UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Suporte para Transmissão de Mídia Contínua entre Servidor e Clientes Cooperativos"

Marcio Akio Shimoda

São Carlos

2005

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Suporte para Transmissão de Mídia Contínua entre Servidor e Clientes Cooperativos"

Marcio Akio Shimoda

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes.

São Carlos
2005

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S335st

Shimoda, Marcio Akio.
Suporte para transmissão de mídia contínua entre servidor e clientes cooperativos / Marcio Akio Shimoda. -- São Carlos : UFSCar, 2006.
73 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Sistemas distribuídos. 2. Peer-to-peer. 3. Distribuição de mídia. I. Título.

CDD: 005.43 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Suporte para Transmissão de Mídia Contínua entre Servidor e Clientes Cooperativos"

MARCIO AKIO SHIMODA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Luis Carlos Trevelin
(Orientador – DC/UFSCar)

Prof. Dr. Sérgio Donizetti Zorzo
(DC/UFSCar)

Prof. Dr. Edgard Jamhour
(PUC/PR)

São Carlos
Agosto/2005

À minha família e aos meus amigos...

companheiros de todas as horas...

AGRADECIMENTOS

A minha família, pela confiança e apoio, principalmente nos momentos mais difíceis.

Aos amigos, minha segunda família, pela força e pela motivação em relação a esta jornada.

Ao Prof. Dr. Luis Carlos Trevelin, orientador neste trabalho, bem como aos demais professores do curso.

A Érico de Mattos e Fernando Jardim, que em muito contribuíram nas etapas iniciais deste trabalho, e aos outros colegas de curso e de laboratório.

A todos que, com boa intenção, colaboraram para a realização deste trabalho.

"A imaginação é mais importante que o conhecimento."

Albert Einstein

RESUMO

Neste trabalho discutimos o problema da distribuição de conteúdo multimídia para um grande número de computadores. Especificamente, consideramos o problema da sobrecarga do servidor, que ocorre devido ao rápido aumento do volume de requisições para receber conteúdo. Como conseqüência tem-se o aumento do tempo de resposta e até negação de serviço. Apresentamos como solução um modelo de comunicação para complementar o tradicional cliente-servidor. Ele é baseado nas redes cooperativas, onde os clientes cooperam para distribuir a mídia, fazendo com que eles atuem como uma espécie de servidor temporário a novos clientes. Com esse redirecionamento, reduz-se o tráfego no servidor, o que alivia a sua carga. A solução proposta é apresentada em detalhes, mostrando as modificações no sistema tradicional.

Palavras-chave: distribuição de mídia; *peer-to-peer*; clientes cooperativos.

ABSTRACT

In this work the problem of the multimedia content distribution to a large number of computers is addressed. Specifically, we consider the problem of the server overloading that occurs due to the increase of the volume of requests to receive content. The consequence is the increase of response time and denial of service. As solution a communication model complementing the traditional client-serve is presented. It is based on the cooperative networks, where the clients cooperate to distribute the media, where they act as a kind of temporary server, until the server come back to a normal stage. Using such redirectioning scheme, the data networked traffic on server is decreased which alleviates its load. The proposed solution is shown in detail, pointing the modifications in the traditional model.

Key-words: media distribution; *peer-to-peer*; cooperative clients.

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

CAPÍTULO I

1 INTRODUÇÃO

É grande a contribuição que faz a transmissão multimídia. Isso se deve ao fato de que áudio, vídeo e imagens são um meio mais natural para a comunicação humana do que dados alfanuméricos, além de serem mais ricas em informação. Por esses motivos, a comunicação e a computação multimídia possuem aplicações em todas as áreas em que esses tipos de mídias são requeridos.

Encontram-se exemplos de seu uso nas áreas da educação, do entretenimento, da medicina, entre outras. Multimídia permite que sessões de teleconferência sejam abertas entre profissionais distantes para que eles possam trocar informações a respeito de procedimentos em tempo real de uma forma mais eficiente que a simplesmente escrita. Ela também permite que estudantes assistam à palestra de um professor a quilômetros de distância sem sair de sua sala de aula. Com ela, ao se tornarem *on-line*, as rádios podem ir além do que seus transmissores podem alcançar. E pensar que até a pouco tempo a única forma de se distribuir esse tipo de conteúdo era física, via disquetes, CDs, etc.

Contudo, o modelo cliente-servidor, o mais usado na Internet, se mostra inadequado para a transmissão desse tipo de dado. Isso se deve, principalmente, à alta largura de banda requerida para o envio e recebimento de informação multimídia. Com isso o número de clientes que podem ser servidos acaba sendo limitado.

Muito têm se pesquisado para se criar uma forma do modelo cliente-servidor atuar para melhor acomodar o tráfego multimídia. Alguns defendem que nenhum tipo de alteração nos protocolos e serviços é necessário devendo apenas adicionar mais banda. Outros se opõem observando o alto custo de tal atitude, e que em breve surgirão novas aplicações que consumirão toda a banda oferecida, gerando assim um ciclo vicioso de disponibilidade e exaustão de largura de banda.

Uma visão mais interessante é a que pequenas mudanças em pontos fundamentais da comunicação do modelo cliente-servidor são suficientes

para que este acomode a transmissão multimídia. Mudanças essas que permitam ao sistema se adaptar às alterações nas condições de transferência de dados, para conseguir um melhor aproveitamento da banda. É essa abordagem que iremos utilizar.

Neste trabalho discute-se o problema da distribuição de conteúdo multimídia, com foco especial na transmissão de mídia contínua ao vivo, para um grande número de computadores de uma forma escalável. O problema a ser tratado é a sobrecarga do servidor, que pode ocorrer devido à chegada de um número de requisições de recebimento de conteúdo maior do que ele pode suportar. Como conseqüência tem-se o aumento do tempo de resposta e em alguns casos até negação de serviço. Deve ser ressaltado que como se trata de um grande volume de dados a ser transmitido, até mesmo um pequeno aumento no número de requisições pode provocar uma sobrecarga.

Em vista desses problemas, apresentamos um modelo de comunicação que visa complementar o tradicional cliente-servidor. Nesse novo modelo os clientes cooperam entre si para distribuir a mídia como se fossem uma espécie de servidor para os clientes que chegarem durante a sobrecarga do servidor. Os clientes que se dispõem a atender outros clientes são chamados de clientes cooperativos, *peers* servidores ou ainda *servents*. Com o redirecionamento espera-se, reduzir o tráfego no servidor, aliviando a sua carga.

1.1 Motivação

A disponibilidade de computadores multimídia, capazes de processar vários tipos de mídia, tem feito crescer a demanda por aplicações distribuídas multimídia. Mas como a Internet não foi originalmente construída para atender a transmissão de conteúdo multimídia, torna-se um desafio implementar uma solução que faça isso com uma qualidade razoável. Além disso, multimídia envolve um grande volume de dados e a necessidade de transmissão síncrona em tempo real.

Por esses motivos, não é difícil encontrar servidores de mídia na *web* que não conseguem atender eficientemente seus clientes. Um caso especial

desse problema é a sobrecarga devido a um volume de requisições de conteúdo maior do que o suportado por um servidor típico. Um exemplo disso é o que aconteceu durante os ataques terroristas em 11 de setembro de 2001, quando, em busca de informação, vários usuários tentaram, ao mesmo tempo, acessar os *sites* de notícias mais populares da rede, tais como o UOL [Uol 2005] e o Terra [Terra 2005] que acabaram tendo problemas devido à incapacidade de seus servidores lidarem com tamanho volume de requisições. Um outro caso desse tipo, comum atualmente, é o do *site* de comunidades *on-line* Orkut [Orkut 2005], que por não possuir servidores com largura de banda suficiente para atender ao grande de usuários acaba retornando mensagens de erro para suas solicitações em momentos de maior tráfego.

No mercado já é possível encontrar diversos *softwares* capazes de disponibilizar um fluxo de vídeo pela Internet como, por exemplo, o Windows Media Services [WMS 2005], porém eles não conseguem lidar satisfatoriamente com o crescimento de requisições.

1.2 Objetivos

O objetivo deste trabalho é propor uma solução para os problemas causados por um número de requisições do serviço de transmissão de mídia ao vivo muito maior que o servidor pode suportar. Elaboramos um novo modelo de comunicação que visa o trabalho cooperativo entre os clientes. Em específico, tratamos do problema da sobrecarga do servidor e o conseqüente aumento no tempo de resposta do servidor e a recusa de serviço.

1.3 Organização do texto

No capítulo 2 alguns conceitos básicos pertinentes ao estudo de distribuição de conteúdo são vistos.

No capítulo 3 o modelo proposto é apresentado em detalhes. Questões relacionadas ao uso de clientes cooperativos são analisadas.

O capítulo 4 descreve as técnicas de validação e os resultados dos

testes realizados.

No capítulo 5 detalhes de implementação são mostrados. Os problemas encontrados e as soluções adotadas durante esta fase são apresentados.

Por fim, no capítulo 6, são feitas as considerações finais, as conclusões e os trabalhos futuros são apresentados.

CAPÍTULO II

2 CONCEITOS BÁSICOS

Com o intuito de situar o trabalho realizado, neste capítulo estaremos vendo alguns fundamentos que cercam a pesquisa de distribuição de conteúdo. Discutiremos os conceitos inerentes a sistemas multimídia e de comunicação de dados. Isso forma a base para o estudo desenvolvido. Também veremos alguns métodos para analisar a capacidade de máquinas dentro de um grupo, algo essencial para o sistema de balanceamento que estamos propondo. Por fim, veremos alguns trabalhos relacionados.

2.1 Aplicações de Multimídia Distribuída

O avanço e popularização das redes de computadores fizeram surgir ambientes computacionais distribuídos, onde os componentes estão fisicamente dispersos. Entretanto, o desenvolvimento em tais ambientes é mais complexo que em centralizados.

Para facilitar o desenvolvimento de aplicações distribuídas, foram propostas tecnologias de *middleware* para suporte à comunicação. Interpostas entre as aplicações e os sistemas operacionais, elas fornecem uma interface que esconde do programador as operações de baixo nível.

A disseminação do *middleware*, aliada aos avanços na área, tem favorecido o uso de arquiteturas distribuídas para aplicações e serviços. Isto acarretou em uma sofisticação das aplicações e na criação de novas categorias de aplicações, tais como a multimídia distribuída.

As aplicações de multimídia distribuída caracterizam-se pela transmissão e recebimento de diversos tipos de mídia na comunicação entre elementos da aplicação e seus usuários. O uso desse tipo de dado deve-se ao fato dele ser mais rico em informação que dados alfanuméricos. A idéia é emular a comunicação humana, fazendo com que os usuários recebam informação através de pelo menos dois sentidos humanos, tais como visão e audição.

O termo mídia refere-se a tipos de informação ou tipos de portadores de informação, tais como dados alfanuméricos, imagens, áudio e vídeo. Mídia pode ser classificada de diversas maneiras, mas nesse trabalho vamos usar dimensão de tempo para isso.

Temos dois tipos de mídia: estática e dinâmica. A mídia estática não possui uma dimensão de tempo, e seu conteúdo e significado não dependem do tempo de apresentação. São exemplos deste tipo de mídia os dados alfanuméricos, os gráficos e as imagens. A mídia dinâmica possui uma dimensão de tempo e seu significado e precisão depende da taxa na qual são apresentados. Mídias deste tipo têm que ser exibidas continuamente a uma taxa (geralmente fixa), e freqüentemente chamadas de mídia contínua. Este tipo de mídia também é referido por alguns autores como mídia isócrona devido à relação fixa entre cada unidade de mídia e tempo.

Aplicações nessa categoria possuem alguns requisitos adicionais não presentes nas aplicações mais tradicionais, como o suporte a *streams*, essenciais à transmissão de mídia contínua, e a sincronia, tanto da informação transmitida por cada mídia (intramídia), quanto entre as diferentes mídias utilizadas (intermídia).

2.2 *Stream* (fluxo) de Mídia

O uso de mídia contínua na comunicação entre elementos distantes requer que as transferências sejam ininterruptas por períodos relativamente longos, além de que uma relação temporal entre os elementos de dados transmitidos sucessivamente seja obedecida.

Na transmissão de voz, como uma conversa de telefone, o sinal é codificado como uma seqüência de pacotes. Nesse cenário, a freqüência em que os pacotes são enviados deve ser mais ou menos constante para que quando o sinal for reconstituído do outro lado não haja distorção. Se cada pacote representar 200 milisegundos de áudio, a taxa média de chegada de dados deve ser de 5 pacotes por segundo (o intervalo máximo de chegada entre um pacote e outro deve ser de 200 milisegundos) para não haver interrupção no som.

O modelo de comunicação baseado em chamada de procedimento não é capaz de atender a estes requisitos. Tanto a chamada síncrona quanto à assíncrona são voltadas à transmissão de unidades discretas de informação, sem qualquer relação ou dependência entre os dados.

O *streaming* é uma técnica para transferência de dados usada para transmitir áudio e/ou vídeo através da rede, onde toda informação pode ser processada logo após o seu recebimento. O sincronismo entre a taxa de recebimento e a taxa de processamento garante o fluxo correto da mídia. Normalmente, os dados recebidos são descartados quando não mais necessários.

Até o seu surgimento, a transmissão de mídia não ocorria em tempo real, os vídeos precisavam ser inteiramente transferidos pela rede (e escritos em uma unidade de armazenamento local) antes de serem visualizados.

O *streaming* de mídia vem se tornando uma "*killer application*" (aplicação de grande utilidade e demanda). Isso porque elas não só têm demanda comercial, oferecendo oportunidades de treinamento e transmissão eficaz de informação, como também, aumentam a procura por maior largura de banda e elevam os níveis de qualidade de serviço. Sistemas de vídeo conferência e televisão digital são alguns exemplos de sua utilização.

2.2.1 Streaming Progressivo

O *streaming* progressivo (*progressive streaming*), também conhecido como *webcasting*, refere-se à mídia que usuários podem assistir à medida que os arquivos são copiados. Neste método o usuário pode ver a parte do arquivo já copiada, mas não pode saltar para porções à frente que ainda não foram transferidas. Considera-se que os dados não são mais necessários quando o reprodutor é fechado. Um exemplo deste tipo de streaming é o QuickTime [Quicktime 2004].

Alguns autores também se referem ao *streaming* progressivo como HTTP *streaming*, pois servidores HTTP padrão podem ser usados para distribuir mídia desta maneira, sem o envolvimento de nenhum protocolo especial. Servidores FTP também podem ser usados.

Esta abordagem é útil para computadores com pouca banda. Ela permite que o cliente armazene grande parte do vídeo antes de visualizá-lo. Com isso é possível obter um vídeo de qualidade superior à que um modem comum consegue obter em tempo real. Porém, faz com que os usuários experimentem um grande atraso antes da reprodução começar.

O *streaming* progressivo adequa-se muito bem a vídeos de tamanho pequeno, conseguindo garantir uma alta qualidade a eles. Este método garante a qualidade do filme, pois a porção vista do arquivo é copiada com pouquíssima perda antes de ser reproduzida.

Contudo, este modo de *streaming* não é uma boa solução para filmes de longa duração ou que precisem de acesso aleatório. Além disso, ele não permite adaptação de conteúdo.

2.2.2 Streaming em Tempo Real

Utilizado pelo Windows Media Player [WM 2005], o *streaming* em tempo real (*real-time streaming*) refere-se a tecnologias que mantêm a taxa de reprodução em sincronia com a de recepção. Por isso, é recomendado para transmissões ao vivo (mídia aberta).

Uma vantagem desta abordagem é que ela permite acesso aleatório ao conteúdo permitindo pular para trechos ainda não recebidos. Contudo, para rever alguns trechos, eles devem ser copiados novamente, pois eles são descartados a medida em que são exibidos.

Outra vantagem do *streaming* em tempo real é o suporte à adaptação do conteúdo. Isso permite ajustar a qualidade da mídia transmitida de acordo com a banda do receptor, inclusive durante a transmissão. Por exemplo, se alguma alteração na velocidade de recebimento for detectada, o sistema pode reduzir a qualidade da mídia para melhor acomodá-la às novas condições.

Contudo, este método também tem seus problemas. Quando a rede está congestionada ou com algum tipo de problema que provoque uma perda ou mesmo corrupção dos pacotes a qualidade do vídeo deve sofrer uma grande queda. A informação perdida em face a erros na transmissão é ignorada.

