
- o seu nó pai.

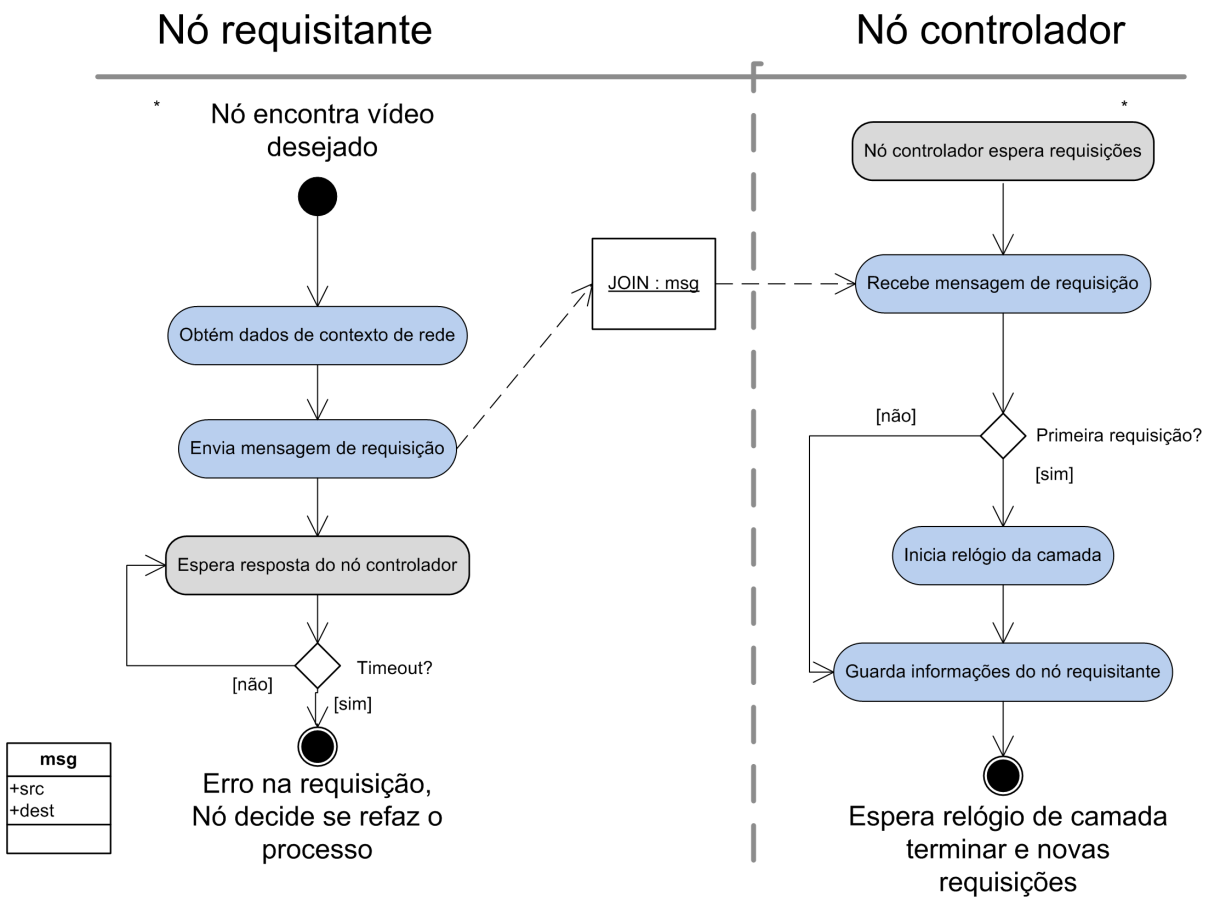


Figura 3.6: Fluxograma de requisição de um nó entrante.

Por fim, o nó provedor envia uma mensagem (*INFO*) para cada nó com estas informações. Este fluxo no nó controlador é ilustrado na Figura 3.7. Ao receber a resposta do nó controlador, o nó requisitante envia uma mensagem de requisição (*REQNODE*) ao nó que foi designado como seu pai, informando-o que está pronto para receber o fluxo multimídia a partir do segmento i, neste caso, como o nó está entrando na rede, este requisita o primeiro segmento. Paralelamente, este nó requisitante aguarda que seus nós filhos requisitem o fluxo multimídia. Caso o fluxo multimídia requisitado ainda não esteja disponível, ou seja, o nó ainda não tenha recebido o segmento i, este envia uma mensagem (*REPLYNODE*) ao nó filho informando que ainda não está preparado e, portanto, o nó filho esperará um intervalo de tempo aleatório para então requisitar o fluxo multimídia novamente. Após o recebimento e envio do segmento i, o nó pai encaminha todos os segmentos posteriores para seus filhos até que receba uma mensagem de interrupção do nó controlador, ou até que o fluxo de vídeo termine. Após o recebimento da resposta do nó controlador, o funcionamento de um nó requisitante pode ser descrito através de dois fluxos. Um responsável pelo controle dos filhos, e outro pelo envio do fluxo multimídia.

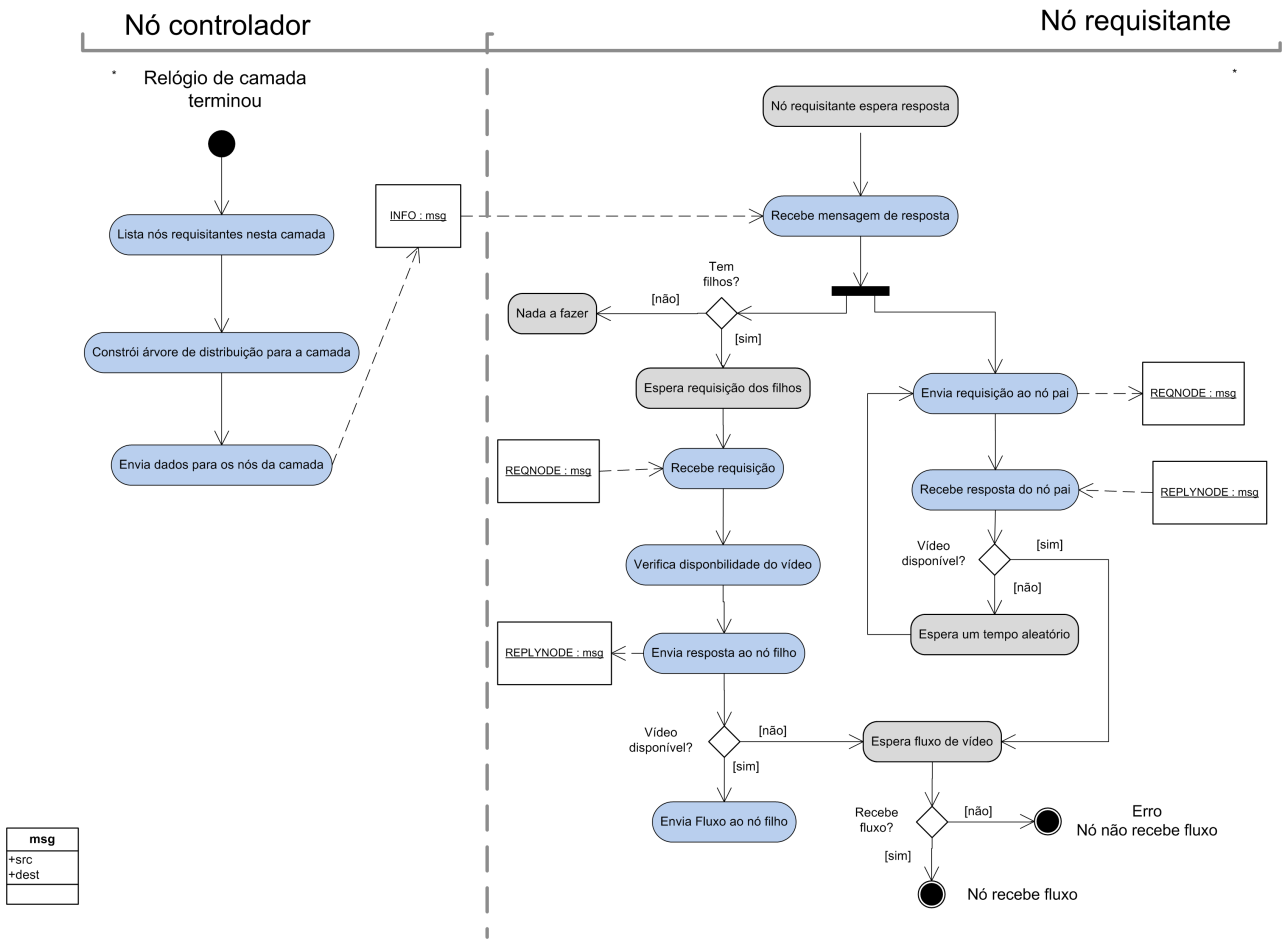


Figura 3.7: Fluxograma de formação de uma camada de distribuição.

Nó saindo na rede

Como ilustrado na Figura 3.9, Para um nó sair da rede de distribuição, este envia uma mensagem de saída (*LEAVE*) ao nó controlador. Ao receber esta mensagem, o nó controlador retira o nó da árvore de distribuição e constrói uma nova árvore a partir da localização do nó ausente. Ou seja, determina-se o subconjunto de nós que estão abaixo do nó ausente, e com este subconjunto é construída uma nova árvore de distribuição, como ilustrado na Figura 3.8. Após a criação desta nova árvore, o nó controlador envia uma mensagem de atualização (*UPDATE*), com as mesmas informações de *INFO*, aos nós que sofreram alguma atualização, seja esta um novo pai atribuído, ou novos filhos a serem suportados. Ao receberem esta atualização os nós primeiramente checam se o conjunto de filhos é o mesmo, e então:

- caso tenha sido removido algum filho, o fluxo multimídia que estava sendo enviado para esse é interrompido.
- caso tenha sido adicionado um novo filho, o nó espera a requisição vinda deste novo filho.

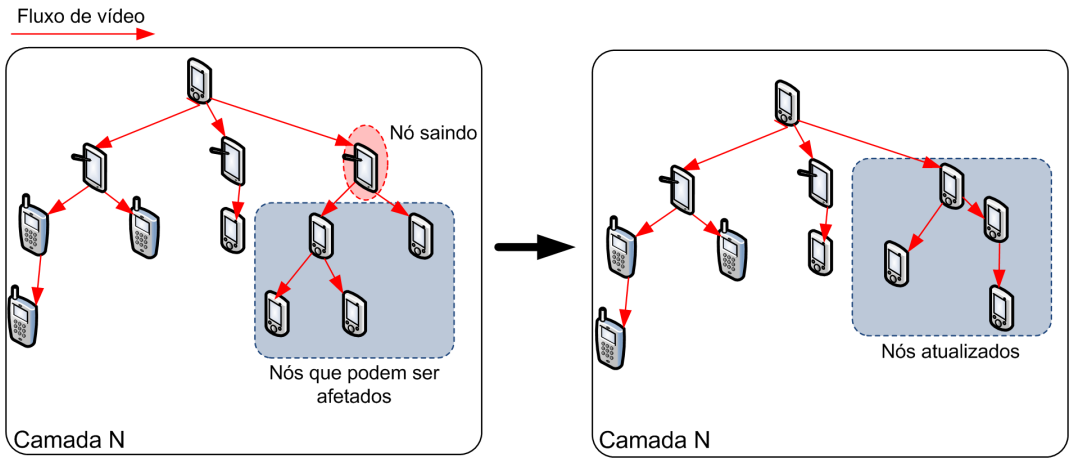


Figura 3.8: Nó saindo da rede.

Em seguida, o nó atualizado verifica qual o último segmento de vídeo recebido de seu antigo pai e, então, envia uma requisição ao novo pai informando o próximo segmento a ser recebido.

O nó que está saindo recebe uma mensagem de atualização informando que este pode sair da rede sem deixar inconsistência na mesma.

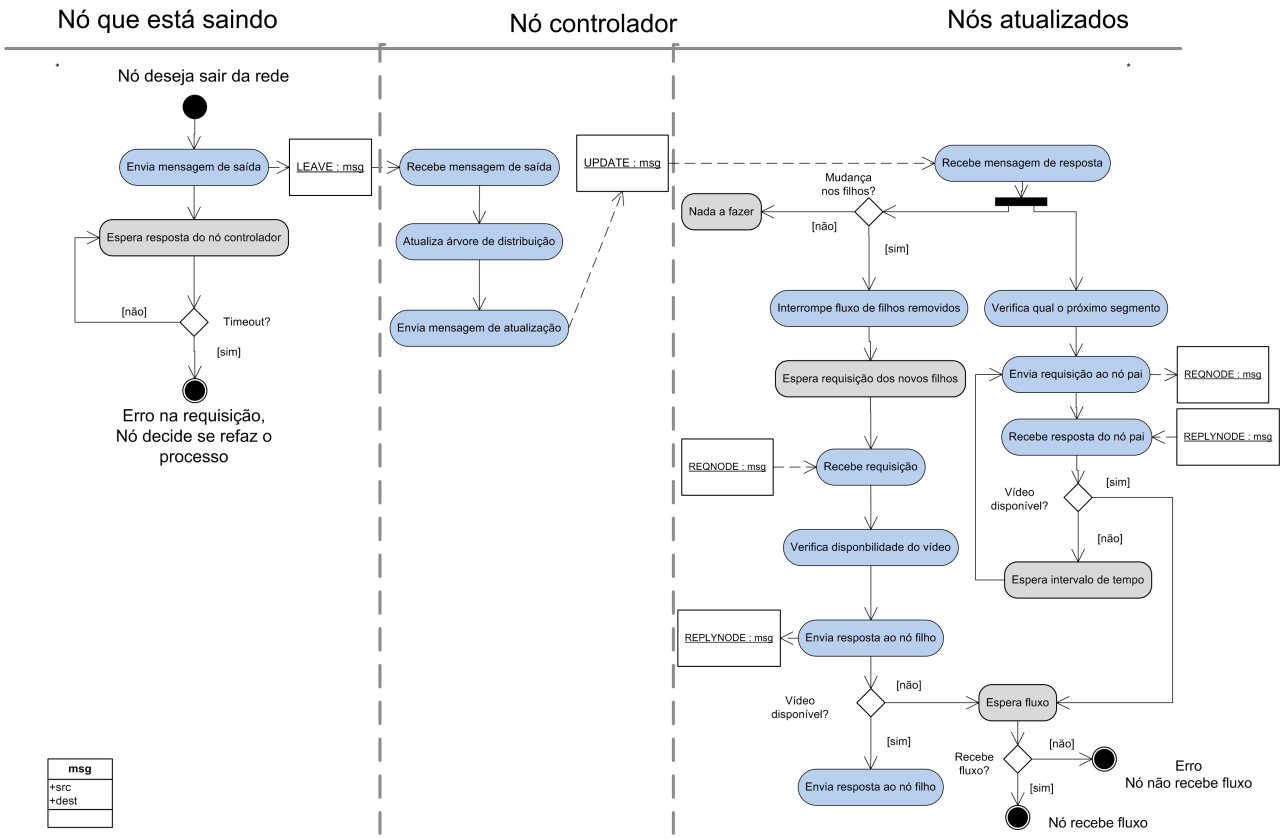


Figura 3.9: Fluxograma do nó saindo da rede.

Falha de um nó

Antes de tomar as devidas ações é necessário detectar que um determinado nó falhou, ou seja, que este tenha saído da rede de modo inesperado devido a algum tipo de falha, seja por causa de uma desconexão repentina ou uma falha de *software*, por exemplo. Após esta detecção de falha, o nó controlador executa o mesmo procedimento de quando um nó sai espontaneamente da rede, procedimento este, detalhado na seção anterior.

Portanto, é necessário determinar um procedimento de detecção de falha de nó. Dois procedimentos podem ser utilizados:

- o nó controlador monitora todos os nós participantes da rede de distribuição enviando mensagens de verificação (*PING*). Entretanto, este método gera uma sobrecarga na rede, principalmente no nó controlador caso o número de nós participantes seja alto, em torno dos milhares.

- os nós pais monitoram seus filhos, ou seja, cada nó que está compartilhando seu fluxo multimídia com outros nós, monitora estes nós filhos através de mensagens de verificação (*PING*). Ao constatar que um nó não esta mais presente na rede, este nó pai informa o nó controlador sobre esta falha. Para isto é enviada ao nó controlador uma mensagem de informação de nó (*NODEINF*) com o endereço do nó que falhou.

Atualização de informações de um nó

O protocolo aqui introduzido se comporta de maneira adaptativa. Com isto, cada nó periodicamente atualiza seus dados de contexto, e caso exista alguma modificação, estes dados são informados ao nó controlador. O período de atualização é determinado por cada nó. Nesta primeira versão do protocolo, serão atualizados apenas os dados relativos a quantidade de fluxos multimídia que um determinado nó pode compartilhar. Os dados relativos a posição relativa do nó na rede, mesmo que sejam alterados, não são utilizados para atualizar a rede de distribuição nesta primeira versão do protocolo. Isto se deve ao fato de que, por usar coordenadas de rede, nós móveis podem ter suas coordenadas alteradas, porém, se mantendo conectados ao mesmo roteador de rede por exemplo.

Para detectar uma mudança no número de fluxos multimídia que um nó é capaz de compartilhar, usam-se ferramentas que estimam a banda de transmissão disponível. Como estas ferramentas apenas estimam a banda disponível em um caminho fim-a-fim, o nó verifica a banda disponível para a transmissão do fluxo para cada um dos seus nós filhos. Caso a banda disponível de um destes caminhos esteja saturada, ou seja, próxima de zero, o nó atualiza o número de fluxos os quais ele é capaz de compartilhar. Como ilustrado na Figura 3.11, após esta atualização o nó envia uma mensagem de atualização (*NODEUPDATE*) ao nó controlador. O nó controlador, por sua vez, atualiza os dados desse nó em sua base de dados, e constrói uma nova árvore a partir daquele nó, como ilustrado na Figura 3.10.

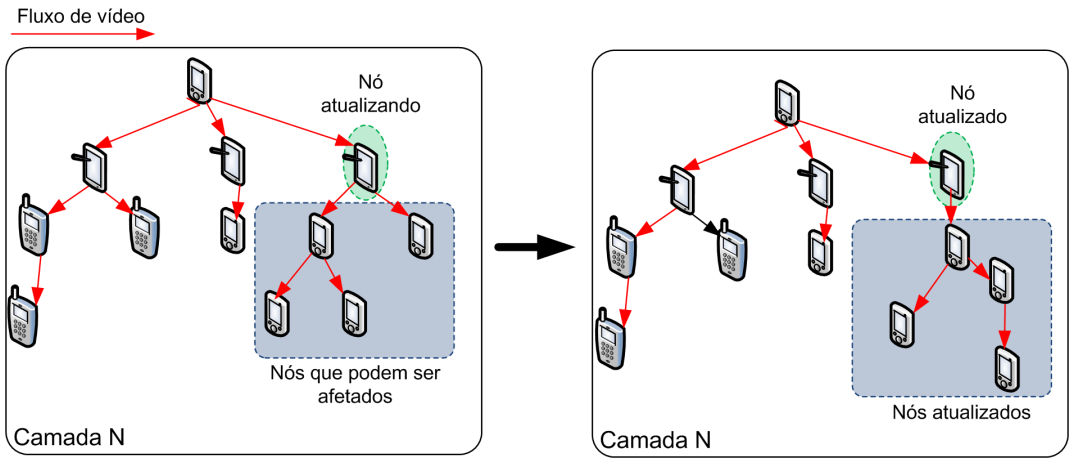


Figura 3.10: Atualizando árvore em relação ao um Nó.

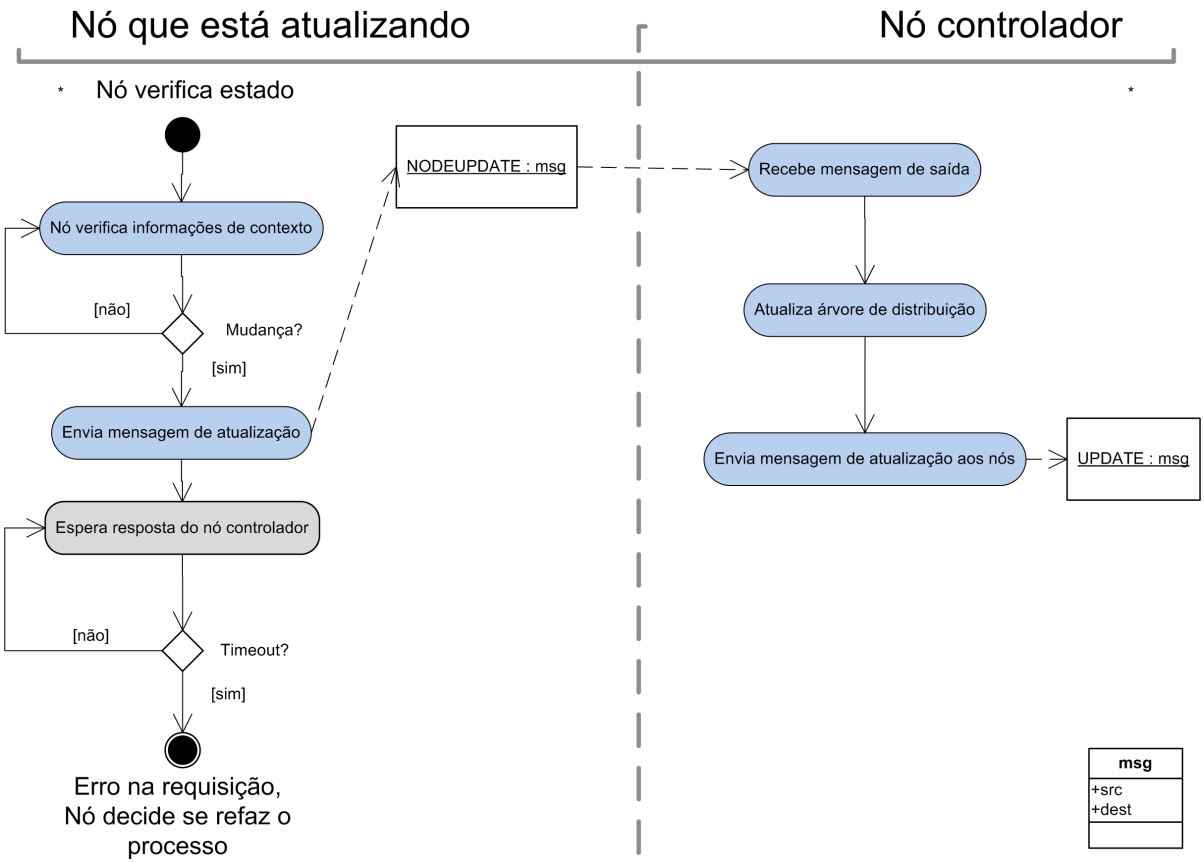


Figura 3.11: Fluxograma de atualização.

Após está atualização da árvore, todos os nós atualizados são informados através de uma mensagem *UPDATE* enviada pelo nó controlador. Ao receber esta mensagem, os nós se comportam como descrito anteriormente.

3.3.2 Algoritmos para Construção de Árvores de Distribuição

Após o fim do tempo definido para a criação de uma camada, o nó controlador organiza todos os nós requisitantes durante este intervalo de tempo em uma árvore de distribuição *multicast*, utilizando como parâmetros de entrada para cada nó: o número de filhos para o qual este nó pode suprir um fluxo de vídeo; e as coordenadas de rede de cada nó.

Durante o desenvolvimento deste trabalho foram propostos três algoritmos para a construção das árvores de distribuição [8] [7]:

- um algoritmo que apenas utiliza o número de nós filhos de cada nó para construir a árvore de distribuição (*Simple Bandwidth Aware algorithm - SBA*);
- um algoritmo que utiliza tanto o número de nós filhos quanto a posição relativa do nó na rede (*Bandwidth and Position Aware algorithm - BPA*). Este algoritmo constrói uma árvore com a menor distância média entre os nós requisitantes e o nó controlador. Este algoritmo, então, tem uma custo elevado e, portanto, requer mais capacidade de processamento do nó controlador, como será apresentado na próxima seção, a qual faz uma avaliação experimental desses algoritmos;
- um algoritmo que é uma variação do BPA (*Fast Bandwidth and Position Aware algorithm - FBPA*), entretanto, com menor custo. Apesar deste algoritmo calcular uma distância média maior entre os nós e o nó controlador, este requer menos capacidade de processamento do nó controlador.

A seguir são apresentados em detalhes cada um destes algoritmos. Em relação aos papéis dos nós participantes de cada camada, considere que o último nó visitado por cada algoritmo é designado como nó sorvedouro ($sink$), e que o nó fonte (src) de cada camada é o nó mais próximo do nó sorvedouro da camada superior. No caso do uso destes algoritmos na atualização da árvore quando um nó sai da rede, o nó fonte utilizado na execução do algoritmo será o pai do nó que saiu da rede. No caso da atualização das informações de um nó, o próprio nó é utilizado como nó fonte na execução do algoritmo.

No final desta seção é apresentada uma avaliação experimental dos protocolos.

Simple Bandwidth Aware Tree - SBA

Este algoritmo é baseado no algoritmo de Busca em Amplitude (*Breadth-First Search algorithm - BFS*) [29]. O princípio do BFS é que o processamento é feito por níveis, analisando-se primeiro os níveis mais próximos do vértice inicial, e deixando por último os mais distantes. Este algoritmo é utilizado para o cálculo do menor caminho entre dois vértices em um grafo onde todas as arestas (ligação entre dois vértices) têm a mesma distância.

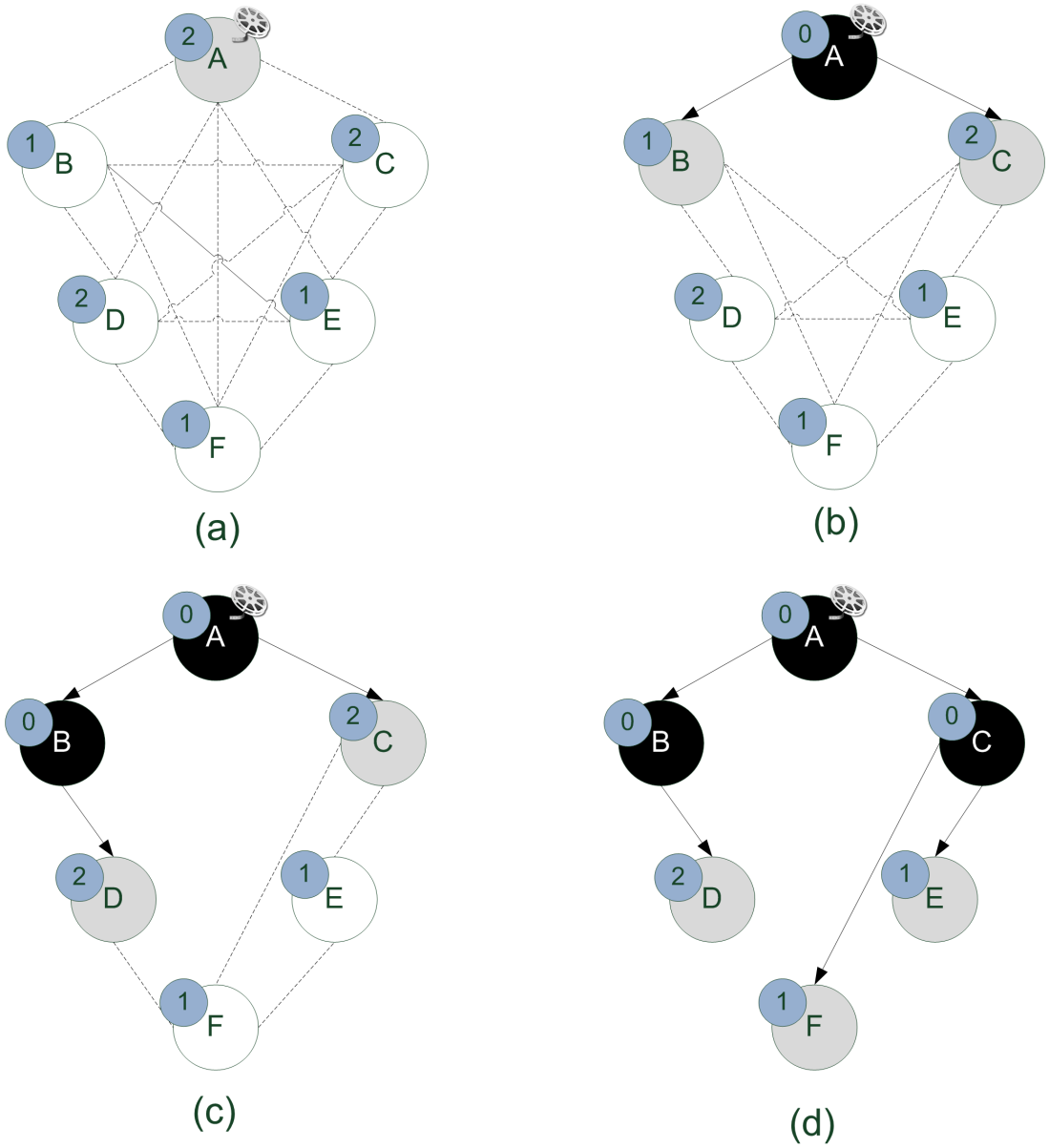


Figura 3.12: Ilustração da formação da árvore de distribuição com o SBA.

Para este algoritmo considera-se um grafo onde todos os vértices (nós) podem se comunicar entre si, ou seja, têm arestas entre si, como ilustrado na Figura 3.12a. Como exemplo, o procedimento de construção da árvore em SBA é ilustrado na Figura 3.12.

1. O algoritmo inicia verificando quantos nós filhos (dois) o nó fonte A suporta. Como a distância entre todos os nós é a mesma, SBA determina dois nós quaisquer (B e C) para serem filhos de A.

2. A seguir, o algoritmo seleciona um dos dois filhos de A, neste caso B, e verifica quantos filhos (um) o nó B pode suportar. SBA, então, seleciona entre os nós não visitados um nó filho para B.

Algoritmo 3.1 Algoritmo SBA

Entrada(s): $todosNos, noFonte$

1: **para cada nó** n **em** $todosNos$ **faça**
2: $cor[n] := BRANCA$
3: $dsrc[n] := infinito$
4: **fim para**
5: $cor[noFonte] := CINZA$
6: $dsrc[noFonte] := 0$
7: $insere(listaCinza, noFonte)$
8: **enqto** $tamanho(listaCinza)! = 0$ **faça**
9: $u := retiraPrimeiro(listaCinza)$
10: **para** $i = 1$ **até** $N[u]$ **faça**
11: $n := retiraPrimeiro(listaBranca)$
12: $pai[n] := u$
13: $cor[n] := CINZA$
14: $dsrc[n] := dsrc[u] + 1$
15: $insere(listaCinza, n)$
16: **fim para**
17: $cor[u] := PRETA$
18: $insere(listaPreta, u)$
19: **fim enqto**

3. A seguir o procedimento é repetido para o outro nó no mesmo nível de B, o nó C.

4. Depois de determinados os filhos do nó C, os filhos dos nós no nível seguinte são determinados, e assim por diante.

Para um controle no processamento dos nós durante a execução do algoritmo, os nós são caracterizados a partir de três cores:

- nós de cor branca: são os nós que não foram visitados pelo algoritmo;
- nós de cor cinza: são os nós que já foram visitados pelo algoritmo (são filhos de outro nó), porém não têm filhos designados pelo algoritmo;
- nós de cor preta: são os nós que já foram visitados pelo algoritmo e têm filhos designados. Um nó folha (sem filhos), também é caracterizado pela cor preta.

O algoritmo utiliza uma estrutura de dados para armazenar cada grupo de nós. São utilizadas uma lista branca, uma lista cinza e uma lista preta.

Como apresentado no Algoritmo 3.1, SBA é iniciado inserindo-se o nó fonte na lista cinza (**linhas 5 a 7**), em seguida define-se a distância entre os outros nós e o nó fonte, $dsrc$, como infinita e caracteriza-se estes nós como brancos (**linhas 1 a 4**). Após este processo de inicialização, escolhe-se um nó u da lista cinza (o primeiro nó) (**linha 9**). Como o nó u compartilha banda suficiente para suprir outros N nós como filhos, são escolhidos e removidos N nós da lista branca (**linhas 10 e 11**). A estes nós é atribuído o nó u como nó pai (**linha 12**), e em seguida estes N nós são caracterizados como nós cinzas e inseridos na lista cinza (**linhas 13 e 15**). O nó u é então carcaterizado como nó preto e inserido na lista preta (**linhas 17 e 18**). Esta operação é repetida até que a lista cinza esteja vazia, ou seja, até que todos os nós sejam visitados e tenham nós filhos atribuídos caso seja possível.

Bandwidth and Position Aware Tree - BPA

Este algoritmo é baseado no algoritmo de Dijkstra [9], que é utilizado para a computação do menor caminho entre dois vértices em um grafo, dado que as arestas possam ter pesos diferentes. Estes pesos podem ser relativos ao comprimento de cada aresta, por exemplo.

Como descrito no algoritmo anterior, em BPA considera-se um grafo, como ilustrado na Figura 3.12a, onde todos os vértices (nós) podem se comunicar entre si, ou seja, têm arestas entre si. Entretanto, além de considerar que cada nó pode suportar um número N de filhos, considera-se o peso de cada aresta entre os nós. O procedimento de construção da árvore de distribuição do algoritmo BPA é ilustrado na Figura 3.13, e descrito a seguir:

1. Inicialmente computa-se o peso do caminho entre cada nó e o nó fonte A;
2. Em seguida escolhe-se o nó com menor peso (mais próximo), neste caso, o nó B. A seguir estes dois (A e B) nós são marcados como visitados (nós cinzas);
3. A seguir calculam-se os pesos (distância) entre os demais nós (nós brancos) e o nó fonte através dos nós visitados (nós cinzas). Por exemplo, o nó C tem uma distância de peso 3 diretamente até o nó A, e uma distância de peso 2 através do nó B (peso 1 entre C e B mais peso 1 entre B e A). Assim, computa-se todos esses pesos para todos os nós não visitados (nós C, D, E, e F), semelhante ao processo de relaxamento do algoritmo de Dijkstra. Após isto, escolhe-se o nó com menor peso, neste exemplo o nó C;
4. O processo é repetido até que todos os nós sejam visitados e tenham nós pais atribuídos.

O algoritmo BPA, portanto, sempre computa a menor distância entre um determinado nó X e o nó fonte.

Como o algoritmo anterior, o BPA caracteriza os nós em três cores, branco, cinza e preto. Além das três listas apresentadas anteriormente, é também utilizada uma lista que guarda os nós cinza que têm banda de transmissão suficiente para suportar outros filhos. Esta lista é chamada de quase-preta (*almostBlackList*).