Este mecanismo de *streaming* requer um tipo especial de servidor, mais complexo que o HTTP, mas que dá maior controle sobre a transmissão da mídia. Além disso, geralmente usam protocolos especiais como o *Real-Time Streaming Protocol* [RTSP 1998].

2.3 O Padrão MPEG

Tipicamente, para se representar áudio e vídeo em forma digital é preciso um grande montante de dados, o que aumenta os requisitos de largura de banda para a transmissão desse tipo de conteúdo. Por isso, diversas tecnologias para compactação de dados foram desenvolvidas. Essas tecnologias conseguem altas taxas de compactação com qualidade de áudio e vídeo aceitáveis.

Contudo, apenas a compactação não é suficiente, também é preciso que haja uma padronização, que possibilita a interoperabilidade entre aplicações e equipamentos de diferentes fabricantes. Assim, em 1988 o *Joint* ISO/IEC *Technical Commitee on Information Tecnology*, encarregado de desenvolver padrões de representação de figuras em movimento e áudio associado, criou o formato MPEG. Hoje, a sigla MPEG designa o grupo que trabalha com esse formato [MPEG 2005].

Inicialmente, o grupo tinha três idéias, que foram chamadas de MPEG-1, MPEG-2 e MPEG-3. O MPEG-1 tem como objetivo codificar vídeo em qualidade VHS (360 x 280 *pixels* a uma taxa de 30 figuras por segundo) a uma taxa de bits (*bit rate*) em torno de 1.5 Mbps. Já o MPEG-2 tinha a intenção de codificar vídeo de qualidade de televisão digital CCIR 601 (720 x 480 pixels a uma taxa de 30 figuras por segundo) a um bit rate de 2 a 10 Mbps. Agora, o MPEG-3 deveria codificar vídeo com qualidade HDTV a um bit rate em torno de 40 Mbps, contudo, como os requisitos do MPEG-2 cobrem os do MPEG-3, ele acabou sendo abandonado e mais tarde, um novo formato foi proposto, o MPEG-4. Há ainda o MPEG-7 (oficialmente conhecido como *Multimedia Content Description Interface*) que diferentemente dos outros padrões, define a descrição do conteúdo e não os métodos de codificação para compressão de dados.

2.3.1 MPEG Systems

O formato MPEG-2, extensão do MPEG-1, é composto de diversos padrões, sendo que os mais conhecidos são o ISO/IEC 13818-1 [ISO 1996a] para sistemas MPEG (MPEG *Systems*), o ISO/IEC 13818-2 [ISO 1996b] para codificação de vídeo, o ISO/IEC 13818-3 [ISO 1998a] para áudio com compatibilidade regressiva e o ISO/IEC 13818-6 [ISO 1998b] para comando e controle de armazenamento de mídia digital.

O padrão 13818-1 para sistemas de áudio e vídeo MPEG-2 define as codificações PS (*Program Stream*) e TS (*Transport Stream*). A diferença entre as duas é que a primeira é voltada para a utilização em ambiente controlado e livre de erros, enquanto que a segunda é preparada para ser transportada em redes, carregando informações úteis em ambientes onde há ruído ou perda de pacotes. Claramente, essas codificações foram desenhadas para diferentes tipos de aplicações.

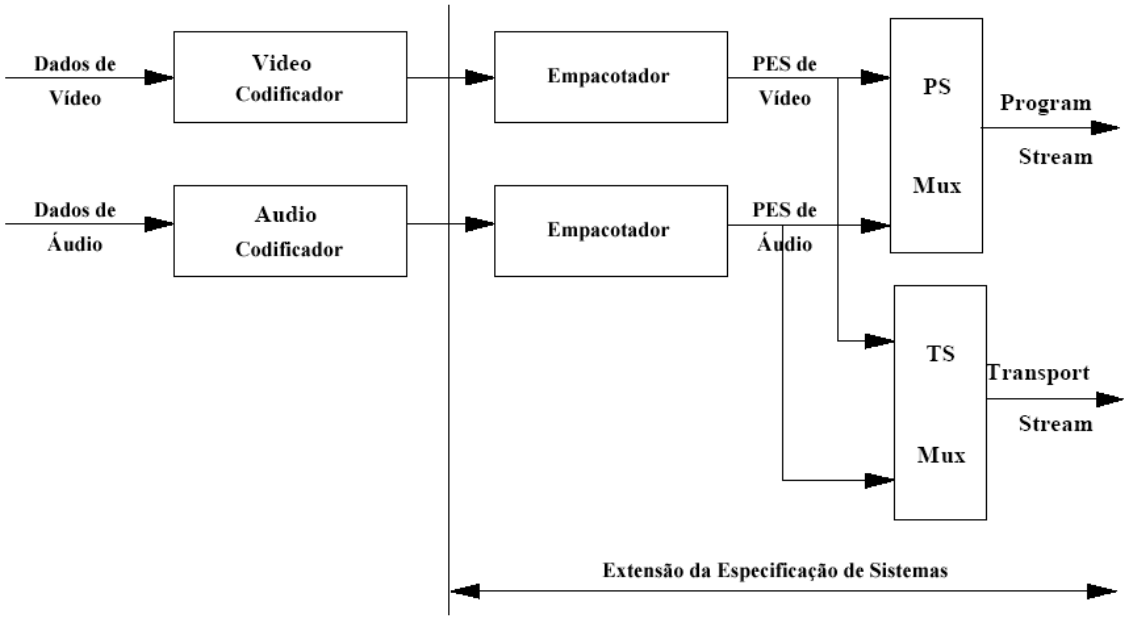

Figura 1 - Visão simplificada da codificação TS e PS [ISO 1996a]

Tanto a codificação TS quanto a PS são formatos de multiplexação de pacotes de fluxos de áudio e vídeo. Na figura 1 é possível ver a abordagem

básica para a multiplexação dos pacotes. No exemplo, têm-se as codificações de áudio e vídeo sendo realizadas independentemente uma da outra, gerando dois fluxos: um de áudio e outro de vídeo. Cada um desses fluxos passa por um empacotador que gera os pacotes PES (*Packetized Elementary Streams*). Esses pacotes podem gerar tanto *streams* de codificação TS quanto PS após a multiplexação. *Streams* elementares (Elementary Streams - ES) é um termo genérico para vídeo, áudio ou qualquer outro tipo de *bitstream* codificado em pacotes PES.

Um diagrama da estrutura de um pacote PES pode ser visto na figura 2. Um pacote PES é composto de vários campos, mas os mais importantes são os três primeiros: prefixo de código de inicio de pacote (*packet start code prefix*), identificador de *stream* (*stream id*) e tamanho do pacote (PES *packet length*). O primeiro é prefixo que indica o início de um pacote, o segundo é um byte que identifica a *stream* e o terceiro diz o tamanho do pacote. Seguindo esses campos há um cabeçalho com alguns campos opcionais e após eles vêm os dados do pacote.

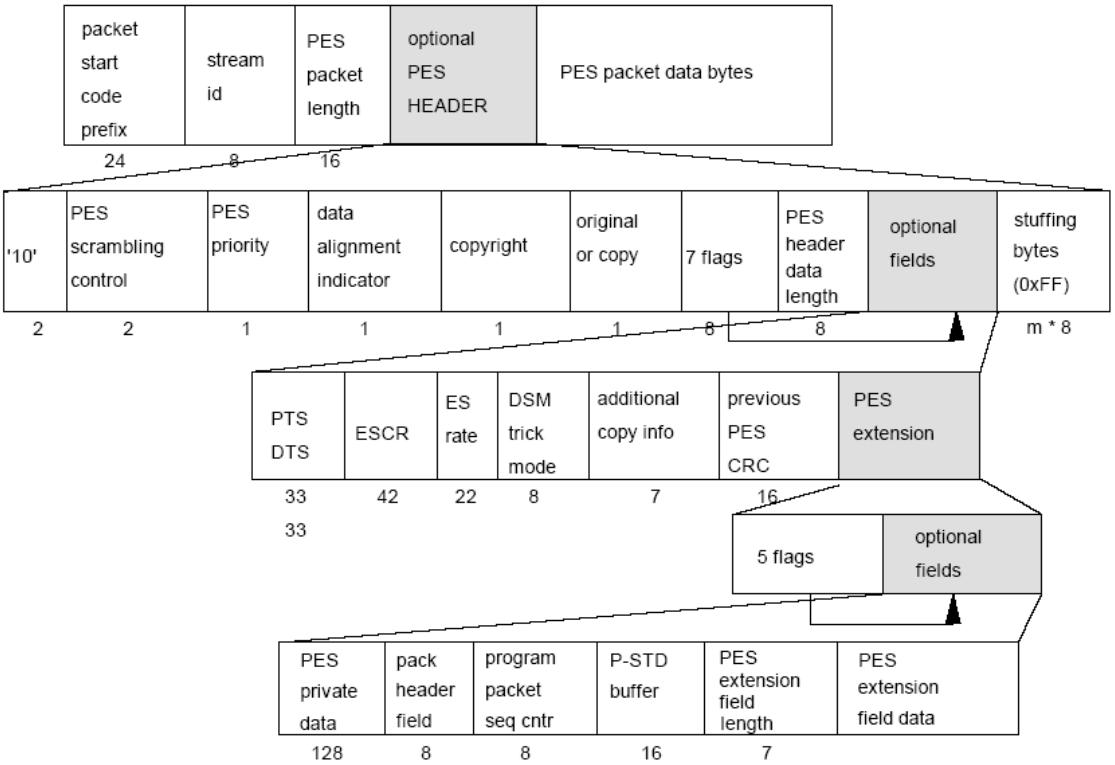

Figura 2 – Estrutura de um pacote PES [ISO 1996a]

Como as *streams* TS e PS são constituídas de pacotes PES é possível realizar conversões entre as duas codificações. Na maioria dos casos, os pacotes PES podem ser extraídos diretamente do bloco de dados de uma *stream* de *bits* multiplexada para outra. Certas informações como a relação entre as *Elementary Streams*, necessárias para a conversão, estão disponíveis em ambas *streams*.

2.3.2 Transport Streams

Um pacote TS possui tamanho fixo (188 bytes) e é composto de um cabeçalho e um bloco de dados (*payload*) ou campo de adaptação (*adaptation field*). Cada pacote pode carregar informações de áudio ou vídeo, ou de referência de tempo, ou descrição da *stream*, ou ainda, quaisquer outros tipos de dados que possam ser encapsulados no bloco de dados.

O cabeçalho, como pode ser visto na figura 3, inicia com um byte de sincronização (*sync byte*) que possui um valor fixo (0x47) que serve para indicar o início de um pacote TS. A seguir vem um bit chamado Indicador de Erro de Transporte (*Transport Error Indicator*) que indica a ocorrência de pelo menos um erro irrecuperável. Um outro campo de um bit vem logo em seguida Unidade Indicadora de Início de Bloco de Dados (*Payload Unit Start Indicator*), se ele possuir o valor 1, o primeiro byte do bloco de dados é o primeiro byte de um pacote PES, senão nenhum pacote PES inicia com esses dados. O outro bit que vem a seguir, quando possuir o valor 1, indica que este pacote tem prioridade maior que os outros de mesmo PID (*Packet Identifier*), que é o campo que vem seguida e indica o tipo de dado que o bloco de dados carrega. Um outro campo chamado contador de continuidade (*Continuity Counter*) informa o número de pacotes com o mesmo PID incluindo o atual. O controle de campo de adaptação (*Adaptation Field Control*) indica se o pacote é seguido por um campo de adaptação ou bloco de dados.

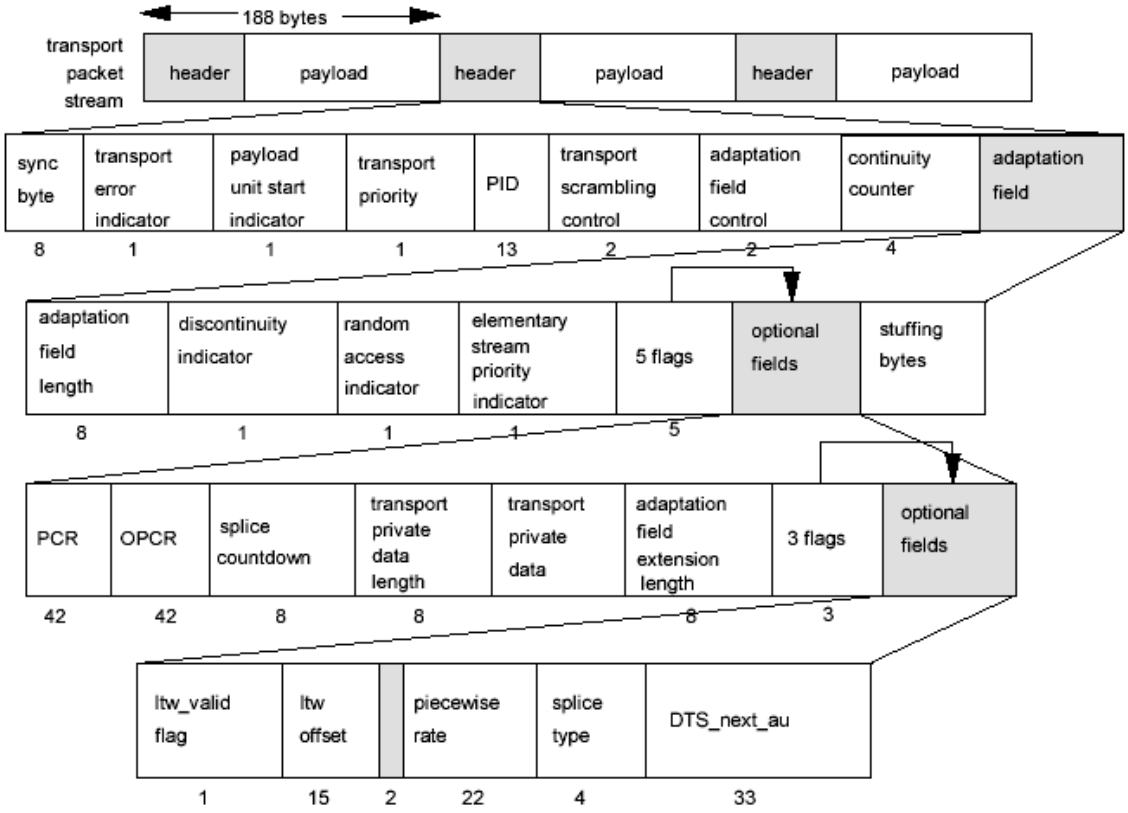

Figura 3 – Estrutura de um pacote TS [ISO 1996a]

O campo de adaptação contém informações a respeito de como o programa deve decodificar e reproduzir o vídeo. Tal campo deve possuir o Relógio de Referência do Programa (*Program Clock Reference* - PCR), utilizado para sincronizar as operações de decodificação e reprodução. Um outro exemplo é o campo Indicador de Descontinuidade (*Discontinuity Indicator*), um único bit que aponta uma descontinuidade de tempo ou do contador de continuidade.

O bloco de dados corresponde aos dados que vêm logo após o cabeçalho. O campo de adaptação não é considerado um bloco de dados. No caso do pacote TS, o bloco de dados carrega os dados de um pacote PES ou um campo ponteiro (*field pointer)* e sessões PSI (*Program Specific Information*), ou ainda, dados privados. O campo ponteiro é um campo de 8 bits que indica o número de *bytes* até o início de uma nova sessão. A sessão PSI é uma sessão que contém informações necessárias para a demultiplexação das *Transport Streams* e recuperação de ES ou qualquer outro tipo de *stream* de dados.

2.3.3 Program Streams

As *Program Streams* são constituídas de pacotes cujo cabeçalho é chamado de *pack header* e o bloco de dados é conhecido como *pack*. Os campos do *pack header* guardam informações que facilitam algumas tarefas do processo de decodificação e reprodução da mídia. Todos esses campos podem ser vistos na figura 4.

O *pack header* é iniciado com um campo de 3 *bytes* conhecido como Código de Início de *Pack* (*Pack Start Code*), que, como o próprio nome diz, indica que a partir daquele ponto inicia um *pack*. Esse campo é seguido de 2 bits de valor fixo (01) e uma Referência de *Clock* de Sistema (*System Clock Reference* – SCR). A taxa de multiplexação de programa (*Program Mux Rate*) é informada por um campo de 22 bits. Há ainda um campo chamado Tamanho do Preenchimento do *Pack* (*Pack Stuffing Length*) que indica o número de *bytes* de preenchimento (*Pack Stuffing Byte*) posicionados imediatamente após ele.

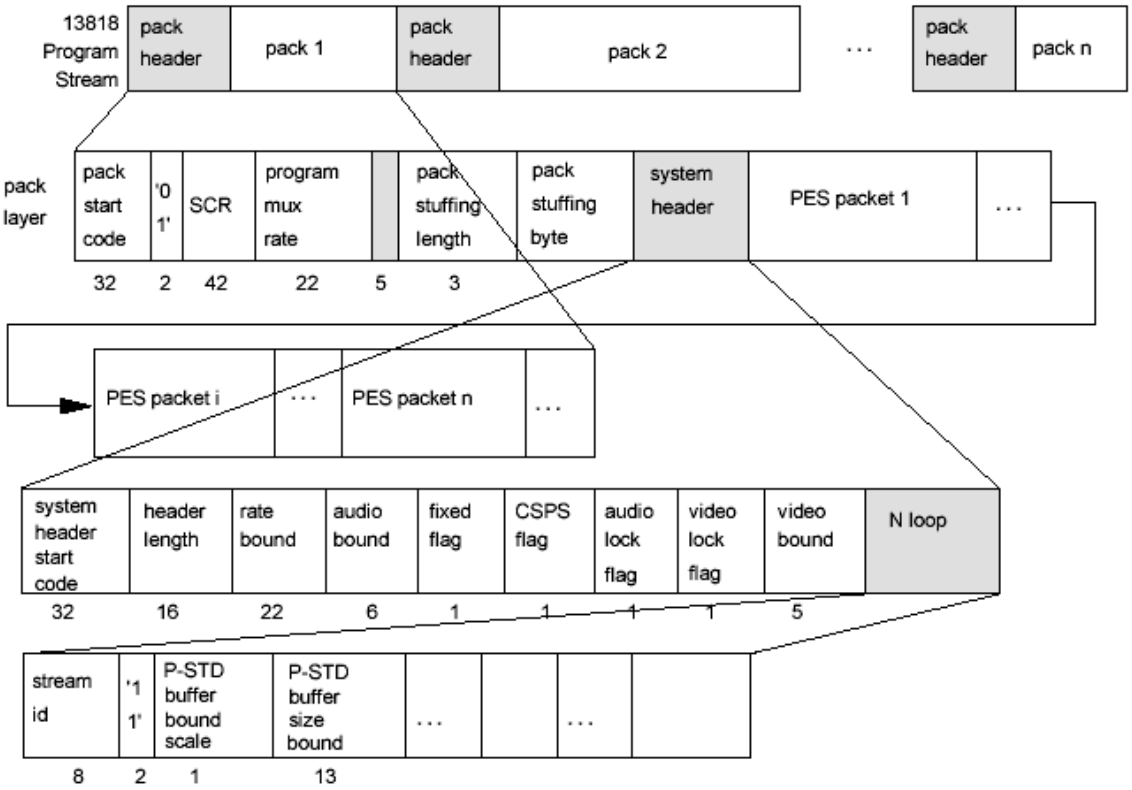

Figura 4 - Estrutura de um pacote PS [ISO 1996a]

Os pacotes das *Program Stream*s carregam um Cabeçalho de Sistema (*System Header*) que é seguido por zero ou mais pacotes PES. Ele inicia com um inteiro de 4 *bytes* com o valor de 0x000001BB. O tamanho do cabeçalho é indicado logo em seguida por um inteiro de 2 *bytes*. O campo limite (*rate bound*) e o limite de áudio (*audio bound*) indicam o maior valor para o taxa de multiplexação de programa em qualquer *pack* da *Program Stream* e o maior valor para as streams de áudio, respectivamente. O bit que vem logo a seguir indica uma taxa de bit fixa ou não. Os outros dois bits a seguir informam se há ou não uma relação entre as amostras de áudio e a freqüência do *clock* do sistema apontada pelo decodificador alvo, e entre a taxa de imagens de vídeo e a freqüência do *clock* do sistema, respectivamente. O campo chamado limite de vídeo (*video bound*) indica o número máximo de *streams* de vídeo. Identificador de *Stream* indica a codificação e o número de *streams* elementares aos quais os campos que vêm logo a seguir se referem. O bit escala do limite do *buffer* P-STD (P-STD *buffer bound scale*) indica a escala para o valor do campo limite do tamanho do *buffer* P-STD (P-STD *buffer size bound*) que informa o tamanho máximo para o *buffer* P-STD. P-STD é o Decodificador da *Program Stream* Alvo (*Program System Target Decoder*).

2.4 Modelos de Comunicação

Quando há diversas entidades que desejam comunicar-se, torna-se necessária adoção de um protocolo comum de comunicação. É o que acontece num debate televisivo, onde há um moderador que dá a palavra a cada um dos participantes através de um conjunto de regras pré-estabelecidas, ou numa conversa via rádio, em que cada parte fala alternadamente, ou numa entrevista, em que um interlocutor coloca questões e outro limita-se a responder. Todas essas estruturas se baseiam em um mecanismo de pergunta e resposta.

Os modelos para troca de dados em uma rede seguem essa estrutura entrevistador-entrevistado. A comunicação ocorre apenas entre duas entidades. Há sempre uma entidade que toma a iniciativa, enquanto a outra espera pelo pedido. A entidade que recebe o pedido responde a um conjunto perguntas pré-definidas, e que constituem a interface do entrevistado.

2.4.1 Cliente-Servidor

No modelo de comunicação cliente-servidor [Tanenbaum 2003] cada computador pode ser configurado como cliente ou servidor. Os servidores são máquinas poderosas dedicadas a oferecer algum tipo de serviço, tais como armazenamento de arquivos (servidores de arquivos), impressão (servidor de impressão) ou de acesso à rede (servidores de rede). Clientes são computadores nos quais os usuários executam seus programas e solicitam serviços aos servidores.

Comparando com a estrutura entrevistador-entrevistado, o entrevistado pode ser visto como a entidade que presta serviço, sendo que para cada pedido há uma resposta definida, dai a designação servidor. O entrevistador é o cliente, que usa os serviços.

O funcionamento dessa estrutura é baseado em um protocolo do tipo solicitação/resposta. Primeiramente, o cliente envia uma mensagem contendo a solicitação de um serviço do servidor, como por exemplo, o valor de um determinado campo no banco de dados do sistema descrito anteriormente. Em seguida, o servidor executa o serviço associado ao pedido do cliente. Por fim, o servidor devolve ao cliente uma mensagem contendo a resposta ao pedido (o resultado do serviço requerido pelo cliente).

Neste modelo o cliente é quem sempre toma a iniciativa, enquanto que, o servidor, fica passivamente esperando pelos pedidos. Dois fatos devem ser ressaltados: um é que o modelo não se preocupa com o modo como a comunicação é executada, outro é que os pedidos permitidos (serviços oferecidos) são descritos pela interface do servidor.

2.4.2 Peer-to-peer

Popularizado pela sua utilização em redes de compartilhamento de arquivos, o modelo *peer-to-peer* [Schollmeier 2002], freqüentemente referido como P2P, é um modelo de comunicação “cliente-cliente”, no qual cada computador conectado à rede, comumente chamado de *peer* (par), possui equivalentes

capacidades e responsabilidades.

Em um primeiro momento a definição pode parecer um tanto quanto obscura, mas há uma grande diferença entre o modelo cliente-servidor e o P2P. No primeiro há uma clara distinção entre a parte que provê e a que usa o serviço. Já no segundo, não existe uma definição fixa entre clientes e servidores. Um outro fato que difere os dois modelos é que no P2P não há um ponto central em suas redes, por isso são geralmente simples. Isso quer dizer que ele não possui um ponto central de falhas.

O *peer-to-peer* possibilita que as pessoas possam fazer conexões diretas para interação em tempo real. Essa interação pode ser uma simples conversa via mensagens instantâneas, ou uma teleconferência, ou mesmo um compartilhamento de arquivos.

Contudo, o P2P tem os seus problemas. Ele não oferece o mesmo desempenho do cliente-servidor. Em geral a localização do recurso necessário é custosa, pois exige a busca em diversos nós, ao passo que no cliente-servidor, basta procurar no servidor.

Diversas aplicações implementando esse modelo foram propostas, sendo que as mais famosas são o Napster, o Gnutella e o BitTorrent. As três aplicações são vistas nas sessões seguintes.

2.4.3 Napster

Criado pelo americano Shawn Fanning, o Napster [Napster 2003] é um sistema para troca de arquivos de forma direta entre a fonte e o destino. O sistema é limitado, pois não permite reinício de *download*, cópia a partir de múltiplas fontes e distribuição do conteúdo sendo copiado. Mas isso não impediu a sua popularização. Ele atingiu mais de 50 milhões de usuários registrados para distribuir arquivos pela rede. Os arquivos copiados podem ser redistribuídos para outros usuários, gerando a maior rede de distribuição já vista.

Os participantes, para localizar os arquivos que desejam copiar, dependem de uma consulta a um servidor central - na verdade um cluster de servidores - onde ficam armazenadas listas de arquivos que cada usuário possui.

Para se conectar ao sistema, o usuário deve primeiro se conectar ao servidor central e fornecer a ele a lista dos arquivos que estará compartilhando. Dessa forma, o servidor monta um banco de dados com os nomes dos arquivos sendo compartilhados na rede para consulta. Eventualmente um arquivo disponibilizado anteriormente poderá ser listado como uma resposta para uma busca de algum usuário que poderá optar por iniciar a sua transferência. Na figura 5 é mostrada como toda essa rede opera.

Os arquivos são transferidos via HTTP entre fonte e origem sem intervenção do servidor. Um detalhe é que o servidor não armazena nenhuma cópia do arquivo, só a lista.

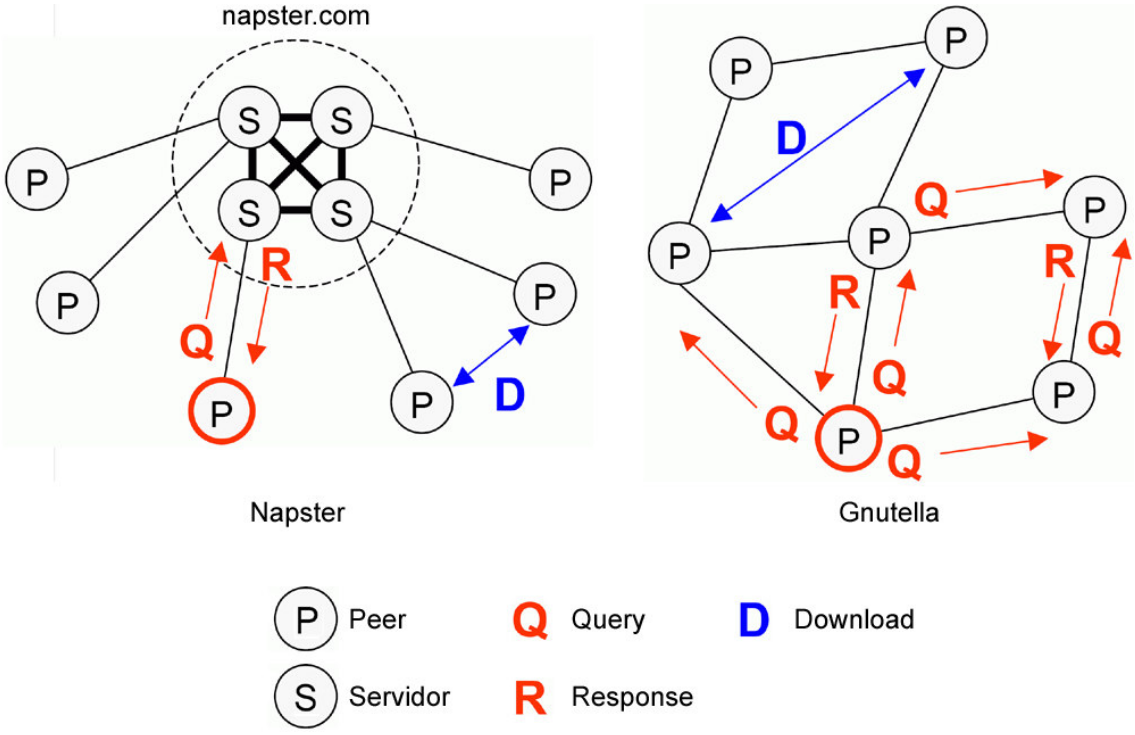

Figura 5 - Redes Napster e Gnutella [Rocha 2003]

2.4.4 Gnutella

O Gnutella [Gnutella 2004] diferencia-se do Napster pelo fato de ser um P2P puro, ou seja, não depende de qualquer tipo de servidor. No Napster embora a transferência dos arquivos ocorra de forma direta, as buscas são

respondidas no modelo cliente-servidor tradicional. Isso fica evidente na figura 5 onde há uma visão dessas duas redes.

No Gnutella, cada *servent*, nome dado ao *peer* nesta rede, mantém um banco de dados local para armazenar listas de arquivos que os outros usuários estão disponibilizando, além de ser responsável por prover uma lista de *servents* próximos. Isso elimina a necessidade de um servidor central. O nome servent é surgiu da união dos primeiros caracteres da palavra *server* e dos caracteres finais da palavra *client*.

Nesse sistema, cada *servent* pode atuar como cliente ou servidor. Eles possuem uma interface servidora para responder buscas e uma interface cliente para realizar suas próprias buscas. Como resultado, uma rede de *servents* Gnutella é altamente redundante e o serviço não será interrompido mesmo se vários nós estiverem desligados ou com problemas.

Quando deseja entrar na rede, um *servent* estabelece conexão com outro *peer* já conectado. Para adquirir o endereço de um *host* ativo na rede, normalmente são utilizados arquivos *cache* contendo dados de *peers* encontrados em conexões anteriores, ou se for a primeira conexão após a instalação do cliente, endereços de *peers default*. Feito isso, uma mensagem CONNECT é enviada para o *servent* e uma conexão TCP é aberta. Se o *servent* aceitar o novo *peer*, uma mensagem OK será recebida. Através dessa conexão, o usuário pode ir a qualquer outro membro, simplesmente inspecionando a lista de *servents* de cada um dos *peers* conhecidos.

Assim como o Napster, inicialmente esse sistema também era limitado a uma conexão para cada *download* e não tinha recurso de retomada de transmissão interrompida. Também possui a restrição de distribuição de somente arquivos completos.

2.4.5 BitTorrent

O BitTorrent (BT) [BitTorrent 2004] otimiza o uso da banda com o princípio “eu te ajudo se você me ajudar”. Ele permite que usuários que estão fazendo o *download* de um mesmo arquivo, ao mesmo tempo, possam se ajudar

compartilhando os pedaços que já possuem.

A busca é feita com a ajuda de *websites* especializados em localizar arquivos .torrent. Esses arquivos guardam o nome, tamanho e dados de *hashing*, além da URL de um rastreador (*tracker*). O rastreador é o responsável por ajudar os *downloaders* a encontrar uns aos outros.

Após iniciar o *download* o usuário envia informações ao rastreador para que este lhe informe o endereço de outros *peers* que estejam copiando o mesmo arquivo, e assim localizar mais fontes de *download*. Dentre as informações enviadas para o rastreador estão o nome do arquivo sendo copiado e qual porta está sendo usada. Essa comunicação realizada entre os usuários e o rastreador é feita através de um protocolo bem simples construído sobre o HTTP.

Para que possam ser distribuídos mais facilmente, os arquivos são divididos em pedaços, os quais, para facilitar a identificação, possuem um tamanho fixo. Cada *peer* informa aos demais qual parte do arquivo ele possui, assim, quem está fazendo *upload* também pode obter o conteúdo daqueles que estão copiando. A integridade dos dados é garantida por *hashing* SHA1, sendo que os *hashes* de todos os pedaços são incluídos no arquivo .*torrent* e nenhum *peer* informa que tem uma determinada parte do arquivo sem antes checar o *hash*.

No BitTorrent cada *peer* é responsável por maximizar sua taxa de recepção. Isso é feito copiando de quem for possível e limitando para quais *peers* enviar. Essa decisão de enviar ou não é feita com o algoritmo de *choking*. *Choking* é uma recusa temporária de *upload*. O *peer* pára o *upload*, mas o *download* ainda pode ser feito. Quando a taxa de *download* volta ao normal uma operação de *unchoke* é executada para reativar o *upload*.

Para que essa rede possa funcionar, alguém precisa disponibilizar o (primeiro) arquivo completo. Esse arquivo é o início de tudo, por isso ele é conhecido como semente (*seed*). Clientes que disponibilizam cópias inteiras do arquivo são chamados de *seeders* (semeadores).