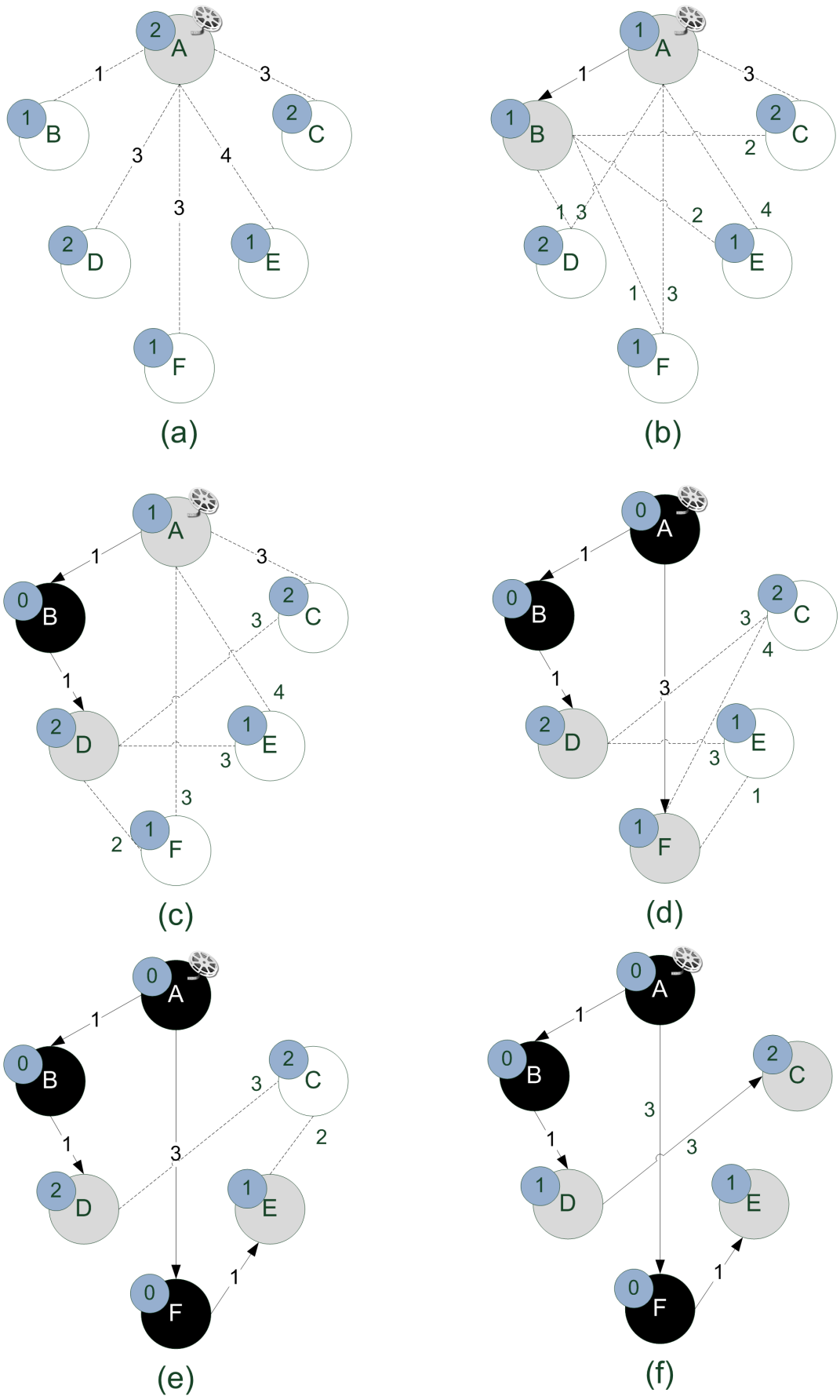


Figura 3.13: Ilustração da formação da árvore de distribuição com o BPA

O BPA, apresentado no Algoritmo 3.3, inicia-se inserindo o nó fonte na lista cinza, definindo a distância entre cada nó e o nó fonte como infinita e inserindo estes nós na lista branca (**linhas 1 a 7**). Após este processo de inicialização é escolhido (e removido) da lista cinza o nó u mais próximo do nó fonte (**linha 9**). Na primeira interação o próprio nó fonte é escolhido como o mais próximo. Este nó escolhido é inserido na lista quase-preta (**linha 10**). Então, para cada nó na lista cinza e na lista branca (estas duas listas são unidas em uma lista branca-e-cinza) é calculada a distância até o nó fonte através de cada elemento da lista quase-preta (**linhas 13 a 26**). Este é o processo de relaxamento do algoritmo Dijkstra, descrito no Algoritmo 3.2. Todos os nós visitados, então, são incluídos na lista cinza e removidos da lista branca. Quando um nó na lista quase-preta utiliza todas as N conexões suportadas (não suporta mais filhos), este é removido da lista quase-preta e todos os nós na lista branca-e-cinza atualizam sua informação de distância até a fonte (processo de relaxamento) (**linhas 13 a 23**). Toda vez que é atribuído um filho a um nó p é verificado se este nó ainda suporta mais filhos (**linha 13**). Esta interação é repetida até que a lista cinza seja esvaziada.

Algoritmo 3.2 Algoritmo de Relaxamento

```
 1: RELAX(u, v)
 2: se distanciaEntre(u, v) + dsrc[u] menor que dsrc[v] então
 3:     dsrc[v] := distanciaEntre(u, v) + dsrc[u]
 4:     pai[v] = u
 5:     se cor[v] é BRANCA então
 6:         cor[v] := CINZA
 7:         insere(listaCinza, v)
 8:     senão
 9:         atualiza(listaCinza, v)
10:     fim se
11: fim se
```

Algoritmo 3.3 Algoritmo BPA

Entrada(s): $todosNos, noFonte$

para cada nó n **em** $todosNos$ **faça**
  $cor[n] := BRANCA$
  $dsrc[n] := infinito$
fim para
$cor[noFonte] := CINZA$
$dsrc[noFonte] := 0$
$insere(listaCinza, noFonte)$
enqto $tamanho(listaCinza)! = 0$ **faça**
  $u := retira - no - com - menor - distancia(listaCinza)$
  $insere(listaQuasePreta, u)$
  $p := pai[u]$
  $N[p] = N[p] - 1$
  se $N[p]$ **iqual** 0 **então**
    $remove(listaQuasePreta, p)$
    $cor[p] := PRETA$
    para cada nó v **em** $listaBrancaECinza$ **faça**
      se $pai[v]$ **é** p **então**
        para cada nó v **em** $listaQuasePreta$ **faça**
          $RELAX(u, v)$
        fim para
      fim se
    fim para
  senão
    para cada nó v **em** $listaBrancaECinza$ **faça**
      $RELAX(b, v)$
    fim para
  fim se
fim enqto

Fast Bandwidth and Position Aware Tree - FBPA

Apesar do algoritmo anterior computar a menor distância média entre os nós e o nó fonte, este executa muitas interações para o cálculo da distância, resultando em uma maior custo. Resultados experimentais serão apresentados na próxima seção.

Portanto, propõe-se outro algoritmo, o FBPA. Este algoritmo é baseado no algoritmo SBA, entretanto, utiliza o processo de relaxamento utilizado no algoritmo anterior.

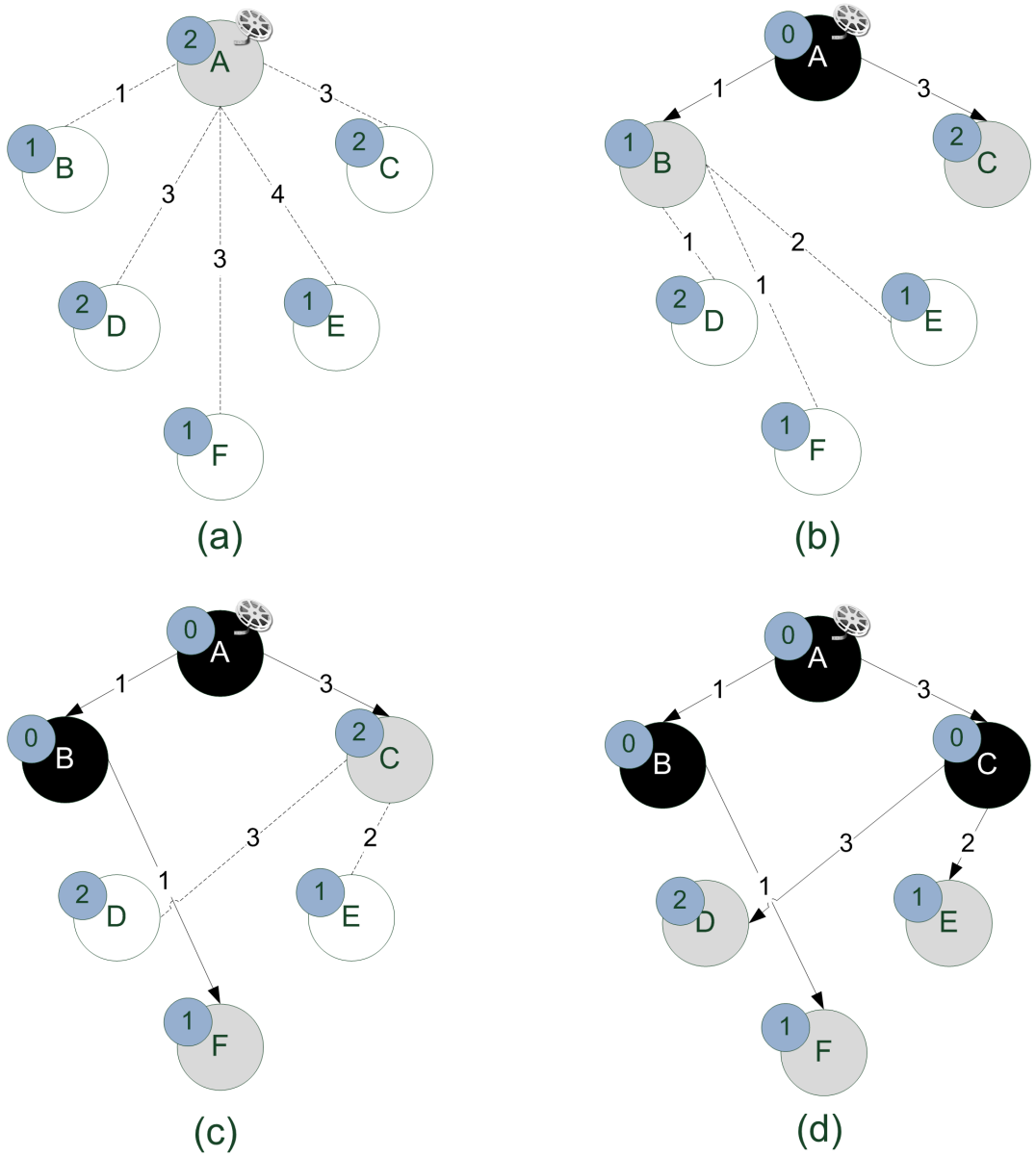


Figura 3.14: Ilustração da formação da árvore de distribuição com o FBPA.

Como no algoritmo anterior, considera-se um grafo onde todos os vértices (nós) podem se comunicar entre si, ou seja, como ilustrado na Figura 3.12a. Como para SBA, utiliza-se uma abordagem por camadas. O procedimento de construção da árvore de distribuição é ilustrado na Figura 3.14 e descrito a seguir.

1. Inicialmente computa-se o peso do caminho entre cada nó e o nó fonte A, como ilustrado na Figura 3.14a;

2. Em seguida são escolhidos os nós com menor peso (mais próximos), neste caso, o nó B e o nó C. A seguir estes dois nós (B e C) são marcados como visitados (nós cinzas) e o nó A é marcado como nó preto. Este procedimento define uma camada, como ilustrado na Figura 3.14b;

Algoritmo 3.4 Algoritmo FBPA

Entrada(s): $todosNos, noFonte$

1: **para cada nó** n **em** $todosNos$ **faça**
2: $cor[n] := BRANCA$
3: $dsrc[n] := infinito$
4: **fim para**
5: $cor[noFonte] := CINZA$
6: $dsrc[noFonte] := 0$
7: $insere(listaCinza, noFonte)$
8: **enqto** $tamanho(listaCinza)! = 0$ **faça**
9: $u := retiraPrimeiro(listaCinza)$
10: **para cada nó** v **em** $listaBranca$ **faça**
11: $dsrc[v] := distanciaEntre(u, v) + dsrc[u]$
12: **fim para**
13: **para** $i = 1$ **até** $N[u]$ **faça**
14: $n := retira - no - com - menor - distancia(listaBranca)$
15: $cor[n] := CINZA$
16: $pai[n] := u$
17: $insereNoFinal(listaCinza, n)$
18: **fim para**
19: $cor[u] := PRETA$
20: $insere(listaPreta, u)$
21: **fim enqto**

3. A seguir são computados os pesos (distância) entre os demais nós (nós brancos) para o nó cinza mais próximo da fonte, neste exemplo, o nó B. São escolhidos os nós mais próximos de B, neste exemplo o nó F. Para este nó F é atribuído como pai o nó B. O nó B, portanto, é marcado como nó preto, como ilustrado na Figura 3.14c;

4. Em seguida este procedimento é repetido para os outros nós na mesma camada de B, neste caso o nó C, como ilustrado na Figura 3.14d;

5. O processo é repetido até que todos os nós sejam visitados e tenham nós pais atribuídos.

Como nos algoritmos anteriores, para o FBPA, apresentado no Algoritmo 3.4, os nós são caracterizados por três cores, branco, cinza e preto e são organizados em suas respectivas listas. Este algoritmo inicia inserindo o nó fonte na lista cinza, definindo a distância entre cada nó e o nó fonte como infinita e inserindo estes nós na lista branca (**linhas 1 a 7**). Após este processo de inicialização é escolhido (e removido) da lista cinza o nó u mais próximo do nó fonte (**linha 9**). Na primeira interação o próprio

nó fonte é escolhido como o mais próximo. Este nó u compartilha banda de transmissão suficiente para suprir outros N nós como filhos. Portanto, para todos os nós na lista branca é calculado a distância até este nó u (**linhas 10 a 12**). Então, os N nós mais próximos de u são escolhidos e inseridos na lista cinza, portanto, removidos da lista branca (**linhas 13 a 18**). O u é marcado como nó preto e removido da lista cinza (**linhas 19 e 20**). Em seguida, é escolhido o primeiro nó u na lista cinza, e a operação é repetida até que a lista cinza esteja vazia.

3.3.3 Avaliação Experimental dos Algoritmos

Como descrito anteriormente, os algoritmos introduzidos neste trabalho apresentam custos diferentes. Portanto, nesta seção é apresentada uma avaliação de desempenho entre os algoritmos propostos, fazendo uma análise do tempo necessário para o cálculo da árvore e da distância média entre os nós e o nó fonte.

Os nós foram distribuídos em um plano Cartesiano com coordenadas entre (0,0) e (1000,1000). Adicionalmente, os nós podem compartilhar o fluxo multimídia com no máximo outros três nós. A distribuição dos nós no plano Cartesiano, e a definição do número de fluxos a compartilhar foram definidas de maneira aleatória.

Foram executados cálculos de árvores de distribuição com topologias com 10, 100, 1000, 2500, 5000, 7500 e 10000 nós. O gráfico apresentado na Figura 3.15 ilustra a distância média entre os nós e o nó fonte para cada algoritmo. O gráfico apresentado na Figura 3.16 ilustra o tempo relativo necessário para o cálculo da árvore para cada algoritmo. Este tempo é relativo ao tempo necessário para o cálculo da árvore do algoritmo SBA, o qual é o mais rápido.

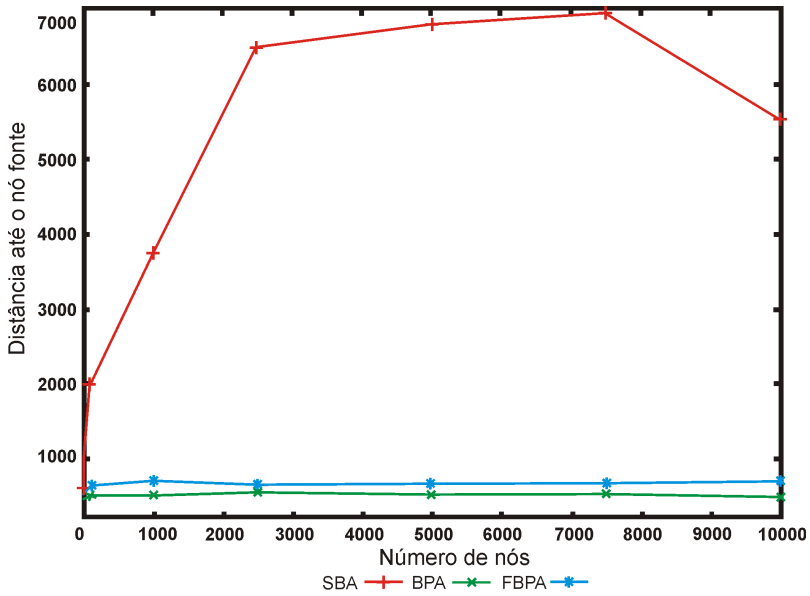


Figura 3.15: Gráfico da distância média ao nó fonte.

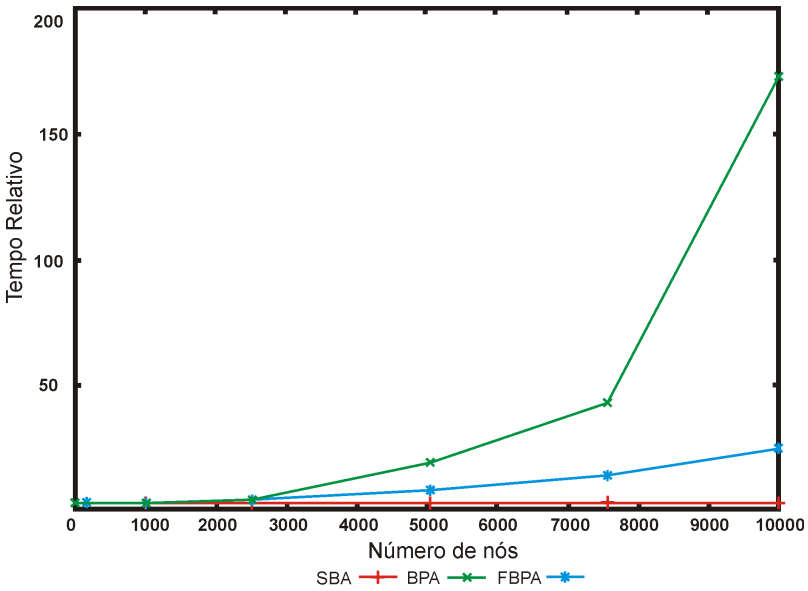


Figura 3.16: Gráfico do tempo relativo para o cálculo da árvore.

Com estes resultados pode-se concluir que:

- O algoritmo SBA, apesar de executar o cálculo em um menor intervalo de tempo, apresenta uma distância média maior. Isto se deve ao fato deste algoritmo não levar em consideração a distância entre os nós em seu cálculo;
- O algoritmo BPA computa sempre a menor distância média entre os nós e o nó fonte, entretanto, para um número maior de nós, o tempo relativo para o cálculo cresce de maneira mais rápida do que os outros algoritmos;
- O algoritmo FBPA computa uma distância média um pouco maior do que o algoritmo BPA, entretanto, este computa a árvore em um tempo relativo menor e com crescimento mais lento do que o BPA.

Com estes resultados, pode-se sugerir um sistema de construção de árvores de distribuição o qual utiliza como entrada o número de nós a ser utilizado, e então, decide qual o algoritmo a ser utilizado. Portanto, a depender do cenário de aplicação do sistema, é necessário escolher qual algoritmo utilizar e quais os limiares de decisão na escolha do algoritmo. Esta decisão fica a critério de cada implementação, por exemplo, em um sistema que será utilizado entre comunidades de dispositivos que não ultrapassem uma centena de usuários, pode-se utilizar o algoritmo BPA sempre. Entretanto, em um sistema com um número de usuários que pode variar entre uma dezena e milhares de usuários, pode-se utilizar um sistema de decisão que, por exemplo, define:

- utilizar o algoritmo BPA até 100 usuários;
- utilizar o algoritmo FBPA entre 100 e 500 usuários;

- utilizar o algoritmo SBA com mais de 500 usuários.

Estes valores do número de usuários devem ser calculados dependendo do tipo de dispositivo onde o algoritmo será utilizado, portanto, dependendo de onde o sistema será implantado.

3.4 Sumário

Neste capítulo foi apresentado em detalhes o protocolo introduzido neste trabalho, chamado de Granola. Foram apresentados os procedimentos de entrada, saída e atualização de um nó na rede. Também foram apresentados os algoritmos utilizados para a computação das árvores de distribuição de conteúdo. Ao final do capítulo, foi apresentada uma avaliação experimental dos algoritmos apresentados.

Capítulo 4

Avaliação do Protocolo

Neste capítulo é feita uma avaliação do protocolo apresentado neste trabalho. Apresenta-se o ambiente de simulação utilizado, e os módulos desenvolvidos para a simulação de um sistema de Vídeo sob-Demanda que utilize o protocolo Granola. Em seguida são apresentadas as métricas utilizadas para a avaliação, e o método estatístico utilizado para obtenção dos resultados. Em seguida, são apresentados os resultados das simulações de três possíveis cenários. Ao final do capítulo, apresenta-se os aspectos e ferramentas utilizadas para a implementação de um protótipo de avaliação.

4.1 Simulação

Para a simulação de um sistema de Vídeo sob-Demanda que utiliza o protocolo apresentado neste trabalho, foi utilizado o ambiente de simulação OMNET++[1]. O OMNET++ é uma ambiente de simulação de redes orientado a objetos com uma estrutura modular cujo alvo inicial era a simulação de redes de comunicação, entretanto, devido a sua arquitetura genérica e flexível, este também é utilizado em outras áreas como simulação de arquiteturas de hardware, e redes em fila.

A estrutura modular do OMNET++ faz uso de dois tipos de módulos: módulos simples e módulos compostos, como ilustrado na Figura 4.1. Os módulos simples não podem ser divididos, e suas tarefas mais freqüentes são o envio e recebimento de mensagens. Mensagens, por sua vez, podem ser enviadas através de *gates* ou diretamente para outro módulo. Os *gates* são as interfaces de entrada e saída dos módulos, e podem ser conectados entre si através de conexões. A estas conexões podem ser atribuídas propriedades como atraso de propagação, taxa de transmissão de dados e taxa de erro de bits. Por fim, os módulos compostos são formados por outros módulos, sejam estes módulos simples ou compostos. A composição destes módulos compostos pode ser feita através da linguagem NED disponibilizada para o ambiente OMNET++. Os módulos simples, entretanto, devem ser implementados na linguagem de programação nativa do OMNET++, a linguagem C++.

[1] http://www.omnetpp.org

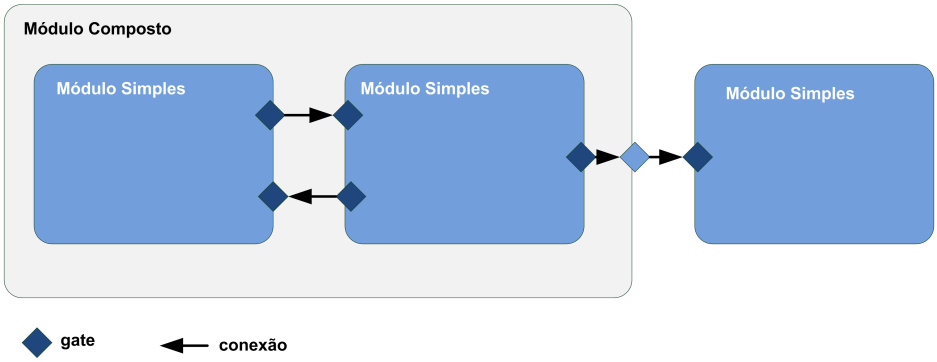


Figura 4.1: Estrutura modular do OMNET++.

Mais especificamente, neste trabalho foi utilizado o ambiente INET do OMNET++ o qual oferece módulos para a simulação de protocolos como IPv4, IPv6, TCP, e UDP, além de vários modelos para o desenvolvimento de aplicações para o próprio ambiente, módulos que simulam a camada de enlace para protocolos como Ethernet, entre outras características desejáveis.

4.1.1 Desenvolvimento dos Módulos de Simulação

No desenvolvimento dos módulos utilizados para a simulação neste trabalho foi utilizado o módulo *StandardHost* disponível no INET, o qual serve como base para a simulação de nós ou dispositivos que utilizem o protocolo IP (*Internet Protocol*). Este módulo composto é formado por módulos que implementam protocolos como TCP, UDP e Ethernet. Especificamente, neste trabalho foram utilizados dois módulos base: o *TCPApp* que serve como base para o desenvolvimento de aplicações que utilizam o protocolo TCP; e o *UDPApp* que serve como base para o desenvolvimento de aplicações que utilizam o protocolo UDP.

Portanto, foram desenvolvidos dois tipos de nós para a simulação, os quais derivam diretamente do módulo composto *StandardHost*: um módulo servidor, *GranolaServer*, o qual funciona como nó provedor de conteúdo e nó controlador; e um módulo cliente, *GranolaClient*, o qual funciona como nó requisitante de conteúdo.

Para o módulo servidor foram desenvolvidos dois módulos simples que utilizam os módulos bases fornecidos pelo *StandardHost*: um módulo para uma aplicação TCP, *GranolaTCPServer*, que espera requisições e controla os nós clientes através da troca de mensagens pré-definidas pela implementação do protocolo na simulação; e um módulo para uma aplicação UDP, *GranolaUDPServer*, que envia o fluxo de vídeo aos nós requisitantes. Este fluxo de vídeo é enviado através de mensagens numeradas e com informação de tempo (*timestamp*) de modo a se comportar de maneira semelhante ao protocolo RTP (*Real Time Protocol*). Além disto, a aplicação *GranolaTCPServer* faz uso de uma biblioteca, *GranolaTreeBuilderLib*, que contém os algoritmos de controle e construção das árvores de distribuição do protocolo definido neste trabalho.

Para o módulo cliente foram também desenvolvidos dois módulos simples: um para uma aplica-

ção TCP, *GranolaTCPClient*, que envia a requisição ao nó servidor, verificar a localização relativa do nó na rede, verificar a capacidade de transmissão do nó, e implementa toda a comunicação de controle e negociação com o nó servidor e outros nós clientes; e um módulo para uma aplicação UDP, *GranolaUDPClient*, responsável por receber o fluxo de vídeo vindo de outros nós e transmiti-lo para os nós filhos. Além disto, este módulo coleta as informações necessárias para a obtenção dos resultados a serem avaliados. Na Figura 4.2 ilustra-se como estes módulos interagem entre si.

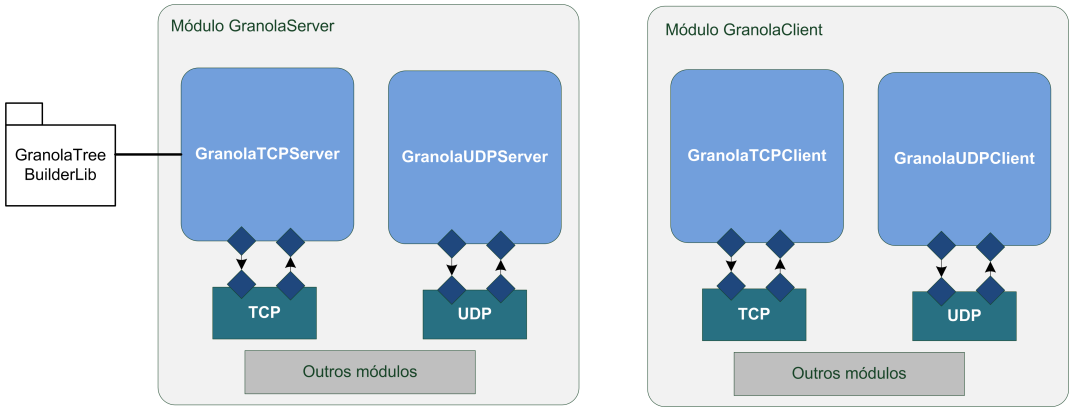


Figura 4.2: Esquema para os módulos implementados para a simulação.

Por fim, foram criadas topologias de redes utilizando uma modificação do software PSGEN[2], as quais são formadas por roteadores disponibilizados pelo ambiente INET e os módulos desenvolvidos para a simulação do sistema de Vídeo sob-Demanda. Detalhes sobre estas topologias são apresentados na seção 4.1.4.

4.1.2 Métricas Avaliadas

No processo de simulação foram avaliadas duas métricas relacionadas a transmissão de fluxos multimídia: o *jitter* que é um valor relativo a variação do atraso no recebimento de pacotes; e o percentual relativo ao número de pacotes recebidos por cada nó. Além destas métricas, também avaliou-se uma métrica importante para a análise do protocolo introduzido neste trabalho: o atraso médio para o início do recebimento do fluxo multimídia.

O *jitter* é uma medida que determina o atraso médio no recebimento dos pacotes em uma transmissão multimídia. O *jitter*, por sua vez, é calculado no recebimento de cada pacote, e é definido como [18]:

$$J_i = J_{i-1} + \frac{(|D(i-1,i)| - J_{i-1})}{16} \tag{4.1}$$

onde: J_i é o *jitter*, D é a diferença do atraso entre o recebimento do pacote anterior e o pacote atual, e i é p número do pacote recebido.

[2]http://www1.inf.tu-dresden.de/ rf913578/psgen.html

Neste trabalho avalia-se o *jitter* médio para cada cenário, portanto, calcula-se o *jitter* médio para cada nó, o qual é a média do *jitter* para cada pacote recebido:

$$J_{no} = \frac{\sum_{i=1}^{n} J_i}{n} \tag{4.2}$$

onde: n é o número de pacotes recebidos.

Em seguida calcula-se o *jitter* médio para a simulação, o qual é a média do *jitter* para cada nó.

O atraso médio para o início do recebimento do fluxo multimídia é calculado como se segue. Cada nó calcula o tempo para o recebimento do fluxo, o qual é o tempo decorrido depois da requisição do fluxo multimídia até o recebimento do mesmo, t_{start}. O atraso médio para cada simulação é a média do t_{start} de todos os nós da topologia.

$$T_{startmedio} = \frac{\sum_{i=1}^{N} t_{start}}{N} \tag{4.3}$$

onde: N é o número nós na simulação e $T_{startmedio}$ é o atraso médio da simulação.

Em todos os resultados apresentados nas próximas seções, obteve-se uma média entre os resultados obtidos em N_s simulações. O número N_s de simulações foi determinado através de um procedimento estatístico apresentado na seção que segue.

4.1.3 Método Estatístico para Obtenção dos Resultados

No processo de simulação realizado no contexto deste trabalho foram calculadas médias dos resultados obtidos em séries de experimentos para cada métrica. Entretanto, para se obter uma estimativa exata dos resultados desejados seria necessário realizar um número infinito de simulações, o que é inviável, logicamente. Portanto, neste trabalho foi utilizado um método para obter um intervalo de confiança de valores para um determinado coeficiente de confiança escolhido. Este intervalo é calculado utilizando os resultados obtidos através de um número limitado de simulações.

Mais especificamente, neste trabalho foram obtidos resultados de modo a obter um intervalo de confiança para um coeficiente de confiança de 90%. Ou seja, foi obtido um intervalo de valores de modo que a média real daquela métrica tenha 90% de chance de estar contida nesse intervalo. Vale ressaltar que isto não significa dizer que 90% daz vezes que este resultado for obtido, este esteja contido naquele intervalo.

Intervalo de confiança

Antes de determinar o intervalo de confiança para os resultados obtidos, é preciso caracterizar estes resultados em uma distribuição de variável aleatória. Através o Teorema do Limite Central [16], se uma variável aleatória X pode ser representada pela soma de quaisquer outras n variáveis aleatórias independentes, então esta soma, para n suficientemente grande, terá uma distribuição aproximadamente *Normal* [16]. Portanto, como neste trabalho os resultados obtidos são uma média das variáveis

obtidas em cada simulação, por exemplo o *jitter* médio obtido em uma simulação é a média dos *jitters* médios obtidos para cada nó, pode-se considerar que estes resultados são variáveis aleatórias com uma distribuição *Normal*:

$$N(\mu, \sigma^2). \tag{4.4}$$

onde: μ representa a expectância (ou média), e σ^2 a variância. O valor σ também pode ser interpretado como o desvio padrão.

Como se pode considerar que a variável X resultante de uma simulação tem uma distribuição *Normal*, sua média amostral $\overline{X}$ é também uma variável aleatória com distribuição *Normal*:

$$N(\mu, \frac{\sigma^2}{n}) \tag{4.5}$$

De modo a utilizar a função tabelada [31] que caracteriza uma função *Normal* reduzida $N(0, 1)$ com média $\mu = 0$ e desvio padrão $\sigma = 1$, para o cálculo das probabilidades associadas a $\overline{X}$, é definida a variável:

$$Z = \frac{(\overline{X} - \mu)\sqrt{n}}{\sigma} \tag{4.6}$$

onde: Z tem uma distribuição $N(0, 1)$.

Deste modo podemos determinar que:

$$P(\overline{X} - \frac{z\sigma}{\sqrt{n}} \leq \mu \leq \overline{X} + \frac{z\sigma}{\sqrt{n}}) = 2\Phi(z) - 1 \tag{4.7}$$

onde: $\Phi(z)$ é o valor da função de distribuição reduzida para z.

Esta expressão define que a probabilidade que a média μ esteja contida no intervalo $(\overline{X} - \frac{z\sigma}{\sqrt{n}}, \overline{X} + \frac{z\sigma}{\sqrt{n}})$ é igual $2\Phi(z) - 1$. Este intervalo, então, é conhecido como o intervalo de confiança do parâmetro μ [31].

Para determinar este intervalo é necessário determinar o valor de z. Este valor, portanto, é determinado através da definição do coeficiente de confiança desejado. Por exemplo, para um coeficiente de confiança de 90% ou seja $(1 - a) = 0,9$ faz-se:

$$2\Phi(z) - 1 = 0,9 \rightarrow \Phi(z) = 1,9/2 \rightarrow \Phi(z) = 0,95 \tag{4.8}$$

Então, faz-se uso da tabela de valores para a função $N(0, 1)$, de modo a obter um $z = 1,65$. Portanto, como $\Phi(z) = 1 - a/2$, denotando $z = K_{1-a/2}$, tem-se que o intervalo de confiança para um coeficiente de confiança $(1 - a)$ é:

$$(\overline{X} - \frac{K_{1-a/2}\sigma}{\sqrt{n}}, \overline{X} + \frac{K_{1-a/2}\sigma}{\sqrt{n}}) \tag{4.9}$$

Utilizando a distribuição de t de Student

O intervalo de confiança definido anteriormente depende inteiramente do fato da variância σ^2, ou desvio padrão σ, serem conhecidos. Entretanto, nas simulações realizadas neste trabalho não são conhecidos esses valores a priori. Uma maneira de estimar σ^2 de maneira não-tendenciosa é:

$$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2 \tag{4.10}$$

Deste modo, a variável Z determinada anteriormente pode ser agora representada pela variável t:

$$t = \frac{(\overline{X} - \mu)\sqrt{n}}{\hat{\sigma}} \tag{4.11}$$

Portanto, a distribuição de probabilidade de t difere da distribuição de Z, pois, ambos, numerador e denominador, são variáveis aleatórias. A função de distribuição de probabilidade de t é dada por:

$$h_k(t) = \frac{\Gamma[(k+1)/2]}{\Gamma(k/2)\sqrt{(\pi k)}}(1 + \frac{t^2}{k})^{-(k+1)/2}, -\infty < t < \infty \tag{4.12}$$

onde: Γ = função gama.