Um exemplo de uma configuração para a rede BitTorrent com 3 semeadores e 3 clientes está na figura 6. Os semeadores são os círculos com retângulos coloridos preenchidos. Cada coluna preenchida desse retângulo representa um bloco do arquivo que o *peer* possui, sendo que cada bloco é

representado por uma cor. Os pontos que ligam um *peer* ao outro representam o bloco sendo transmitido.

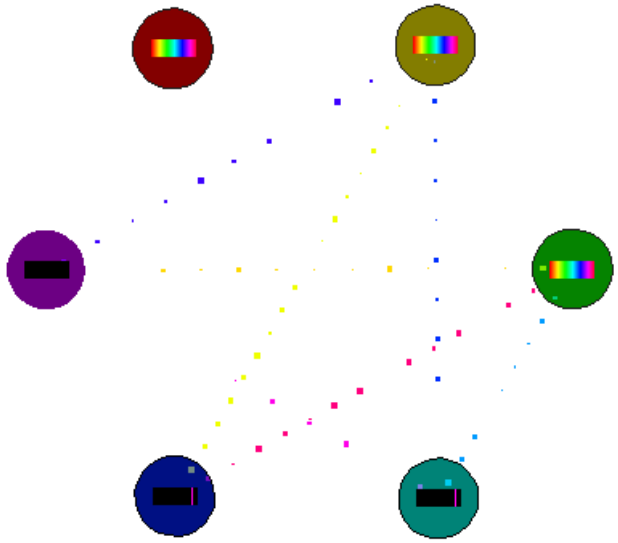

Figura 6 - Rede BitTorrent

2.5 Estimativa de Características dos Nós da Rede

Conhecer a velocidade com a qual um determinado elemento da rede pode transmitir é de grande importância para vários tipos de aplicações. O conhecimento de tal informação pode ajudar a melhorar o desempenho de aplicações baseadas em taxa de recepção de dados, sistemas de controle de admissão, mecanismos de seleção de rotas, controle de congestionamento e adaptação de conteúdo.

Por isso, diversos trabalhos foram desenvolvidos na tentativa de desenvolver técnicas para estimar a capacidade atual e largura de banda disponível. Por capacidade atual entenda-se a largura de banda de um link apresenta no momento e por largura de banda disponível entenda-se a taxa de transmissão máxima que um novo fluxo pode ser iniciado sem reduzir a velocidade dos já existentes. Nas subseções seguintes são abordados alguns dessas técnicas.

2.5.1 Estimativa de Capacidade

A maioria dos mecanismos usados para estimar a capacidade de um elemento da rede utiliza a técnica de pares de pacotes. Nessa técnica os pacotes de dados são transmitidos em pares, assim caso haja um ponto entre a origem e o destino que apresente a menor capacidade de transmissão (gargalo da rede), ele irá provocar um espaçamento (dispersão) entre os pacotes. Esse espaçamento pode ser medido e utilizado para inferir a largura de banda entre origem e destino.

Entretanto, numa rede real, o tráfego concorrente dificulta a utilização de pares de pacotes de forma precisa, especialmente quando se utilizam filas do tipo FIFO, que é a mais utilizada na Internet. Para solucionar esse problema, foi proposta uma modificação incluindo a transmissão de vários pacotes enfileirados (trem de pacotes ou *packet train*), surgindo assim o bprobe [Carter 1996].

Outra forma de contornar o problema do tráfego concorrente é o uso de técnicas de filtragem para descartar amostras, com faz o nettimer [Lai 2000]. Ele usa um mecanismo chamado "*packet tailgating*", onde o primeiro pacote do par é um ICMP com TTL reduzido, sendo encaminhado juntamente com outro de tamanho pequeno, fazendo com que o segundo pacote seja sempre enfileirado junto com o primeiro, até o descarte do primeiro pelo roteador, quando seu TTL expira.

Uma alternativa ao sistema de pares e de trem é a técnica empregada em algumas ferramentas como pathchar [Downey 1999a], clink [Downey 1999b] e pchar [Mah 2000], onde a capacidade do link é inferida através da relação do tamanho do pacote e atraso. A idéia é calcular a capacidade do link através da variação do atraso à medida que se aumenta o tamanho do pacote. É a chamada técnica de um pacote (*one-packet technique*) [Lai 2000].

Nas ferramentas citadas, pacotes UDP são enviados sucessivamente incrementando o valor do tempo de vida do pacote para receber mensagens ICMP de tempo excedido dos roteadores. Os atrasos de recebimento de resposta desses pacotes são coletados e relacionados com os respectivos

tamanhos de pacote, e através de regressão linear, é obtida a aceleração da reta do gráfico formado por esses dados (figura 7). A inversa da aceleração representa a largura de banda.

Essa abordagem tem a vantagem de não requerer que nenhum tipo especial de software seja desenvolvido para os roteadores, contudo a sua aplicabilidade acaba sendo limitada, pois alguns roteadores, devido ao uso mal intencionado de mensagens ICMP, filtram esse tipo de mensagem, o que produz medições imprecisas. Além disso, como a geração de respostas ICMP requer o processamento de diversos caminhos em um roteador, a capacidade do *link* acaba sendo subestimada [Dovrolis 2001].

Um outro problema da técnica de um pacote é que uma porção considerável da banda da rede é consumida para efetuar as medições e todo o processo de calculo de capacidade pode acabar sendo muito lento para alguns tipos de aplicações.

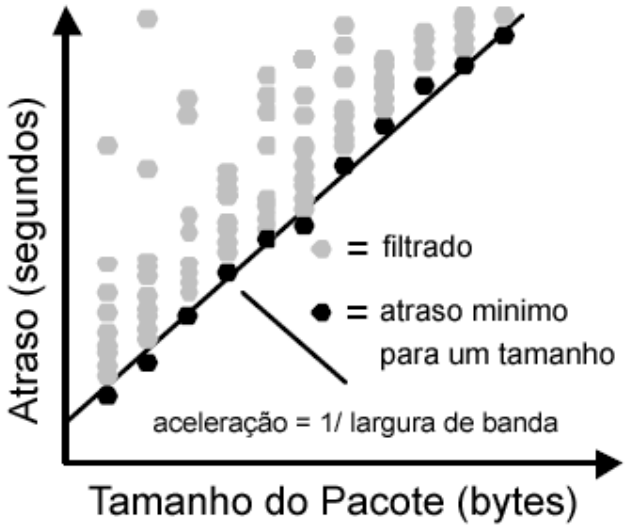

Figura 7 - Obtenção da largura de banda através do método de um pacote [Lai 2000]

2.5.2 Estimativa de Largura de Banda Disponível

De acordo com [Lakshminarayanan 2004], há duas classes de técnicas para estimativa de largura de banda disponível, a baseada em modelos estatísticos de tráfego concorrente, conhecida como PGM (*Packet Gap Method*), e

as baseadas em congestionamento auto-induzido, chamada de PRM (*Packet Rate Method*).

No PGM pares de pacotes de mesmo tamanho são enviados, espaçados de acordo com o tempo de transmissão. Se nenhum tráfego concorrente se interpor entre os pacotes, o espaço utilizado no envio será preservado. Caso contrário, o espaçamento sofrerá um aumento e com essa variação é possível calcular o volume de tráfego concorrente. A banda disponível é calculada subtraindo o tráfego concorrente da capacidade. Esse método é utilizado com sucesso pelas ferramentas Spruce [Strauss 2003] e Delphi [Ribeiro 2000].

As técnicas PRM utilizam um princípio bem simples. Se os pacotes forem enviados a uma taxa superior à da banda disponível entre origem e destino, os mesmos serão colocados em uma lista de espera em algum roteador ao longo do caminho, resultando no aumento do tempo de transferência. Por outro lado, se a taxa de envio for mais baixa, os pacotes serão recebidos, em média, na mesma freqüência que o envio. A banda disponível é a taxa de envio apresentada no momento em que a fila nos roteadores começa a ser formada. As ferramentas pathload [Jain 2002] e pathChirp [Ribeiro 2003] usam este método.

2.6 Trabalhos Relacionados

Muitos esforços têm sido feitos no sentido de se criar novas abordagens para a distribuição de conteúdo. Os novos trabalhos nessa área se dividem basicamente em três grupos: a distribuição centralizada de conteúdo, a distribuição de conteúdo baseada em infraestrutura e o popular *peer-to-peer.*

Na distribuição centralizada utilizando o modelo cliente-servidor, um computador é configurado para servir mídia para aqueles que a ele requisitarem conteúdo. É uma arquitetura simples e de fácil implementação, mas apresenta problemas de escalabilidade.

As redes de distribuição de conteúdo (CDN - *Content Distribuition Network* ou *Content Delivery Network*) [Wikipedia 2005] [Pc-News 2005] são baseadas em infra-estrutura e empregam um conjunto de máquinas dedicadas a armazenar e ajudar o servidor a distribuir conteúdo para os clientes. Os nós de uma

CDN cooperam entre si para atender o usuário de forma satisfatória, movendo transparentemente o conteúdo para outras localizações de forma a otimizar o processo de distribuição. É uma excelente forma de se garantir alta disponibilidade e um bom desempenho. Contudo, devido ao seu custo, pequenos provedores podem não estar em posição de implementar toda a infra-estrutura.

No *peer-to-peer*, modelo de comunicação que ganhou fama com programas de distribuição de arquivos como o Napster e o Gnutella, a proposta é utilizar os clientes para armazenar e distribuir o conteúdo para outros clientes. Tipicamente, não possui um servidor central que hospeda o conteúdo, e ao contrário do que acontece com a baseada em infra-estrutura substitui o modelo cliente-servidor.

O SpreadIt [Deshpande 2001] foi um dos primeiros trabalhos de *peer-to-peer* envolvendo a transmissão de vídeo. Nessa arquitetura, uma abordagem voltada às transmissões ao-vivo, o *stream* é transmitido aos clientes através de uma árvore *multicast* implementada ao nível de aplicação. Na árvore, cada nó possui uma camada de *peering* responsável pela manutenção da árvore. Contudo, descobriu-se que a qualidade de serviço (QoS - *Quality of Service*) é degradada de acordo com o número de clientes [Vasconcelos 2002].

Também voltado para a transmissão ao vivo, o Zigzag [Tran 2004] organiza seus *peers* em uma hierarquia de *clusters* e constrói uma árvore *multicast* acima desta hierarquia. Em cada *cluster* um nó cabeça (*head* ou líder) é responsável por gerenciar a entrada de novos *peers* e um nó associado (*associate-head* ou co-líder) tem a responsabilidade de transmitir os dados para os outros membros.

Ao contrário do que acontece no SpreadIt e no Zigzag, o GloVE (*Global Vídeo Environment*) [Pinho 2002], é uma abordagem estritamente sob-demanda. Ele propõe a utilização da técnica de reuso de fluxo denominada Cache de Vídeo Cooperativa (CVC) para adicionar maior escalabilidade a servidores de vídeo sob-demanda. Nessa técnica, os *buffers* locais dos clientes integram uma memória global distribuída, usada como fonte de conteúdo. Ao chegar a primeira requisição, o servidor inicia um novo fluxo para o cliente. Quando uma segunda requisição chegar, o gerente da CVC procura em sua tabela por um cliente, chamado provedor, que possua a parte inicial do vídeo em seu buffer local. Caso

seja encontrado, o cliente é inserido no grupo de *multicast* servido por este provedor.

Um trabalho semelhante ao GloVE é o P2VoD [Do 2004]. Trata-se de uma arquitetura P2P, focada na transmissão sob-demanda, onde os clientes são agrupados em uma hierarquia de gerações. Pertencem a uma mesma geração aqueles que estão reproduzindo o mesmo trecho (ou um instante muito próximo do vídeo). Os *peers* da primeira geração são conectados diretamente ao servidor, os da segunda geração conectam-se a *peers* da primeira, e assim por diante. O nó pai é escolhido na geração anterior através de algoritmos de *round-robin*, menor atraso ou semelhança de perfil.

O *Cooperative Networking* (CoopNet) [Padmanabhan 2001] suporta tanto a transmissão ao vivo quanto a sob-demanda. Desenvolvido por um grupo de pesquisa da Microsoft [MSReseach 2005], o CoopNet nasceu como uma proposta para aliviar o volume de requisições à servidores *web* utilizando a capacidade de *upload* de alguns de seus clientes para servir outros. Nesse sistema, quando o servidor está sobrecarregado, o mesmo verifica quais os outros clientes que acessaram a mesma URL (página *web* ou mídia) recentemente, monta uma lista com seus endereços e a envia para o cliente que fez o pedido, este ao receber a mensagem requisita conteúdo para os clientes na lista.

Nos trabalhos descritos acima, o servidor precisa ter conhecimento da rede, pois é dele a tarefa de redirecionamento dos nós. No GnuStream [Jiang 2002] e na arquitetura proposta em [Vasconcelos 2002], a descoberta de recursos é feita com um algoritmo de inundação. Sempre que um nó desejar entrar na rede, o mesmo deverá enviar uma mensagem de *broadcast* solicitando uma fonte. Contudo, esse método apresenta problemas de escalabilidade quando se deseja alcançar todos os *peers* [Ritter 2001].

CAPÍTULO III

3 MODELO PROPOSTO

Com o objetivo de minimizar os problemas apresentados pelo tradicional modelo cliente-servidor na transmissão de mídia pela Internet, apresenta-se um novo modelo de comunicação, o **WaveNet.**

De forma semelhante ao CoopNet (*Cooperative Networking*), o modelo criado utiliza alguns de seus clientes para servir conteúdo para aqueles que requisitarem o recebimento da mídia quando o servidor estiver sobrecarregado. O novo modelo apresenta um sistema de agrupamento dos clientes de acordo com o trecho da mídia que estão compartilhando. Esse trecho da mídia é o que chamamos de onda (ou *wave)*.

Essa divisão em ondas possibilita que a rede assuma uma configuração semelhante na transmissão de vídeos sob-demanda e na transmissão ao vivo, diferindo basicamente no número de ondas simultâneas. Nas sessões seguintes discutimos isso mais detalhadamente.

No modelo proposto, chamamos de *peer* o nó da rede que atua como cliente ou cliente cooperativo (servindo ou não outros *peers*), em outras palavras, é todo nó que está conectado à rede excetuando o servidor.

O modelo WaveNet consiste basicamente de duas partes: cliente e servidor. O cliente é a entidade que solicita e recebe a mídia. O servidor é quem distribui a mídia. Porém, este não é um servidor comum, ele possui um elemento adicional, o *tracker*, que cataloga cada *peer* da rede e informa o endereço de cada um deles.

A interação entre esses elementos ocorre como mostrado no diagrama de seqüência da figura 8. O cliente que requisitar a mídia quando o servidor estiver sobrecarregado recebe uma lista de endereços de clientes que já estão recebendo conteúdo. Recebida a lista, o cliente escolhe um dos clientes na lista e negocia com ele a transmissão.

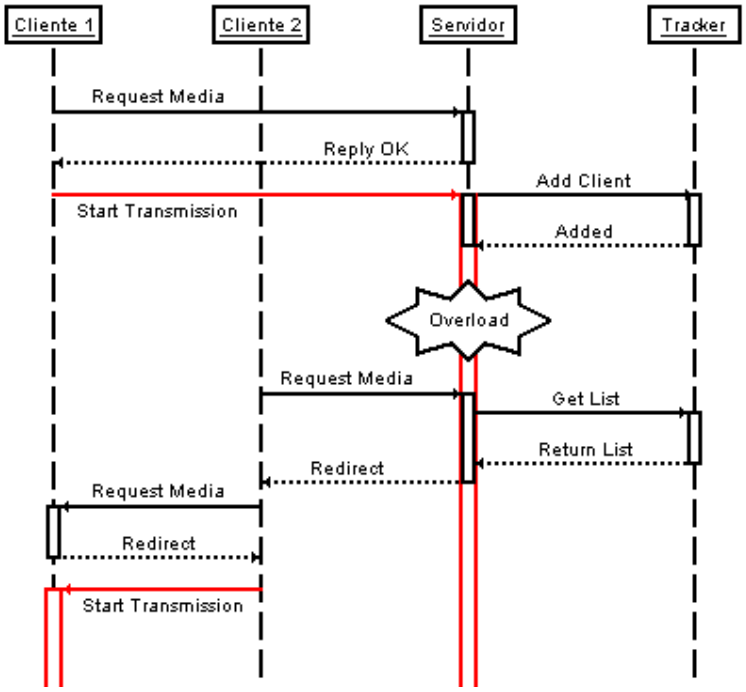

Figura 8 - Seqüência de operação de WaveNet

O conteúdo transmitido pelo servidor pode ser obtido de diferentes formas. Ele pode vir de uma mídia pré-gravada armazenada localmente no servidor ou de um servidor de arquivos conectado à rede. Já no caso específico de eventos ao vivo, o conteúdo pode ser capturado através de uma câmera ou microfone, e processado por um codificador antes de ser passado para o servidor. Na figura 9 é possível ter uma visão geral da operação de uma rede utilizando Wavenet.