Esta distribuição é conhecida como distribuição de t de *Student* com k graus de liberdade.

Portanto, de maneira semelhante ao cálculo com a função *Normal*, obtém-se que o intervalo de confiança para μ, com um coeficiente de confiança $(1 - a)$ é:

$$(\overline{X} - n^{(-1/2)}\hat{\sigma}t_{n-1,1-a/2}, \overline{X} + n^{(-1/2)}\hat{\sigma}t_{n-1,1-a/2}) \tag{4.13}$$

Este intervalo tem a mesma estrutura do anteriormente definido, com a diferença que o valor de σ foi substituído por sua estimativa $\hat{\sigma}$ e que a constante $k_{1-a/2}$ foi substituída pela constante $t_{n-1,1-a/2}$ que é obtida através das tábuas da distribuição t. Neste caso, n é o número de repetições de um determinado processo para se obter o grau de liberdade $k = n - 1$.

Com uso da distribuição t de *Student* pode-se determinar o intervalo de confiança com um número reduzido de amostras [16], pois o cálculo do intervalo apenas depende dos valores das amostras, do número de amostras obtidas, e de uma constante tabelada a qual também depende do número de amostras obtidas.

Procedimento de Obtenção dos Resultados

Obter um intervalo de confiança (l_1, l_2) para um coeficiente de confiança $(1 - a)$ não ajuda na determinação de um número satisfatório de resultados a serem obtidos. Por exemplo, para um número de experimentos $n = 4$ pode-se obter um intervalo de confiança $(25, 55)$ para um coeficiente de confiança de 90%, e para um número de experimentos $n = 10$ obtém-se um intervalo $(30, 35)$ para o mesmo coeficiente de confiança de 90%.

Portanto, além de escolher um coeficiente de confiança desejado, é necessário determinar um limiar de variação do intervalo de confiança. Com isto, neste trabalho foi determinado que os intervalos de confiança obtidos tenham uma variação de $\pm$ 10% em relação ao seu valor central, ou seja:

$$(l_1 < VC < l_2) = (0.9 \times VC < VC < 1.1 \times VC) \tag{4.14}$$

onde $VC = valorcentral = (l_1 + l_2)/2$

Deste modo, o procedimento adotado neste trabalho foi o seguinte:

1. Inicialmente foi realizado um número mínimo $n_{min} = 4$ de simulações para cada caso.
2. Em seguida foi calculado o intervalo de confiança para um coeficiente de confiança de 90% para cada caso.
3. Então, foi verificado se estes intervalos estão dentro do limiar de variação de $\pm$ 10% determinado.
4. Caso estes intervalos não estejam dentro do limiar, foram realizadas outras simulações de maneira incremental até que o limiar fosse alcançado.

4.1.4 Cenários de Simulação

No processo de avaliação através de simulação foram escolhidos três cenários e criadas três topologias de rede:

- No primeiro cenário, foi gerada uma topologia de rede com 20 nós e 10 roteadores, de modo a simular uma pequena rede sem fio local;
- No segundo cenário, foi gerada uma topologia de rede com 200 nós e 10 roteadores, de modo a simular uma rede sem fio local semelhante a anterior, porém com um número maior de usuários, de modo que a rede opere perto de sua capacidade limite;
- No terceiro cenário, foi gerada uma topologia de rede com 200 nós e 100 roteadores, de modo a simular uma rede de maior porte, semelhante a Internet, onde os nós estão distribuídos com uma distância relativa maior.

Em todos os cenários, os nós estão conectados aos roteadores através de enlaces com taxas de transmissão entre 250 Kbps e 800 Kbps com características de redes sem fio, como maior atraso para a entrega de mensagens e probabilidade maior de perda de pacotes. Os roteadores, por sua vez, estão conectados entre si através de enlaces com 54 Mbps com características de uma rede Ethernet.

	Cliente-Servidor	SBA	FBPA	BPA
Número Médio de Pacotes Recebidos	3290,47	9593,08	9779,29	9835,47
Atraso Médio para o início da recepção	1,01 s	4,2 s	3,78 s	3,58 s

Tabela 4.1: Comparação entre os algoritmos e a arquitetura cliente-servidor para o primeiro cenário

4.1.5 Resultados

Para todos os cenários foram feitas comparações entre:

- O desempenho de uma arquitetura cliente-servidor e um sistema de Vídeo sob-Demanda que utiliza os três algoritmos apresentados neste trabalho;
- Os três algoritmos, avaliando-se o desempenho destes em situações as quais os nós fazem uma estimativa errada de sua capacidade de transmissão, quando os nós atualizam ou não suas informações durante a execução do fluxo, e quando os nós saem da rede de modo inesperado (uma falha) durante a execução do fluxo.

Na simulação foi compartilhado um fluxo de dados de modo a representar um fluxo de vídeo de tamanho 10 Mb, o qual é disponibilizado pelo nó provedor de conteúdo. Esse fluxo, por sua vez, é transmitido em pacotes de tamanho de 1024 bits com uma taxa de transmissão de 102.4 Kbps, portanto é oferecido um fluxo de vídeo com duração de 100 segundos. Em todos os cenários, os nós enviam requisições ao nó controlador durante os primeiros 15 segundos de simulação.

A seguir são apresentados os resultados para cada cenário.

Primeiro cenário

O objetivo da avaliação deste cenário é verificar o desempenho dos algoritmos apresentados neste trabalho em uma rede local com um número de nós que simule uma situação aparentemente comum de funcionamento do sistema. Como o número de nós é menor do que os dos outros cenários, o nó controlador não lida com um número excessivo de requisições. Por exemplo, como os nós enviam suas requisições durante os primeiros 15 segundos de simulação, o nó controlador irá receber em média quatro requisições a cada 3 segundos.

A seguir são apresentados os resultados para este cenário. Estes resultados podem ser divididos em três estágios. No primeiro estágio é feita uma avaliação comparativa entre o desempenho dos três algoritmos e uma arquitetura cliente-servidor, na qual o nó controlador tem que prover o fluxo de vídeo diretamente para cada nó requisitante. Estes resultados são apresentados na Tabela 4.1.5.

Avaliando os resultados obtidos, pode-se verificar a degradação do desempenho da arquitetura cliente-servidor em relação aos algoritmos propostos neste trabalho. Apesar da topologia de rede não ser considerada crítica em relação aos outros cenários, verifica-se um número médio menor de

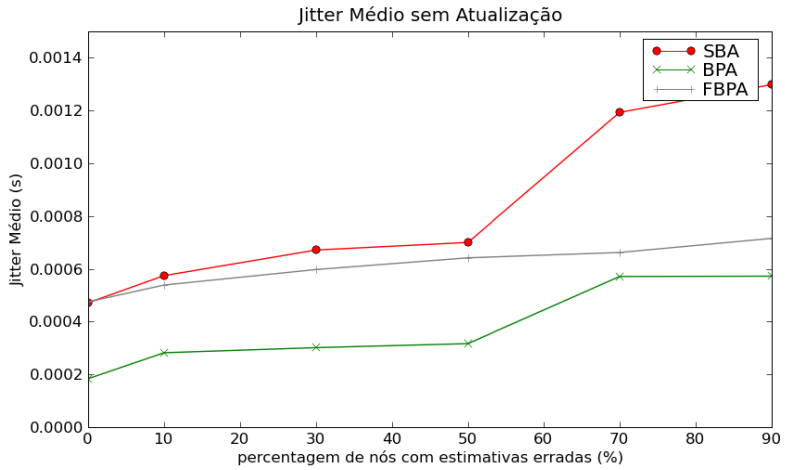
pacotes recebidos. Isto não se deve ao fato da rede estar saturada, por exemplo, mas sim ao fato do gargalo de transmissão que aparece no enlace que conecta o nó controlador a rede. Este enlace é limitado e, portanto, o nó controlador não é capaz de prover o fluxo de vídeo a todos os nós filhos de maneira simultânea. Em relação aos outros três protocolos é observado o melhor desempenho do protocolo BPA em relação ao FBPA e SBA, respectivamente. Um resultado importante a ser observado é o relativo ao atraso médio para o início do recebimento do fluxo multimídia. Como discutido no Capítulo anterior, o protocolo apresentado neste trabalho impõe uma penalidade, um atraso maior para o início do recebimento do fluxo de vídeo. Os resultados apresentados aqui refletem esta penalidade, mesmo com um número reduzido de pacotes recebidos, a arquitetura cliente-servidor entrega os primeiros pacotes com um atraso menor.

No segundo estágio de avaliação é feita uma comparação entre os três algoritmos apresentados. Os algoritmos são comparados em situações onde uma determinada percentagem dos nós estima de maneira errada sua capacidade de transmissão. Por exemplo, um nó A verifica que tem uma capacidade de transmissão de 350 Kbps e que o vídeo compartilhado tem uma taxa de 100 Kbps, porém, ao invés de estimar que é capaz de compartilhar o fluxo com outros três nós, o nó A estima que é capaz de compartilhá-lo com outros quatro nós. Deste modo, no decorrer da transmissão o nó A deverá atualizar esta estimativa e atualizar suas informações com o nó controlador. Portanto, neste estágio, além de se comparar os três algoritmos apresentados, é feita uma comparação entre duas situações:

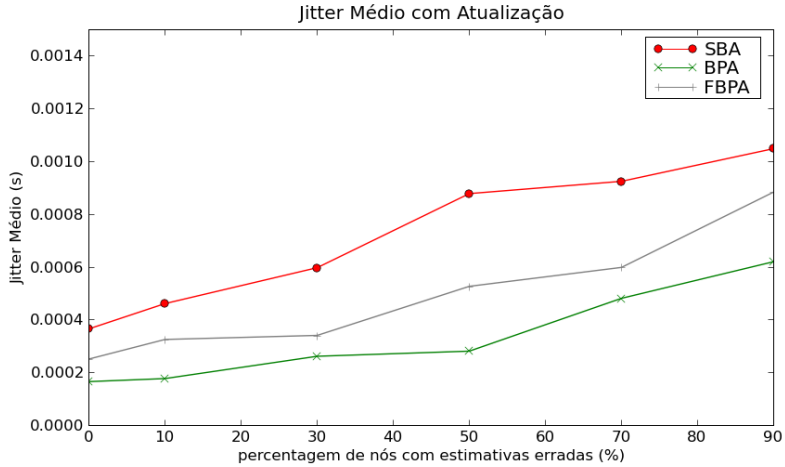
- Na primeira situação os nós estimam de maneira errada sua capacidade de transmissão, entretanto, **NÃO** enviam atualizações ao nó controlador;
- Na segunda situação os nós estimam de maneira errada sua capacidade de transmissão, e no decorrer da execução do protocolo vão enviando informações de atualização ao nó controlador.

Os resultados para este estágio são apresentados nos gráficos das Figuras 4.3 e 4.4. Foram comparadas situações onde a percentagem de nós com estimativas erradas varia entre 0% e 90% dos nós.

Em relação aos gráficos de *jitter* obtidos, pode-se verificar um *jitter* médio menor quando do uso de atualizações da rede, principalmente onde a percentagem dos nós com estimativas erradas é menor, até 30% por exemplo. Após esta percentagem, os valores médios dos *jitters* se equivalem quando se compara as duas situações, com ou sem atualização. Isto se deve ao fato de que, apesar de na situação com envio de atualizações o sistema como um todo tende a se estabilizar com um baixo *jitter*, por exemplo, durante o processo de atualização o *jitter* médio tende a variar bastante. Como exemplo, um nó pode ter seu nó pai alterado várias vezes e, em cada troca de pai, um período de transição da sobrerede ocorre, o que acarreta variação no atraso dos pacotes recebidos, ocasionando maior *jitter* naquele instante. Destes resultados do *jitter* médio, destaca-se o menor *jitter* médio obtido pelo algoritmo BPA. Este resultado se deve ao fato desse algoritmo sempre calcular uma árvore com menor distância relativa entre os nós e o nó fonte, portanto, os nós são agrupados de maneira mais próxima



(a)



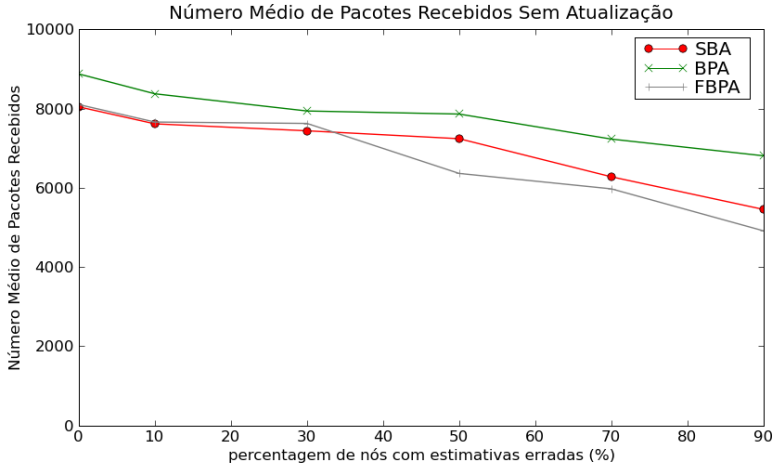
(b)

Figura 4.3: Resultados da avaliação com estimativas erradas para o primeiro cenário - *jitter*

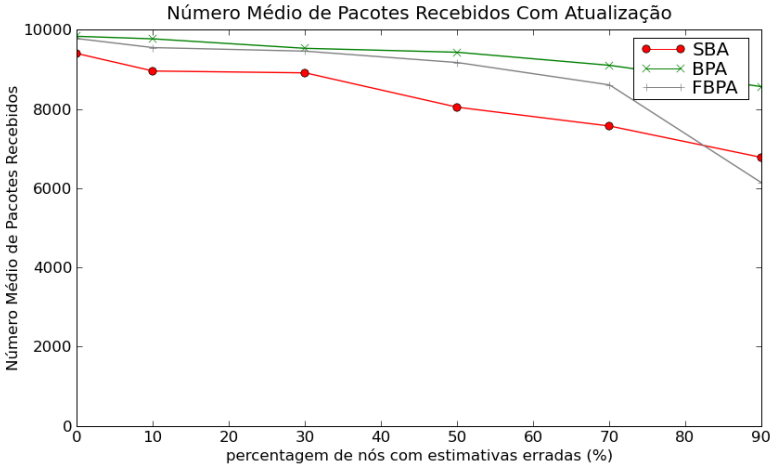
um dos outros em relação aos outros algoritmos. O resultado mais expressivo aparece nos gráficos relativos ao número médio de pacotes recebidos. Na situação onde os nós enviam atualizações sobre sua capacidade de transmissão é observado um ganho significativo em relação à situação contrária. Destaca-se o algoritmo BPA que sempre entrega um número maior de pacotes em relação aos outros algoritmos.

No terceiro e último estágio é avaliado o desempenho de cada algoritmo em situações onde uma determinada percentagem dos nós sai da rede de maneira inesperada. Neste caso, os nós começam a sair da rede após 50 segundos de simulação decorridos. Os resultados para este estágio são apresentados nos gráficos da Figura 4.5. Foram comparadas situações onde a percentagem de nós que saem da rede varia entre 10% e 50% dos nós.

Nestes últimos resultados deste cenário, em relação ao *jitter* médio pode-se observar um crescimento bastante acentuado em relação ao *jitter* médio obtido no estágio de avaliação anterior. Por



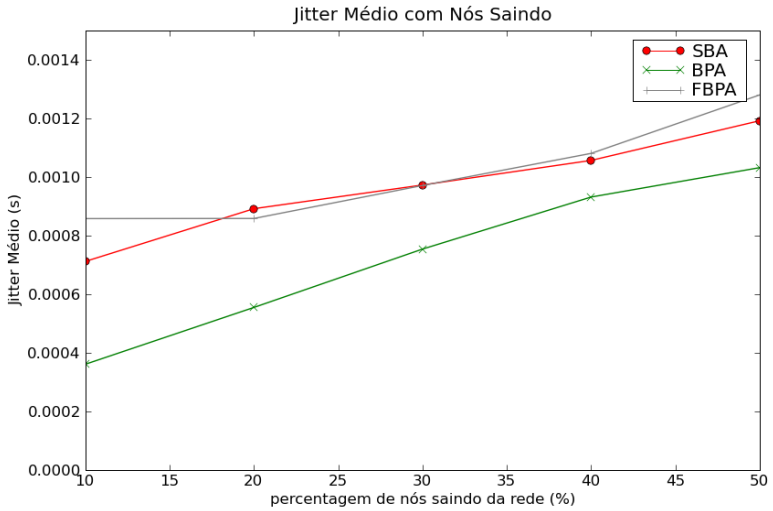
(a)



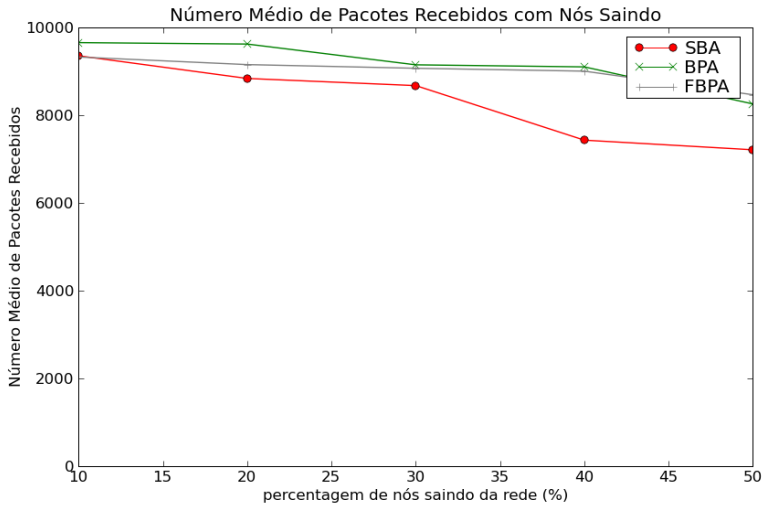
(b)

Figura 4.4: Resultados da avaliação com estimativas erradas para o primeiro cenário - pacotes recebidos

exemplo, em relação ao algoritmo SBA, no caso onde 90% dos nós efetuam estimativas erradas o *jitter* médio se aproxima de $0,6ms$, enquanto que no caso onde 50% dos nós saem da rede o *jitter* se aproxima de $1,0ms$. Estes valores maiores do *jitter* se devem ao fato de que além do período de transição após o nó controlador ser informado da saída de um nó, existe um período de tempo entre a saída do nó e o registro da saída do nó ser recebido pelo nó controlador. Ou seja, existe um período de tempo em que os nós filhos do nó que saiu ficam sem receber o fluxo de vídeo. Este período de tempo é somado no cálculo da variação do atraso de chegada dos pacotes, o que resulta em um *jitter* médio maior. Em relação ao número médio de pacotes recebidos, é observado um comportamento semelhante ao ocorrido na segunda situação.



(a)



(b)

Figura 4.5: Resultados da avaliação com nós saindo da rede para o primeiro cenário.

Segundo cenário

Este cenário tem como objetivo avaliar o protocolo proposto neste trabalho em uma topologia de rede menor, semelhante a topologia do cenário anterior, entretanto, com um número maior de nós. Deste modo é possível avaliar o protocolo em uma rede congestionada, por exemplo. Foram avaliadas as três situações apresentadas no cenário anterior. Semelhantemente ao cenário anterior, os nós requisitantes enviam suas requisições ao nó controlador durante os 15 segundos iniciais de simulação. Entretanto, neste cenário o nó controlador irá lidar com cerca de 40 requisições a cada três segundos, por exemplo. Deste modo é possível avaliar o protocolo em uma situação mais crítica, como no caso de um usuário oferecer um conteúdo ”quente“ durante um intervalo de tempo curto.

Os resultados do primeiro estágio de avaliação (comparação com a arquitetura cliente-servidor)

	Cliente-Servidor	**SBA**	**FBPA**	**BPA**
Número Médio de Pacotes Recebidos	359,25	7788,16	7932,22	8315,65
Atraso Médio para o início da recepção	14,4	79,65	67,46	66,24

Tabela 4.2: Comparação entre os algoritmos e a arquitetura cliente-servidor para o segundo cenário

são apresentados na Tabela 4.1.5.

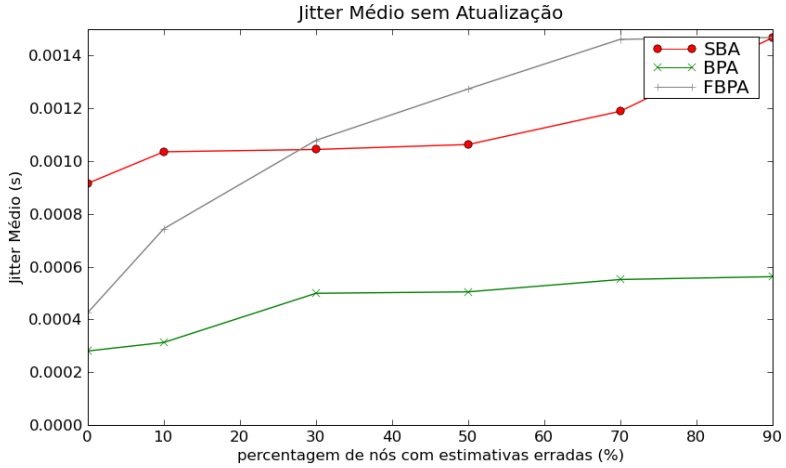
Com estes resultados fica evidente como a arquitetura cliente-servidor degrada com um número maior de nós. Entretanto, mesmo neste caso, a penalidade imposta pelo protocolo apresentado neste trabalho aparece. O atraso médio para o início do recebimento do fluxo multimídia é menor na arquitetura cliente-servidor. Como no cenário anterior, o algoritmo SBA tem um desempenho melhor no que diz respeito ao número médio de pacotes recebidos. Entretanto, neste caso, é observada uma redução em relação ao cenário anterior. Isto se deve ao fato de que este cenário apresenta uma topologia de rede com o mesmo número de roteadores que o cenário anterior, entretanto, como um número de nós 10 vezes maior. Portanto, neste cenário a topologia de rede sofre de congestionamentos maiores, os quais são causados por enlaces próximos da saturação (100% de capacidade utilizada) o que resulta em uma taxa de perda de pacotes maior. Esta conclusão fica mais evidente com os próximos resultados.

Os resultados relativos ao segundo estágio, onde é feita uma comparação entre os algoritmos variando-se a percentagem de nós com estimativas erradas, são apresentados nos gráficos das Figuras 4.6 e 4.7.

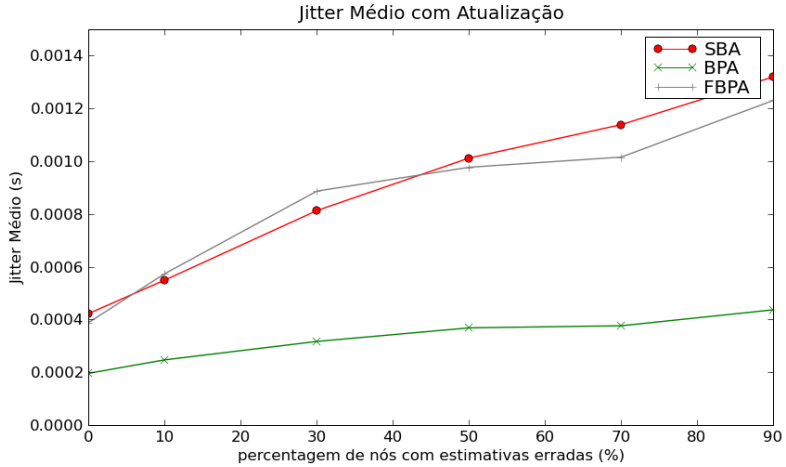
Em relação aos resultados do *jitter* médio, observa-se um comportamento semelhante ao cenário anterior, porém, com um valor médio maior principalmente para as maiores percentagens. Isto se deve ao fato da rede estar mais congestionada em relação ao cenário anterior. Destaca-se neste cenário também o desempenho superior do algoritmo BPA. No caso entre o comparativo do envio de informações de atualização de rede ou não, fica evidente o quanto a atualização da rede beneficia a entrega de pacotes. Por exemplo, para o algoritmo BPA no caso onde 90% dos nós fazem estimativas iniciais erradas, observa-se um ganho de 20% com o envio de informações de atualização. Em relação ao cenário anterior, observa-se um número inferior de pacotes entregues neste segundo cenário devido, portanto, ao congestionamento da rede com um número maior de nós.

Os resultados relativos ao desempenho da rede quando da saída dos nós são apresentados nos gráficos da Figura 4.8. Como no cenário anterior, os nós começam a sair da rede após os 50 segundos iniciais de simulação.

Neste último estágio analisado, observa-se um comportamento semelhante ao cenário anterior. Destaca-se um maior valor médio do *jitter* pois, além do *jitter* resultante do período de transição ocorrido durante a mudança de um nó pai, se soma ao *jitter* médio os valores de *jitter* resultante da perda de pacotes ocasionados pela rede congestionada. Em relação ao número de pacotes entregues,

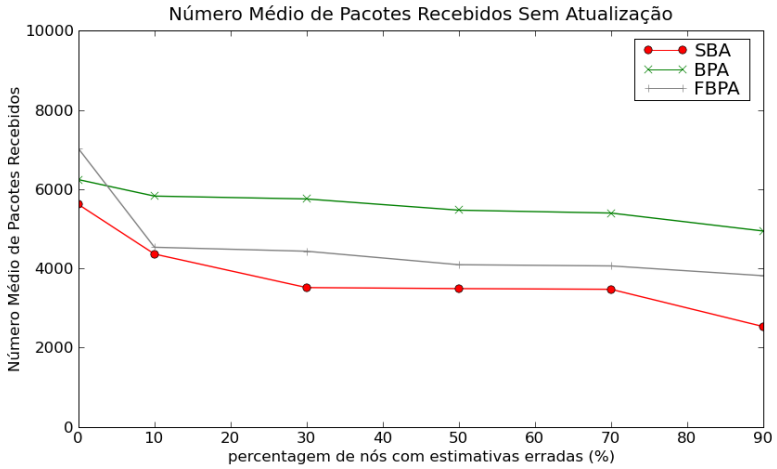


(a)

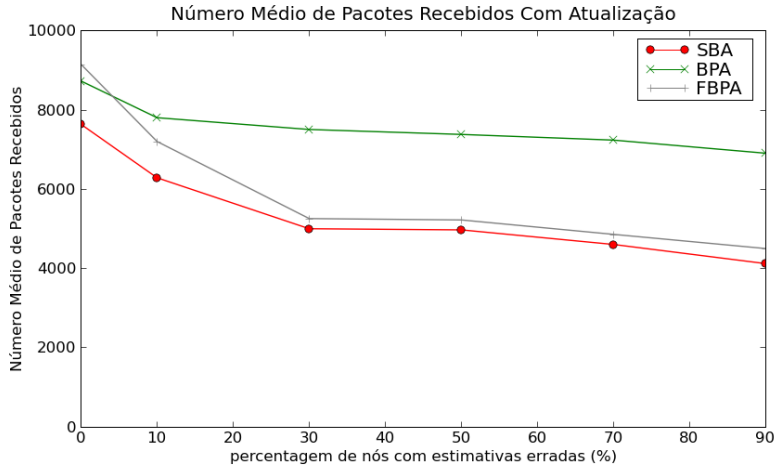


(b)

Figura 4.6: Resultados da avaliação com estimativas erradas para o segundo cenário - *jitter*

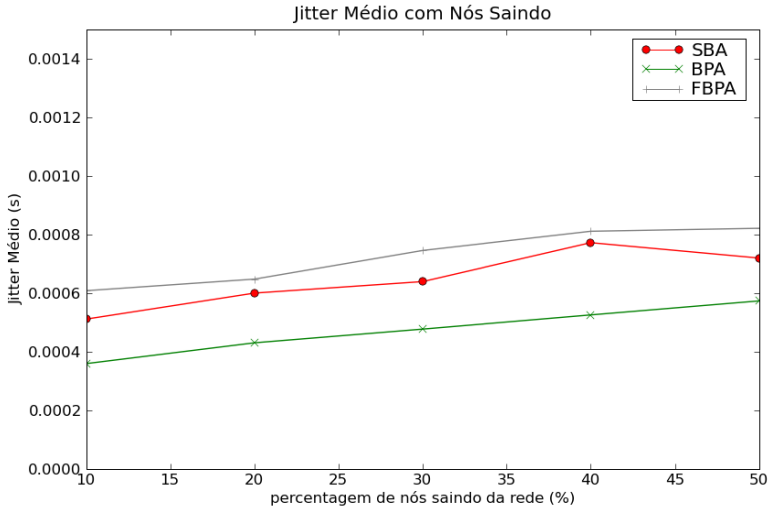


(a)

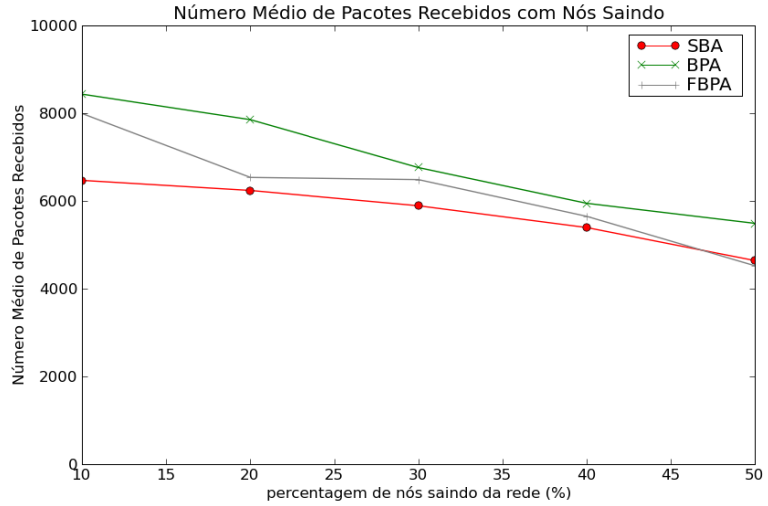


(b)

Figura 4.7: Resultados da avaliação com estimativas erradas para o segundo cenário - pacotes recebidos



(a)



(b)

Figura 4.8: Resultados da avaliação com nós saindo da rede para o segundo cenário.

os valores médios obtidos se assemelham aos valores obtidos na avaliação anterior.

Terceiro cenário

O objetivo deste cenário é avaliar o desempenho do protocolo em uma rede maior onde os nós estão mais distantes uns dos outros. Para tanto, a topologia de rede para este cenário é composta por um número maior de roteadores. Com isto, para que seja possível a comunicação entre dois nós quaisquer, em média, uma mensagem passa por um número maior de enlaces. Como nos cenários anteriores, a avaliação deste cenário passa por três estágios. Os resultados da comparação com uma arquitetura cliente-servidor são apresentados na Tabela 4.1.5.

Neste primeiro estágio de avaliação observa-se o mesmo comportamento dos cenários anteriores,

	Cliente-Servidor	**SBA**	**FBPA**	**BPA**
Número Médio de Pacotes Recebidos	353,67	7744,49	7635,09	8035,98
Atraso Médio para o início da recepção	19,9	73,5	68,25	59,57

Tabela 4.3: Comparação entre os algoritmos e a arquitetura cliente-servidor para o terceiro cenário

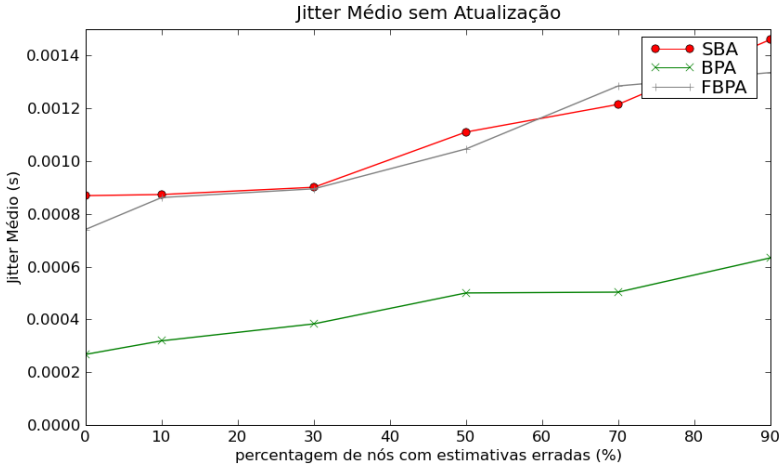
portanto, é obtida a mesma conclusão em relação à degradação da arquitetura cliente-servidor. De maneira semelhante, fica evidente a penalidade imposta pelo protocolo apresentado neste trabalho em relação ao tempo médio necessário para o início do recebimento do fluxo de vídeo. Portanto, após a análise dos três cenários, no instante de implantação de um sistema de Vídeo sob-Demanda que utilize o protocolo apresentado neste trabalho, deve-se levar em consideração requisitos relativos ao atraso para o início do recebimento do fluxo de vídeo. Caso este atraso seja crítico para o sistema desejado, outro protocolo de VoD deve ser considerado.

Em relação ao segundo estágio de avaliação, os resultados são apresentados nos gráficos das Figuras 4.9 e 4.10.

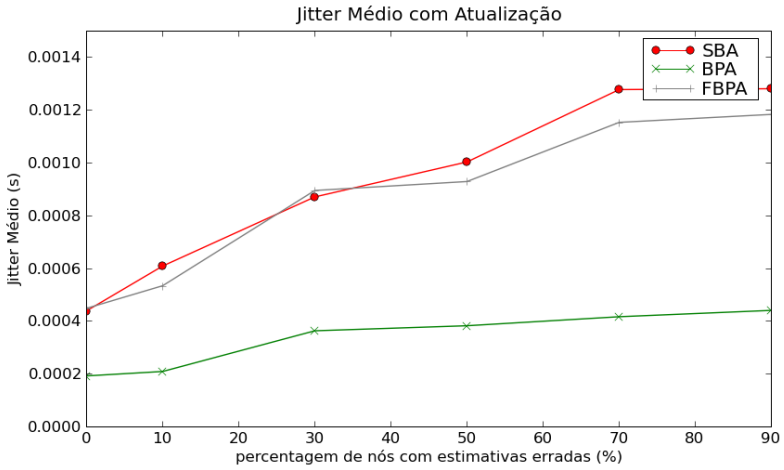
Com base nestes resultados observa-se pouca diferença no comportamento do *jitter* em relação ao segundo cenário. Entretanto, pode-se observar um desempenho pior do algoritmo FBPA nestes dois últimos cenários em relação ao primeiro cenário. Isto se deve ao fato do algoritmo SBA apenas calculas a árvore de distribuição com distâncias maiores em situações onde o número de nós é maior (mais de 1000 nós, como apresentado no capítulo 2). Em topologias com um menor número de nós, o desempenho dos algoritmos SBA e FBPA se equivale, o que fica evidente nos gráficos do *jitter* médio tanto para este cenário como para o cenário anterior. Em relação ao resultado obtido relativo ao número de pacotes recebidos, observa-se um melhor desempenho quando do uso de atualizações de rede, como esperado. Entretanto, este melhor desempenho é mais discreto se comparado com os cenários anteriores. Isto se deve ao fato da topologia de rede ser mais distribuída e, portanto, a comunicação entre dois nós distintos percorrer em média um número maior de enlaces de modo que é maior a probabilidade de um desses enlaces está congestionado.