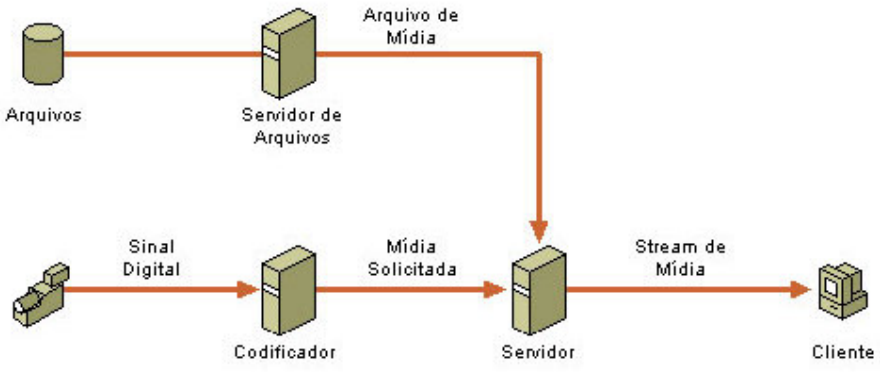

Figura 9 - Visão geral da rede WaveNet

3.1 Modos de Transmissão

Há duas formas de oferecer conteúdo para os clientes através da rede. Uma dessas formas é usar a transmissão ao vivo, a outra é usar a transmissão sob-demanda. O modelo aqui descrito assume um comportamento semelhante nos dois casos, ao contrário do que acontece nos sistemas de distribuição de conteúdo tradicionais. Nas sessões seguintes veremos isso com mais detalhes.

3.1.1 Transmissão Ao Vivo

A transmissão de mídia ao vivo é caracterizada pela distribuição de conteúdo sendo obtido de algum dispositivo de captura, como a cobertura de um evento, ou pré-gravado e distribuído apenas em um único instante, como acontece na televisão.

Nesse tipo de transmissão, o conteúdo que um usuário recebe e vê é muito próximo do que os outros usuários recebem e vêem. Explicando de uma outra forma, todos os clientes devem estar reproduzindo o mesmo trecho da mídia quase ao mesmo tempo.

Devido a essa sincronia de conteúdo a ser exibido, a transmissão de mídia ao vivo compõe um cenário favorável para a aplicação do modelo proposto. Quando ocorre uma sobrecarga no servidor, os clientes passam a compartilhar o conteúdo que já possuem em seus *buffers* com aqueles que solicitarem a mídia.

A figura 10 mostra a configuração de uma rede WaveNet quando nesse tipo de transmissão. Na figura, o M[1] ao lado das setas indica a transmissão do instante 1 e o $M[1]^{-}$ representa um instante anterior a 1.

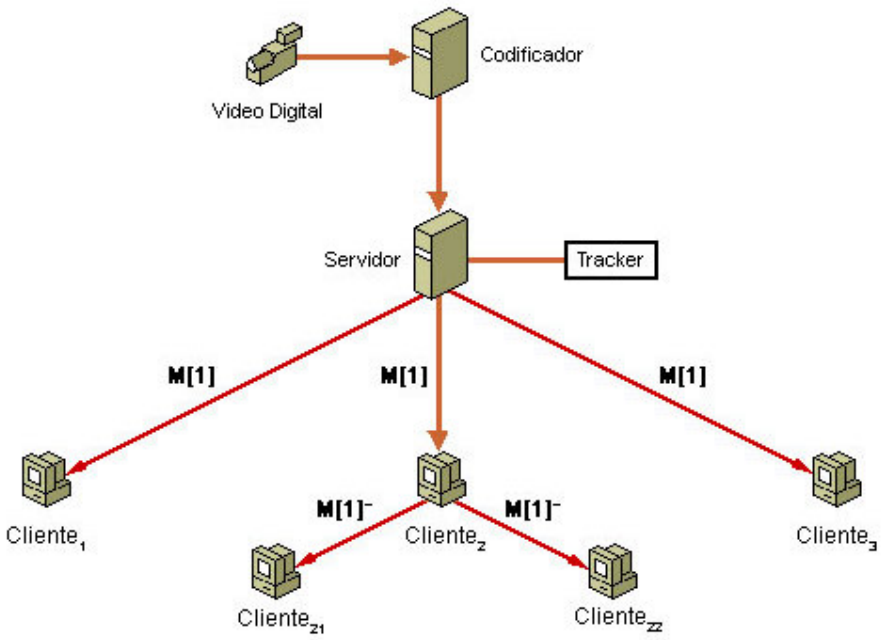

Figura 10 - WaveNet na trasmissão ao vivo

3.1.2 Transmissão Sob-Demanda

A transmissão de mídia sob-demanda refere-se àquela que inicia quando o conteúdo é solicitado. São os próprios usuários que decidem quando a transmissão deve iniciar.

Diferentemente do que acontece na transmissão ao vivo, na sob-demanda raramente todos os usuários estão reproduzindo o mesmo trecho da mídia. Para o caso do *streaming* progressivo isso não chega a ser um problema, mas para o *streaming* de tempo real tal fato reduz a potencialidade de um cliente servir conteúdo para os demais. Como o conteúdo é descartado após algum tempo, o trecho que um cliente possui pode não ser o que os demais estão precisando. Por exemplo, se o buffer de um cliente guardar até 10 segundos de transmissão, o seu conteúdo não será o que mais precisam, aqueles que conectaram 10 segundos mais tarde.

Contudo, em situações onde há uma sobrecarga devido ao volume de requisições, podemos encontrar vários usuários que iniciaram a transmissão ao mesmo tempo. Em WaveNet esses usuários sincronizados são separados em

grupos, de acordo com a onda em que estão. Dessa forma, usuários na mesma onda podem compartilhar conteúdo entre si. Repare que no caso da transmissão ao vivo o que temos é um sistema de apenas uma onda.

A figura 11 apresenta o funcionamento de WaveNet na transmissão de mídia sob-demanda. Na figura, o M[i] ao lado das setas indica o envio do instante i (onda i) da mídia e o $M[i]^{-n}$ indica que o que está sendo transmitido é um instante n unidades de tempo anterior a i, sendo i>0. Para cada onda i há um *tracker* i associado.

No que diz respeito ao funcionamento e ao gerenciamento dos *buffers*, tudo ocorre da mesma forma que na transmissão de mídia ao vivo. Um outro ponto que deve ser ressaltado é que ao contrário de CoopNet, o nosso modelo não faz *cache* na transmissão sob-demanda, pois isso exige um considerável espaço em disco.

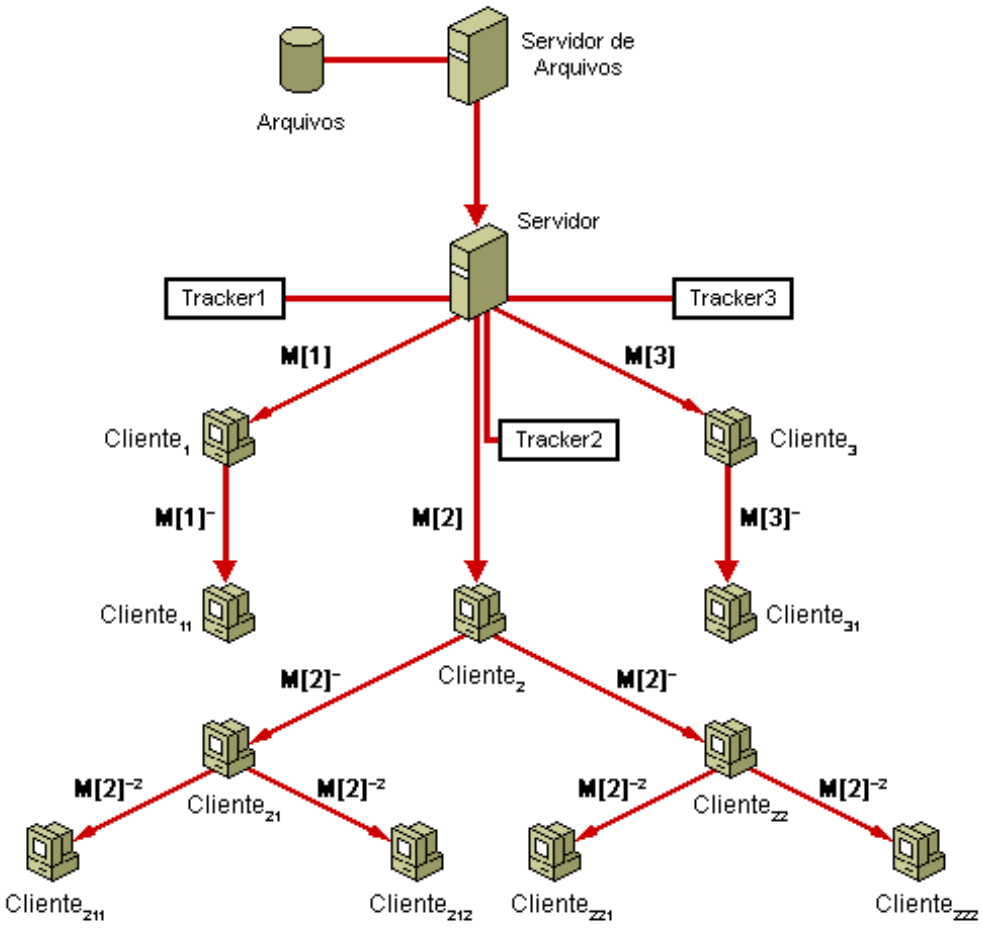

Figura 11 - WaveNet na transmissão sob-demanda

3.2 Gerenciamento dos Buffers

No *streaming* de tempo real, geralmente, o conteúdo não é guardado em *cache* e é descartado à medida que o mesmo é exibido. Por esse motivo, o modelo WaveNet utiliza um mecanismo de gerenciamento de *buffer* a fim de evitar que o conteúdo não seja descartado quando os demais ainda precisam dele.

O *buffer* usado no WaveNet é do tipo circular, isso significa que ao ser totalmente preenchido, ele passa a sobrepor os dados mais antigos, porém sem sobrescrever informações ainda não lidas pelo reprodutor de mídia.

Caso o usuário precise do espaço ocupado por dados que algum cliente esteja solicitando para armazenar conteúdo, a conexão com o solicitante é fechada, o que força este a buscar outro cliente que possua o trecho solicitado, só então os dados são sobrescritos.

3.3 Redirecionamento por Perfil

Da mesma forma que no Napster, os clientes devem reportar o seu perfil (banda) para o servidor junto ao seu pedido de conexão. Assim, quando o servidor solicita ao *tracker* uma lista de endereços, ele seleciona dentre os *peers* que conectaram à rede mais recentemente, aqueles que possuem um perfil semelhante ao daquele que está requisitando a mídia.

São considerados de perfil semelhante aqueles que possuem conexões com largura de banda próxima e pertencem ao mesmo grupo. Na tabela 1 são mostrados os grupos que utilizamos. Tal classificação é baseada na perda de sincronismo durante a reprodução da mídia. Os usuários de banda estreita são aqueles que tipicamente perdem a sincronia diversas vezes. Os usuários classificados como de largura de banda média podem ocasionalmente perder a sincronia. Enquanto que os de banda larga não perdem sincronia.

Tabela 1 - Classificação dos clientes de acordo com a banda e grupo

Perfil	Grupo
14.4 Kbps modem	Banda Estreita
28.8 Kbps modem	
56 Kbps modem	
64 Kbps ISDN	
112 Kbps ISDN	Banda Média
256 Kbps DSL/cable modem	Banda Larga
512 Kbps DSL/cable modem	
Acima de 512 Kbps	

A escolha dessa estratégia se deve ao fato de que para um cliente utilizando ADSL não há vantagem em tentar obter conteúdo de um usuário de um modem de 56 KB. O mesmo vale para o caso oposto, o usuário do modem não tem muito a ganhar conectando-se ao cliente com ADSL, pois o aquele está limitado à sua própria a banda. Mesmo os dados tendo a possibilidade de serem enviados com rapidez, eles seriam recebidos lentamente. Além disso, caso os *peers* sejam todos redirecionados para os de banda larga, quando outro com o mesmo tipo de conexão tentar entrar na rede, os *peers* que poderiam o atender melhor teriam pouca banda disponível, já que estariam repletos de usuários de banda estreita.

Contudo, *peers* com maior largura de banda podem suportar vários clientes de conexões mais lentas, equilibrando melhor o sistema. Por isso, também são incluídos na lista, *peers* com maior largura de banda, para caso não haja peers de perfil semelhante que possam atender ao solicitante. E para que clientes de banda larga possam ser atendidos satisfatoriamente, mesmo com outros clientes conectados ao mesmo *peer*, todos os eles são atendidos a uma velocidade proporcional ao seu perfil.

No entanto, sem qualquer restrição na velocidade de distribuição, permitir que *peers* com menor largura de banda obtenham conteúdo de *peers* com conexões mais rápidas pode fazer com que o *buffer* seja esvaziado mais rapidamente do que o solicitante pode receber. Por exemplo, um *peer* com uma conexão DSL de 512 Kbps recebendo uma mídia a 256 Kbps, deve ter outros 258 Kbps livres para *upload* (ignorando a banda utilizada para o tráfego que não seja do WaveNet). Caso um outro *peer* de banda inferior a 256 Kbps conecte-se a ele, e o cliente cooperativo distribuir a mídia nessa velocidade, parte do conteúdo pode ser perdido provocando uma séria queda na qualidade da exibição da mídia. Para evitar

essa situação, em WaveNet os *peers* servem conteúdo de acordo com o perfil do solicitante ou na velocidade estimada, o que for maior. O algoritmo completo com o processo de montagem da lista de *peers* de onde receber conteúdo é mostrado na figura 12.

```
do forever
  m_q ← receive(*)
  if (m_q = = REQUEST_MEDIA)
    if (getUsedBandwidth() < SERVER_BANDWIDTH)
       T = T ∪ {(A_q,P_q)}
       streamMedia(q)
    else
       L ← getMostRecentlyConnected(P_q)
       send(REDIRECT, L) to q
  else if (m_q = = CONFIRM_REDIRECTION)
    T = T ∪ {(A_q,P_q)}
```

Figura 12 - Algoritmo do servidor. Notação: m_q é a mensagem recebida de q, A_q é o endereço de q, P_q é o perfil de q, L é a lista e T é a lista de peers no tracker

3.4 Seleção do Peer

Tendo recebido a lista, o cliente deve tomar uma importante decisão, que é a escolha de qual *peer* obter conteúdo. Obviamente, ele deve escolher aquele que está em melhores condições para distribuir a mídia, contudo, descobrir qual é, não é tarefa trivial.

Alguns trabalhos como [Krishnamurthy 2001] e [Xiang 2004] propõem que os melhores *peers* são aqueles que estão mais próximos, mas para tráfego pesado como mídia, a largura de banda é uma medida melhor. Em [Bernstein 2003] os autores propõem o uso de árvores de decisão para classificar e um algoritmo de aprendizado baseado em informações coletadas de *downloads* anteriores para selecionar os *peers*. Porém isso exige uma grande coleta de dados. Do mesmo problema sofrem alguns trabalhos de estimativa de capacidade e banda disponível, como aqueles citados na sessão 2.5, que poderiam servir para classificar os *peers*, e assim mais facilmente escolher o melhor dentre eles.

De forma a evitar todos esses problemas, escolhemos os *peers* que de acordo com a banda disponível, para isso adotamos uma estratégia mais

simples. Primeiramente, o cliente solicita para até 5 *peers* da lista recebida do servidor que estimem a banda que poderão disponibilizar e enviem o resultado. A banda disponível é estimada dividindo a velocidade nominal que é possível transmitir pelo número de clientes já conectados ao *peer* mais um. Se em 10 segundos nenhum dos *peers* responder ou se as respostas forem muito baixas, o cliente escolhe outros 5 *peers* na lista e pede a mesma informação para eles. Todo o processo de seleção do *peer*, como os algoritmos do cliente e do servidor é mostrado na figura 13.

```
do forever
  send(REQUEST_MEDIA) to s
  mS ← receive(s)
  if (mS == REDIRECT)
    n ← min(5, |LS|)
    C = <pick n nodes form LS>
    while (C ≠ VAZIO ^ NOT found)
      i = 0
      while (i < n)
        send(ESTIMATE_SPEED) to Ci
        bi ← receive(Ci)
        if (bmax < bi) bmax ← bi

      if (bmax is enough to stream) found ← true
      else C = <pick n nodes form LS>

  else if (mS == RECEIVE_MEDIA)
    receiveMedia()
```

(a) Thread ativa

```
do forever
  mq ← receive(*)
  if (mq = = ESTIMATE_SPEED)
    do ∀ r ∈ CLIENTS
      l ← l + <download speed of r>
    send(lP/2 - l) to q

  else if (mq = = REQUEST_MEDIA)
    CLIENTS ← CLIENTS ∪ {q}
    send(CONFIRM_REDIRECTION) to s
    streamMedia(q)
```

(b) Thread passiva

Figura 13 – Algoritmos do cliente. Notação: s é o servidor, ms é a mensagem recebida de s, Ls é a lista recebida de s, bi é a banda disponível em i, l é a carga total do peer, l_P é a carga nominal do perfil P e CLIENTS é o conjunto de clientes conectados ao peer.

3.5 Gerenciamento da Árvore de Distribuição

Em WaveNet, conforme os clientes vão conectando à rede, uma árvore de distribuição vai sendo formada, tendo o servidor como raiz. Nessa árvore cada nó é um cliente que retransmite a *stream* recebida para cada um de seus nós filhos. A quantidade de filhos que o nó pode ter, ou seja, seu grau de saída, é condicionado à banda que o nó possui.

O gerenciamento da árvore resume-se a administrar eficientemente

as chegadas e saídas de nós, de forma que ao entrar na rede cada cliente seja redirecionado rapidamente para o nó mais apropriado e que as saídas sejam mais transparentes quanto possível para os demais nós da árvore.

Quando um cliente deseja entrar no sistema, ele primeiro avisa o servidor. Essa informação é então repassada para o *tracker* que monta uma lista com os *peers* de acordo com os critérios discutidos na sessão 3.3, e que não façam parte de sua sub-árvore. Essa lista é então enviada para o cliente, que dentre os que estão na lista, escolhe aquele em melhores condições.

Quanto à saída de nós da árvore, um tratamento mais cuidadoso é necessário para evitar que os nós órfãos inundem o servidor com mensagens de solicitação de um novo nó pai. A saída pode ocorrer de duas formas: intencionalmente ou não. No primeiro caso, o nó que deseja sair informa sua intenção antes de deixar a rede. Já no segundo caso, o nó sai por algum defeito, e por isso, fica impossibilitado de avisar.

Na saída intencional, assim que o servidor recebe a notificação de que um nó vai sair, ele imediatamente seleciona um novo nó pai (ou uma lista de possíveis pais) e envia essa informação para o nó que está saindo, e este retransmite a mensagem para seus filhos.

Já na situação em que um dos nós falhar, o problema é tratado junto à perda de pacotes. A falha do nó corresponde a uma condição especial onde há uma perda de 100% dos pacotes. Sendo assim, quando a taxa de perda atingir um ponto inaceitável (ou estiver muito próximo disso), o nó contata seu pai para ver se este está tendo o mesmo problema. Se a resposta for positiva, a fonte do problema está acima dele e o melhor é deixar que os nós que estiverem acima resolvam o problema. Agora, se a resposta for negativa ou se ele não responder, o próprio nó envia uma mensagem para o servidor solicitando um novo pai.

3.6 Balanceamento Dinâmico de Carga

De forma a contornar eventuais problemas decorrentes de variações nas condições de recebimento e transmissão dos clientes na rede, é feito um monitoramento da velocidade de recepção dados. Isso permite que os próprios

clientes possam buscar sempre maximizar suas taxas de *download*, monitorando-as e solicitando ao *tracker* clientes em melhores condições de lhe servir conteúdo em tempo de execução.

Para o caso específico da transmissão sob-demanda essa troca de *peer* servidor ainda pode envolver a mudança de onda. Caso na mesma onda não haja clientes em condições de lhe servir conteúdo, pode-se deslocar o cliente para uma onda próxima. Se as condições dos *peers* permitirem, ainda pode-se introduzir um atraso para o cliente esperar outra onda.

CAPÍTULO IV

4 SIMULAÇÃO

Na área de redes e sistemas distribuídos, para avaliar o desempenho de protocolos ou algoritmos, os pesquisadores utilizam modelagem analítica, experimentação (*benchmark*) ou simulação. Em [Fernandes 1992] o autor ainda cita mais duas técnicas: medição (feita em sistemas reais operando em condições normais) e prototipação (uma construção do sistema real simplificada é avaliada).

Na modelagem analítica o objetivo é conseguir, através da variação de parâmetros de entrada de um modelo estocástico, determinar o comportamento do algoritmo ou protocolo em situações variadas. Para que o modelo seja considerado válido, é necessário que as diferenças entre os valores obtidos analiticamente e através de medição sejam pequenas o suficiente para que o modelo seja considerado de boa aproximação. Essa técnica funciona bem para modelos simples que permitam modelagem através de um conjunto de equações. Ela é inadequada para casos onde, dada sua complexidade, não é possível determinar e testar todas as possibilidades de entrada. É útil, nestes casos, apenas para demonstrar propriedades gerais do protocolo. Diversos trabalhos como [Mesquita 1999] empregam essa técnica. Em [Ge 2003] os autores apresentam um modelo matemático genérico para sistemas peer-to-peer de compartilhamento de arquivos.

A experimentação consiste em testar sobre uma rede real. Define-se um conjunto de testes de resultados conhecidos, a serem executados para, em seguida, analisar os resultados. Geralmente, os testes são específicos de uma configuração, devido à grande influência da topologia, dos sistemas e dos recursos utilizados. Os resultados produzidos em uma rede podem não ser os mesmos em outra.

De uso mais comum, a simulação tem sido uma ferramenta poderosa no processo de projetar e avaliar protocolos de comunicação. Enquanto a avaliação analítica requer modelos simplificados da realidade e os experimentos práticos são de baixa reprodutibilidade (realizar testes de redes P2P em um

ambiente real torna-se uma tarefa quase impossível devido ao alto grau de crescimento desse tipo de rede), a simulação se encontra no nível intermediário, permitindo que o projetista do protocolo ajuste o nível de detalhe, atendo-se apenas aos recursos desejados. Além disso, simulação permite o aumento gradual de detalhamento, o que é ideal para o nosso trabalho. Os valores obtidos por simulação também podem ser usados como base para a validação do modelo analítico.

Diversos tipos de simuladores (programas desenvolvidos para promover a simulação desses sistemas) estão disponíveis hoje como ferramentas de apoio à análise de sistemas P2P. Podemos citar como exemplos o PeerSim [Jelasity 2004] e o P2psim [P2psim 2005].

4.1 Simuladores de Redes Peer-to-Peer

Além do fato das redes *peer-to-peer* poderem atingir grandes dimensões, os seus nós podem entrar e sair dinamicamente, tanto definitivamente quanto temporariamente. Simuladores de redes tradicionais como o *Network Simulator* 2 (ns-2) [Ns2 2005] e o SimmCast [Muhammad 2004] não lidam com tal fato. O ns-2 apenas oferece a opção de interromper os *links* entre os pontos comunicantes.

Existe, atualmente, uma série de simuladores para redes P2P como o GnutellaSim [He 2003] voltado a uma rede especifica. Mas ainda não se chegou a um consenso sobre um sistema de simulação genérico para redes P2P [Rocha 2004].

Entretanto, existe um consenso a cerca das principais características que precisam ser avaliadas ao se construir ou utilizar um simulador para redes P2P. Essas características dizem respeito à distribuição de conteúdo (volume e variedade de arquivos compartilhados), ao comportamento dos nós (número de *downloads*, entrada e saída rápida da rede) e configuração da rede física (topologia da rede e largura de banda) [Rocha 2004].

O PeerSim procura oferecer essas características. Parte do projeto BISON [Bison 2005], ele foi desenvolvido tendo em mente o suporte à larga

escalabilidade e dinamicidade necessárias para simular uma rede P2P. Para permitir a alta escalabilidade, são ignorados alguns detalhes da camada de comunicação da pilha de protocolos.

4.2 Metodologia

Para validar a nossa abordagem, diversos experimentos foram realizados com o PeerSim. A intenção era conferir o comportamento do modelo proposto com respeito aos seus parâmetros, verificar o tempo de comunicação associados à execução de um sistema usando o WaveNet e saber o quanto ele pode melhorar o modelo cliente-servidor tradicional.

Foram executados 20 experimentos independentes. Salvo quando dito o contrário, nesses experimentos os parâmetros eram fixos. Dois tamanhos de rede foram testados: 500 nós e 1000 nós. Cada um dos nós foi configurado aleatoriamente como tendo conexão de 110 Kbps, 256 Kbps, 512 Kbps ou 1Mbps.

Como é uma característica do simulador escolhido, os resultados são expressos em ciclos, sendo que o δ, que representa a duração de cada um deles, define a complexidade de convergência. A escolha de um valor pequeno para δ resulta em uma rápida convergência, porém com um alto custo de comunicação por unidade de tempo. Caso com um pequeno número de ciclos seja possível obter uma configuração otimizada, valores altos para o δ são permitidos.

No apêndice A é apresentado o arquivo de configuração do simulador para a realização dos testes neste trabalho. Também no apêndice está uma breve explicação dos parâmetros principais do simulador e como proceder para simular a execução de um protocolo.

Como configuração inicial, foi escolhido um servidor, sem clientes conectados, com capacidade para atender seus clientes a uma taxa total de 100 Mbps e então começamos a adicionar os *peers*. Foi assumido que os *peers* chegam a uma taxa *A*, fixa, e que cada requisição é processada em um tempo *t*.

É considerado que os *peers* estão conectados através de uma rede roteada, tal como a *Internet*, onde cada um pode comunicar-se com os demais.

Para que essa comunicação seja possível, basta que o *peer* saiba o identificador do destinatário. A relação entre os *peers* define a topologia da rede.

4.3 Resultados

A primeira característica a ser testada é o número de clientes que podem ser atendidos com o sistema de redirecionamento. Em média, cerca de 210 *peers* podem ser atendidos pelo servidor, e outros quase 290 *peers* podem ser servidos por clientes. Os dados desse teste podem ser vistos no gráfico da figura 14, onde a parte de baixo de cada coluna indica o número de clientes atendidos pelo servidor e a parte de cima indica o número de clientes atendidos por outros clientes.

Nos testes, o tamanho do ciclo é configurado como o tempo entre duas chegadas consecutivas mais o tempo gasto para processar a requisição, ou seja, $\delta = (1/A)+t$. Assim, com esse mesmo teste pode-se constatar o tempo que o servidor consegue operar normalmente até ser sobrecarregado e os *peers* que forem chegando começarem a ser redirecionados. Então, temos que, em média, o servidor foi sobrecarregado após $210 * \delta$.

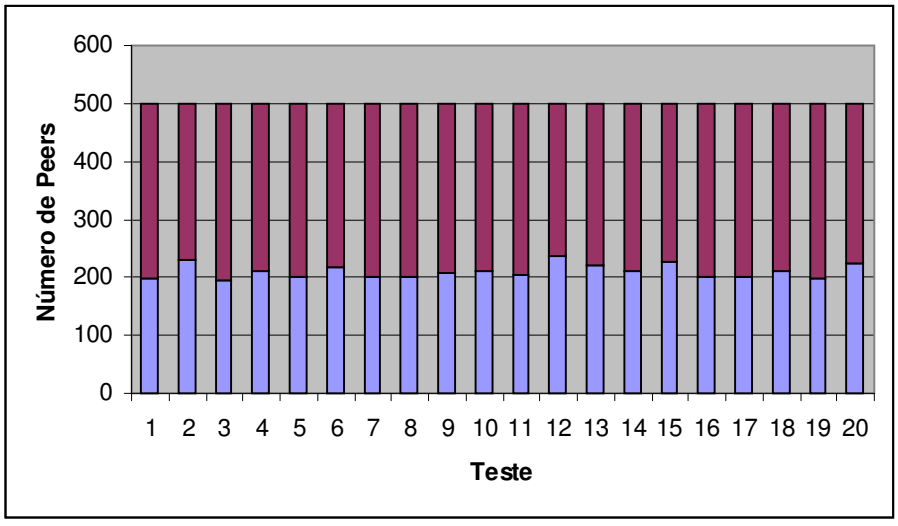

Figura 14 - Número de Peers atendidos pelo Servidor e pelos Clientes em cada teste

Esse mesmo teste foi repetido para um número maior de clientes. Escolhendo 1000 chegadas consecutivas, tivemos uma média de 683,6 *peers* atendidos por clientes e quase 213 *peers* conectados diretamente ao servidor. Os demais tiveram serviço negado e não puderam ser acomodados pelo sistema, mesmo após 2000 ciclos de execução.

CAPITULO V

5 IMPLEMENTAÇÃO

Por proporcionar um ambiente mais interessante para a aplicação do modelo, o estudo de caso ficou concentrado na transmissão de conteúdo ao vivo. A aplicação desenvolvida neste trabalho é composta de um módulo cliente que solicita o recebimento de mídia contínua para um servidor com capacidade para redirecionamento de requisições para outros clientes, segundo o modelo WaveNet definido no capítulo anterior.

5.1 Servidor

O servidor, desenvolvido em linguagem Java, é alimentado com arquivos MPEG2-TS, criados com o software VLC [VideoLan 2005]. Esse tipo de arquivo é um formato de multiplexação de pacotes de sinais de áudio e vídeo, preparado para ser transportado em redes, carregando informações úteis em ambientes onde há ruído ou perda de pacotes.

Usamos arquivos de vídeo já codificados porque a implementação de um sistema de captura e codificação em tempo real exigiria um alto grau de código extra em nosso servidor. Além disso, programas e dispositivos de codificação de vídeo capturado, em geral, produzem conteúdo mais lentamente do que o nosso servidor pode transmitir [Schulter 2005]. O próprio software utilizado para codificar os arquivos testados possui um módulo de captura, codificação e envio da mídia através da rede, mas como ele necessita de um certo tempo de para montar o vídeo, era necessário que o cliente requisitasse a mídia somente alguns segundos após a transmissão iniciar.

A transmissão da mídia é iniciada quando o primeiro cliente conecta-se a rede e encerrada no momento em que a rotina de envio atingir o fim do arquivo. Nesse momento, um sinal de fim de *stream* é enviado.

Como especificado no capítulo anterior, um servidor WaveNet em cada onda há um *tracker* para armazenar e recuperar o endereço dos *peers*

conectados. Contudo, como nosso estudo de caso está centrado na transmissão de mídia ao vivo, temos apenas um *tracker*.

5.2 Cliente

Assim como o servidor, o cliente também foi desenvolvido em plataforma Java, sendo que a reprodução das *streams* ficou a cargo da *Java Media Framework* API (JMF) [JMF 2005]. Como as classes presentes nessa API não são capazes de manipular *streams* corretamente, foram implementados um novo PullDataSource e um novo PullSourceStream com *buffer* circular de 512KB para manipular a mídia da forma desejada. O diagrama de classes do *buffer* circular está na figura 15.