O último estágio de avaliação é o relativo a saída dos nós da rede. Os resultados são apresentados nos gráficos da Figura 4.11.

Relacionado ao *jitter*, observa-se um valor médio maior em relação aos cenários anteriores. Isto se deve ao fato da distância média entre os nós ser maior e, portanto, a variação no atraso na entrega dos pacotes se torna maior também. Em relação ao número de pacotes recebidos observa-se um comportamento semelhante ao do estágio anterior.

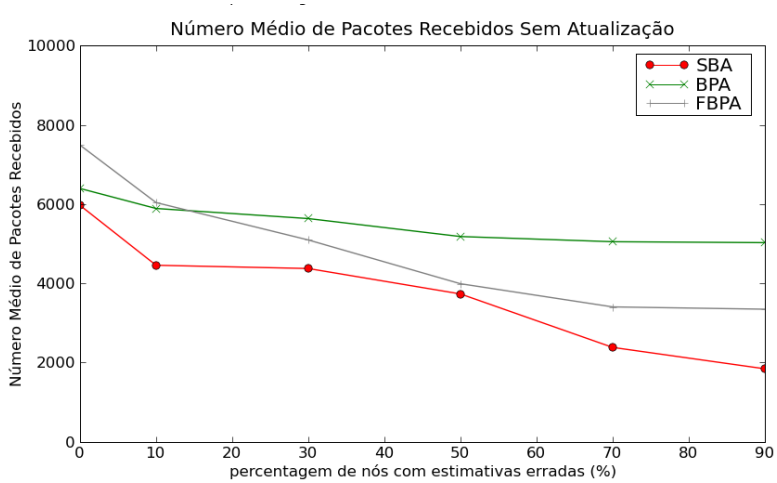


(a)

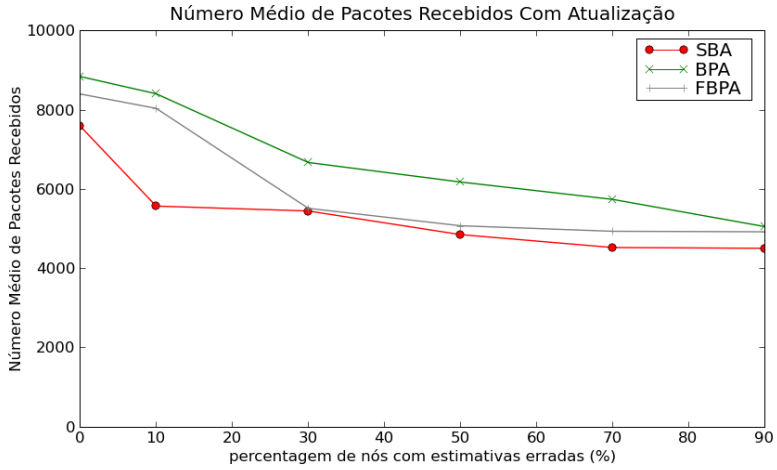


(b)

Figura 4.9: Resultados da avaliação com estimativas erradas para o terceiro cenário - *jitter*

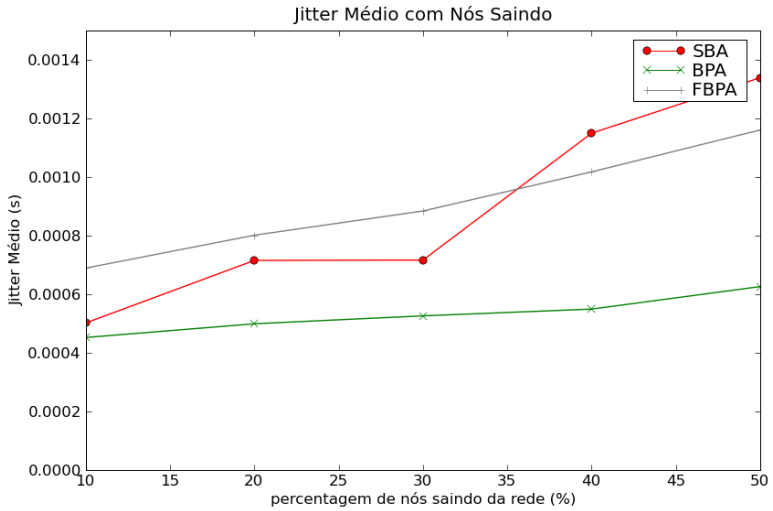


(a)

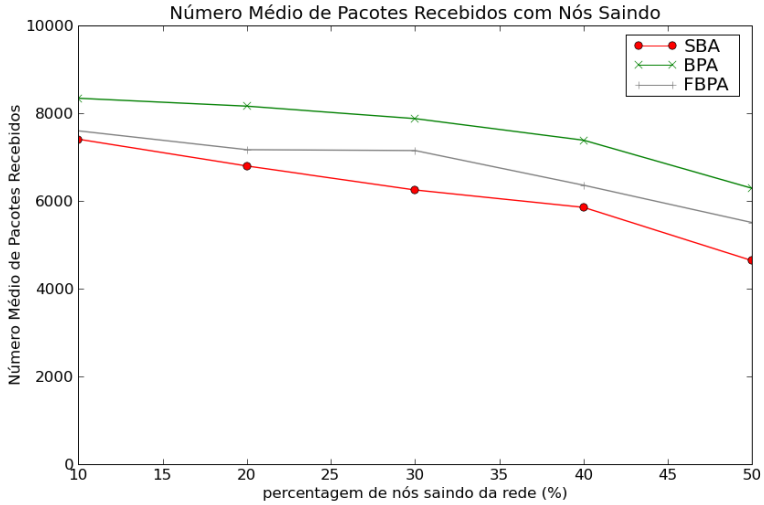


(b)

Figura 4.10: Resultados da avaliação com estimativas erradas para o terceiro cenário - pacotes recebidos



(a)



(b)

Figura 4.11: Resultados da avaliação com nós saindo da rede para o terceiro cenário.

4.2 Protótipo de Avaliação

Neste trabalho foi desenvolvida uma aplicação protótipo que tem como objetivo avaliar o funcionamento de ferramentas auxiliares, como as utilizadas para obter as coordenadas de rede, em conjunto com o protocolo apresentado neste trabalho. Este protótipo foi desenvolvido na linguagem de programação Python[3]. Os dispositivos alvos deste protótipo são os que utilizam a plataforma Linux, mais especificamente a plataforma maemo[4], como os dispositivos portáteis Nokia N800[5]. O cenário de aplicação foi uma rede sem fio IEEE 802.11g.

[3]http://www.python.org
[4]http://www.maemo.org
[5]http://www.nseries.com

Como ilustrado na Figura 4.12, a aplicação desenvolvida pode ser dividida em quatro módulos:

- Módulo de controle, implementa a comunicação entre os nós;
- Módulo de localização, que calcula as coordenadas de rede;
- Módulo de capacidade, que avalia a capacidade de transmissão de um nó;
- Módulo multimídia, que envia e recebe o fluxo de vídeo.

Neste primeiro protótipo a aplicação pode se comportar de duas maneiras distintas e separadas: como aplicação provedora de conteúdo, ou como aplicação cliente. O modo de utilização deve ser definido no início da execução da aplicação.

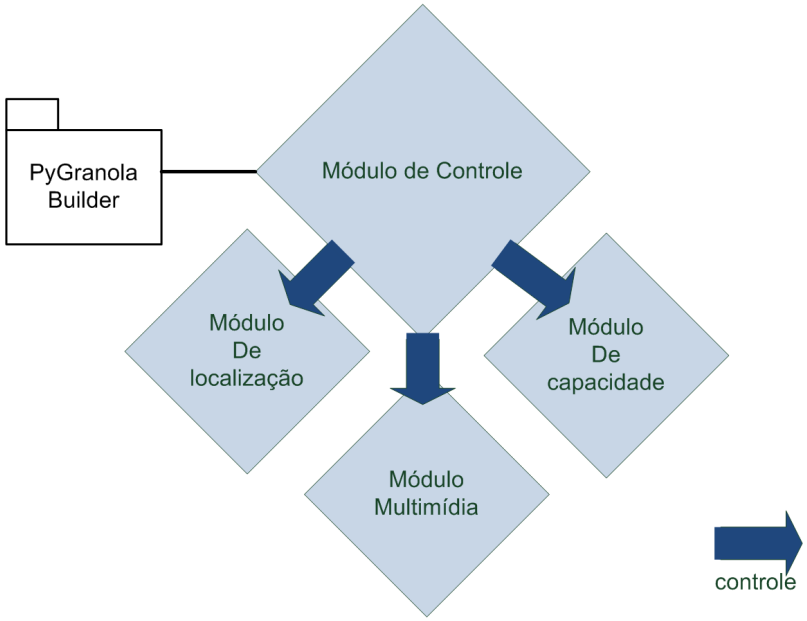


Figura 4.12: Diagrama ilustrando os módulos desenvolvidos para o protótipo.

4.2.1 Módulo de controle

A biblioteca *GranolaTreeBuilderLib* foi desenvolvida na linguagem de programação C++. Para ser utilizada pelo módulo de controle da aplicação provedora de conteúdo, foram desenvolvidos *bindings* para a linguagem Python utilizando a ferramenta Pyrex[6]. Deste modo, foi oferecida uma ligação direta entre a interface do *GranolaTreeBuilderLib* em C++ e a interface em Python, chamada de *PyGranolaBuilder*.

Para a comunicação entre as aplicações cliente e a aplicação provedora de conteúdo é utilizado o protocolo XML-RPC[7]. XML-RPC é uma das especificações mais simples de *web services*. Nessa,

[6] http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/
[7] http://www.xmlrpc.com

o formato XML descreve chamadas de procedimentos remotos (RPC), as quais são transmitidas entre computadores através do protocolo HTTP. Para tanto, é necessário que a aplicação cliente tenha conhecimento do endereço IP e porta da aplicação provedora de conteúdo. Estas informações são fornecidas como parâmetros de entrada da aplicação cliente em sua inicialização. O módulo de controle, por final, é o responsável pela organização do fluxo e comunicação entre os outros módulos.

4.2.2 Módulo de localização

O protocolo escolhido para a utilização no protocolo de VoD apresentado neste trabalho foi o Vivaldi [6], como descrito no Capítulo 2. Existe disponível uma implementação do protocolo Vivaldi que funciona de maneira distribuída e é de software livre e aberto, o Pyxida[8]. Entretanto, o Pyxida é desenvolvido na linguagem de programação Java[9], a qual não é a linguagem do protótipo desenvolvido, e também não é suportada nativamente pela plataforma maemo.

Portanto, foi realizado um trabalho para adaptar o Pyxida para plataforma maemo. Inicialmente foi utilizado o Jalimo[10], o qual é um projeto que mantém uma máquina virtual Java para dispositivos móveis com distribuição Linux, como o maemo. Entretanto, o Pyxida utiliza alguns pacotes disponibilizados pela Sun1, os quais não são oferecidos pelo Jalimo. Depois de identificados estes pacotes, sua utilização foi isolada no código fonte de Pyxida e substituída pelo uso de outros pacotes.

O módulo de localização, então, foi implementado na linguagem Python e utiliza XML-RPC para se comunicar com o Pyxida, o qual é executado em outro processo no mesmo dispositivo. Toda vez que o módulo de controle requisita informações de localização ao módulo de localização, este envia a requisição ao Pyxida via XML-RPC.

Para determinar a suas coordenadas de rede o Pyxida precisa se comunicar com outro dispositivo que esteja o executando e, através do processo descrito no Capítulo 2, determinar suas coordenadas de rede. Portanto, nos testes executados com o protótipo desenvolvido, as aplicações cliente sempre determinavam suas coordenadas em relação a aplicação provedora de conteúdo.

4.2.3 Módulo de capacidade

Para estimar a capacidade de transmissão do dispositivo, foram analisadas duas ferramentas de modo a se comparar os dois modelos apresentados no Capítulo 2:

- O Spruce [45], o qual utiliza o Modelo de Análise de Lacunas;
- O Pathload [25], o qual utiliza o Modelo de Análise de Taxa.

[8] http://pyxida.sourceforge.net
[9] http://www.sun.com
[10] https://wiki.evolvis.org/jalimo/

Após uma análise inicial, o Spruce foi escolhido cálcular a estimativa rápidamente. Para efetuar a estimativa da capacidade de transmissão disponível entre dois dispositivos, o Spruce necessita ser executado em ambos os dispositivos. Portanto, o módulo de localização desenvolvido neste protótipo, tem como principal objetivo executar o processo do Spruce no dispositivo.

Ao ser requisitado para o cálculo de uma estimativa da capacidade de transmissão entre dois dispositivos, o módulo de capacidade executa outro Spruce, o qual envia uma requisição ao outro dispositivo e faz o cálculo da estimativa. Como o ambiente de testes foi uma rede local sem fio, WLAN, a qual oferece taxas elevadas de transmissão de dados, o módulo de capacidade também oferece um modo de execução de teste, o qual faz uma estimativa errônea da capacidade de transmissão. Deste modo, é possível testar características do módulo de controle relativas a atualização de informações de um dispositivo, por exemplo.

4.2.4 Módulo multimídia

O módulo multimídia foi desenvolvido utilizando o arcabouço multimídia de código aberto, GStreamer[11]. Apesar de ser desenvolvido na linguagem de programação C, o GStreamer permite seu uso através de vários *bindings* para outras linguagens, como Perl, Java, e Python.

O Gstreamer tem uma arquitetura baseada em *plugins*. Cada *plugin* pode conter um ou mais elementos. Estes elementos, por sua vez, implementam funcionalidades específicas como a codificação ou decodificação de um fluxo MP3, a abertura e leitura de arquivo qualquer, ou a execução de um fluxo de som na saída padrão de áudio de um dispositivo. Portanto, o Gstreamer permite que estes elementos sejam ligados entre si formando um *pipeline*. Este *pipeline*, então, executa as funções dos elementos em um fluxo contínuo, por exemplo, ao se criar um *pipeline* com o objetivo de executar uma música codificada em MP3 primeiramente é usado um elemento para abrir o arquivo desejado. Em seguida, este elemento envia pedaços desse arquivo ao elemento posterior, neste caso um decodificador MP3, o qual decodifica o áudio MP3 e o transforma em áudio bruto. Por fim este áudio bruto é enviado a um elemento que o executa na saída de áudio do dispositivo.

No que diz respeito ao protótipo desenvolvido, foram utilizados elementos que enviam e recebem dados usando o protocolo RTP (*Real Time Protocol*). Portanto, o módulo multimídia faz uso de três tipos de *pipelines*:

- O primeiro *pipeline* recebe o fluxo multimídia vindo de outro dispositivo e o salva em um arquivo temporário;
- O segundo *pipeline* executa localmente o arquivo temporário que esta sendo salvo pelo primeiro *pipeline*;

[11] http://www.gstreamer.net/

Figura 4.13: *Screenshot* da tela de requisição.

Figura 4.14: *Screenshot* da tela da aplicação recebendo o fluxo de vídeo.

- O terceiro *pipeline* lê o arquivo temporário e o envia a outros dispositivos utilizando o RTP (*Real-Time Protocol*).

4.2.5 Cenário de Teste

Para o teste do protótipo, foi utilizado um *notebook* com uma câmera embutida como provedor de conteúdo e três dispositivos Nokia N800 como consumidores de conteúdo. O *notebook* gera o conteúdo de vídeo a partir de sua câmera de vídeo, e o compartilha com os outros dispositivos. No teste, os dispositivos N800 entraram na rede em diferentes intervalos de tempo e, em todos os casos, estes acessaram o vídeo a partir do início do fluxo gravado. A Figura 4.13 apresenta o *screenshot* da tela de requisição quando o dispositivo entra na rede. A Figura 4.14 apresenta o *screenshot* da tela quando o dispositivo está recebendo o fluxo de vídeo.

4.3 Sumário

Neste capítulo foram apresentados e descritos os procedimentos utilizados para a simulação do protocolo Granola. Foram simulados três cenários, de modo a avaliar o desempenho do protocolo em diferentes situações. Em todos os cenários foi observado o melhor desempenho do protocolo Granola em relação a uma arquitetura cliente-servidor. Ao final do capítulo, foi apresentada uma descrição de um protótipo de avaliação, que teve como objetivo integrar as diferentes ferramentas necessárias para o funcionamento do Granola.

Capítulo 5

Considerações Finais e Trabalhos Futuros

Recentemente os usuários de dispositivos móveis, como *smartphones* e PDAs, demandam cada vez mais novos tipos serviços. Esta demanda vem do fato desses dispositivos oferecerem um conjunto de possibilidades, sejam estas criadas pelo oferecimento de novos dispositivos agregados, como câmeras digitais, ou pelas várias possibilidades de conectividade, como redes celulares com transmissão de dados de banda larga ou tecnologias de rede sem fio local e pessoal, como *Wi-Fi* e *Bluetooth*, respectivamente. Estas novas tecnologias agregadas aos dispositivos móveis fazem com que os usuários tenham a possibilidade de criar conteúdo digital e, além disto, compartilhar este conteúdo pela Internet. Neste contexto, um novo serviço que surge é o oferecimento de Vídeo sob-Demanda para estes usuários.