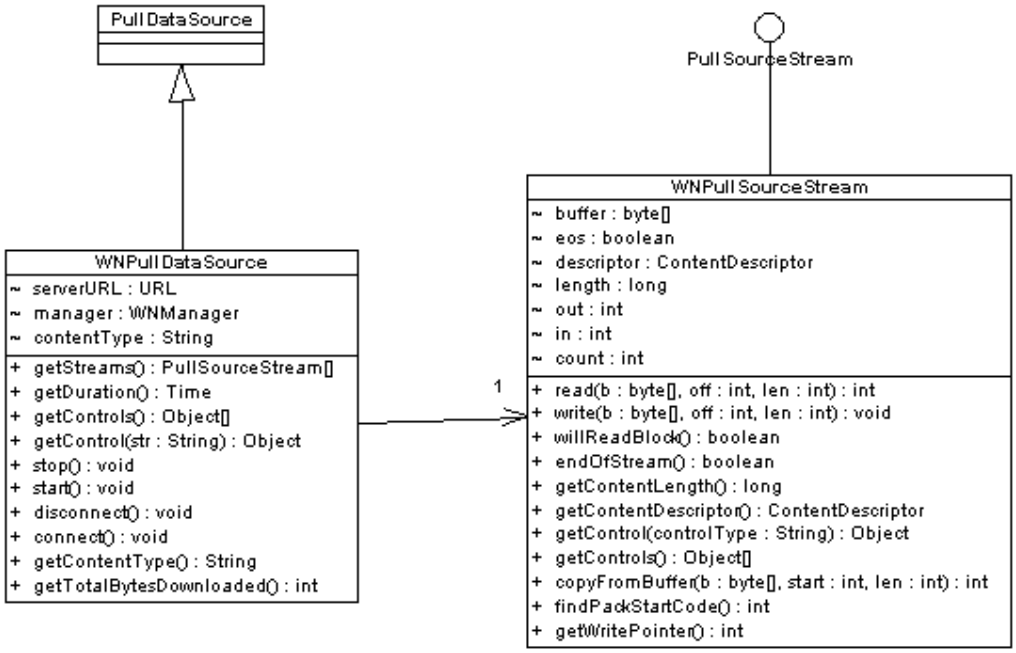

Figura 15 - Diagrama de classes do buffer circular

A JMF possui uma interface chamada Player que consegue reproduzir vídeo em formato MPEG-1, mas ela somente aceita mídia provida desde os seus primeiros *bytes*. Ela precisa de algumas informações contidas no cabeçalho MPEG para conseguir decodificar corretamente o arquivo.

Contudo, esse problema foi facilmente resolvido pela utilização do formato MPEG2-TS. Como o bloco de dados de um pacote TS carrega pacotes PES que possuem uma estrutura semelhante à dos pacotes MPEG-1, bastou que ao receber os primeiros *bytes*, o ponteiro de leitura do buffer fosse posicionado no início de um desses pacotes que a Player conseguiu reproduzir a mídia. Para isso, existe um método chamado findPackStartCode() na classe WNPullSourceStream, que busca pelo prefixo de código de início de *pack* e pelo identificador de *stream*. Esses dois valores, em conjunto, formam um inteiro de 4 *bytes* cujo valor é igual ao código de inicio de pacote do MPEG-1 (0x000001BA). Como a interface Player da JMF não foi modificado para aceitar a codificação TS (apenas um novo SourceStream foi criado para manipular localizar alguns dados que os pacotes carregam), os campos dessa codificação que não estão presentes no formato MPEG-1 são ignorados pela Player conforme sua implementação original.

Seguindo o modelo proposto, quando o cliente deseja receber a mídia, ele solicita ao servidor e caso o mesmo não possa atender, uma lista de possíveis fontes é recebida e o *peer* é escolhido de acordo com as estratégias descritas anteriormente.

Tão logo a conexão seja estabelecida, um processo de preenchimento do *buffer* (*buffering*) tem início. O cliente segue copiando os dados até que o tempo para preencher o restante do buffer seja menor que o tempo para reproduzir o conteúdo já obtido. Por exemplo, caso o vídeo exija uma taxa de transmissão de 128 Kbps e o cliente possua uma conexão de 112 Kbps, conhecendo o tamanho do *buffer*, com um simples cálculo é possível saber que a exibição do conteúdo nele presente não deve ser interrompida se o tempo de espera for de pelo menos 19,6 s. Consideramos que caso o cliente não possua conexão rápida, interrupções na transmissão são toleráveis. Caso o cliente tenha mais banda que a taxa de codificação, a exibição começa imediatamente após os primeiros *bytes* serem recebidos.

Em cada conexão, há um tempo máximo de espera por dados. Se o cliente permanecer 10 segundos sem receber qualquer dado de sua fonte, ele deve procurar um outro *peer* na lista que possa atendê-lo. A lista também deve ser atualizada regularmente. Seja N o número de endereços recebidos na lista anterior, caso já tenha se passado mais de 6 * N segundos desde a recepção, uma nova lista

atualizada é solicitada ao servidor. O diagrama de classes do gerenciador de conexões está na figura 16.

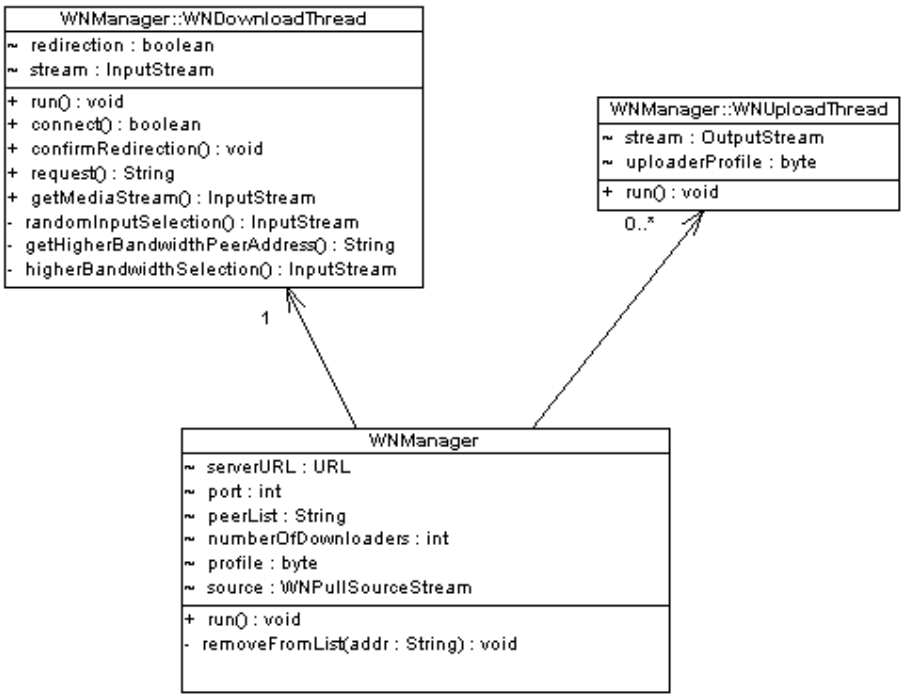

Figura 16 - Diagrama de classes do gerenciador de conexões

5.3 Testes Experimentais

Concluída a fase de projeto e implementação do sistema de distribuição de mídia com o WaveNet, procedeu-se uma avaliação do sistema obtido.

Os testes foram executados em um ambiente controlado de uma rede local com dois Athlon XP 1600 com 256 MB de memória RAM e dois computadores Athlon XP 2200+ com 256 MB de memória RAM. Um desses computadores foi designado para ser o servidor. Os demais computadores foram limitados a conexões de 110 Kbps, 256 Kbps, 512 Kbps através do uso do software *NetPeeker* 2.83 [NetPeeker 2005]. Larguras de banda menores não foram utilizadas por não serem suficientes para receber o vídeo e apresentá-lo adequadamente.

No total, foram feitos 20 experimentos independentes, com o servidor transmitindo uma mídia de 799857 bytes com 50 segundos de duração. Foi escolhido esse tamanho de arquivo, pois ele oferece uma boa qualidade de imagem e os clientes configurados com 110 Kbps conseguem recebê-lo forma aceitável sem comprometer muito a exibição (uma taxa de cerca de 124 Kbps é exigida).

Os dados dos testes foram coletados com um software de monitoramento de tráfego chamado *Bandwidth Monitor Pro* 1.29 [BMP 2005]. O cliente implementado também possui um relatório de comportamento que é apresentado na tela assim que as tarefas de *download* e *upload* terminam.

5.3.1 Resultados

O principal elemento testado foi o buffer circular. Verificamos a influência do tamanho do buffer na taxa de transferência do vídeo entre o servidor e cliente. Diferentes velocidades de conexão foram testadas, porém, como pudemos constatar nos testes, para os clientes de maior largura de banda mesmo que haja algum acréscimo no tempo de transferência, a velocidade de recebimento é superior a de consumo. Os dados destes testes são apresentados na figura 17. Por estarmos em uma rede local os valores obtidos acabaram sendo bastante altos, pois não estivemos sujeitos a atrasos e perdas de pacotes características de um acesso via Internet.

Utilizando um *buffer* de 512 KB, em média a transmissão usando o perfil de 110 Kbps ocorreu a 13,12 KBps, enquanto que com um perfil de 256 Kbps ela ocorreu a 32,58 KBps em média. Com um *buffer* de 256 KB, no perfil 110 Kbps a transmissão atingiu uma média muito próxima à obtida com um *buffer* de 512 KB, mas com interrupções na transmissão após cerca de 36 segundos. No perfil 256 Kbps houve uma redução na taxa de transmissão, atingindo uma média de 26,92 KBps. O tamanho de 1024 KB também foi testado para o *buffer*, porém os valores obtidos foram muito próximos aos dos testes para buffer de 512 KB.

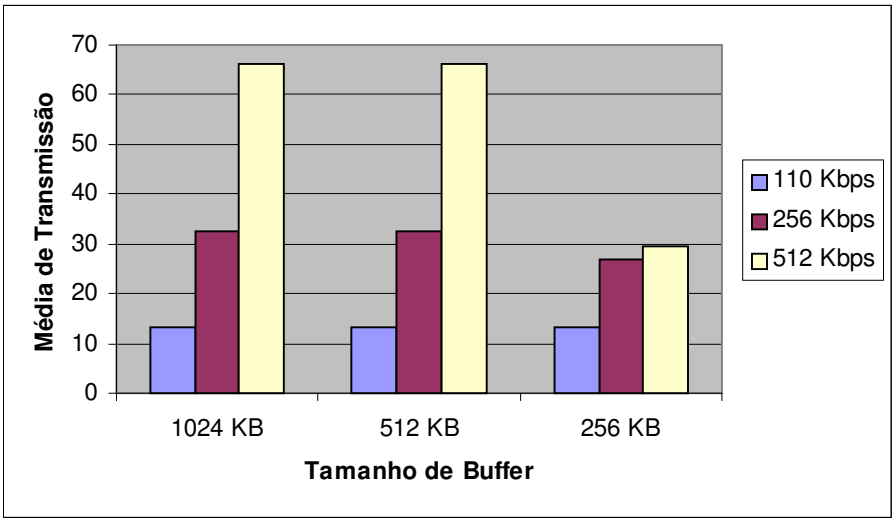

Figura 17 - Média de transmissão para diferentes tamanhos de buffer

CAPITULO VI

6 CONCLUSÕES

Antes de apresentar as conclusões tiradas dos testes realizados através de simulação e de experimentação, algumas considerações devem ser feitas. Primeiramente, deve ser salientado que o objetivo das discussões realizadas nesse trabalho era fazer uma análise sobre a utilização de clientes cooperativos nos sistemas de distribuição de mídia. As limitações técnicas dos elementos envolvidos são respeitadas. Em segundo lugar, a escolha da linguagem Java deu-se pelo fato dela ser portável, o que permitiu um desenvolvimento independente de plataforma, além de possuir uma API que tornou mais fácil criar o reprodutor de mídia, e uma boa integração com o simulador (também em Java). O *overhead* característico dessa linguagem é aceito.

Pelos resultados dos testes simulados é possível concluir que o modelo proposto faz com que um número maior de clientes seja atendido, mas há limites. Como os clientes não são originalmente criados com o intuito de servir conteúdo, a capacidade deles é limitada, e dependendo da configuração da rede que se formar, pode não haver muitos clientes com largura de banda suficiente para servir conteúdo em certos momentos.

Apesar disso, o número de clientes atendidos é muito maior que quando o sistema opera no modo cliente-servidor tradicional. Nos testes com 500 *peers*, em média o sistema acomodou mais que o dobro de nós, e nos teste com 1000 *peers* o tamanho da rede chegou aumentar mais de três vezes.

No que diz respeito aos testes experimentais, devido às limitações impostas pelo ambiente de trabalho, no que diz respeito ao número de computadores disponíveis para serem utilizados como clientes, não foi possível exaurir o sistema da mesma forma que na simulação. Contudo, isso é uma limitação bem conhecida dos testes experimentais, como discutido nesse trabalho.

Quanto aos testes com o buffer, é possível concluir quais são os melhores tamanhos para se obter o melhor desempenho. Obviamente, o tamanho ideal para o *buffer* seria o tamanho do arquivo de mídia, mas como não podemos contar que tanto espaço esteja disponível no cliente, devemos procurar alocar a

menor quantidade de memória que consiga nos atender satisfatoriamente. Sendo assim, para uma mídia com a qualidade utilizada, o menor tamanho de *buffer* que proporcionou as melhores taxas de transmissão para diferentes velocidades de conexão foi de 512 KB.

Uma limitação do reprodutor presente na API utilizada que influenciou no desenvolvimento diz respeito à duração da mídia. A interface Player permite reproduzir até 1 hora 42 minutos e 25 segundos de mídia no formato MPEG. Aparentemente, a JMF não consegue manipular corretamente a memória alocada e acaba interrompendo a reprodução. Após esse tempo, o cliente continua em execução e caso o *buffer* ainda possua espaço livre, ele segue transferindo conteúdo até preenchê-lo totalmente. Isso prova que o sistema tem condições de continuar funcionando para arquivos de maior tamanho, mas ele está condicionado às limitações da API.

Tratar da qualidade de imagem e som apresentados pelo vídeo não está no escopo deste trabalho, mas durante a geração dos arquivos ficou visível que mesmo clientes que podem ser considerados de conexão com velocidade mediana como os de 110 Kbps não são capazes de lidar com fluxos de mídia de boa qualidade. Uma mídia de boa qualidade exige pelo menos 256 Kbps. Na figura 18 são mostradas as imagens de arquivos para essas duas velocidades de transmissão.

Acreditamos que as pesquisas para encontrar novas maneiras para aproveitar as estruturas já existentes e assim melhor acomodar o tráfego multimídia na rede serão ainda mais elaboradas no futuro. O protótipo desenvolvido é apenas o primeiro passo nesse sentido.

Figura 18 - Imagens de arquivos de mídia com 799857 bytes (coluna da esquerda) e 1466338 bytes (coluna da direita). Percebe-se que há menos distorções nas imagens da direita.

6.1 Trabalhos Futuros

O assunto abordado nesse trabalho não está totalmente esgotado e muitas questões ainda podem ser vistas, testadas e analisadas. Para tanto, algumas implementações ainda devem ser feitas:

1) Um ponto que ainda pode ser estudado diz respeito à qualidade do conteúdo exibido quando alguns nós falham. Problemas, como a saída ou redução repentina das capacidades de um cliente da rede, podem fazer com que os nós abaixo dele da árvore de distribuição fiquem sem receber dados até que a árvore seja reparada. Tal fato acarreta em dois problemas: a parada completa da transmissão e a interrupção de seqüência.

1.1) A parada da transmissão foi parcialmente tratada neste trabalho. Quando há uma queda na transmissão o cliente imediatamente reinicia um processo de entrada na rede solicitando a lista e buscando um novo cliente cooperativo. Contudo esse processo pode ser agilizado se cada peer tiver uma

noção de como estão os demais peers. O SG-1 [Montresor 2004] utiliza um algoritmo que propaga informações dos peers uns para os outros diretamente através da rede e pode ser utilizado para minimizar os efeitos desse problema.

1.2) Para contornar o problema da interrupção de seqüência, uma das saídas é utilizar o *Multiple Description Coding* (MDC) [Goyal 2001] [Padmanabhan 2002]. Ele pode ser utilizado para codificar sinal de áudio e vídeo em *streams* separadas, que chamamos de descrições, onde cada subconjunto dessas descrições pode ser recebido e decodificado em um sinal com ou sem distorção, de acordo com os dados recebidos. Nesses casos, a renegociação da qualidade de serviço também pode ser vista.

2) Um sistema para adaptação de conteúdo de acordo com o perfil indicado pelo cliente pode ser uma boa adição ao modelo proposto. Como uma extensão da aplicação de MDC, mas agora para adaptação de conteúdo, alguns pacotes poderiam ser suprimidos para clientes de banda estreita reduzindo a qualidade da mídia. Em [Braga 2003] o vídeo é transmitido em duas camadas sendo uma com o fluxo básico e outra com pacotes opcionais que apenas adicionam qualidade. Dessa forma a recepção e visualização da mídia pode ser melhor acomodada em conexões mais lentas.

3) Como testes experimentais em grande escala que garantem uma visão realista da operação do modelo proposto são difíceis de serem feitos, pode ser interessante realizar os testes já feitos em plataformas abertas distribuídas de grande escala tais como o PlanetLab [PlanetLab 2005]. Atualmente, o PlanetLab possui cerca de 580 nós espalhados pelo mundo, pouco se comparado ao que uma rede *peer-to-peer* pode atingir, mas apropriado para um teste de operação com elementos distantes.

4) No que diz respeito à implementação, o cliente ainda pode ser estendido de forma a se tornar um *plugin* para *browsers*, e assim permitir que qualquer usuário possa instalar o reprodutor de mídia e se tornar um cliente WaveNet.

REFERÊNCIAS

[Bernstein 2003] Bernstein, D. S.; Feng, Z.; Levine, B. N.; Zilberstein, S. "Adaptive peer selection". Proceedings of the 2nd International Workshop on Peer-to-Peer Systems. 2003.

[Bison 2005] Bison: Biology-Inspired techniques for Self-Organization in dynamic Networks. 2005. http://www.cs.unibo.it/bison/. Consultado em 06/07/05.

[BMP 2005] Bandwidth Monitor Pro. 2005. http://www.bandwidthmonitorpro.com/. Consultado em 18/07/2005.

[Braga 2003] Braga, H. de L.; Andrade, M. de; Roesler, V.; Lima, V.; "SAM - Sistema adaptativo para multimídia". ERAD - Escola Regional de Alto Desempenho. Janeiro/2003.

[Carter 1996] Carter, R. L.; Crovella, M. E. "Measuring Bottleneck Link Speed in Packet Switched Networks". Performance Evaluation, 27-28, 1996.

[Do 2004] Do, T. T.; Hua, K. A.; Tantaoui, M. "P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment". IEEE International Conference on Communications, Paris, Junho/2004.

[Deshpande 2001] Deshpande, H.; Bawa, M.; Garcia-Molina, H. "Streaming Live Media over a Peer-to-Peer Network", Technical Report, Stanford University, Stanford, CA, EUA, Agosto/2001. http://dbpubs.stanford.edu:8090/pub/2001-31.

[Dovrolis 2001] Dovrolis, C.; Ramanathan, P.; D. Moore. “What do Packet Dispersion Techniques Measure?” Proceedings of IEEE INFOCOM 2001.

[Downey 1999a] Downey, A. B. "Using pathchar to Estimate Link Characteristics". Procceedings of ACM SIGCOMM, 1999.

[Downey 1999b] Downey, A. B. "Clink: a tool for estimating Internet link characteristics". 1999. SIGMETRICS '99.

[Fernandes 1992] Fernandes, M. M. “Modelagem analítica de desempenho de sistemas multiprocessadores: aplicação ao multiprocessador CPER”. Departamento de Computação. Universidade Federal de São Carlos. São Carlos. 1992. 104p. Dissertação de Mestrado.

[Ge 2003] Ge, Z.; Figueiredo, D. R.; Jaiswal, S.; Kurose, J.; Towsley, D. . "Modeling Peer-to-Peer File Sharing Systems". Proceedings of INFOCOMM 2003.

[Gnutella 2005] Gnutella.com, 2005, http://www.gnutella.com/. Consultado em 04/07/05.

[Goyal 2001] Goyal, V. K. "Multiple Description Coding: Compression Meets the Network". IEEE Signal Processing Magazine, vol. 18, no. 5, p. 74-93. Setembro/2001.

[He 2003] He, Q.; Ammar, M.; Riley, G.; Raj, H.; Fujimoto, R. "Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems". 11th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2003). Outubro/2003.

[ISO 1996a] "ISO/IEC 13818-1 - Information Technology - Generic Coding of Moving Pictures and Associated Audio Information - Part 1: Systems (MPEG-2 Systems)", 1996.

[ISO 1996b] "ISO/IEC 13818-2 - Information Technology - Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video (MPEG-2 Video)", 1996.

[ISO 1998a] "ISO/IEC 13818-3 - Information Technology - Generic Coding of Moving Pictures and Associated Audio Information - Part 3: Audio (MPEG-2 Audio)", Second Edition, 1998.

[ISO 1998b] "ISO/IEC 13818-6 - Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Extensions for Digital Storage Media Command and Control". 1998.

[Jain 2002] Jain, M.; Dovrolis, C. "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput". In Proceedings of ACM SIGCOMM. 2002.

[Java 2005] Sun Microsystems. 2005. http://java.sun.com/. Consultado em 04/06/2005.

[Jelasity 2004] Jelasity, M.; Montresor, A; Babaoglu, O. “A Modular Paradigm for Building Self-organizing Peer-to-peer Applications”. Engineering Self-Organising Systems, volume 2977 of Lecture Notes in Artificial Intelligence, pages 265-282. 2004.

[Jiang 2002] Jiang, X.; Dong, Y.; Xu, D. "GnuStream: A MPEG Video Streaming System Over Peer-to-Peer Network". CS590N P2P Networks and Services. Dezembro/2002.

[JMF 2005] Java Media Framework API, Sun Microsystems, 2005, http://java.sun.com/products/java-media/jmf/index.jsp. Consultado em 30/06/05.

[Krishnamurthy 2001] Krishnamurthy, B; Wang, J. “On Network-Aware Clustering of Web Clients”, ACM SIGCOMM, Agosto/2001.

[Lai 2000] Lai, K; Baker, M. "Measuring link bandwidths using a deterministic model of packet delay". Proceedings of ACM SIGCOMM 2000. Agosto/2000.

[Lakshminarayanan 2004] Lakshminarayanan, K.; Padmanabhan, V. N.; Padhye, J. "Bandwidth Estimation in Broadband Access Networks". Internet Measurement Conference 2004 (IMC’04). Outubro/2004.

[Lara 2003] Lara, C. R. F. "Projeto de um Servidor de Vídeos Sob Demanda Paralelo e Distribuído". Departamento de Computação. Universidade Federal de São Carlos. São Carlos. 2003. 105p. Dissertação de Mestrado.

[Mah 2000] Mah, B. A. "Estimating Bandwidth and Other Network Properties". Internet Statistics and Metrics Analysis Workshop on Routing and Topology Data Sets: Correlation and Visualization. 2000.

[Mesquita 1999] Mesquita, T. M. S. "Modelagem e Análise de Desmpenho do Serviço vídeo-sob-demanda em Redes de Alta Velocidade utilizando o Protocolo RSVP sobre Redes ATM". Departamento de Computação. Universidade Federal de São Carlos. São Carlos. 1999. 110p. Dissertação de Mestrado.

[Montresor 2004] Montresor, A. "A Robust Protocol for Building Superpeer Overlay Topologies". Proceedings of the 4th International Conference on Peer-to-Peer Computing. Agosto/2004.

[MPEG 2005] Moving Picture Experts Group. 2005. http://www.chiariglione.org/mpeg/.Consultado em 19/07/2005.

[MSResearch 2005] Microsoft Research. 2005. http://research.microsoft.com/. Consultado em 06/07/05.

[Muhammad 2004] Muhammad, H. H.; Bedin, G. B.; Facchini, G.; Barcellos, M. P. "Quebrando a Barreira entre Simulação e Experimentação Prática em Redes de Computadores". SBRC 2004.

[Napster 2005] Napster.com, 2005, http://www.napster.com/. Consultado em 04/07/2005.

[Ns2 2002] The Network Simulator 2. 2005. http://www.isi.edu/nsnam/ns/. Consultado em 06/07/2005.

[Orkut 2005] Orkut. 2005. http://www.orkut.com/. Consultado em 15/07/2005.

[P2psim 2005] "p2psim: A Simulator for Peer-to-peer Protocols". 2005. http://pdos.csail.mit.edu/p2psim/. Consultado em 19/06/2005.

[Padmanabhan 2002] Padmanabhan, V. N.; Sripanidkulchai, K. "The Case for Cooperative Networking". Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS). Março/2002.

[Padmanabhan 2002] Padmanabhan, V. N.; Wang, H. J.; Chou, P. A.; Sripanidkulchai, K. "Distributing Streaming Media Content Using Cooperative Networking". ACM NOSSDAV, Maio/2002.

[Pc-News 2005] CDN (Content Delivery Network), Pc-news.com. 2005. http://www.pc-news.com/detalle.asp?sid=&id=44&Ida=750. Consultado em 04/06/05.

[Pinho 2002] Pinho, L. B. "Implementação e Avaliação de um Sistema de Vídeo Sob-demanda Baseado em Cache de Video Cooperativa", COPPE. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2002. 71 p. Dissertação de Mestrado.

[PlanetLab 2005] PlanetLab. 2005. http://www.planet-lab.org/. Consultado em 06/07/05.

[Real 2004] "Introduction to Streaming Media with RealPlayer 10". RealNetworks, Inc. 2004.

[Ribeiro 2000] Ribeiro, V.; Coates, M.; Riedi, R.; Sarvotham, S.; Hendricks, B.; Baraniuk, R. "Multifractal Cross-traffic Estimation". ITC Conference on IP Traffic, Modeling and Management. 2000.

[Ribeiro 2003] Ribeiro, V. J.; Riedi, R. H.; Baraniuk, R. G.; Navratil, J.; Cottrell, L. "pathChirp: Efficient Available Bandwidth Estimation for Network Paths". Proceedings of PAM Workshop. 2003.

[Ritter 2001] Ritter, J. "Why Gnutella can't scale. No, really.", Fevereiro/2001,http://www.darkridge.com/jpr5/doc/gnutella.html. Consultado em 04/06/2005.

[Rocha 2004] Rocha, J.; Domingues, M.; Callado, A.; Souto, E.; Silvestre, G.; Kamienski, C.; Sadok, D. "Peer-to-Peer: Computação Colaborativa na Internet". SBRC 2004.

[RTSP 1998] Request For Comments 2326 - Real Time Streaming Protocol (RTSP). 1998. http://www.ietf.org/rfc/rfc2326.txt. Consultado em 27/04/2004.

[Schollmeier 2002] Schollmeier, R. "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications". Proceedings of the First International Conference on Peer-to-Peer Computing. IEEE Computer Society. 2002.

[Schulter 2003] Schulter, A.; Ribeiro, L. A.; Becker, V.; Caetano, M. F.; Montez, C.; Melo, E.; Fröhlich, A. A. M. "Um Ambiente de Distribuição de Vídeo MPEG2 com Suporte Multicast em Código Aberto para o Projeto I2TV". 4º Workshop RNP2. 2003.

[Strauss 2003] Strauss, J.; Katabi, D.; Kaashoek, F. "A Measurement Study of Available Bandwidth Estimation Tools". In Proceedings of IMC. 2003.

[Tanenbaum 2003] Tanenbaum, A. S. "Computer Networks". Prentice Hall. 2003.

[Terra 2005] Terra. http://www.terra.com.br. 2005.

[Tran 2004] Tran, D. A.; Hua, K. A.; Do, T. T. “A Peer-to-Peer Architecture for Media Streaming”, IEEE Journal of Selected Areas in Communications, Vol. 22, No. 1, pp. 121-133, Janeiro/2004.

[Uol 2005] Uol. http://www.uol.com.br. 2005.

[Vasconcelos 2002] Vasconcelos, M. A.; Almeida, V.; Campos, S. “Análise da Distribuição de Streaming Media em Arquiteturas do Tipo Peer-to-Peer”. VI Semana de Pós-Graduação em Ciência da Computação. Setembro/2002.

[VideoLan 2005] VideoLan. http://www.videolan.org. 2005. Consultado em 23/04/2005.

[Wikipedia 2005] Content Delivery Network, Article, Wikipedia. 2005. http://en.wikipedia.org/wiki/Content_Delivery_Network. Consultado em 04/06/2005.

[WM 2005] Windows Media Website. 2005. http://www.windowsmedia.com/. Consultado em 19/07/2005.

[WMS 2005] Windows Media Services. 2005. http://www.microsoft.com/ windows/windowsmedia/serve/wmservices.aspx. Consultado em 19/07/2005.

[Xiang 2004] Xiang, Z.; Zhang, Q.; Zhu, W.; Zhang, Z.; Zhang, Y. "Peer-to-Peer Based Multimedia Distribution Service", IEEE Transactions on Multimedia, Vol. 6, No. 2, Abril/2004.

ANEXO A

A.1 A arquitetura do PeerSim

O PeerSim é um simulador de protocolos de redes peer-to-peer construído para ser modular e altamente configurável. Cada componente é facilmente trocado por outro que tenha uma função similar, ou seja, implemente a mesma interface.

O seu componente principal é a entidade Configurator, responsável por ler os arquivos de configuração. Esses arquivos de configuração são em formato texto, compostos basicamente de duas partes chave.

Na simulação três tipos de componentes podem estar presentes: protocolo (protocol), dinâmica (dynamic) e observador (observer). Cada um deles implementado por uma classe Java.

A rede é formada por uma coleção de nós, sendo que cada nó pode possuir mais de um protocolo. A comunicação entre os protocolos é baseada em chamada de métodos. Um protocolo deve implementar a interface Protocol ou CDProtocol.

A dinâmica é implementada com a interface Dynamics e a sua execução é agendada pelo próprio motor do simulador para ser executada periodicamente. Com esse componente é possível modificar qualquer aspecto do protocolo. Todos os parâmetros internos do protocolo podem ser alterados. Também é possível provocar a falha do nó.

O observador tem como função monitorar e analisar a rede, por isso são executados periodicamente da mesma forma que o componente de dinâmica. Esse componente deve implementar uma interface Observer.

O motor do simulador é o componente que executa todos os passos da simulação de acordo com as instruções no arquivo de configuração. Dois modelos de execução do motor podem ser usados: o modelo baseado em ciclo e o modelo baseado em evento. No primeiro caso, a cada passo todos os nós são selecionados randomicamente e cada um dos nós é invocado por vez dentro do ciclo. No segundo caso, um suporte à concorrência é provido e um conjunto de

eventos (mensagens) são agendados a cada momento e os protocolos são executados de acordo com a ordem na qual as mensagens são transmitidas.

A.2 Implementando WaveNet no PeerSim

Para implementar um protocolo no PeerSim, é necessário escreve-lo em linguagem Java. Como já dito anteriormente um componente do tipo protocolo no PeerSim precisa implementar a classe CDProtocol, ou pelo menos a Protocol. wavenet.WNProtocol, o nome escolhido para o protocolo implementado, implementa a primeira opção. Para armazenar os identificadores de perfil estendemos a classe SingleValueHolder.

Contudo, apenas escrever o protocolo não é suficiente. Alguns componentes auxiliares são requeridos. Um componente para inicialização dos peers e outro para o observador foram implementados. O primeiro, o wavenet.RandomProfileInitializer implementa a interface Dynamics. O segundo, o wavenet.WNObserver, implementa a interface Observer.

A.3 Arquivo de Configuração

```
1   # WaveNet Example
2   simulation.cycles 550
3   #simulation.cycles 2000
4   simulation.shuffle
5
6   overlay.size 500
7   #overlay.size 1000
8   overlay.maxsize 100000
9
10  protocol.0 peersim.core.IdleProtocol
11  protocol.0.degree 20
12
13  protocol.1 wavenet.WNProtocol
14  protocol.1.linkable 0
15
16  init.0 peersim.dynamics.WireRegularRandom
17  init.0.protocol 0
18  init.0.degree 20
19
```

```
20    init.1 wavenet.RandomProfileInitializer
21    init.1.protocol 1
22
23    observer.0 wavenet.WNObserver
24    observer.0.protocol 1
```

Da linha 3 até a 7 alguns parâmetros globais são impostos. O número máximo de ciclos e o tamanho da rede são configurados. O parâmetro simulation.shuffle indica que os nós devem ser visitados em ordem diferente a cada ciclo. Ele não possui parâmetro porque é usado como uma *flag*. Na linha 8 o tamanho máximo da rede é configurado.

As linhas seguintes configuram os protocolos. Dois protocolos são usados: peersim.core.IdleProtocol (linha 10) e wavenet.WNProtocol (linha 13). O primeiro não executa qualquer tipo de operação, sendo usado apenas pela sua capacidade de acessar a topologia. O segundo executa as operações do protocolo de comunicação do WaveNet.

Da linha 16 até a 21 todos os componentes previamente declarados são inicializados. Dois são componentes os inicializados: peersim.dynamics.WireRegularRandom e wavenet.RandomProfileInitializer. Este inicializa cada um dos peers com um perfil escolhido aleatoriamente, enquanto aquele liga randomicamente os nós.

E finalmente, nas linhas 23 e 24 o último componente é declarado e configurado, o observador. Apenas um observador, o wavenet.WNObserver, para o protocolo WaveNet é usado.