Neste trabalho foi introduzido um protocolo para o oferecimento de Vídeo sob-Demanda para dispositivos móveis, chamado de Granola. Este protocolo permite que estes dispositivos sejam tanto provedores quanto consumidores de conteúdo, neste caso, vídeo. Para isto, na definição do protocolo leva-se em consideração duas características: a posição relativa do dispositivo na rede, e sua capacidade de transmissão. Com isto, é possível construir árvores de distribuição de conteúdo de modo a posicionar os dispositivos mais próximos uns dos outros sem ultrapassar sua capacidade de transmissão.

Poucos trabalhos tem como objetivo disponibilizar VoD para dispositivos móveis. Destes, nenhum trabalho considera uma implantação na Internet onde estes dispositivos possam ser os provedores de conteúdos. Além disto, a maioria dos trabalhos relacionados a disponibilização de VoD para dispositivos móveis [42] [13] [15] [23] [15], até a escrita deste trabalho, oferecem dados suficientes para se fazer uma análise comparativa entre seus resultados e os obtidos neste trabalhos.

No contexto deste trabalho, em relação as árvores de distribuição, no capítulo 3 foram introduzidos três algoritmos: o SBA (*Simple Bandwidth Aware*), o FBPA (*Fast Bandwidth and Position Aware*), e o BPA (Bandwidth and Position Aware). Os algoritmos foram descritos detalhadamente, e uma avaliação experimental sobre eles foi apresentada.

Para uma melhor avaliação do protocolo e seus algoritmos, foi simulado um sistema de Vídeo

sob-Demanda que utiliza o protocolo em diferentes topologias de rede. No desenvolvimento deste sistema utilizou-se uma biblioteca desenvolvida em C++, denominada de *GranolaTreeBuilderLib*, que implementa todas as funções do protocolo e os três algoritmos. Em relação a simulação, foi feita uma comparação entre um sistema de vídeo que utiliza o protocolo Granola, e um sistema de vídeo que utiliza uma arquitetura cliente-servidor. Nesta comparação, detalhada no capítulo 3, fica evidente a eficácia de utilizar uma arquitetura P2P, neste caso uma árvore de distribuição, em relação a uma arquitetura cliente-servidor. Além desta comparação, foram feitas comparações entre os três algoritmos apresentados. Como esperado, o algoritmo BPA se comporta melhor em relação aos outros algoritmos em todos os cenários. Isto se deve ao fato do BPA sempre computar uma árvore de distribuição com menor distância entre os nós e o nó fonte. O uso do BPA, portanto, deve ser revisado apenas em situações onde seja possível formar a rede com um número elevado de dispositivos, na ordem dos milhares. Esta restrição se deve ao fato do maior custo de processamento que este algoritmo impõe, como foi discutido na avaliação experimental dos algoritmos apresentada no capítulo 2.

Para avaliar a viabilidade de implementação de uma aplicação que utilize o Granola, foi implementado um protótipo de avaliação. Este protótipo integrou várias ferramentas necessárias para o funcionamento do protocolo. Para o desenvolvimento do protótipo foi desenvolvida uma biblioteca em Python, chamada de *PyGranolaBuilder*. O protótipo integrou várias ferramentas como *Pyxida* para a determinação das coordenadas de rede, e o *Spruce* para avaliação da capacidade de transmissão de um dispositivo. O protótipo foi desenvolvido tendo como cenário uma rede sem fio local.

5.1 Trabalhos Futuros

O trabalho desenvolvido e apresentado é um primeiro passo para a disponibilização de um sistema totalmente funcional de Vídeo sob-Demanda para dispositivos móveis. Portanto, alguns tópicos para trabalhos futuros são apresentados a seguir.

5.1.1 Aplicação de VoD

Para uma completa validação do protocolo introduzido neste trabalho, é necessário desenvolver e oferecer ao usuário final uma aplicação que compartilhe vídeos entre uma comunidade de pessoas, por exemplo. Deste modo, pode-se avaliar o desempenho do protocolo em um cenário real, como a própria Internet, onde os usuários estarão espalhados em vários locais do mundo. Para isto, uma aplicação de código aberto e livre está sendo desenvolvida para a plataforma maemo[1]. Deste modo, a aplicação será utilizada por vários usuários, e contribuições de outros desenvolvedores de código livre poderão ocorrer.

Entretanto, desenvolver uma aplicação P2P para dispositivos móveis não é uma tarefa simples.

[1] http://www.maemo.org

Alguns problemas surgem neste contexto relativos ao acesso destes dispositivos através de redes celulares e algumas redes sem fio locais. A maioria destas redes utiliza NAT (*Network Access Translation*) e apenas oferecem um endereço IP local para o acesso dos usuários. Portanto, dispositivos que estão atrás de roteadores com NAT não são capazes de oferecer conectividade fim-a-fim, o que é extremamente necessário para o protocolo Granola, por exemplo.

Portanto, é necessário encontrar uma solução para este problema de modo a oferecer uma aplicação de VoD que utilize um protocolo P2P. Algumas soluções foram propostas, como a utilizada no *Symella*[2], que foi o primeiro cliente do Gnutella para o sistema operacional *Symbian*[3]. Esta solução faz uso de um servidor *proxy*, chamado de *Connector Server*, que tem como principal tarefa oferecer um endereço IP público para os clientes que estão se conectando a rede [11]. Este servidor oferece duas portas para os clientes. Uma porta disponibiliza acesso a funcionalidades de controle, onde uma conexão de controle é criada e mantida durante toda a sessão a qual o cliente esta conectado. E uma outra porta, denominada de porta de encaminhamento, disponibiliza mecanismos para encaminhar conexões para os clientes móveis. Uma conexão de encaminhamento é criada por um cliente, e faz a transmissão entre o dispositivo que está se conectando com o servidor *proxy* e o cliente móvel. Além deste exemplo prático, outras técnicas para comunicação P2P através de redes com tradução de endereços são conhecidas, como o *Hole punching* [12], por exemplo.

Portanto, um estudo detalhado das várias soluções para a comunicação fim-a-fim através de NATs deve ser feito, de modo a escolher a melhor alternativa para a implantação de um sistema de VoD que utilize o protocolo Granola.

5.1.2 Sistema de VoD Híbrido

Como descrito no Capítulo 2, vários protocolos para o oferecimento de Vídeo sob-Demanda existem. A maioria destes protocolos faz uso de mecanismos de difusão *multicast*. Entretanto, devido a limitação do uso de *multicast* na Internet [3], a utilização destes protocolos se restringe a redes locais ou privadas. Este foi um dos fatores relevantes para o desenvolvimento de um protocolo P2P para VoD, no contexto deste trabalho, o protocolo Granola.

Entretanto, os protocolos que utilizam difusão *multicast* são eficazes, pois compartilham um único fluxo multimídia com vários dispositivos. Portanto, em uma rede local, por exemplo, é mais eficaz compartilhar um fluxo multimídia utilizando um dos protocolos de difusão *multicast*, do que utilizando um protocolo P2P como o Granola.

Neste contexto, um outro ponto para um trabalho futuro seria estender o protocolo Granola, de modo que ele se comportasse de maneira híbrida, como ilustrado na Figura 5.1. Neste cenário, o protocolo permitiria organizar os nós em relação a suas redes. Caso fosse identificado que um conjunto de nós está em uma mesma rede local, por exemplo, estes nós iriam se comunicar através de

[2]http://symella.aut.bme.hu

[3]http://www.symbian.com

um protocolo de difusão *multicast*. Dentre estes nós, um dispositivo receberia e encaminharia o fluxo multimídia para os outros nós distribuídos pela Internet utilizando uma árvore P2P, por exemplo

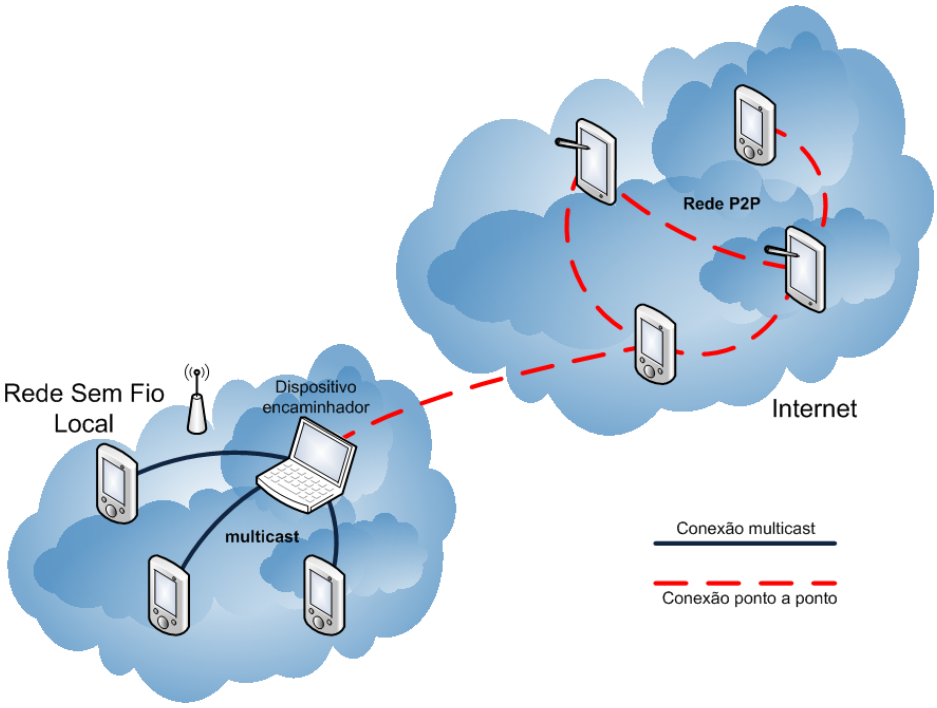


Figura 5.1: Ilustração de um cenário para o protocolo Granola Híbrido

Referências Bibliográficas

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *IEEE International conf. on Multimedia Computing and Systems*. IEEE, 1996.

[2] Charu Aggarwal, Joel Wolf, and Philip S. Yu. On optimal piggyback merging policies for video-on-demand systems. In *SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 200–209, New York, NY, USA, 1996. ACM Press.

[3] Mostafa H. Ammar. Why johnny can't multicast: lessons about the evolution of the internet. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 1–1, New York, NY, USA, 2003. ACM.

[4] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: high-bandwidth multicast in cooperative environments. *SIGOPS Oper. Syst. Rev.*, 37(5):298–313, 2003.

[5] Yi Cui, Baochun Li, and Klara Nahrstedt. ostream: Asynchronous streaming multicast in application-layer overlay networks. *IEEE Journal on Selected Areas in Communications*, 22:91–106, 2004.

[6] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system, 2004.

[7] Danilo F. S. Santos, Angelo Perkusich. A Location and Bandwitdh Aware P2P Video on Demand System for Mobile Devices. In *Proceedings of International Conference on Wireless Information Networks and Systems (WINSYS 2008)*, 2008.

[8] Danilo F. S. Santos, Angelo Perkusich. Granola: A Location and Bandwidth Aware Protocol for Mobile Video on-Demand Systems. In *Proceedings of Softcom 2008, 16th International Conference on Software, Telecommunications and Computer Networks*. Piscatway, IEEE Communication Society, 2008.

[9] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1959.

[10] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure?

[11] F. H. P. Fitzek and F. Reichert. *Mobile Phone Programming and its Application to Wireless Networking*. Springer, 2007.

[12] Bryan Ford. Peer-to-peer communication across network address translators. In *In USENIX Annual Technical Conference*, pages 179–192, 2005.

[13] Panayotis Fouliras, Spiros Xanthos, Nikolaos Tsantalis, and Athanasios Manitsaris. Lemp: Lightweight efficient multicast protocol for video on demand. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1226–1231, New York, NY, USA, 2004. ACM Press.

[14] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. Idmaps: a global internet host distance estimation service. *IEEE/ACM Trans. Netw.*, 9(5):525–540, 2001.

[15] Yi Qi Gui, Eysuk Jung, Young Choi, and Hwang Kyu Choi. An efficient periodic broadcasting scheme for mobile video-on-demand system. In *ITNG '07. Fourth International Conference on Information Technology*, pages 888–889, 2007.

[16] Gunter Bolch, Stefan Greiner, Hermann de Meer, Kishor S. Trived. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. 2006.

[17] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2cast: peer-to-peer patching scheme for vod service. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 301–309, New York, NY, USA, 2003. ACM Press.

[18] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RFC 3550 - RTP: A Transport Protocol for Real-Time Applications, July 2003.

[19] A. Hu. Video-on-demand broadcasting protocols: a comprehensive study. In *Proceedings. INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 508 – 517. IEEE, 2001.

[20] Ningning Hu, Student Member, Peter Steenkiste, and Senior Member. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 21:879–894, 2003.

[21] Kien A. Hua, Ying Cai, and Simon Sheu. Patching: a multicast technique for true video-on-demand services. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 191–200, New York, NY, USA, 1998. ACM Press.

[22] Kien A. Hua and Simon Sheu. Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. In *SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 89–100, New York, NY, USA, 1997. ACM Press.

[23] Regant Y. S. Hung and H. F. Ting. An optimal broadcasting protocol for mobile video-on-demand. In *CATS '07: Proceedings of the thirteenth Australasian symposium on Theory of computing*, pages 79–84, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[24] S. W. Carter J.-F. Paris and D. D. E. Long. A hybrid broadcasting protocol for video on demand. In *Multimedia Computing and Networking Conference (MMCN 99)*, pages 317 – 326, 1999.

[25] M. Jain and C. Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth, 2002.

[26] Xuxian Jiang, Yu Dong, Dongyan Xu, and B. Bhargava. Gnustream: a p2p media streaming system prototype. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 325–328, Washington, DC, USA, 2003. IEEE Computer Society.

[27] L. Juhn and L. Tseng. Harmonic broadcasting for video-on-demand service. In *IEEE Transactions on Broadcasting*, volume 3, pages 268–271. IEEE, 1997.

[28] L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. In *IEEE Transactions on Broadcasting*, pages 100 – 105. IEEE, 1998.

[29] Knuth, Donald E. *The Art Of Computer Programming*. 1997.

[30] Huadong Ma and Kang G. Shin. Multicast video-on-demand services. *SIGCOMM Comput. Commun. Rev.*, 32(1):31–43, 2002.

[31] Meyer, Paul L. *Probabilidade: Aplicações à Estatística*. 1975.

[32] T. S. Eugene Ng and Hui Zhang. Global network positioning: a new approach to network distance prediction. *SIGCOMM Comput. Commun. Rev.*, 32(1):73–73, 2002.

[33] T. S. Eugene Ng and Hui Zhang. A network positioning system for the internet. In *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 11–11, Berkeley, CA, USA, 2004. USENIX Association.

[34] J.-F. Paris. A simple low-bandwidth broadcasting protocol for video-on-demand. In *8th International Conference on Computer Communications and Networks (IC3N 99)*, pages 118–123, 1999.

[35] J.-F. Paris, S. W. Carter, and D. D. E. Long. A low bandwidth broadcasting protocol for video on demand. In *IEEE International Conference on Computer Communications and Networks (IC3N 98)*. IEEE, 1998.

[36] Jehan-Francois Pâris, Steven W. Carter, and Darrell D. E. Long. Efficient broadcasting protocols for video on demand. In *MASCOTS '98: Proceedings of the 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 127, Washington, DC, USA, 1998. IEEE Computer Society.

[37] Ho Kyun Park and Hwang Bin Ryou. Multicast delivery for interactive video-on-demand service. In *Proceedings Twelfth International Conference on Information Networking, 1998. (ICOIN-12)*, pages 46–50, 1998.

[38] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[39] Vinay J. Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. 2003.

[40] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.

[41] K. Sato, M. Katsumoto, and T. Miki. P2mvod: peer-to-peer mobile video on-demand. In *ICACT 2006. The 8th International Conference on Advanced Communication Technology*, volume 3. IEEE, 2006.

[42] K. Sato, M. Katsumoto, and T. Miki. Peer-to-peer based mobile video on-demand with layered video distribution. In *CISIS 2007. First International Conference on Complex, Intelligent and Software Intensive Systems*, pages 119 – 126. IEEE, 2007.

[43] Sridhar Srinivasan and Ellen Zegura. M-coop: A scalable infrastructure for network measurement. In *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*, page 35, Washington, DC, USA, 2003. IEEE Computer Society.

[44] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[45] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 39–44, New York, NY, USA, 2003. ACM.

[46] Wolfgang Theilmann and Kurt Rothermel. Dynamic distance maps of the internet. In *In IEEE INFOCOM 2000, Tel Aviv*, 2000.

[47] F. Thouin and M. Coates. Video-on-demand networks: Design approaches and future challenges. *IEEE Network*, 21:42 – 48, 2007.

[48] D. A. Tran, M. Le, and K. A. Hua. Mobivod: A video-on-demand system design for mobile ad hoc networks. In *Proceedings of the 5th IEEE International Conference on Mobile Data Management (MDM)*, pages 212–223. IEEE, 2004.

[49] S. Viswanathan and T. Imielinski. Pyramid broadcasting for video on demand service. In *IEEE Multimedia Computing and Networking Conference*, pages 66 – 77. IEEE, 1995.

[50] Xing Jin Wai-Pun Ken Yiu and Shueng-Han Gary Chan. Challenges and approaches in large-scale peer-to-peer media streaming. In *IEEE Multimedia, Vol. 14, No. 2*, pages 50–59, 2007.

[51] Xinyan Zhang, Jiangchuan Liu, and Tak shing Peter Yum. Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In *in IEEE Infocom*, 2005.

[52] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, Berkeley, CA, USA, 2001.

[53] M. Zhou and J. Liu. A hybrid overlay network for video-on-demand. In *Proceedings of IEEE International Conference on Communications (ICC'05)*, pages 1309–1313. IEEE, 2005.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo