DIEGO SANCHEZ GALLO

Arquitetura de IPTV com Suporte à Apresentação Deslocada no Tempo
Baseada em Distribuição *Peer-to-Peer*

Dissertação apresentada à Escola Politécnica da
Universidade de São Paulo para a obtenção do
Título de Mestre em Engenharia

São Paulo

2009

DIEGO SANCHEZ GALLO

Arquitetura de IPTV com Suporte à Apresentação Deslocada no Tempo
Baseada em Distribuição *Peer-to-Peer*

Dissertação apresentada à Escola Politécnica da
Universidade de São Paulo para a obtenção do
Título de Mestre em Engenharia

Área de concentração:
Sistemas Digitais

Orientadora:
Prof^a. Dr^a. Tereza Cristina Melo de Brito Carvalho

São Paulo

2009

FICHA CATALOGRÁFICA

Gallo, Diego Sanchez

Arquitetura de IPTV com suporte à apresentação deslocada no tempo baseada em distribuição Peer-to-Peer / D.S. Gallo -- São Paulo, 2009.

122 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Redes de computadores 2. Redes multimídia 3. Sistemas colaborativos 4. Sistemas distribuídos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

Aos meus pais, irmãos, namorada e mestres.

AGRADECIMENTOS

À minha orientadora, Tereza Cristina Melo de Brito Carvalho, pela orientação e apoio nas diversas etapas desta pesquisa.

Ao Frank Schaffa, pela co-orientação, ainda que informalmente, pela atenção e empenho em me ajudar sempre que precisei, mesmo sem ter que dizer, e pela possibilidade de uma experiência profissional e pessoal valiosíssima no início de meu mestrado, na IBM Research em Hawthorne, NY.

Ao Professor Wilson, pelos valiosos comentários durante o exame de qualificação.

Aos colegas de pós-graduação e do LARC, em especial ao Charles, Carlos, Marcio, Marcos, Flavio, Joelle e Fernando, pelo incentivo e ajuda durante estes anos.

À minha namorada por agüentar os momentos de estresse e sempre dar forças para eu continuar o trabalho, e aos meus pais e minha família em geral, responsáveis pela minha formação e por me apoiarem incondicionalmente.

Aos meus amigos, em especial André e Henrique, pela força dada nos momentos de desespero e pelos momentos de diversão ao longo de todos estes anos em SP, e Vlad e Leila, pelas longas conversas após as caronas e por todo o incentivo à conclusão deste trabalho.

À Ericsson Telecomunicações do Brasil e à FDTE, pelo suporte financeiro, e aos colegas da Ericsson Research da Suécia, em especial Per, Victor, Ayodele, Karl-Ake, Hareesh, Sami, Bruce, Stefan e Josilene, pelos comentários, sugestões, e auxílio durante minha estada na Suécia, propiciando um ambiente de pesquisa que foi essencial para a conclusão deste trabalho.

E a tantos outros que colaboraram, direta ou indiretamente, mesmo que sem saber, com a conclusão desta pesquisa.

RESUMO

Com o aumento da concorrência sofrido pelas operadoras de telecomunicações frente à entrada de diversas empresas de outros ramos no mercado de comunicação, como, por exemplo, os Provedores de Serviço de Internet (ISPs - *Internet Service Providers*) através da oferta de serviços de voz sobre IP, tais operadoras viram-se obrigadas a diversificar sua oferta de serviços para gerar novas fontes de receita. Por possuírem ampla infra-estrutura instalada, as operadoras de telecomunicações passaram a oferecer, também, serviço de TV aos usuários, através de suas redes (convergentes) de telefonia e dados já existentes, o chamado IPTV. O objetivo deste trabalho foi possibilitar, neste cenário, que estas empresas consigam oferecer, além dos serviços convencionais de TV (e.g., transmissões lineares dos conteúdos nos canais de TV), serviços diferenciados empregando-se a mesma infra-estrutura. O foco deste trabalho é a oferta do serviço de apresentação deslocada no tempo dos conteúdos transmitidos linearmente nos canais de TV, sem a necessidade de configuração prévia por parte do usuário. Desta maneira, dá-se maior flexibilidade ao usuário, possibilitando-o assistir aos conteúdos que lhe interessam, no horário mais conveniente, sem ter que se preocupar com isso antecipadamente (i.e., sem a necessidade de configurar algum equipamento para gravar o conteúdo ou saber antecipadamente quais programas lhe interessam). Para isso foram pesquisadas e analisadas tanto tecnologias de transmissão e distribuição de conteúdos, como também o paradigma *peer-to-peer*, muito utilizado atualmente no compartilhamento de arquivos na Internet. A partir daí, foi concebida uma arquitetura capaz de oferecer tanto o serviço tradicional de transmissão linear de TV, quanto de apresentar vídeos deslocados no tempo (i.e., vídeos cuja transmissão linear já foi iniciada ou até concluída, a partir de qualquer posição já transmitida), combinando-se técnicas de multidifusão de dados, armazenamento distribuído e protocolos *peer-to-peer*. Desta maneira, obteve-se uma solução eficiente, utilizando-se os recursos disponíveis em todo o sistema, incluindo recursos ociosos dos usuários finais, para auxiliar no armazenamento e distribuição dos conteúdos deslocados no tempo. Finalmente, um protótipo foi desenvolvido como prova de conceito da arquitetura proposta neste trabalho, e, juntamente com os testes

realizados, comprovam a viabilidade de se utilizar redes P2P para a distribuição dos conteúdos para a apresentação deslocada no tempo.

Palavras chaves: Sistemas IPTV. Apresentação deslocada no tempo. Vídeo sob demanda. Redes *peer-to-peer*. Protocolos *peer-to-peer*. Redes de distribuição de conteúdos. Multimídia. Sistemas distribuídos. Sistemas colaborativos.

ABSTRACT

Telecommunication companies are suffering from the increasing offer of cheap and reliable voice over IP services, being forced to diversify their services looking for new revenue possibilities. Since these companies have a vast infrastructure, they are now providing TV services through the same telephony and data infrastructure, using their IP networks to offer IPTV. The goal of the present work is to allow, in this scenario, that such companies offer, additionally to the traditional TV services (e.g., the linear transmissions of the TV channels), differentiated services through the same infrastructure. The focus of the present work is, therefore, the offering of the time-shift service, allowing users to watch linear transmitted contents, time-shifted, without the need for any in-advance configuration. This approach gives more flexibility to the users, allowing them to choose the most appropriate time to watch some content without having to specify their interests in advance (i.e., without configuring some equipment to record the content or knowing in advance which programs will interest themselves). To achieve this goal, technologies for content transmission and distribution, as well as the peer-to-peer paradigm for file sharing were studied, resulting in the development of an architecture capable of offering the traditional linear transmission's service as well as the possibility of time-shift, combining multicast, distributed caching and peer-to-peer technologies. Accordingly, an efficient solution was envisioned, making use of all available resources in the system, including idle resources in the user equipments, to help in the caching and distribution of the time-shifted contents. Finally, a prototype was developed as a proof-of-concept for the designed architecture, which together with the performed tests, shows the viability of utilizing P2P networks in the distribution of time-shifted contents.

Keywords: IPTV systems. Time-shift TV. Video on Demand (VoD). Peer-to-peer networks. Peer-to-peer protocols. Content distribution network. Multimedia. Distributed systems. Collaborative systems.

LISTA DE ILUSTRAÇÕES

Figura 3.1 - Taxonomia de arquiteturas para <i>multicast</i> na Internet (LIU, J. ET AL., 2008)....	28
Figura 3.2 - Cliente ingressando em um grupo <i>multicast</i>	32
Figura 3.3 - Tráfego inicial através do RP	32
Figura 3.4 - Rota final	33
Figura 3.5 - SplitStream (CASTRO, MIGUEL ET AL., 2003)	36
Figura 3.6 - CoolStreaming/DONet (ZHANG, X. ET AL., 2005).....	38
Figura 3.7 - Captura do <i>buffer</i> no (a) BitTorrent e (b) CoolStreaming (LIU, J. ET AL., 2008)	39
Figura 3.8 - Bullet (KOSTIĆ ET AL., 2003)	41
Figura 3.9 - mTreeBone (WANG, F. ET AL., 2007).....	43
Figura 3.10 - Rede de distribuição de conteúdo (CDN) (PALLIS; VAKALI, 2006)	44
Figura 3.11 - Akamai Media Delivery (para conteúdo sob demanda) (AKAMAI TECHNOLOGIES, 2008a).....	46
Figura 3.12 - Akamai Stream OS (AKAMAI TECHNOLOGIES, 2008b).....	46
Figura 3.13 - Plano de dados do Prism (CRANOR ET AL., 2001)	48
Figura 3.14 - Plano de controle do Prism (CRANOR ET AL., 2001)	49
Figura 3.15 - Serviço de mapeamento hierárquico (CRANOR ET AL., 2001)	50
Figura 3.16 - Porcentagem relativa de tráfego P2P (SCHULZE; MOCHALSKI, 2007)	51
Figura 3.17 - Estatística de tráfego P2P (SCHULZE; MOCHALSKI, 2007).....	53
Figura 3.18 - Troca de mensagens entre <i>peers</i>	54
Figura 3.19 - Mecanismo de <i>unchoke</i>	55
Figura 4.1 - Arquitetura proposta	66
Figura 4.2 - Comunicação entre os componentes da arquitetura.....	68
Figura 5.1 – Problemas de sincronismo dos dados armazenados no <i>cache</i>	75
Figura 5.2 - Alinhamento dos dados armazenados no <i>cache</i>	76
Figura 5.3 - Janela do EPG (<i>Electronic Program Guide</i>)	80
Figura 5.4 - Janela principal do Cliente.....	81
Figura 6.1 - Topologia de testes	90
Figura 6.2 - Taxa de bits do conteúdo	92
Figura 6.3 - Componentes da latência para o início da exibição.....	93
Figura 6.4 - Parcela da latência correspondente a cada componente	94
Figura 6.5 - <i>Overhead</i> de sinalização e controle	95
Figura 6.6 - Tempo total de obtenção do conteúdo	96
Figura 6.7 - Impacto da capacidade de processamento do cliente na latência para o início da exibição.....	97
Figura 6.8 - Impacto do número de <i>peers</i> servindo o conteúdo na latência para o início da exibição.....	98
Figura 6.9 – Impacto do número de <i>peers</i> no <i>overhead</i> de sinalização e controle.....	99
Figura 6.10 – Impacto do uso de informações de localidade na escolha de <i>peers</i> na latência para o início da exibição.....	100
Figura 6.11 - Redução do tráfego nos enlaces de rede decorrentes do uso de localidade.....	101

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BM	Buffer Map
CDN	Content Distribution Network
DHT	Distributed Hash Table
DNS	Domain Name System
DSHT	Distributed Sloppy Hash Table
DSL	Digital Subscriber Line
DVR	Digital Video Recorder
EPG	Electronic Program Guide
FTP	File Transfer Protocol
GoP	Group of Pictures
HD	High Definition
HDD	Hard-Disk Drive
HDTV	High Definition TeleVision
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Socket Layer
IGMP	Internet Group Management Protocol
IPTV	Internet Protocol TeleVision
ISP	Internet Service Provider
jBitTorrent	Java BitTorrent
JMF	Java Media Framework
jVLC	Java bindings for VideoLan Client
MDC	Multiple Description Code
NAT	Network Address Translation

P2P	Peer-to-Peer
PIM-SM	Protocol Independent Multicast – Sparse Mode
PVR	Personal Video Recorder
QoE	Quality of Experience
RP	Rendezvous Point
RSS	Really Simple Syndication
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol
SCAMP	SCAlable Membership Protocol
SD	Standard Definition
SLA	Service Level Agreement
SPT	Shortest Path Tree
SSM	Source Specific Multicast
STB	Set-Top Box
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
URN	Uniform Resource Name
URL	Uniform Resource Locator
VCR	Video Cassette Recorder
VLC	VideoLan Client
VoD	Video on Demand
VoIP	Voice over Internet Protocol
XML	eXtensible Markup Language
XORP	eXtensible Open Router Platform

SUMÁRIO

1	Introdução	14
1.1	Motivação e Objetivos	15
1.2	Descrição do Problema	16
1.3	Método	18
1.4	Organização do Trabalho	19
2	Definições	20
2.1	Transmissão Linear	20
2.2	PVR (Personal Video Recorder)	21
2.3	Vídeo sob Demanda	21
2.4	Apresentação Deslocada no Tempo	22
2.5	Considerações Finais	23
3	Distribuição de Vídeo em Redes de Dados	24
3.1	Requisitos para Distribuição de Vídeo	24
3.2	<i>Multicast</i>	26
3.2.1	Taxonomia de Arquiteturas para <i>Multicast</i> na Internet	28
3.2.2	<i>Multicast</i> IP nativo	29
3.2.3	<i>Multicast</i> em Nível de Aplicação	33
3.2.3.1	Abordagem Baseada em Árvores	34
3.2.3.2	Abordagem Dirigida por Dados	37
3.2.3.3	Abordagem Híbrida	40
3.3	Redes de Distribuição de Conteúdos	43
3.3.1	Akamai	45
3.3.2	Prism	47
3.4	Compartilhamento de Arquivos <i>Peer-to-Peer</i>	51
3.4.1	Protocolo BitTorrent	52
3.5	Considerações Finais	56
4	Arquitetura de um Sistema IPTV com Apresentação Deslocada no Tempo	59
4.1	Soluções Existentes	59
4.2	Especificação de Requisitos	63
4.2.1	Requisitos Funcionais	63
4.2.2	Requisitos Não-Funcionais	64
4.3	Descrição da Arquitetura de Proposta	66
4.4	Casos de Uso do Sistema	69
4.4.1	Exibição de Fluxo de Conteúdo Durante a Transmissão Linear	69
4.4.2	Exibição de Fluxo de Conteúdo Deslocado no Tempo	69
4.5	Considerações Finais	71
5	Implementação do Protótipo	72
5.1	Detalhes de Implementação	72
5.1.1	Ingestão de Conteúdo pelo <i>Proxy</i>	73
5.1.2	Alinhamento dos Dados Armazenados nos <i>Caches</i>	74
5.1.3	Verificação de Integridade do Conteúdo	77
5.1.4	Operação do Módulo Cliente	78
5.1.5	Mecanismo de Seleção de <i>Peers</i>	81
5.2	Limitações	83
5.2.1	OpenChord DHT	83

5.2.2	Java BitTorrent	84
5.2.3	Java Bindings for VideoLan Client (jVLC)	85
5.2.4	Configurações Estáticas dos Módulos no Protótipo	86
5.3	Considerações Finais	87
6	Descrição e Análise dos Resultados	89
6.1	Topologia de Testes	90
6.2	Caracterização do Conteúdo Utilizado nos Testes	91
6.3	Método de Testes	92
6.4	Cenários de Testes e Resultados Obtidos	93
6.4.1	Componentes da Latência para o Início da Exibição	93
6.4.2	<i>Overhead</i> de Sinalização e Controle	94
6.4.3	Tempo Total de Obtenção do Conteúdo	95
6.4.4	Capacidade de Processamento do Cliente	96
6.4.5	Replicação do Conteúdo em <i>Caches</i> e Outros Clientes	97
6.4.6	Impacto do Uso de Informações de Localidade	99
6.5	Considerações Finais	101
7	Considerações Finais	103
7.1	Contribuições e Inovações da Dissertação	104
7.2	Trabalhos Futuros	104
	Referências	106
	Apêndice A – Estrutura do Arquivo de Metadados do BitTorrent	112
	Apêndice B – Parâmetros da Comunicação com o Rastreador BitTorrent	115
	Apêndice C – Detalhamento das Mensagens do BitTorrent	119
	Apêndice D – Exemplo do Arquivo XML de Configuração dos Módulos	122

1 INTRODUÇÃO

A chegada da TV a cabo e via satélite atraiu a atenção dos usuários devido à oferta de maior quantidade de conteúdos, incluindo canais de conteúdo específico 24 horas por dia, como, por exemplo, somente de filmes, desenhos, documentários, entre outros.

Após esta atração inicial de consumidores, as operadoras de TV a cabo começaram buscar maneiras de gerar maior receita por usuário, criando, então, serviços de vídeo sob demanda (VoD – *Vídeo On Demand*), no qual o usuário pode assistir a um determinado conteúdo (e.g., filme, seriado, etc.) no horário em que lhe for mais adequado. Além disso, novos serviços começaram a ser providos em outras áreas, tais como a oferta de banda larga para acesso à Internet e de telefonia VoIP (*Voice over IP*), aproveitando-se a mesma infra-estrutura.

Por outro lado, as empresas de telecomunicações, que investiram massivamente em infra-estrutura muitos anos antes, viram sua rentabilidade diminuindo devido à entrada das operadoras de TV a cabo em mercados antes dominado por aquelas. Com isso, viram-se obrigadas a criarem novos serviços que fizessem uso da infra-estrutura ociosa, estendendo esta infra-estrutura até a casa do usuário, o que possibilitou a oferta de serviços de TV utilizando-se a rede de dados para distribuição de conteúdos, e adentrando, assim, no mercado de entretenimento televisivo.

Com a convergência da TV à rede de dados e voz (*triple play*), surge a possibilidade de ofertar muitos outros serviços além dos convencionais (i.e., a transmissão linear dos conteúdos e vídeo sob demanda), tal como interatividade, com a possibilidade de seleção de ângulo da câmera, votação influenciando o “fluxo” do conteúdo e requisição de mais informações sobre produtos ou serviços sendo anunciados, entre outros.

Em alguns locais, empresas de telecomunicações já estão ofertando IPTV (*Internet Protocol Television*), como, por exemplo, nos Estados Unidos através da AT&T com o serviço chamado “U-Verse” que contava com um milhão de assinantes¹ em dezembro de 2008 e em Hong Kong através da PCCW Limited com o “Now TV” que

¹ <http://www.att.com/gen/press-room?pid=5838>

em junho de 2008 possuía 927.000 assinantes² (em uma cidade com população estimada em 2007 de 6.963.100³).

No Brasil, as empresas de telecomunicações são proibidas de ofertar serviço de TV por assinatura devido a barreiras regulatórias (Lei do Cabo⁴ e Plano Geral de Outorga⁵). Isso atrasa o início da oferta deste serviço por estas empresas, uma vez que somente agora estas regulamentações começam lentamente a ser revistas. Porém, com a revisão destas regulamentações, as empresas de telecomunicações que atuam no país devem iniciar a oferta deste serviço.

1.1 Motivação e Objetivos

Nos serviços de TV convencionais o usuário pode assistir a uma certa variedade de canais, além de, dependendo do tipo de serviço que possui, poder escolher e assistir vídeo sob demanda. Muitos dos sistemas atuais utilizam um Guia de Programação Eletrônico (EPG – *Electronic Program Guide*) para permitir aos usuários a visualização dos nomes dos conteúdos que foram, estão sendo, e serão transmitidos em cada canal. Além disso, estes sistemas exibem, independentemente do anterior, a lista de conteúdos sob demanda disponíveis para compra ou exibição. Porém, se um usuário percebe que um conteúdo que lhe interessa começou a ser exibido em algum canal, independentemente desta exibição já ter acabado ou não, o usuário não consegue assistir este conteúdo desde o início.

Sendo assim, usuários são forçados a adequar suas próprias agendas para conseguir assistir os conteúdos que lhes interessam, ou ao menos programar previamente para que determinado conteúdo seja gravado localmente quando há explícito interesse futuro em assisti-lo (por exemplo, com equipamentos de DVR – *Digital Video Recorder* ou Gravador de Vídeo Digital – ou PVR – *Personal Video Recorder* ou Gravador de Vídeo Pessoal).

Mas o que acontece quando o usuário chega em casa, liga a TV, e descobre que perdeu um conteúdo muito interessante, ou que algum conteúdo que começou a ser

² <http://www.pccw.com/eng/AboutUs/CompanyProfile.html>

³ http://www.censtatd.gov.hk/hong_kong_statistics/statistics_by_subject/index.jsp

⁴ Lei nº 8.977, de 6 de Janeiro de 1995 (http://www.planalto.gov.br/Ccivil_03/LEIS/L8977.htm)

⁵ Decreto nº 2.534, de 2 de abril de 1998 (http://www.planalto.gov.br/ccivil_03/decreto/D2534.htm)

transmitido há algum tempo atrás é muito interessante, mas perdeu o começo do mesmo? O objetivo central deste trabalho é possibilitar a oferta deste serviço de apresentação deslocada no tempo dos conteúdos transmitidos nos canais em um sistema de IPTV, com eficiência na utilização dos recursos, sem a necessidade de qualquer configuração prévia por parte do usuário.

Com este objetivo, uma arquitetura de IPTV que possibilita a apresentação deslocada no tempo é proposta, utilizando *caches* distribuídos e tecnologia *peer-to-peer* para permitir a distribuição destes conteúdos em qualquer momento após o início de sua exibição em determinado canal de TV (antes ou após o término da transmissão), sendo que o conteúdo deve ficar disponível por um período determinado pelo provedor de IPTV ou pelo provedor do conteúdo.

1.2 Descrição do Problema

Os contínuos avanços nas redes de computadores e na conectividade com a Internet proporcionaram nos últimos anos um crescimento acelerado no número de transmissões de conteúdo multimídia pelas redes. Além disso, fizeram com que a expectativa do usuário crescesse constantemente no que tange à qualidade e diversidade dos serviços ofertados (GRAHAM-ROWE, 2008). Esta possibilidade tecnológica aliada à expectativa dos usuários tem forçado os provedores de TV por assinatura (provedores de TV por satélite, TV a cabo e, mais recentemente, IPTV) a diversificar suas ofertas.

De maneira a assegurar qualidade de experiência (QoE – *Quality of Experience*) satisfatória aos usuários, qualquer serviço de vídeo baseado em redes impõe fortes requisitos com relação à largura de banda necessária e à latência aceitável. Congestionamentos e gargalos podem surgir na rede devido ao grande volume transmitido de dados, levando a um nível de serviço inadequado. Estes problemas ficam mais evidenciados em serviços de IPTV do que em TV sobre Internet⁶, uma vez que no primeiro a resolução dos conteúdos deve ser melhor, necessitando ainda

⁶ IPTV assume transmissão de conteúdos através de uma rede proprietária, de maneira equivalente à TV a cabo, enquanto que TV sobre Internet se refere realmente à transmissão de conteúdos de vídeo – em geral diferentes dos transmitidos nos canais de TV – sobre a Internet, sem qualquer garantia na transmissão.

mais banda, e deve haver garantia de qualidade do serviço. Isto não ocorre na TV sobre Internet onde não existe garantia de qualidade de serviço, utilizando-se a técnica *best-effort*⁷ na transmissão dos dados, i.e., se a Internet estiver congestionada haverá problemas na transmissão não existindo garantia ao usuário de que o sistema funcionará adequadamente em determinado momento (SIMPSON; GREENFIELD, 2007).

Garantir um nível de QoE adequado (equivalente ao oferecido pelos serviços de TV por assinatura existentes) no provimento do serviço de apresentação deslocada no tempo é uma tarefa desafiadora. As soluções existentes (e.g., serviço “*Start Over*” e “*Look Back*” da Time Warner⁸) utilizam transmissão *unicast* entre servidores de rede e usuários, sendo caracterizados pela falta de escalabilidade (limitações de largura de banda e quantidade de usuários por servidor), e demandando conseqüentemente grande infra-estrutura para atender todas as requisições dos usuários. Estes problemas também se aplicam a muitas soluções de VoD e nPVR (*network-based Personal Video Recorder*), os quais se baseiam, do mesmo modo, na transmissão *unicast* de servidores para usuários. (HUANG ET AL., 2006)

Para evitar a demanda demasiada de recursos de infra-estrutura na rede, aumentando a escalabilidade do sistema na oferta de conteúdos defasados no tempo, propõe-se aproveitar o fato de que estes conteúdos já estão sendo transmitidos para outros usuários (durante a transmissão linear dos mesmos ou quando requisitado deslocado no tempo), utilizando-se os recursos ociosos dos equipamentos destes usuários e da própria rede para, em um primeiro momento, armazenar os conteúdos transmitidos em unidades de armazenamento dispersas na rede e, quando solicitado, auxiliar na distribuição defasada destes conteúdos utilizando o paradigma P2P, o que permite reduzir os problemas descritos de demanda excessiva de recursos na rede (LEE, JACK Y. B.; LEUNG, 2002).

Desta maneira, é proposta neste trabalho a utilização de um algoritmo modificado de P2P, que não depende da pré-existência do conteúdo todo para permitir a distribuição do mesmo e é utilizado para solucionar os problemas de escalabilidade na oferta de conteúdo defasado no tempo e reduzir a demanda de recursos de infra-

⁷ *Best-effort* se refere ao modelo de serviço de rede no qual não há qualquer garantia quanto à entrega dos dados ou qualquer garantia de nível de qualidade de serviço ou prioridade aos usuários.

⁸ http://www.timewarnercable.com/Corporate/Products/DigitalCable/enhanced_tv_services.html

estrutura. Além disso, a solução proposta possibilita a distribuição de conteúdo VoD com funcionalidades de PVR (i.e., pausar, retroceder e avançar) empregando-se a mesma arquitetura de distribuição P2P.

1.3 Método

O método utilizado neste trabalho envolve pesquisa aplicada, com o estudo e a análise das diversas tecnologias correlatas, o desenvolvimento de uma arquitetura inovadora para fornecer diversos serviços de IPTV, em especial o serviço de apresentação deslocada no tempo, e a aplicação prática das tecnologias estudadas na implementação de um protótipo da arquitetura especificada.

Realizando-se um levantamento extensivo sobre as diversas possibilidades de como tratar cada aspecto do sistema de IPTV (i.e., transmissão linear, apresentação deslocada no tempo, VoD e funcionalidade de PVR) foi especificada uma arquitetura para o provimento de serviços multimídia sobre redes de dados IP. O primeiro passo consistiu no estudo e na avaliação dos requisitos para distribuição de vídeo em redes de dados, bem como da viabilidade em utilizar *multicast* IP nesta distribuição e de possíveis alternativas de *multicast* em nível de aplicação para suprir esta necessidade. Em seguida, um estudo sobre redes de distribuição de conteúdos foi realizado para identificar técnicas que pudessem ser empregadas na solução dos problemas de escalabilidade e eficiência na distribuição de conteúdos em sistemas IPTV e na oferta de novos serviços. Por último, foi realizado um estudo de protocolos P2P para compartilhamento de arquivos, permitindo compreender em detalhes como estes funcionam e quais partes dos mesmos poderiam ser aproveitadas na construção de uma solução P2P para IPTV, que utilizasse o fato de usuários e rede possuírem ociosidade de recursos que poderiam contribuir para a distribuição de conteúdos e conseqüente melhoria na qualidade do serviço oferecido pelo provedor de IPTV.

Após esta etapa de estudos, tornou-se possível analisar as soluções existentes e propostas especificamente na área de IPTV, apurando-se as vantagens e limitações de cada sistema. A partir daí, foi especificada uma arquitetura detalhada para fornecer os serviços tradicionais de TV (i.e., transmissão linear dos canais e vídeo

sob demanda), e também a possibilidade do usuário de obter e assistir, desde o início, um conteúdo cuja transmissão começou no passado, independentemente da mesma já ter terminado ou não.

Como etapa final deste método de pesquisa aplicada, utilizaram-se os conhecimentos obtidos nas etapas anteriores para implementar um protótipo como prova de conceito da arquitetura proposta, identificando-se soluções de código aberto que puderam ser utilizadas como base do desenvolvimento e implementação do referido protótipo. Este protótipo foi submetido a uma série de testes para se obter a relação do desempenho da solução com diversos parâmetros.

1.4 Organização do Trabalho

Após esta breve introdução do trabalho, descrição da motivação e objetivos, e apresentação do problema e do método utilizado, no capítulo 2 são apresentadas definições importantes utilizadas ao longo do texto, seguindo com a apresentação, no capítulo 3, de uma revisão da literatura importante para a compreensão tanto da arquitetura de IPTV proposta neste trabalho, bem como das possibilidades de evolução deste trabalho e das decisões que foram tomadas no decorrer do seu desenvolvimento. No capítulo 4 são descritas soluções existentes que oferecem serviços de IPTV, as tecnologias utilizadas, os requisitos que devem ser atendidos pela arquitetura proposta, e, finalmente, a arquitetura proposta de IPTV para prover o serviço de apresentação deslocada no tempo dos conteúdos.

No capítulo 5, são expostos detalhes de implementação do protótipo, assim como as limitações do mesmo. Dando continuidade ao trabalho, um cenário de testes foi montado para possibilitar a avaliação de desempenho da solução proposta, do impacto de diversos parâmetros (e.g., tamanho do bloco de vídeo, *peers* com o conteúdo disponível para *upload*, número de servidores de *cache* e capacidade de processamento do equipamento do usuário) na latência de início de exibição, e, por último, do *overhead* de controle gerado pelo sistema. O método de testes, os cenários e os resultados obtidos são apresentados no capítulo 6. E, finalmente, o capítulo 7 contém discussões a respeito da arquitetura proposta e dos resultados de desempenho obtidos, considerações finais e possíveis trabalhos futuros.

2 DEFINIÇÕES

Analisando-se o rápido crescimento na oferta de conteúdos desde os primórdios da difusão de TV até a atual disponibilidade quase que ilimitada de fontes de vídeos, tanto on-line como off-line, observa-se que a diversidade de conteúdos e as maneiras de se assistir conteúdos multimídia evoluíram num ritmo impressionante. No entanto, quando se analisam os tipos de serviço oferecidos ao usuário pode-se perceber que relativamente poucos são conceitualmente novos. Numa abordagem levemente diferente da utilizada por Simpson e Greenfield (2007), identifica-se um conjunto de somente quatro classes de serviços, as quais juntas definem todos os tipos de serviço para o provimento de conteúdos multimídia ao usuário atualmente existentes em sistemas de TV.

2.1 Transmissão Linear

Trata-se de transmissão convencional de conteúdos linearmente (continuamente), sem qualquer possibilidade de interação por parte do usuário. Originalmente oferecido pelas estações de difusão de TV, foi por muito tempo o único serviço disponível aos consumidores. Enquanto que ao passar dos anos tal serviço começou a oferecer maior diversidade de conteúdos, a característica primordial deste serviço não mudou: o usuário somente pode assistir o conteúdo que está sendo transmitido linearmente pela estação de TV naquele instante, sem qualquer opção de interromper a transmissão, retroagí-la ou avançá-la. No entanto, a transmissão linear ainda é responsável por grande parte dos conteúdos transmitidos nos diversos sistemas de TV e sobreviveu a muitas mudanças de paradigma, tais como da TV em branco e preto para a colorida, e da era analógica para a digital.

Com relação ao tipo de conteúdo transmitido linearmente, pode-se diferenciar entre a transmissão de conteúdo pré-gravado (i.e., conteúdos que existem em sua completude no distribuidor quando a transmissão se inicia, como durante a difusão de um filme ou documentário), e a transmissão de conteúdo ao vivo (i.e., conteúdos que estão sendo produzidos simultaneamente com a transmissão, como eventos esportivos).

2.2 PVR (Personal Video Recorder)

Devido às limitações impostas pela transmissão linear de conteúdos (i.e., a impossibilidade de assistir um determinado conteúdo em outro instante, ou interromper a transmissão e continuá-la posteriormente, ou até mesmo repetir alguma cena), o surgimento da funcionalidade de gravação de conteúdos foi um sucesso entre os consumidores, com rápida adoção de equipamentos de VCR (*Video Cassette Recording*) em meados da década de 1970. Ao contrário destes equipamentos analógicos de VCR, o termo “PVR” é freqüentemente utilizado para se referir à gravação digital de vídeo (e referido muitas vezes pelo acrônimo DVR – *Digital Video Recorder*). PVR pode, no entanto, ser utilizado para caracterizar o conceito de gravação de vídeo para uso pessoal como um serviço.

A gravação de vídeo não somente sobreviveu a transição da tecnologia analógica para a digital, mas também evoluiu tremendamente com este processo. Dispositivos como o popular “*TiVo recorder*”⁹ permitem gravação paralela de dois canais ao mesmo tempo em que um terceiro pode ser assistido.

2.3 Vídeo sob Demanda

Embora a funcionalidade de PVR dê ao usuário maior flexibilidade, é necessário configurar o sistema antecipadamente para que um conteúdo específico seja gravado e possa ser assistido posteriormente. O serviço de vídeo sob demanda (VoD – *Video on Demand*) ameniza esse problema, permitindo que usuários assistam os conteúdos oferecidos sem a necessidade de qualquer configuração prévia. Porém, tipicamente não há correlação entre os conteúdos oferecidos através do serviço de VoD e os transmitidos linearmente nos canais de TV. Caso o usuário perca a transmissão linear de um conteúdo em um canal (e.g., o noticiário do dia), ou perca o início de um programa, mesmo com o serviço de VoD este usuário geralmente não conseguirá obter as partes perdidas.

Vídeo sob demanda é caracterizado pela entrega de conteúdos de vídeo aos

⁹ [http:// www.tivo.com](http://www.tivo.com)

usuários, iniciado em um momento escolhido pelo usuário. A transmissão é freqüentemente realizada por um servidor através de um fluxo *unicast* para cada um dos usuários. Outros mecanismos de distribuição existentes para VoD consideram *buffers* para armazenar parcialmente o conteúdo antes de iniciar a exibição do mesmo, e outros ainda distribuem estes conteúdos através de redes P2P, nas quais os usuários compartilham os conteúdos que obtêm, auxiliando na sua distribuição (WAUTERS ET AL., 2006).

2.4 Apresentação Deslocada no Tempo

O mais recente dos quatro serviços aqui definidos, e foco deste trabalho, o serviço de **apresentação deslocada no tempo** pode ser definido como um serviço que possibilita ao usuário **assistir a um conteúdo**, que foi transmitido linearmente em determinado canal, **deslocado no tempo**. Ou seja, o usuário pode começar a assistir determinado conteúdo **desde o início** (ou qualquer instante já transmitido) embora a transmissão linear do mesmo já tenha começado ou até mesmo terminado (WAUTERS ET AL., 2006). O único exemplo comercial desta funcionalidade é um serviço da empresa de TV a cabo Time Warner, chamado “*Start Over*”, que começou a ser lançado em novembro de 2005¹⁰.

Embora sob a perspectiva do usuário em alguns casos estes serviços possam ser percebidos de maneira parecida, duas distinções com relação à solução tecnológica devem ficar claras: primeiro com relação às soluções de PVR com armazenamento local ou remoto, uma vez que as mesmas dependem da configuração do usuário para armazenar ou não determinado conteúdo que está sendo transmitido linearmente; e segundo com relação ao tipo de conteúdo sendo transmitido linearmente, entre conteúdos ao vivo sendo transmitidos e conteúdos pré-gravados ou cuja transmissão já tenha sido finalizada, uma vez que para estes últimos a apresentação deslocada no tempo poderia ser reduzida tecnologicamente a distribuição de vídeo sob demanda, enquanto que para as transmissões de conteúdo ao vivo em andamento isso não é possível devido a não existência do conteúdo completo no provedor quando o usuário solicita o serviço.

¹⁰ <http://www.timewarner.com/corp/newsroom/pr/0,20812,1192749,00.html>

2.5 Considerações Finais

Neste capítulo foram definidos os quatro tipos de serviços que cobrem a atual oferta das operadoras de TV no que se refere à distribuição de vídeo. É importante observar que definições levemente diferentes para os mesmos termos podem ser encontradas tanto na literatura e principalmente nos nomes comerciais utilizados pelos provedores de TV, porém muitas vezes se referindo a apenas um caso particular de determinado serviço. Por esse motivo, definiram-se neste trabalho os quatro serviços apresentados e ao longo deste texto segue-se esta definição.

Note ainda que a definição do serviço de apresentação deslocada no tempo adotada neste trabalho é a mais abrangente possível, não se restringindo apenas à apresentação dos conteúdos após o término dos mesmos (o que poderia ser reduzido simplesmente a distribuição destes através de VoD), nem a simples possibilidade de pausar a exibição, fazer replay ou retroceder a algum trecho assistido anteriormente (equivalente a DVR, com armazenamento local em *buffer* dos conteúdos assistidos), nem ainda necessitando pré-configuração por parte do usuário para a gravação (também equivalente a DVR, com armazenamento em disco para exibição futura).

Outra consideração importante com relação às definições adotadas neste trabalho é que, embora o termo IPTV seja utilizado algumas vezes para descrever somente o serviço de **transmissão linear** sobre redes IP, neste trabalho IPTV é interpretado como o sistema como um todo, capaz de oferecer os diversos serviços de distribuição de vídeo, e não somente a transmissão linear, através de uma infraestrutura de rede IP proprietária.

3 DISTRIBUIÇÃO DE VÍDEO EM REDES DE DADOS

Neste capítulo, apresenta-se a revisão da literatura sobre diversos assuntos correlatos a esta dissertação e utilizados no desenvolvimento da arquitetura de IPTV proposta. Inicialmente, são especificados os requisitos para distribuição de vídeo (e conteúdos multimídia em geral) em redes de dados, seguindo-se com a apresentação e a discussão de diversas técnicas de multidifusão em redes IP, tanto de *multicast* nativo como de *overlay*¹¹. Após isso, são introduzidos os conceitos de Redes de Distribuição de Conteúdo (CDN – *Content Distribution Network*) e, posteriormente, são apresentados protocolos P2P empregados em sistemas de compartilhamento de arquivos.

3.1 Requisitos para Distribuição de Vídeo

Tipicamente, um sistema de multidifusão de vídeo tem uma única fonte de transmissão dedicada que está presente durante toda a sessão. O endereço da fonte é conhecido previamente, servindo como ponto de conexão para novos usuários que ingressam na sessão.

As características deste tipo de sistema englobam: larga escala, correspondente a dezenas de milhares de usuários simultaneamente; demanda de desempenho, envolvendo requisitos de largura de banda de centenas ou milhares de *kilobits* por segundo por usuário; transmissão ininterrupta e sincronizada; e degradação gradual da qualidade para permitir a entrega do conteúdo de maneira flexível e adaptativa, acomodando largura de banda heterogênea e dinamicidade da rede e do sistema como um todo.

Com relação aos requisitos de largura de banda, atualmente a maioria dos vídeos sobre Internet requerem taxa de transmissão variando de 100 kbps a 1 Mbps (e.g., vídeos do YouTube¹² utilizam aproximadamente 320 kbps, sendo que os vídeos mais recentes possuem qualidade melhor associada a maior taxa de transmissão, e

¹¹ *Overlay* se refere a técnicas alternativas de multidifusão que utiliza os próprios participantes do grupo para distribuir os dados de maneira mais eficiente do que o modelo cliente-servidor em redes onde não existe *multicast* IP nativo, como por exemplo na Internet.

¹² <http://www.youtube.com>

o Joost¹³ trabalha com taxas de transmissão em torno de 750 kbps, dependendo do conteúdo e do canal). Para vídeos com qualidade de TV, a taxa de transmissão, dependendo da codificação utilizada, fica entre 1 e 4 Mbps. E, finalmente, quando se consideram conteúdos de alta definição (HDTV – *High Definition Television*), visto que a oferta destes está crescendo rapidamente e os sistemas de IPTV devem estar preparados para tratar tal resolução, as taxas de transmissão podem ultrapassar os 10 Mbps, trabalhando-se hoje em dia com taxa de transmissão entre 4 e 12 Mbps (XIAO, Y. ET AL., 2007).

Pode-se dimensionar melhor essa necessidade de largura de banda citando dois dentre os eventos de maior escala no que tange a multidifusão de vídeo pela Internet: a transmissão pela AOL do concerto “*Live 8*” em julho de 2005, que chegou a ter 175.000 expectadores simultâneos¹⁴, e a transmissão do campeonato da NCAA pela CBS em março de 2006, que alcançou a marca dos 268.000 expectadores simultâneos¹⁵. Mesmo considerando uma taxa de transmissão relativamente baixa de 400 kbps, a transmissão da CBS precisaria, num modelo cliente-servidor tradicional, mais de 100 Gbps de largura de banda e servidores capazes de sustentar tal carga, transmitindo para todos aqueles clientes simultaneamente. Como comparação, a Akamai, maior provedora de serviço de CDN (*Content Distribution Network* – Rede de Distribuição de Conteúdo) comercial, alega possuir capacidade agregada de 200 Gbps com suas dezenas de milhares de servidores. (LIU, J. ET AL., 2008)

Com relação aos requisitos de tempo real, enquanto interatividade pode não ser crítica e pequenos atrasos são toleráveis através de armazenamento em *buffer*, é essencial exibir o vídeo ininterruptamente.

Estas características de um sistema de transmissão linear de conteúdo ao vivo levam a um cenário de aplicação único que difere de aplicações de vídeo sob demanda, vídeo conferências e transferência de arquivos. Dentre estas aplicações, vídeo sob demanda necessita largura de banda, porém, os usuários são assíncronos e a latência não é crítica como em transmissões ao vivo. Por outro lado, aplicações de vídeo conferência são interativas, sendo a latência ainda mais crítica que na

¹³ <http://www.joost.com>

¹⁴ <http://www.thewhir.com/marketwatch/aol070505.cfm>

¹⁵ <http://www.techweb.com/wire/networking/183700547>

transmissão linear, além de serem, tipicamente, multiponto, devendo possibilitar que qualquer participante transmita dados para todos os outros. No entanto, estas aplicações são geralmente de menor escala, envolvendo somente algumas centenas de participantes quando muito. Finalmente, aplicações de compartilhamento de arquivos P2P envolvem, também, a distribuição de informações para dezenas de milhares de participantes, porém os requisitos de tempo-real tornam a transmissão linear de vídeo muito mais desafiadora.

3.2 **Multicast**

Para permitir a distribuição de conteúdos num sistema de IPTV de maneira escalável se faz necessário que o provedor do conteúdo transmita este conteúdo para múltiplos destinos, sendo que se uma conexão para cada cliente for estabelecida para a transmissão deste conteúdo, um número muito grande de conexões seria necessário e a banda necessária cresceria linearmente com o número de usuários assistindo a determinado conteúdo. Esta transmissão ponto-a-ponto acarretaria sobrecarga do servidor onde está o conteúdo bem como da rede como um todo, pois múltiplas cópias do mesmo conteúdo estariam sendo transmitidas pelos diversos enlaces ao mesmo tempo. Com isso, tal tipo de transmissão não escala para um grande número de usuários assistindo a um mesmo conteúdo que um sistema de IPTV pode ter. A fim de tratar tal problema, algum método de transmissão ponto-multiponto pode ser utilizado, tal como o *multicast* IP.

Ainda hoje muitos provedores de serviço de Internet (ISPs – *Internet Service Providers*) estão relutantes em habilitar o serviço de roteamento *multicast* por diversas razões. O modelo de serviço e arquitetura do *multicast* IP não provê, eficientemente, muitas funcionalidades necessárias para uma implantação de *multicast* para fins comerciais, tais como gerenciamento de grupo, incluindo autorização para criação de grupos e controle de acesso para limitar quem pode receber e quem pode transmitir dados de e para um determinado grupo *multicast*; alocação distribuída de endereços; segurança, incluindo proteção contra ataques em roteadores ou sessões *multicast* (e.g., ataque de inundação ou *flooding* – no qual altas taxas de dados inúteis são transmitidas em um grupo *multicast* causando

congestionamento e perda de pacotes – ou forjando uma fonte de dados e transmitindo dados alternativos, alterando-se o conteúdo de uma sessão), bem como suporte a mecanismos de verificação de integridade; e suporte para gerenciamento de rede. (DIOT ET AL., 2000)

Devido aos problemas citados acima, a implantação global de *multicast* IP na Internet ainda não é uma realidade, e inúmeros estudos de soluções alternativas para serviços de comunicação de grupo foram e estão sendo desenvolvidos, visando suprir a falta de roteamento *multicast* IP nativo. (CHU ET AL., 2002; EL-SAYED; ROCA; MATHY, 2003; LAO ET AL., 2005; SRIPANIDKULCHAI ET AL., 2004)

Nestas soluções alternativas, alguns nós formam uma rede sobreposta (*overlay*), e estruturas de distribuição tais como árvores *multicast* são construídas sobre esta rede. Os pacotes são replicados somente nos nós participantes, e transmitidos através de túneis entre estes nós. No entanto, existem diferenças significativas entre as diversas soluções, não somente com relação aos algoritmos utilizados, mas também na concepção do tipo de equipamento utilizado como nó na criação da rede sobreposta (i.e., equipamento de usuário final ou servidor dedicado no interior da rede). A taxonomia de arquiteturas para *multicast* utilizada neste trabalho para distinguir os diversos tipos de solução é descrita no tópico 3.2.1.

Comparativamente, estas soluções são mais simples de implantar na Internet em nível global do que o *multicast* IP, uma vez que não dependem da habilitação de *multicast* nos roteadores por parte de todos os ISPs, nem de protocolos de roteamento *multicast* inter-domínio. No entanto, a eficiência do *multicast* é sacrificada, pois pacotes de dados podem ser transmitidos através de um mesmo enlace físico múltiplas vezes dependendo de onde os participantes estão localizados e como são organizados, além dos caminhos utilizados serem possivelmente não-ótimos, e, principalmente devido à dinamicidade dos participantes, a complexidade de se manter a rede sobreposta é muito alta, sendo que as chances de falha na entrega para parte dos participantes é muito mais alta do que no *multicast* IP nativo.

Além disso, dentro de um domínio de rede específico, controlado por um único provedor (que é o caso de um provedor que deseja oferecer IPTV sobre sua rede), a implantação de *multicast* IP é viável e não acarreta os problemas que as soluções alternativas de *multicast* apresentam, principalmente com relação à estabilidade, ao desempenho ótimo e à não dependência de equipamentos dedicados adicionais.

Adicionalmente, as deficiências do *multicast* IP apresentadas anteriormente são facilmente contornáveis dentro de um domínio para o cenário de IPTV, podendo-se proibir a ingestão de dados *multicast* pelos usuários do sistema através de políticas de *firewall*, evitando os problemas relacionados a gerenciamento de grupo, limitação de quem pode transmitir dados para um determinado grupo *multicast* e ataques em roteadores ou sessões *multicast*. Ainda com relação às deficiências descritas anteriormente, uma vez que todo o tráfego *multicast* está sob controle do provedor de rede, os problemas com relação a suporte para gerenciamento de rede do *multicast* também desaparecem.

Sendo assim, pelos motivos apresentados acima, mostra-se que para o cenário de IPTV, onde o serviço é oferecido por um provedor dentro de seu domínio de rede, a utilização de *multicast* IP nativo é o mais indicado. Nos tópicos a seguir, apresenta-se uma taxonomia de arquiteturas de multidifusão para *multicast* na Internet e detalhes do *multicast* IP e das alternativas de *multicast* em nível de aplicação.

3.2.1 Taxonomia de Arquiteturas para *Multicast* na Internet

Existem diversas abordagens para se fornecer serviço de *multicast* na Internet. A classificação utilizada neste texto segue a taxonomia apresentada por Liu et al. (2008). Esta taxonomia é ilustrada na Figura 3.1.

Figura 3.1 - Taxonomia de arquiteturas para *multicast* na Internet (LIU, J. ET AL., 2008)

As arquiteturas *Router-Based* são as construídas sobre uma rede IP que emprega a capacidade nativa dos roteadores de lidar com o tráfego *multicast*, independentemente do protocolo utilizado por estes. Do outro lado estão as arquiteturas que não dependem da capacidade dos equipamentos de rede (*No Router Support*).

Para prover funcionalidade de *multicast*, sem suporte dos roteadores, existem três diferentes abordagens: o “centrado em infra-estrutura” (*Infrastructure-Centric*), que utiliza um conjunto de servidores para criar uma rede sobreposta (sobre a rede física) na qual estes servidores lidam com o tráfego *multicast* através de transmissões *unicast* entre os servidores no nível de rede (e.g., CDNs); o “em nível de aplicação sem infra-estrutura” (*Application End-points Only*), no qual os equipamentos dos usuários finais se organizam a fim de criar *multicast* no nível de aplicação utilizando transmissões *unicast* entre eles; e uma mistura destas duas abordagens, o “em nível de aplicação com infra-estrutura” (*Application End-points with infrastructure support*), que utiliza equipamentos dedicados (*overlay proxy nodes*) posicionados estrategicamente na rede, além dos equipamentos dos usuários finais com o objetivo de facilitar o gerenciamento dos grupos *multicast* e a formação da estrutura de distribuição *multicast*.

Vale notar, no entanto, que nem toda CDN provê a funcionalidade de *multicast*, pois existem CDNs com o simples objetivo de replicar conteúdos próximo a borda da rede, sem necessariamente lidar com transmissão linear dos dados, i.e, sem manter o fluxo entre servidores e clientes sincronizado. Porém, as CDNs que tratam da distribuição de fluxos ao vivo de vídeo para usuários pertencem à categoria centrado em infra-estrutura, como por exemplo, o Akamai Stream OS¹⁶.

3.2.2 **Multicast IP nativo**

É um esquema de distribuição para grupos em redes de computadores que teve muitas propostas desenvolvidas no passado. Apesar de existirem vários padrões de transmissão *multicast*, este esquema de distribuição ainda permanece marginalizado na maioria das redes. Porém, com a crescente demanda de transmissões de vídeos

¹⁶ http://www.akamai.com/html/solutions/stream_os.html

– inclusive em alta definição – através das redes de dados, começa-se a reconhecer a necessidade da utilização do *multicast*, e pesquisadores tentam adaptar protocolos antigos e até mesmo desenvolver novos, para suprir esta necessidade de *multicast* em redes convergentes.

Levando em consideração os problemas do *multicast* IP, apresentados anteriormente (falta de autorização para criação de grupos e controle de acesso, alocação distribuída de endereços *multicast*, proteção contra ataques, suporte a mecanismos de verificação de integridade, e suporte para gerenciamento de rede) (DIOT ET AL., 2000), a implantação global do mesmo é bastante difícil. Contudo, a sua implantação por um provedor de serviço em sua rede de acesso (rede proprietária) é perfeitamente plausível e evita os problemas de eficiência remanescentes nas alternativas de *multicast* em nível de aplicação e a necessidade de servidores de infra-estrutura como no caso das CDNs.

As deficiências do *multicast* IP apresentadas anteriormente são facilmente contornáveis dentro de um domínio para o cenário de IPTV, podendo-se proibir a ingestão de dados *multicast* pelos usuários do sistema através de políticas de *firewall*, evitando os problemas relacionados a gerenciamento de grupo, limitação de quem pode transmitir dados para um determinado grupo *multicast* e ataques em roteadores ou sessões *multicast*. Além disso, uma vez que todo o tráfego *multicast* está sob controle do provedor de rede, os problemas com relação a suporte para gerenciamento de rede do *multicast* também desaparecem.

Por isso, a seguir são descritos brevemente dois protocolos para se obter *multicast* IP: o IGMP (*Internet Group Management Protocol*), protocolo de gerenciamento de grupos, e o PIM-SM (*Protocol Independent Multicast – Sparse Mode*), protocolo de roteamento de dados *multicast*. Estes dois protocolos são os utilizados pelo sistema implementado como prova de conceito da arquitetura proposta nesta dissertação.

2.2.2.1 IGMP (*Internet Group Management Protocol*)

Os nós de rede (roteadores e comutadores/*switches*) necessitam um protocolo para controlar a admissão e o egresso de dispositivos nos grupos *multicast* em um ambiente de rede IP. O protocolo de gerenciamento de grupos IGMP (FENNER, W., 1997) foi desenvolvido com esta finalidade, utilizando tabelas para armazenar

informações sobre os dispositivos que estão conectados em um grupo *multicast*. Estas tabelas são armazenadas nos próprios nós.

Além das tabelas, o IGMP especifica as mensagens que são trocadas entre dispositivos e elementos de rede. Estas mensagens são responsáveis por informar os dispositivos de rede sobre o estado dos dispositivos e dos grupos *multicast*. Atualmente, o protocolo IGMP encontra-se em sua terceira versão (IGMPv3) (CAIN ET AL., 2002).

2.2.2.2 PIM-SM (*Protocol Independent Multicast - Sparse Mode*)

O protocolo PIM (*Protocol Independent Multicast*), no modo esparsa (SM – *Sparse Mode*) (ESTRIN ET AL., 1998), foi um dos primeiros protocolos desenvolvidos para roteamento *multicast* e pode ser considerado o mais popular, estando implementado na maioria dos roteadores de grande porte disponíveis atualmente no mercado.

O PIM-SM é um protocolo de roteamento para *multicast* baseado em um ponto central chamado *Rendezvous Point* (RP). Cada dispositivo que pretende ingressar em um grupo *multicast* pede ao RP primeiro e, somente após isso, o tráfego é redirecionado para o melhor caminho entre origem e destino. É importante observar que o PIM-SM necessita do IGMP para manter o controle dos dispositivos nos grupos, pois ele trata somente do roteamento *multicast*, não desempenhando tarefas de gerenciamento de grupo.

A distribuição inicia-se quando um dispositivo interessado em transmitir um fluxo (fonte) começa a transmissão dos pacotes ao RP. Este recebe o tráfego e aguarda possíveis clientes. Se nenhum cliente requisita o fluxo deste grupo específico, o RP envia uma mensagem *unicast* para a fonte, pedindo que esta pare de transmitir pacotes enquanto não há interesse.

Quando um cliente quer ingressar em um grupo, este envia uma mensagem de ingresso (*join*) para o RP (Figura 3.2), passando, então, a receber o fluxo encaminhado pelo RP.

Figura 3.2 - Cliente ingressando em um grupo *multicast*

Após começar a receber o fluxo, o cliente fica sabendo o endereço de origem (da fonte), podendo encontrar o melhor caminho através do protocolo SPT (*Shortest Path Tree*) até a origem. Neste momento, o cliente poderá enviar uma mensagem de ingresso diretamente à fonte (Figura 3.3).

Figura 3.3 - Tráfego inicial através do RP

A fonte, ao receber a mensagem de ingresso, redireciona os pacotes utilizando o caminho da SPT, liberando o RP de uma carga desnecessária (Figura 3.4). Após este novo caminho ter sido estabelecido e os pacotes começarem a chegar ao cliente através dele, a fonte envia uma mensagem de poda (*prune*) para o RP como uma notificação de que o caminho pode ser podado.

Figura 3.4 - Rota final

Neste instante, o RP fica livre, mas qualquer outro novo cliente para o grupo *multicast* força o RP a começar a encaminhar os pacotes uma vez mais.

O primeiro estágio de roteamento através do RP pode ser benéfico para a rede, pois ele é o único roteador que precisa manter controle sobre as fontes ativas na rede. Mas, por outro lado, isto pode não ser tão escalável para grandes grupos com milhares ou milhões de usuários, tornando o RP o gargalo do processo uma vez que força o tráfego a ser concentrado inicialmente num único nó da rede. O SSM (*Source Specific Multicast*) simplifica este processo, não possibilitando conexões do tipo multiponto a multiponto (como originalmente suportado). Isto não é um problema, por exemplo, para distribuição de canais de TV em um cenário de IPTV, e facilita todo este processo, eliminando a necessidade do RP, criando as árvores *multicast* diretamente com a raiz no transmissor.

3.2.3 **Multicast em Nível de Aplicação**

Para atender aos requisitos de distribuição de vídeo descritos no item 3.1, a eficiência da rede sobreposta (*overlay*) é absolutamente necessária, tanto com relação à rede quanto à aplicação. No entanto, alguns segundos de atraso podem ser tolerados em alguns casos, quando a aplicação não é interativa. Um elemento chave para o sucesso de uma solução deste tipo é a possibilidade de escalar e fazer distribuição de carga, uma vez que o sistema precisa estar preparado para suportar dezenas de milhares de usuários simultaneamente quando necessário (e.g.,

transmissão de um canal de TV em horário nobre, ou durante a transmissão de um evento esportivo ao vivo), e a rede sobreposta precisa suportar tal carga com *overhead* associado plausível.

Além disso, como a construção da rede sobreposta acontece de maneira distribuída (i.e., não existe controle centralizado) e precisa suportar o dinamismo do grupo (i.e., *peers* entram e saem do sistema quando querem), o sistema precisa se auto-organizar, adaptando-se tanto a esta dinamicidade dos usuários finais quanto a variações de longo prazo nas características dos caminhos da Internet (e.g., tráfego nos enlaces de rede). Tais sistemas precisam também assegurar-se de que a banda total que um usuário deve contribuir não excede sua capacidade inerente de banda. E por último, o sistema precisa levar em consideração que uma grande parcela dos usuários pode estar conectada através de NAT (*Network Address Translation*) ou *firewall*, e tais restrições de conectividade reduzem a capacidade do *overlay*, visto que estes usuários podem não contribuir com seus recursos.

Existem três abordagens para a construção de uma rede sobreposta para permitir tráfego *multicast* em nível de aplicação: baseada em árvores (árvore única ou múltiplas árvores); dirigida por dados (*data-driven*) e híbrida. Nas próximas seções, as diferenças entre estas abordagens são explicadas e algumas propostas de cada abordagem são apresentadas. A abordagem baseada em árvores é exemplificada através do PeerCast (árvore única) e do SplitStream (múltiplas árvores). Como dirigida por dados, o CoolStreaming/DONet e o Chainsaw são apresentados. E por último, são descritos o Bullet, o Chunkyspread e o mTreebone, três exemplos da abordagem híbrida.

3.2.3.1 Abordagem Baseada em Árvores

A idéia básica desta abordagem é construir uma árvore utilizando os elementos envolvidos (*peers*) como nós. Fazendo isto, alguns nós estarão conectados diretamente à fonte dos dados (nó raiz da árvore), outros serão nós internos da árvore, e a maioria será do tipo nó folha.

Construindo uma única árvore (e.g., PeerCast (DESHPANDE; BAWA; GARCIA-MOLINA, 2002)), não são necessários algoritmos de codificação de vídeo

sofisticados, uma vez que o vídeo não precisa ser dividido em múltiplos fragmentos (como será explicado a seguir para o caso de múltiplas árvores). Por outro lado, a estrutura de árvore precisa ser mantida, pois os nós podem ingressar ou deixar o grupo a qualquer instante. Além disso, é crítico assegurar que a estrutura seja otimizada para oferecer bom desempenho a todos os receptores. No caso de falha de um nó, se o mesmo **não** for nó folha, ou seja, o nó possui filhos, a árvore ficará desconexa e os descendentes deste nó pararão de receber a transmissão até que a árvore seja restabelecida. Portanto, o sistema apresenta desempenho ruim em caso de falha de nós (e.g., nós deixando o sistema).

Outro problema desta abordagem é que como a maioria dos nós nesta estrutura são nós folha, suas bandas disponíveis não serão utilizadas. Assim, geralmente, existe um número pequeno de usuários que utilizam muita banda para suportar os nós folhas que não contribuem na distribuição do fluxo. Isto é visto como injusto, uma vez que os usuários de um sistema P2P esperam que todos compartilhem seus recursos.

Para tratar esses problemas, diversas propostas sobre a utilização de múltiplas árvores (e.g., SplitStream (CASTRO, MIGUEL ET AL., 2003), CoopNet (PADMANABHAN ET AL., 2002) e THAG (TIAN ET AL., 2005)) foram desenvolvidas com a idéia de dividir o fluxo (*stream*) em múltiplos fragmentos (*stripes*) e construir uma árvore diferente para cada fragmento. Sendo assim, se um nó deixa o sistema, seus descendentes pararão de receber apenas um fragmento do fluxo, podendo ainda exibir o conteúdo (com qualidade prejudicada ou ainda com a mesma qualidade se existir redundância suficiente nos outros fragmentos). Adicionalmente, com múltiplas árvores é possível organizar o sistema de modo que um nó interno em uma árvore seja folha em todas as outras, fazendo com que todos contribuam na distribuição do conteúdo.

Uma das propostas que faz uso de múltiplas árvores é o SplitStream (CASTRO, MIGUEL ET AL., 2003), que para melhorar a resiliência do *overlay* evita a ruptura completa do fluxo, devido a falhas em nós de alto nível, e a sub-utilização da banda disponível nos nós folhas, construindo múltiplas árvores. A fonte codifica o fluxo de vídeo em sub-fluxos, chamados fragmentos (*stripes*), e distribuí cada fragmento em uma árvore distinta. Ao receber um conjunto de fragmentos de diversas árvores, o receptor consegue reconstruir o fluxo com estes fragmentos de maneira que a

qualidade obtida dependa do número de fragmentos recebidos. A Figura 3.5 ilustra as múltiplas árvores (neste caso duas), distribuindo-se um fragmento por vez através de cada árvore.

Figura 3.5 - SplitStream (CASTRO, MIGUEL ET AL., 2003)

Neste exemplo, o conteúdo original é dividido em dois fragmentos e uma árvore *multicast* independente é construída para cada fragmento de maneira que um nó é interno em uma árvore e folha na outra. Para o fragmento 1, os nós folhas são 5, 6, 7 e 8, enquanto que para o fragmento 2 os nós folhas são 2, 3, 4 e 7.

Desta maneira, a resiliência do sistema é melhorada, sendo que determinado nó não é completamente desconectado devido à falha de um antecessor em uma árvore, e a banda potencial de todos os nós pode ser utilizada desde que cada nó não seja folha em pelo menos uma árvore.

Note, no entanto, que a abordagem de múltiplas árvores envolve o uso de algoritmos de codificação especializados (e.g., *Multiple Description Code* (MDC) (GOYAL, 2001; KIM; LEE, SANG-UK, 2000) ou *Erasure Coding of Bulk Data* (BYERS ET AL., 1998; LUBY ET AL., 1997; LUBY, 2002)), que permitem um receptor ter qualidade degradada quando recebe dados somente de um subconjunto de árvores.

3.2.3.2 Abordagem Dirigida por Dados

As estruturas de *overlay* tentam imitar a estrutura de árvores de roteamento do *multicast* IP, utilizando equipamentos de usuários finais (programáveis e extensíveis) como nós internos na árvore, atuando como repetidores para outros nós. *Overlays* têm se mostrado muito promissores para aplicações *multicast*; no entanto, muitas pesquisas argumentam que estruturas em árvore possuem limitações fundamentais tanto com relação à largura de banda quanto à confiabilidade. Uma das maiores dificuldades com árvores está relacionada ao fato de que a largura de banda decresce descendo a árvore. Qualquer perda no alto da árvore reduzirá a banda disponível para receptores abaixo na árvore. A largura de banda disponível para qualquer usuário é limitada pela banda disponível de seu antecessor na árvore.

Portanto, outra abordagem para distribuição de conteúdo ao vivo, que tem recebido muita atenção, é a **dirigida por dados**, a qual utiliza a disponibilidade de dados para guiar o fluxo de dados ao invés de manter uma estrutura explícita (tal como uma árvore) para distribuição de dados. Para tal finalidade existem dois tipos de algoritmos: os algoritmos “*gossip-based*”, nos quais os nós escolhem outros nós para transmitir dados de maneira aleatória, e os algoritmos “*pull-based*”, nos quais os equipamentos de usuários finais trocam informação sobre os dados que cada um possui, solicitando explicitamente os dados que precisam.

Os requisitos de tempo real de uma transmissão ao vivo implicam que os segmentos do fluxo de vídeo precisam ser obtidos em tempo hábil para a exibição, sendo necessário um mecanismo que consiga agendar os segmentos que devem ser obtidos dos vários pares (*peers*) para garantir que a exibição seja executada em um tempo limite máximo. De outra maneira, a exibição do conteúdo poderia ser interrompida ou degradada.

Existem diversas propostas de solução que utilizam esta abordagem, tais como CoolStreaming (ZHANG, X. ET AL., 2005), Chainsaw (PAI ET AL., 2005) e AnySee (LIAO ET AL., 2006), entre outros. Nesta seção, o CoolStreaming será apresentado com a finalidade de mostrar o funcionamento da abordagem dirigida por dados.

Como o CoolStreaming segue esta abordagem dirigida por dados, não existe uma estrutura formal pré-definida para transmitir os dados através dos clientes/*peers*, i.e., uma estrutura formal para distribuição dos dados. Vale notar que o CoolStreaming

utiliza um algoritmo *pull-based* para obter os dados que ele necessita, i.e., clientes explicitamente requisitam os segmentos que precisam. A vantagem deste tipo de algoritmo comparado aos *gossip-based* está no fato de que a redundância pode ser evitada uma vez que um nó solicita determinado dado somente se ele ainda não o possui.

Um fluxo de vídeo é dividido em segmentos de tamanho uniforme e a disponibilidade dos segmentos ativos no *buffer* de um nó é representada através de um mapa do *buffer* (*Buffer Map* – BM). Os nós mantêm um conjunto de parceiros com os quais trocam informações de disponibilidade de dados (trocando BMs). Isso permite a um nó definir qual segmento solicitar a determinado parceiro. Na Figura 3.6 podem ser vistos diversos nós trocando seus BMs, no qual o nó A é a fonte do conteúdo.

Figura 3.6 - CoolStreaming/DONet (ZHANG, X. ET AL., 2005)

O progresso da exibição do conteúdo nos clientes é semi-sincronizado (i.e., os clientes estão assistindo a mesma parte do conteúdo, porém não **exatamente** o mesmo instante), e qualquer segmento obtido após seu instante de exibição será inútil. Portanto, o desempenho do algoritmo de agendamento é muito importante para a exibição contínua.

Em um sistema de transferência de arquivos, tal como o BitTorrent, cada cliente/*peer* sabe as partes do arquivo completo que ele já possui e as que ele precisa obter, sendo que a ordem na qual estes pedaços são obtidos não faz diferença, pois a meta é obter o arquivo completo antes de utilizá-lo. O CoolStreaming utiliza a mesma idéia que o BitTorrent, porém como um fluxo de vídeo possui requisitos temporais, o conceito de uma janela deslizante representando a parte ativa do fluxo é utilizado, sendo que a parte representada pela janela é a única que realmente

importa num determinado instante, uma vez que os clientes precisam desta parte para exibir o vídeo. A Figura 3.7b mostra a janela deslizante e o ponto de exibição.

Figura 3.7 - Captura do *buffer* no (a) BitTorrent e (b) CoolStreaming (LIU, J. ET AL., 2008)

O CoolStreaming adota uma janela deslizante de 120 segmentos, com um segundo de vídeo cada. Portanto, um BM consiste de um vetor (*array*) de 120 bits, cada um indicando a disponibilidade do segmento correspondente. Um número de seqüência é usado para localizar a janela deslizante no fluxo de vídeo.

O algoritmo de agendamento precisa levar em conta o tempo limite para exibição de cada segmento e a taxa de transmissão heterogênea dos parceiros.

No estudo realizado por Liu et al. (2005, p. 2109, tradução nossa), é concluído que o *CoolStreaming* “apresenta *overhead* de controle relativamente baixo (1% do tráfego de vídeo)”, além de que “a continuidade na exibição dos vídeos é muito melhor comparada a apresentada por *overlays* baseados em árvores, particularmente em ambientes de alta dinamicidade”, e que “o atraso fim-a-fim é comparável com o de *overlays* baseados em árvores”.

Observe que a avaliação acima foi realizada pelos desenvolvedores do CoolStreaming, contudo, não existem detalhes suficientes sobre os critérios de comparação, parâmetros e procedimentos utilizados para comprovar a validade desta avaliação. Adicionalmente, a abordagem baseada em árvore mencionada refere-se única e exclusivamente a uma árvore, e não a abordagem de múltiplas árvores que, como explicado no tópico 3.2.3.1, é uma evolução dessa abordagem que melhora o desempenho do sistema. Finalmente, enquanto que a latência para o início da exibição pode ser da mesma ordem nas duas abordagens, a afirmação com relação ao atraso fim-a-fim não é muito fundamentada, considerando que no

CoolStreaming há um *buffer* de 120 segundos e, portanto, o atraso do instante de exibição com relação ao conteúdo sendo transmitido pela fonte em um dado momento é muito alto (minutos) comparado ao baixo atraso apresentado nas abordagens baseadas em árvore (centenas de milissegundos ou alguns segundos).

3.2.3.3 Abordagem Híbrida

Por um lado, o ponto forte da abordagem dirigida por dados é a robustez, porém a falta da relação ordenada pai-filho implica que os dados precisam ser requisitados aos vizinhos. Isto implica em alguma forma de comunicação de controle entre os participantes, na qual o envio de notificação sobre cada pedaço que os nós obtêm torna o *overhead* excessivo, enquanto que **notificações periódicas** contendo mapa do *buffer* dentro de uma janela deslizante **reduzem o *overhead***, porém **aumentam a latência** (já que os participantes aguardam determinado intervalo antes de notificar que possuem os pedaços obtidos). Por outro lado, o mecanismo de entrega de uma árvore é eficiente, porém quando um nó interno da árvore falha seus descendentes param de receber os dados. Além disso, a direção pré-definida do fluxo impossibilita a utilização total da banda entre os parceiros (e.g., entre dois nós folhas).

Dados os benefícios e as desvantagens de cada abordagem, uma questão natural que surge se refere à possibilidade de combinar ambas as abordagens criando uma solução melhor. Na tentativa de se ter melhor eficiência e resiliência, várias soluções híbridas unindo as vantagens da abordagem **baseada em árvore** com a **dirigida por dados** foram propostas (e.g., Bullet (KOSTIĆ ET AL., 2003), Chunkyspread (VENKATARAMAN; YOSHIDA; FRANCIS, 2006; VENKATARAMAN; FRANCIS; CALANDRINO, 2006) e mTreeBone (WANG, F.; XIONG, YONGGIANG; LIU, J., 2007)).

O Bullet foi concebido sobre a idéia de que os dados deveriam ser distribuídos de maneira disjunta para pontos estratégicos na rede. Cada nó é responsável por localizar e recuperar dados faltantes de maneira colaborativa para transmitir conjuntos distintos de dados para vários pontos da rede ao invés de transmitir cópias idênticas dos dados para todos os nós de uma árvore e necessitar mecanismos escaláveis para recuperação de perdas quando a árvore se rompe.

No Bullet, os nós são iniciados se organizando em uma árvore *overlay*. Começando pela raiz da árvore, cada nó transmite um conjunto disjunto de dados para cada um de seus filhos (dependendo da banda disponível), com a meta de manter o mesmo número de cópias de cada segmento de dado sobre todos os participantes. Feito isso, um mecanismo (“*gossip-based*”, baseado na comunicação com *peers* escolhidos aleatoriamente) é empregado para permitir que um nó localize múltiplos parceiros capazes de transmitir os dados que lhe faltam. Portanto, o Bullet monta um *mesh* sobre esta árvore *overlay* arbitrária.

Assim como no SplitStream, algoritmos de codificação especializados, como o *Multiple Description Code* (MDC) (GOYAL, 2001; KIM; LEE, SANG-UK, 2000) ou *Erasure Codes* (BYERS ET AL., 1998; LUBY ET AL., 1997; LUBY, 2002) podem ser utilizados com a finalidade de “fragmentar” os dados. Para assegurar que o *overlay* se comporte de maneira adequada em caso de congestionamento é empregado o TFRC (*TCP Friendly Rate Control*) (FLOYD ET AL., 2000) para transferir os dados tanto através da árvore como entre parceiros, ajustando a taxa de transmissão de cada conexão individualmente a partir das condições atuais da rede.

A Figura 3.8 ilustra a arquitetura do Bullet, mostrando: a árvore que é utilizada para distribuir conjuntos de dados; e a camada *mesh* que é usada para permitir que os nós recuperem dados faltantes de outros parceiros.

Figura 3.8 - Bullet (KOSTIĆ ET AL., 2003)

O mTreeBone (WANG, F. ET AL., 2007) é outra solução que utiliza a abordagem híbrida misturando árvore e *mesh*, na qual a idéia principal é identificar um conjunto **estável** de nós e utilizar estes para construir um *backbone* baseado em árvore chamado "*treebone*". A maior parte dos dados é transmitida através desse *backbone*. Após isso, tanto os nós considerados "**estáveis**", como os considerados "**instáveis**", são organizados através de um *overlay mesh* auxiliar, o qual facilita acomodar a dinamicidade dos nós e explorar toda a banda disponível entre os nós do *overlay*.

Note que um pequeno subconjunto de nós estáveis é suficiente para suportar o *overlay* inteiro, pois existem poucos nós internos e muitos nós folhas em uma árvore. Portanto, o *overhead* na construção e manutenção desta árvore é relativamente baixo. A dificuldade, então, recai na **identificação dos nós** que podem ser considerados **estáveis**, uma vez que clientes podem entrar ou sair do sistema a qualquer momento.

Alguns estudos mostram que nós que estão há mais tempo no *overlay* tendem a continuar presentes (FREEDMAN; RAO; SRIPANIDKULCHAI, 2006). A arquitetura do mTreeBone, baseando-se nesses estudos, definiu um método baseado na idade dos nós para definir os nós estáveis. O problema desta abordagem é que no início de uma sessão não há informação suficiente sobre a idade dos nós para definição dos estáveis, uma vez que todos os nós possuem praticamente a mesma idade. Para tratar este problema, o mTreeBone introduz um algoritmo de promoção aleatória de nós estáveis no início da sessão. Feito isso, um nó que não pertence a parte estável da árvore (chamada de *treebone*, ou seja, esqueleto da árvore) verifica periodicamente sua própria idade e se esta exceder o limite (que é variável com o tempo), o nó se promove passando a fazer parte do *treebone*.

O esqueleto da árvore, no entanto, não pode eliminar completamente as operações de reparo em sua estrutura, uma vez que mesmo os nós considerados estáveis não são absolutamente persistentes. Além disso, a banda potencial dos nós instáveis é completamente ignorada pelo *treebone*.

Para melhorar a resiliência e a eficiência do sistema, todos os nós são organizados em um *overlay* tipo *mesh*. Similar ao CoolStreaming (ZHANG, X. ET AL., 2005), cada nó mantém uma lista parcial dos nós ativos no *overlay* e seus estados. Esta lista é atualizada com a troca de informações de estado entre os nós, utilizando um

algoritmo leve, escalável e “*gossip-based*” chamado SCAMP (*SCAlable Membership Protocol*) (GANESH; KERMARREC; MASSOULIÉ, 2003). Nós vizinhos no *mesh overlay* trocam, também, periodicamente seus mapas de *buffer*. Diferentemente de outros sistemas **dirigidos por dados**, um nó não agenda ativamente a obtenção de blocos de dados usando as informações contidas nos mapas de *buffer*. Tal obtenção somente é invocada quando ocorre a falha na entrega do dado através do *treebone*.

A Figura 3.9 ilustra a arquitetura híbrida do mTreeBone. Quando um nó instável como o nó A falha ou deixa o sistema, a transmissão dos dados através do *treebone* não é prejudicada. Por outro lado, quando um nó estável falha (e.g., problemas de rede) ou deixa o sistema (e.g., desconexão ou desligamento do equipamento), o impacto pode ser mitigado com a ajuda do *overlay* em *mesh*. Por exemplo, considere que o nó B mostrado na figura deixa o sistema. Enquanto o nó C é afetado, ele pode facilmente obter os dados faltantes de seus vizinhos na rede *mesh* até que ele consiga se reconectar ao *treebone*.

Figura 3.9 - mTreeBone (WANG, F. ET AL., 2007)

3.3 Redes de Distribuição de Conteúdos

Redes de distribuição de conteúdo (CDNs – *Content Distribution Networks*) são compostas de um conjunto de servidores de replicação (*surrogate servers*) distribuídos ao redor do mundo, os quais armazenam os conteúdos provenientes

dos provedores de conteúdo, replicando estes dados em pontos distintos da rede. Quando requisitado por um cliente, o servidor de replicação melhor localizado com relação ao cliente e com recursos disponíveis para atender à requisição fica responsável então pela entrega do conteúdo. A Figura 3.10 ilustra o conceito abstrato de uma CDN.

Figura 3.10 - Rede de distribuição de conteúdo (CDN) (PALLIS; VAKALI, 2006)

O uso de CDNs para transmitir conteúdos a consumidores apresenta muitos benefícios (e.g., melhor aproveitamento de banda, possibilidade de escalar para um grande número de clientes com a adição de novos servidores de replicação) quando comparado ao modelo tradicional cliente-servidor. A qualidade da entrega do conteúdo é melhorada, reduzindo a latência e aumentando a confiabilidade (removendo o ponto único de falha existente no modelo cliente-servidor), e a carga no servidor de origem do conteúdo é reduzida.

As principais idéias e conceitos de CDN podem ser utilizados para definir como os conteúdos podem ser distribuídos entre os nós e trazidos para perto do consumidor, diminuindo os custos de infra-estrutura e aumentando o desempenho do sistema.

Algumas questões são críticas para o correto funcionamento de uma CDN. A determinação da melhor localização dos servidores de replicação na rede é essencial para aperfeiçoar a eficiência do sistema e reduzir custos com infraestrutura. Uma solução que use correlação e frequência de acesso aos conteúdos para decidir onde replicar cada conteúdo, ao invés da simples replicação de todo o conteúdo, impraticável devido à limitação de espaço em disco, é necessária para o funcionamento adequado da CDN.

Para efetuar a replicação dos conteúdos existem diversas maneiras: o conteúdo pode ser enviado pela fonte para os servidores de replicação independentemente da requisição destes; os servidores de replicação requisitam explicitamente o conteúdo para a fonte, obtendo o mesmo após a requisição; ou sob-requisição, e cooperativamente, no qual um servidor de replicação obtém o conteúdo desejado através da cooperação com os servidores mais próximos a ele.

Atualmente existem diversas CDNs comerciais e acadêmicas. A seguir apresentam-se duas delas (Akamai e Prism), com a finalidade de identificar elementos e funcionalidades das mesmas. Estas duas foram escolhidas pela sua importância (a primeira com relação ao número de clientes e a segunda por sua importância acadêmica, por ser uma das primeiras propostas e ter influenciado várias outras propostas) e por lidarem com a distribuição de vídeo, inclusive de conteúdo ao vivo.

3.3.1 Akamai

Akamai é um exemplo de solução comercial de CDN, sendo uma das arquiteturas mais conhecidas para distribuir e acelerar conteúdos *web* e aplicações. Sua arquitetura possibilita diversas abordagens para solução dos problemas de distribuição de conteúdo, tratando tanto de conteúdo sob demanda (*Akamai Media Delivery*) quanto de transmissão ao vivo (*Akamai Stream OS*).

O *Akamai Media Delivery* (AKAMAI TECHNOLOGIES, 2008a), apresentado na Figura 3.11, é um canal de distribuição escalável para uma grande variedade de mídias, incluindo músicas, filmes e jogos. Neste sistema, o cliente publica um conteúdo através de um servidor de origem dos dados (*Origin Server*), sendo que

este conteúdo é então replicado em servidores localizados na borda da rede (*Edge Servers*) para melhor servir o usuário quando este o requisita.

Figura 3.11 - Akamai Media Delivery (para conteúdo sob demanda) (AKAMAI TECHNOLOGIES, 2008a)

Por outro lado, o Akamai Stream OS (*On-line Service*) (AKAMAI TECHNOLOGIES, 2008b), ilustrado na Figura 3.12, apresenta uma solução para controlar fluxos de mídias, provendo e distribuindo os conteúdos através de uma interface de aplicação inteligente. Conteúdos podem ser controlados e protegidos por regras de negócios, autenticação, controle de banda e restrições baseadas em tempo ou localidade geográfica.

Figura 3.12 - Akamai Stream OS (AKAMAI TECHNOLOGIES, 2008b)

No *Akamai Stream OS* existem cinco módulos responsáveis pela distribuição do conteúdo. São eles: o **Content Manager**, responsável pela carga (*upload*), armazenamento, gerenciamento, edição de metadados e entrega do conteúdo; o **RSS (Really Simple Syndication) Manager**, responsável pelo gerenciamento, entrega e distribuição de conteúdo **via feeds RSS**; o **Live Events Module**, que cria, agenda, gerencia e provê fluxos de **eventos e conteúdos ao vivo**; o **Profiling Module**, responsável pela **proteção e controle** do conteúdo baseado em regras de negócio; e o **Enterprise Manager**, que cria e gerencia múltiplas contas de um grupo administrativo específico.

Por ser uma solução comercial com especificação técnica fechada, não existem dados mais específicos e detalhados com relação à arquitetura e ao funcionamento da CDN do Akamai. Porém é interessante apresentá-la, uma vez que é a maior CDN comercial e já proveu serviços para muitas grandes empresas, como por exemplo, para a rede Globo durante a copa do mundo de 2006, para possibilitar que milhares de usuários simultaneamente assistissem aos jogos da copa através do portal globo.com sem a necessidade de aumentar a infra-estrutura de distribuição existente deste portal (AKAMAI TECHNOLOGIES, 2008c).

3.3.2 Prism

Prism é outra arquitetura de CDN para armazenar e distribuir fluxos de mídia de alta qualidade sobre redes IP (CRANOR ET AL., 2001). Além de mídia ao vivo, esta arquitetura permite o armazenamento do conteúdo na rede, e, portanto, uma variedade de serviços sob demanda.

Possui três componentes: “*Live Source*”, “*Portal*”, e “*Client*”. O **Live Source** recebe o conteúdo de um provedor, codifica, empacota e transmite-o através da infra-estrutura do Prism. O **Portal** recebe conteúdo multimídia de fontes ao vivo e outros portais, e transmite para os clientes Prism. Os portais armazenam estes conteúdos ao vivo para permitir funcionalidades de VCR para os clientes, tais como avançar e retroceder o conteúdo. Finalmente, o **Client** recebe o conteúdo do *Portal* e exibe o mesmo para o usuário. Os clientes podem ser dispositivos dedicados (*Set-Top*

Boxes – STBs), com capacidade de comunicação através de IP, ou computadores pessoais (PCs).

A comunicação entre clientes e portais, e entre portais e fontes é feita através do RTSP (*Real Time Streaming Protocol*) para controle da transmissão da mídia.

Estes elementos do Prism comunicam-se através da rede nos planos de dados e controle, como explicado a seguir.

Figura 3.13 - Plano de dados do Prism (CRANOR ET AL., 2001)

No plano de dados (Figura 3.13) são implementadas as funcionalidades de: **distribuição e entrega de conteúdo**, i.e., transferência do conteúdo da fonte ao vivo para um ou mais portais ou entre portais; e **transmissão do conteúdo de um portal para um ou mais clientes**, tentando prover um desempenho aceitável em operações sensíveis à latência, tais como funcionalidades de VCR, sendo importante a escolha de um portal próximo ao cliente e com recursos de processamento e rede disponíveis para atender à requisição.

Figura 3.14 - Plano de controle do Prism (CRANOR ET AL., 2001)

No plano de controle (Figura 3.14) aparecem as funcionalidades de **gerenciamento e descoberta de conteúdo**, além de **redirecionamento sensível ao conteúdo** (*content-aware redirection*). Neste plano, são realizados a coordenação e o mapeamento do armazenamento dos conteúdos nos portais, dependendo do tipo de serviço, dos Acordos de Nível de Serviço (SLA – *Service Level Agreement*) estabelecidos, da capacidade dos portais e da carga causada pelos acessos dos usuários. Este plano determina, também, a existência e a localização de um conteúdo dentro da estrutura do Prism, sendo que a arquitetura de descoberta de conteúdo é construída sobre um serviço de mapeamento hierárquico, que mantém as relações entre conteúdos, identificados por URNs (*Uniform Resource Name*), e um conjunto de URLs (*Uniform Resource Locator*) associado a cada conteúdo/URN, com as informações de localização deste.

Figura 3.15 - Serviço de mapeamento hierárquico (CRANOR ET AL., 2001)

Finalmente, as requisições dos usuários são encaminhadas para o portal que pode satisfazer melhor as necessidades desses (através da funcionalidade de redirecionamento sensível ao conteúdo do plano de controle). Esta decisão de redirecionamento das requisições é baseada na localidade da requisição juntamente com as informações de localização do conteúdo e dos recursos disponíveis nos portais, determinando-se o portal mais adequado para atender à determinada requisição de um usuário.

3.4 Compartilhamento de Arquivos *Peer-to-Peer*

Diversas técnicas para compartilhamento de arquivos, que utilizam recursos dos próprios usuários do sistema para efetuar a distribuição dos dados de maneira colaborativa, são utilizadas atualmente na Internet, facilitando-se a distribuição de livros, músicas, vídeos e *softwares*, entre outros. Obviamente com estas técnicas surge tanto a distribuição de conteúdos legais como ilegais (i.e., com o desrespeito de direitos autorais sobre os conteúdos). Devido à facilidade de transmissão de conteúdos ilegais, existem muitas críticas aos sistemas P2P, além de muitos provedores de acesso tentarem identificar e limitar este tipo de tráfego, que consome grande parte dos recursos de banda da rede. A Figura 3.16 mostra a porcentagem do tráfego P2P com relação ao tráfego total de rede em diversas regiões da Europa.

Figura 3.16 - Porcentagem relativa de tráfego P2P (SCHULZE; MOCHALSKI, 2007)

Por outro lado, levantamentos recentes mostram que em alguns locais como, por exemplo, na América do Norte, após quatro anos consecutivos de aplicações P2P consumindo a maior porcentagem da banda, o tráfego HTTP (*web*) ultrapassou o P2P e continua crescendo, resultado dos fluxos *web* de áudio e vídeo. O tráfego HTTP representa aproximadamente 46% do total, enquanto que o P2P aparece em segundo lugar com 37%. Analisando o tipo de aplicação dentro do tráfego HTTP,

fluxos de vídeo representam 36% deste tipo de tráfego, sendo que o YouTube sozinho representa 20% de todo tráfego HTTP, próximo a 10% de todo o tráfego internet. (ELLACOYA NETWORKS, 2007)

Devido a este impressionante aumento da popularidade de conteúdos multimídia na Internet e o conseqüente crescimento deste tipo de tráfego na rede, o paradigma P2P mostra-se altamente escalável para distribuição de arquivos, principalmente de tamanho grande (e.g., *softwares* e vídeos), e está sendo cada vez mais estudado como meio de distribuição de conteúdos multimídias, como, por exemplo, no Joost e Azureus/Vuze¹⁷.

Os sistemas P2P de compartilhamento de arquivos tornaram-se conhecidos no ano de 2000 com a popularização do Napster, *software* que possibilitava o compartilhamento de músicas no formato MP3. Em seguida, muitos novos sistemas, com basicamente o mesmo propósito, surgiram, como o Gnutella e o KaZaA (SAROIU; GUMMADI; GRIBBLE, 2003; LEIBOWITZ; RIPEANU; WIERZBICKI, 2003). Nestes primeiros sistemas P2P, um grande problema começou a surgir quando muitos usuários somente obtinham conteúdos da rede, não contribuindo com seus recursos para auxiliar na redistribuição (CEBALLOS; GORRICHIO, 2006). Para amenizar estes problemas, os algoritmos mais recentes tais como Emule e BitTorrent utilizam diferentes políticas de incentivo para priorizar o *download* por usuários que auxiliam na distribuição (COHEN, 2003).

3.4.1 Protocolo BitTorrent

Dentre os diversos protocolos P2P existentes atualmente, o BitTorrent mostra-se o mais utilizado, como pode ser observado na Figura 3.17, sendo esta a razão de ter sido escolhido como base para a implementação do protótipo da arquitetura de IPTV definida neste trabalho.

¹⁷ <http://www.azureus.com/>

Figura 3.17 - Estatística de tráfego P2P (SCHULZE; MOCHALSKI, 2007)

Sistemas de compartilhamento de arquivo que utilizam o protocolo BitTorrent¹⁸ são constituídos por quatro elementos principais: um **arquivo de metadados com extensão “.torrent”** (veja Apêndice A para detalhes da estrutura deste arquivo), que contém informações sobre o conteúdo a ser compartilhado, tais como nome do arquivo, tamanho dos pedaços, número de pedaços, endereço do rastreador (*tracker*) e o *hash* de cada pedaço para verificação de integridade após o *download*; um **servidor web**, no qual este arquivo de metadados fica armazenado, funcionando como um repositório para que os usuários consigam as informações necessárias para obter o conteúdo desejado; o **rastreador (*tracker*)**, que auxilia na coordenação dos *peers*, mantendo uma lista dos *peers* ativos no compartilhamento de determinado conteúdo; e finalmente, os ***peers***, que podem possuir parte ou nada do conteúdo associado ao “.torrent” (i.e., usuários interessados em obter o conteúdo), ou o conteúdo completo (i.e., a fonte original do conteúdo ou usuários que concluíram o *download* e continuam compartilhando o arquivo).

No momento em que um usuário do sistema deseja obter um arquivo via P2P, ele precisa do arquivo “.torrent” para poder requisitar ao rastreador (cujo endereço está contido no arquivo “.torrent”) uma lista de *peers* para este arquivo. O arquivo “.torrent”

¹⁸ www.bittorrent.com/protocol.html

contém, além do endereço do rastreador, outros metadados necessários para o correto funcionamento do sistema P2P. A comunicação com o rastreador é feita via HTTP (ou HTTPS). O usuário envia uma requisição do tipo GET, com um conjunto de parâmetros que identificam o arquivo desejado e informam o rastreador sobre o estado atual do *peer*, e recebe, como resposta do rastreador, uma lista contendo um conjunto aleatório de outros *peers* que estão compartilhando o arquivo solicitado além de outras informações de controle. Os detalhes dos parâmetros da requisição e da resposta na comunicação com o *rastreador* são apresentados no Apêndice B.

Com esta lista de *peers*, o usuário passa, então, a conversar diretamente com cada um destes para trocar informações sobre quais partes do arquivo cada um tem e obter os pedaços deste arquivo de maneira colaborativa. Após o término da obtenção de cada pedaço, o *hash* é calculado e verificado com o existente no *torrent*, para conferir se os dados recebidos dos outros *peers* são realmente válidos. Na Figura 3.18 a troca de mensagens entre o usuário e outro *peer* é apresentada para facilitar a compreensão do protocolo.

Figura 3.18 - Troca de mensagens entre *peers*

O usuário (Peer1) envia uma mensagem de handshake para cada um dos outros peers, recebendo uma mensagem que contém a indicação de quais os pedaços o outro peer (Peer2) possui disponíveis (mensagem de bitfield, contendo um conjunto de bits que representa os pedaços que este peer possui, dependendo do valor do bit), e caso haja o interesse em qualquer destes pedaços, é enviada a mensagem de interested notificando o Peer2. Quando este recebe a mensagem indicando o interesse do Peer1, adiciona este em uma lista, juntamente com todos os outros peers que informaram interesse em obter pedaços deste peer, e periodicamente verifica a quais peers, dentre todos os que demonstraram interesse, deve ser dada permissão para requisitar dados, através de um mecanismo chamado de unchoke. Assim que o Peer1 recebe a mensagem de unchoke, este começa a requisitar os pedaços que ele deseja, recebendo estes do Peer2. O mecanismo de unchoke é apresentado na Figura 3.19, e a descrição detalhada de cada campo das mensagens é mostrada no Apêndice C.

```
A cada 10s faça:  
    Ordene peersInteressados por taxaDeUpload  
    Pegue os 3 primeiros da lista  
    Para cada um deles  
        Se choked: unchoke (mude o estado e envie mensagem)  
    Coloque os demais no estado de choke (e envie mensagem)  
Fim
```

Figura 3.19 - Mecanismo de *unchoke*

O objetivo do mecanismo de *unchoke* é fazer o protocolo BitTorrent ser justo com os participantes, baseando-se na política de reciprocidade para decidir quais *peers* devem ficar no estado de *unchoked* (i.e., quais *peers* devem receber permissão para poder solicitar pedaços do conteúdo). Ou seja, os *peers* que contribuem mais para o compartilhamento de arquivos (i.e., quem faz mais *upload*) têm prioridade em relação aos demais.

Essa característica do mecanismo de prioridade do BitTorrent é em parte responsável pelo bom funcionamento deste protocolo na Internet. No entanto, para o sistema proposto neste trabalho, que trata especificamente da distribuição de vídeo através de P2P numa rede proprietária (do provedor do serviço de IPTV), onde se deseja iniciar a exibição do conteúdo com a menor latência possível após a seleção

do mesmo, a velocidade de obtenção do conteúdo não é o fator mais relevante, desde que seja suficiente para obter as partes necessárias antes do instante de exibição das mesmas. Sendo assim, um mecanismo que leva em conta o quão urgente determinado pedaço é para um *peer*, baseado no esvaziamento do *buffer* do mesmo (i.e., quão em breve faltará dados para a exibição e, conseqüentemente, haverá uma interrupção), seria muito mais apropriado para este cenário.

Além disso, na especificação do protocolo do BitTorrent o algoritmo de seleção de pedaços não é explicitamente definido, porém a seleção aleatória é citada como uma abordagem relativamente boa. Diversas aplicações de compartilhamento de arquivos BitTorrent, no entanto, procuram obter os pedaços mais raros primeiro, para evitar que estes desapareçam da rede e o arquivo fique incompleto. Porém, quando se trata da obtenção de conteúdo multimídia através do P2P, é desejável que o *download* seja seqüencial, para permitir o início da exibição do conteúdo o quanto antes.

Finalmente, a grande preocupação das operadoras de rede com relação ao tráfego P2P deve-se ao fato de que este acaba gerando grande volume de tráfego inter-provedor muitas vezes desnecessário, uma vez que os pedaços poderiam ser obtidos de *peers* dentro do mesmo ISP, mas acabam sendo requisitados para *peers* em outro domínio devido à característica de aleatoriedade na seleção dos *peers* do protocolo original. Portanto, faz-se necessário utilizar informações sobre, por exemplo, a topologia física da rede, para que a distribuição P2P de conteúdos seja o mais eficiente possível e realmente benéfica para o provedor em termos da quantidade de recursos utilizados na distribuição (CHEN ET AL., 2007).

3.5 Considerações Finais

Neste capítulo, apresentaram-se os requisitos para distribuição de vídeo em redes de dados (redes IP mais especificamente) além de diversas tecnologias existentes para distribuição de conteúdo em geral. Pelos requisitos citados, observa-se a inviabilidade em se utilizar o modelo cliente-servidor tradicional na distribuição dos conteúdos, pois a largura de banda necessária seria proibitiva. Portanto, para atender a demanda de um sistema de IPTV faz-se necessária a utilização de

técnicas de *multicast* para a transmissão linear dos conteúdos, e de algum tipo de replicação dos conteúdos para acesso sob demanda.

Neste trabalho, considerou-se possível a utilização de *multicast* IP nativo, devido à premissa de que a rede é proprietária, havendo controle desta por parte do provedor de IPTV, ao contrário de TV sobre Internet onde não há controle sobre a rede (i.e., os dados trafegam sobre a rede pública – Internet) e o uso de *multicast* IP é inviável devido às restrições apresentadas anteriormente. Com isso, no decorrer desta dissertação é assumida uma rede com *multicast* IP habilitado, porém mostrou-se neste capítulo que diversas outras técnicas de *multicast* existem e poderiam ser utilizadas caso os roteadores não ofereçam tal suporte (e.g., centrado em infraestrutura – CDNs – ou *multicast* em nível de aplicação).

Para a distribuição de conteúdos sob demanda, técnicas de replicação dos dados em servidores na borda da rede (CDNs) ou de compartilhamento P2P podem ser utilizadas. Porém, vale notar que a abordagem de CDNs gera um alto custo de infraestrutura devido à necessidade de servidores dedicados distribuídos na borda da rede, enquanto que no cenário de IPTV os STBs dos usuários (muitas vezes cedidos em comodato) possuem, em geral, capacidade ociosa, que pode ser aproveitada para auxiliar na distribuição de conteúdos sem gerar um custo adicional de infraestrutura para os provedores de IPTV.

No entanto, vale observar que para a distribuição de VoD, técnicas P2P tradicionais para compartilhamento de arquivos podem ser aplicadas facilmente, com pequenas mudanças para que a obtenção dos dados ocorra de maneira seqüencial e seja possível começar a exibição do conteúdo antes do término do *download*. Porém, no caso da distribuição para apresentação deslocada no tempo, o conteúdo está sendo gerado naquele instante, não existindo ainda em sua completude na rede. Como o compartilhamento P2P tradicional pressupõe a existência completa do arquivo (e.g., o BitTorrent depende da existência do arquivo para gerar o “.torrent” com os *hashes* de cada pedaço para verificação de integridade), estes não podem ser aplicados em sua forma tradicional para permitir a apresentação deslocada no tempo.

Tendo em vista estas restrições técnicas dos sistemas existentes hoje em dia, neste trabalho é proposta uma arquitetura para sistemas IPTV que possibilita a oferta do serviço de apresentação deslocada no tempo sem a necessidade de configuração por parte do usuário para gravar determinado conteúdo, assim como são descritos

os detalhes da implementação de um protótipo que utiliza esta arquitetura para fornecer os serviços tradicionais de IPTV (transmissão linear e VoD) além da possibilidade de apresentação deslocada no tempo.

4 ARQUITETURA DE UM SISTEMA IPTV COM APRESENTAÇÃO DESLOCADA NO TEMPO

No modelo tradicional de IPTV empregado atualmente têm-se a multidifusão dos canais de TV e a oferta de conteúdos sob demanda, geralmente vídeos ou seriados. Porém, pode-se dizer que estes dois serviços são totalmente independentes, sendo que é possível que um conteúdo que foi transmitido em um canal de TV seja ofertado através de VoD, mas do ponto de vista do sistema, não importa se este conteúdo foi transmitido em um canal ou não, ou seja, o sistema não se aproveita do fato de o conteúdo já ter sido transmitido previamente.

Com o intuito de possibilitar aos usuários de um sistema de IPTV a exibição deslocada no tempo dos conteúdos transmitidos nos canais, uma arquitetura que combina servidores de infra-estrutura e equipamentos de usuários para armazenar os conteúdos transmitidos e distribuí-los quando requisitados, é proposta neste trabalho. Neste capítulo são apresentadas, inicialmente, diversas soluções existentes para transmissão linear de conteúdos, distribuição de VoD, oferta de serviços de PVR e apresentação deslocada no tempo, juntamente com as tecnologias utilizadas por estas soluções para os respectivos serviços. Em seguida, são especificados os requisitos de uma arquitetura de IPTV que visa oferecer estes quatro tipos de serviços (i.e., transmissão linear, VoD, PVR e apresentação deslocada no tempo), sendo, então, descrita e detalhada a arquitetura propriamente dita, apresentando-se os componentes da mesma, sua inter-relação e o comportamento destes no sistema.

4.1 Soluções Existentes

Sendo um serviço tão recente, uma quantidade muito pequena de dados a respeito do serviço de apresentação deslocada no tempo está disponível. No entanto, parece claro que este serviço possui grande valor agregado. Apesar de ser oferecido em

apenas seis estados dos Estados Unidos em janeiro de 2007, a Time Warner declarou que o serviço foi utilizado quase quinhentas mil vezes por semana¹⁹.

Tanto esta solução comercial quanto propostas acadêmicas (WAUTERS ET AL., 2006), entretanto, subutilizam recursos do sistema na entrega do conteúdo deslocado no tempo. A arquitetura proposta neste trabalho visa aperfeiçoar o serviço de apresentação deslocada no tempo aproveitando-se dos recursos disponíveis nos equipamentos dos usuários para auxiliar na distribuição dos conteúdos, aumentando a escalabilidade e eficiência do sistema. A arquitetura utiliza *multicast* para a transmissão linear, *caches* distribuídos para armazenar os conteúdos durante a transmissão linear dos mesmos, evitando retransmissão destes, e *peer-to-peer* para distribuir os conteúdos deslocados no tempo quando solicitados por usuários, suportando através da rede P2P inclusive operações de busca (“*seek*”) nestes conteúdos por parte dos usuários.

Enquanto que o uso de redes P2P já foi proposto para resolver problemas de distribuição de VoD (JANARDHAN; SCHULZRINNE, 2007; DE PINHO; DE AMORIM, 2006) e para distribuição de conteúdos ao vivo (SplitStream (CASTRO, MIGUEL ET AL., 2003) e CoolStreaming (ZHANG, X. ET AL., 2005)), esta tecnologia ainda não foi utilizada para distribuição de conteúdos deslocados no tempo, provavelmente devido à necessidade da disponibilidade do conteúdo em sua totalidade nos algoritmos tradicionais de transferência de arquivos P2P. Além disso, as propostas de utilizar P2P para distribuição de VoD apresentam algumas extensões nos protocolos P2P tradicionais (e.g., BitTorrent) para possibilitar o início da exibição antes do término da obtenção do conteúdo, porém os conteúdos precisam estar completamente disponíveis na rede P2P. Em outras palavras, estas técnicas somente poderiam ser aplicadas para o caso de apresentação deslocada no tempo de conteúdos pré-gravados, mas não para conteúdos ao vivo, como definido no capítulo 2.

Tecnologicamente as soluções atuais de PVR podem ser divididas em duas classes. Primeiramente, têm-se as soluções baseadas em discos rígidos (hard-disk drives – HDDs), as quais armazenam localmente o conteúdo para utilização futura (e.g., HDD no *set-top box* da operadora de TV a cabo ou em sistemas auxiliares de PVR, como

¹⁹ www.lightreading.com/document.asp?doc_id=115819&site=cdn

os providos pela TiVo²⁰). A segunda classe consiste das soluções baseadas em armazenamento remoto (também chamado de network-PVR ou nPVR), as quais oferecem aos usuários um espaço para armazenamento de dados remoto. Nos dois casos, o usuário precisa saber a priori sobre os programas que ele deseja assistir no futuro para configurar o dispositivo de maneira adequada. Estes dispositivos armazenam os conteúdos programados durante a transmissão linear dos mesmos e os usuários conseguem visualizar tais conteúdos em qualquer momento futuro (recuperando os dados do armazenamento local ou remoto). As soluções baseadas em armazenamento remoto possuem a desvantagem de utilizar banda de rede para transmitir os fluxos gravados quando o usuário deseja assistí-los, porém é mais eficiente do ponto de vista de armazenamento, uma vez que não é necessário manter múltiplas cópias de um mesmo conteúdo quando diversos usuários configuram este conteúdo para ser armazenado.

Muitos sistemas de TV possuem um guia de programação eletrônico (EPG – Electronic Program Guide), o qual contém não somente a grade de programação linear dos canais de TV, mas também a lista de conteúdos VoD disponíveis. Estes oferecem maior flexibilidade ao usuário para assistir o conteúdo no horário mais conveniente. Historicamente, este serviço começou a ser oferecido numa forma mais simples (“*near VoD*”), na qual os conteúdos são retransmitidos periodicamente, sendo que o usuário espera a transmissão que irá se iniciar. Em sistemas de transmissão de vídeo, no entanto, a oferta de “*true VoD*” está se tornando cada vez mais popular, geralmente utilizando redes de distribuição de conteúdos (CDNs) ou modelos P2P, uma vez que o modelo tradicional cliente-servidor não escala. Independentemente da solução tecnológica adotada, os conteúdos oferecidos através de VoD, no entanto, não têm nenhuma relação com os conteúdos transmitidos linearmente através dos canais. Mesmo quando ambos os serviços são oferecidos pelo mesmo provedor, o usuário não consegue, por exemplo, acessar o noticiário do dia sob demanda. A arquitetura proposta neste trabalho consegue integrar todos os serviços oferecidos, possibilitando que usuários requisitem conteúdos que foram (ou que ainda estão sendo) transmitidos linearmente nos canais de TV, sob demanda, permitindo ainda selecionar qual parte deseja assistir (fazer “busca” no conteúdo).

²⁰ [http:// www.tivo.com](http://www.tivo.com)

A Tabela 4.1 apresenta um resumo de diversos sistemas de distribuição de vídeos que oferecem os serviços discutidos anteriormente, bem como as tecnologias utilizadas em cada caso. Como pode ser observado na tabela, excluindo-se o sistema proposto neste trabalho, a Time Warner é a única a oferecer todos os serviços, utilizando fluxos *unicast* de servidores na rede até os clientes para conteúdos VoD e deslocados no tempo. Esta tecnologia é utilizada em muitas soluções de VoD e nPVR, porém como argumentado anteriormente esta abordagem não é a mais eficiente e implica em alta demanda de infra-estrutura para lidar com as requisições dos usuários (HUANG ET AL., 2006; LEE, JACK Y. B.; LEUNG, 2002). A arquitetura introduzida neste trabalho (seção 4.3), por outro lado, não somente provê os serviços de distribuição de vídeo de maneira acoplada, como combina as tecnologias mais escaláveis e eficientes para cada propósito: *multicast* para transmissão linear dos conteúdos e o paradigma P2P para VoD e, principalmente, apresentação deslocada no tempo.

	Transmissão Linear	PVR	VoD	Apresentação deslocada
Solução proposta	<i>Multicast</i>	HDD	P2P	<i>Multicast</i> caching + P2P
AT&T U-verse	<i>Multicast</i>	HDD	<i>Unicast</i>	-
PCCW Now TV	<i>Multicast</i>	HDD	<i>Unicast</i>	-
SOPCast	P2P	HDD	P2P	-
Time Warner	<i>Multicast</i>	HDD	<i>Unicast</i>	<i>Unicast</i>
YouTube.com	-	-	<i>Unicast</i>	-
PPLive	P2P	-	P2P	-
Joost	P2P	-	P2P	-
CoolStreaming	P2P	HDD	-	-
Verizon P2P VoD	-	HDD	P2P	-
Peercast	P2P	-	-	-
Wauters et al. (2006)	<i>Multicast</i>	-	-	<i>Unicast</i>

Tabela 4.1 – Diversos sistemas de distribuição de vídeo disponíveis e respectivos detalhes técnicos (GALLO; COROAMA; ET AL., 2008)

4.2 Especificação de Requisitos

Nesta seção, primeiramente foram levantados os requisitos funcionais de toda solução de IPTV (e, mais genericamente, de toda solução para oferta de TV). Após este levantamento inicial, foram especificados os requisitos funcionais específicos de uma arquitetura de IPTV para prover o serviço de apresentação deslocada no tempo de conteúdos, com a finalidade de delimitar a solução proposta. Finalmente, os requisitos não funcionais de uma arquitetura de IPTV foram identificados. Tais requisitos são apresentados nas próximas seções.

4.2.1 Requisitos Funcionais

Nesta seção, são apresentados requisitos funcionais de propósito geral e requisitos específicos para sistemas de IPTV com suporte ao serviço de apresentação deslocada no tempo de conteúdos. Têm-se os seguintes requisitos:

- Oferecer o serviço de transmissão linear de conteúdos aos usuários (canais de TV);
- Prover o serviço de apresentação deslocada no tempo dos conteúdos anteriormente transmitidos linearmente pelos canais de TV:
 - Oferecendo flexibilidade do seu horário de apresentação que é definido pelos próprios usuários do sistema;
 - Permitindo, ainda, operações de pausa e busca durante a sua apresentação deslocada no tempo.
- Permitir a distribuição (linear e deslocada no tempo) de conteúdos codificados em diferentes formatos, e com diferentes taxas de transmissão.
 - Por exemplo, conteúdos codificados em H.264 ou MPEG-4, e conteúdos com resolução padrão (SD – *Standard Definition*), com taxa de transmissão entre 1 e 4 Mbps, e de alta definição (HD – *High Definition*), com taxas de transmissão entre 4 e 12 Mbps.

- Possibilitar a configuração do intervalo de tempo de disponibilidade de conteúdo para apresentação deslocada no tempo:
 - Este intervalo de tempo no qual os conteúdos podem ficar disponíveis para apresentação deslocada no tempo depende de acordos feitos com os provedores de conteúdos. Porém o sistema deve ser flexível para permitir a configuração de diferentes intervalos de tempo.

4.2.2 Requisitos Não-Funcionais

Além dos requisitos funcionais descritos anteriormente, foram especificados os seguintes requisitos não funcionais:

- Escalabilidade
 - Atender à crescente demanda de transmissão de conteúdo multimídia pela rede de dados. A solução proposta deve escalar para atender a milhões de usuários.
 - Esta demanda baseia-se no fato de que alguns sistemas de IPTV atualmente em funcionamento já possuem tal quantidade de usuários. Por exemplo, a AT&T, com o serviço “*U-Verse*” nos Estados Unidos, que contava com um milhão de assinantes²¹ em dezembro de 2008, e a PCCW com o “*Now TV*” em Hong Kong, que em junho de 2008 possuía 927.000 assinantes²².
 - Para possibilitar a configuração do intervalo de tempo de disponibilidade de conteúdo para apresentação deslocada no tempo o sistema deve permitir o aumento destes intervalos pela simples adição de capacidade de armazenamento nos dispositivos.
- Tempo de resposta para mudança de canal (tempo de *zapping*):
 - Para o serviço de transmissão linear, a latência na mudança de canal deve ficar abaixo de dois segundos (sendo desejável abaixo de um

²¹ <http://www.att.com/gen/press-room?pid=5838>

²² <http://www.pccw.com/eng/AboutUs/CompanyProfile.html>

segundo), uma vez que latência acima disto causa redução da satisfação do usuário (LEE, JIEUN ET AL., 2007).

- Latência para o início da apresentação deslocada no tempo:
 - Para o serviço de apresentação deslocada no tempo, por se tratar de uma forma de distribuição de vídeo sob demanda, é desejável que a latência para o início da exibição do conteúdo (e em operações de busca, quando o usuário seleciona a parte do conteúdo que deseja assistir) seja de poucos segundos. Vale observar que não há consenso na literatura de qual a latência máxima aceitável pelos usuários para este tipo de serviço, sendo que muitos trabalham com latências muito maiores, da ordem de minutos. Este é o caso das operadoras de TV a cabo que oferecem o chamado “*near VoD*” (ABRAM-PROFETA; SHIN, 1997), através, por exemplo, de técnicas de “carrossel”, onde um mesmo conteúdo é transmitido linearmente, periodicamente, e o usuário precisa aguardar o início da próxima transmissão.
- Custo: Deseja-se minimizar custos de infra-estrutura, considerando-se a premissa de que recursos ociosos dos equipamentos dos usuários (STBs) podem ser utilizados para auxiliar na distribuição defasada dos conteúdos, bem como recursos de outros componentes de transmissão da rede. Além disso, deseja-se minimizar a utilização dos enlaces da rede proveniente da oferta do serviço de apresentação deslocada no tempo de conteúdos.

Na próxima seção, é proposta uma arquitetura para sistemas IPTV visando atender aos requisitos listados acima, mostrando como o paradigma P2P pode ser aplicado para possibilitar a oferta do serviço de apresentação deslocada no tempo utilizando-se o armazenamento distribuído durante a transmissão linear, para evitar a retransmissão dos dados, e os recursos ociosos dos STBs.

4.3 Descrição da Arquitetura de Proposta

Na arquitetura proposta, conforme publicada pelo autor em (GALLO; COROAMA; ET AL., 2008; GALLO ET AL., 2009), um dos objetivos principais é oferecer o serviço de apresentação deslocada no tempo para os conteúdos transmitidos nos canais, independente da transmissão já ter terminado ou ainda estar sendo realizada, e independentemente da origem do conteúdo (i.e., ao vivo ou não). Sendo assim, aproveitando-se do fato do conteúdo ser transmitido através de *multicast*, este é armazenado na rede por servidores de infra-estrutura e equipamentos de usuários (*Set-Top Boxes* – STBs). Como ilustrado na Figura 4.1, a arquitetura proposta é composta de três módulos: *Proxy*, *Cache* e *Cliente*.

Figura 4.1 - Arquitetura proposta

O **Proxy** é responsável pela transmissão do conteúdo na rede de acesso, utilizando-se *multicast* para evitar a duplicação do fluxo de vídeo nos enlaces de rede e sobrecarga dos servidores de vídeo. O **Cache** aproveita-se do fato do conteúdo já estar sendo multidifundido, unindo-se ao grupo *multicast* para receber e armazenar o conteúdo com o intuito de distribuí-lo deslocado no tempo para os usuários do sistema quando requisitado. Finalmente, o módulo **Cliente** é capaz de unir-se a determinado grupo *multicast* associado à transmissão linear de um canal de TV, para a exibição (linear) do conteúdo sendo transmitido. Adicionalmente, o **Cliente** pode obter, via rede P2P, conteúdos cujas transmissões se iniciaram no passado,

independentemente de estas transmissões já terem sido concluídas ou não, possibilitando ao usuário assistir o conteúdo desde o começo.

Desta maneira, dá-se ao usuário maior flexibilidade, possibilitando que o mesmo assista determinado conteúdo no momento mais oportuno para ele, sem que seja necessário seguir a agenda de um canal em específico ou programar previamente algum equipamento para gravar o conteúdo.

Para obter boa eficiência na utilização dos recursos disponíveis além de uma solução escalável, tecnologias de compartilhamento de arquivos P2P são utilizadas na distribuição dos conteúdos deslocados no tempo. Portanto, todos os nós de *cache* e os clientes que possuem ou estejam obtendo determinado conteúdo, estarão também, de forma colaborativa, distribuindo pedaços deste conteúdo quando requisitado.

Quando um cliente deseja obter um conteúdo deslocado no tempo, ele precisa primeiramente descobrir quem possui tal conteúdo, para então solicitá-lo. A informação da localização do conteúdo é mantida de forma distribuída, utilizando-se uma Tabela de *Hash* Distribuída (DHT – *Distributed Hash Table*). Desta maneira, evita-se a necessidade de um servidor específico que lide com essa informação (o chamado rastreador ou *tracker* nos sistemas P2P), e tem-se a distribuição de carga das consultas através da DHT, permitindo que consultas por novos *peers* sejam realizadas com maior frequência sem que haja sobrecarga do rastreador e melhorando a eficiência na obtenção do conteúdo desejado.

Na Figura 4.2 é ilustrada a comunicação entre os componentes do sistema, durante todo o processo de armazenamento distribuído dos fluxos de vídeo e da obtenção deste fluxo deslocado no tempo por parte do cliente.

Figura 4.2 - Comunicação entre os componentes da arquitetura

Como o conteúdo armazenado pelos *caches* pode ter sofrido perdas na transmissão, tanto através de perdas de pacotes na rede como da perda do início do fluxo devido a atrasos no ingresso do grupo *multicast*, um mecanismo de alinhamento dos dados recebidos por todos os *caches* se faz necessário para que posteriormente eles possam compartilhar o conteúdo de maneira colaborativa, inclusive recuperando os pedaços perdidos nos casos descritos acima.

Uma vez que o fluxo de vídeo é encapsulado em RTP (*Real Time Protocol*), cada *cache* consegue verificar a perda de pacotes no meio da transmissão utilizando o número de seqüência do cabeçalho RTP. Porém, ainda existe o problema da perda do início do fluxo, pois neste caso não há como saber qual foi o primeiro número de seqüência utilizado na transmissão do conteúdo. No protótipo implementado para validação do sistema, um mecanismo de alinhamento dos dados armazenados nos *caches* foi desenvolvido e é apresentado em detalhes na seção 5.1.2.

Para permitir este alinhamento, o módulo *Proxy* é também responsável por armazenar o conteúdo localmente, já que ele é a origem do conteúdo sob o ponto de vista da rede de acesso, gerando metadados com o número de seqüência inicial utilizado na transmissão e possibilitando aos *caches*, após obtenção destes metadados, descobrir quanto foi perdido no início do fluxo. Os detalhes de quando o

Proxy gera estes metadados, quais atributos compõem os metadados e como eles são transmitidos entre os módulos do sistema são apresentados em detalhes no processo de ingestão de conteúdos pelo *Proxy*, na seção 5.1.1.

4.4 Casos de Uso do Sistema

A seguir são descritos os casos de uso do sistema proposto, incluindo os casos alternativos e de exceção. O primeiro caso de uso apresentado abrange a exibição de conteúdos durante a transmissão linear destes através do ingresso no grupo *multicast*. O segundo caso de uso apresenta o serviço de apresentação deslocada no tempo, foco deste trabalho, incluindo os casos alternativos e de exceção.

4.4.1 Exibição de Fluxo de Conteúdo Durante a Transmissão Linear

Ator: Usuário do serviço de IPTV

1. Usuário seleciona o canal de TV desejado (através do módulo Cliente no STB);
2. Módulo Cliente envia requisição de ingresso no grupo *multicast* ao roteador de borda;
3. Roteadores atendem a solicitação, fazendo com que o fluxo do conteúdo chegue até o STB;
4. Módulo Cliente inicia decodificação do fluxo, exibindo o conteúdo ao usuário.

4.4.2 Exibição de Fluxo de Conteúdo Deslocado no Tempo

Ator: Usuário do serviço de IPTV.

1. Usuário seleciona um conteúdo para apresentação deslocada no tempo;
2. Módulo Cliente obtém lista de *peers* (*Proxy*, *caches* ou outros clientes) que possuem o conteúdo na rede P2P;

3. Módulo Cliente seleciona *peers* (segundo mecanismo descrito em 5.1.5) e inicia comunicação direta com estes (através do protocolo P2P);
4. Módulo Cliente inicia processo de obtenção de blocos do conteúdo (obtenção seqüencial);
5. Assim que o primeiro bloco de vídeo é obtido, o módulo Cliente inicia a exibição do conteúdo;
6. Os demais blocos são obtidos em paralelo, até a conclusão da obtenção e da exibição.

Caso de exceção: Dados insuficientes para exibição (passo 6).

- 6.1. Módulo Cliente pausa a exibição devido à falta de dados e aguarda até que o próximo bloco de conteúdo seja obtido;
- 6.2. Após a obtenção do bloco em questão, a exibição continua, voltando ao passo 6 do caso de uso.

Caso alternativo 1: Usuário pausa a exibição (passo 6).

- 6.1. Usuário pausa a exibição (através do módulo Cliente no STB);
- 6.2. Obtenção do conteúdo continua normalmente, até que usuário selecione a opção de continuar a exibição, voltando ao passo 6 do caso de uso;

Caso alternativo 2: Usuário seleciona outra parte do conteúdo para apresentação deslocada no tempo (passo 6).

- 6.1. Módulo Cliente interrompe a exibição da posição anterior;
- 6.2. Módulo Cliente verifica a existência do bloco referente à posição selecionada;
- 6.3. Se o bloco ainda não se encontra disponível, o módulo Cliente interrompe a obtenção do bloco que estava obtendo e inicia a obtenção do bloco que contém a posição desejada pelo usuário;
- 6.4. Assim que o bloco é obtido, o módulo Cliente reinicia a exibição, a partir da posição desejada pelo usuário, continuando a obtenção dos demais blocos em paralelo (voltando ao passo 6 do caso de uso).

Caso alternativo 2b: Bloco de conteúdo referente à posição selecionada pelo usuário já disponível (passo 6.3 caso alternativo 2).

- 6.3.1. Se o bloco já se encontra disponível, o processo de obtenção de blocos não é interrompido;
- 6.3.2. Módulo Cliente inicia imediatamente a exibição da posição selecionada pelo usuário, voltando ao passo 6 do caso de uso.

4.5 Considerações Finais

Neste capítulo foram apresentadas inicialmente as soluções existentes para distribuição de vídeos, tanto na internet de maneira geral como em soluções de IPTV, listando-se as tecnologias utilizadas em cada uma destas soluções. Após este levantamento, os requisitos de uma arquitetura de IPTV, visando à provisão do serviço de apresentação deslocada no tempo, foram apresentados. Em seguida, foi apresentada a arquitetura proposta neste trabalho que satisfaz estes requisitos, utilizando *multicast* para distribuição eficiente dos conteúdos durante a transmissão linear dos mesmos e técnicas de armazenamento (*cache*) distribuído e *peer-to-peer* para distribuição dos conteúdos para apresentação deslocada no tempo.

Finalmente, apresentaram-se os principais casos de uso do sistema, abrangendo a exibição de fluxos de conteúdo durante a transmissão linear, via ingresso no grupo *multicast* associado à transmissão de um canal de TV; e a exibição de fluxos de conteúdo deslocados no tempo, através da obtenção deste via rede P2P.

5 IMPLEMENTAÇÃO DO PROTÓTIPO

Neste capítulo, são apresentados os detalhes de implementação do protótipo, incluindo a descrição detalhada do funcionamento de cada módulo implementado, a apresentação das APIs utilizadas e as alterações realizadas nestas APIs para obter as funcionalidades requeridas. Além disso, são discutidas, ao final do capítulo, as limitações relativas à implementação e às APIs utilizadas, e algumas considerações finais.

5.1 Detalhes de Implementação

No desenvolvimento do protótipo, diversas APIs (*Application Programming Interface*) foram utilizadas como base da sua implementação:

- jVLC (*Java bindings for VideoLan Client*)²³, empregado tanto no *Proxy* para transmitir o fluxo de vídeo via *multicast*, como também no Cliente para receber e exibir o fluxo das transmissões lineares e exibir o conteúdo obtido para apresentação deslocada no tempo via rede P2P;
- JMF (*Java Media Framework*)²⁴, utilizado na recepção dos pacotes RTP *multicast*, permitindo o acesso aos seus dados e às informações do cabeçalho (em particular o número de seqüência do RTP, que é utilizado no processo de alinhamento descrito na seção 5.1.2);
- jBitTorrent (*Java BitTorrent*)²⁵, responsável pela implementação do protocolo BitTorrent, possibilitando a obtenção e compartilhamento de pedaços do conteúdo através da rede P2P; e
- OpenChord DHT²⁶, utilizada para armazenar informações referente à localidade dos blocos de conteúdo.

Os processos de ingestão do conteúdo pelo *Proxy*, de alinhamento dos dados armazenados nos *caches*, de verificação de integridade do conteúdo e de operação

²³ <http://trac.videolan.org/jvlc/>

²⁴ <http://java.sun.com/javase/technologies/desktop/media/jmf/>

²⁵ <http://sourceforge.net/projects/bitext/>

²⁶ <http://open-chord.sourceforge.net/>

do módulo Cliente são apresentados a seguir, mostrando qual a função de cada API dentro de cada processo específico do sistema.

5.1.1 Ingestão de Conteúdo pelo *Proxy*

O módulo *Proxy* utiliza o *jVLC* para ler o conteúdo de qualquer endereço de origem (e.g., um arquivo local, remoto, ou ainda um fluxo de vídeo *unicast* proveniente de algum provedor de conteúdo) e transmiti-lo através de *multicast* na rede de acesso. O *Proxy* também armazena, localmente, o conteúdo para gerar os metadados necessários para o processo de alinhamento dos dados armazenados pelos *caches* (descrito na seção a seguir) e para ser o *peer* que possui o conteúdo completo no caso de todos os nós de *cache* terem perdido determinada parte do fluxo. Para armazenar o conteúdo é utilizado o *JMF*, obtendo os dados dos pacotes RTP *multicast* e gravando-os em um arquivo. Periodicamente (este período é definido a partir do tamanho do bloco de vídeo, que é configurável), o *Proxy* lê este arquivo, gerando metadados a partir do mesmo e permitindo a distribuição de cada bloco de conteúdo pela rede P2P.

Uma vez que o BitTorrent está sendo utilizado como protocolo P2P, os metadados são na realidade um arquivo do tipo ***torrent***. Cada ***torrent*** contém informações sobre o bloco de conteúdo, tais como: tamanho total, tamanho dos pedaços e *hashes* dos pedaços (empregados na verificação de integridade pelos *peers*, conforme especificado pelo protocolo do BitTorrent descrito na seção 3.4.1). A estrutura detalhada deste arquivo de metadados é apresentada no Apêndice A.

Para gerar estes metadados e começar a compartilhar o arquivo através da rede P2P, a *jBitTorrent* API é utilizada. Esta API sofreu diversas modificações para atender às necessidades do sistema especificado, incluindo a mudança para usar uma DHT na localização dos conteúdos ao invés de um rastreador centralizado como na implementação original.

Além disso, a *jBitTorrent* API (e o próprio protocolo BitTorrent) foram modificados para incluir o número de seqüência RTP inicial e final de cada bloco **nos metadados** (i.e., em cada arquivo ***torrent***). Esta informação é essencial para o processo de alinhamento dos dados armazenados nos *caches* (descrito na seção 5.1.2).

A faixa de variação do número de seqüência do RTP, no entanto, não é grande o suficiente para este propósito, repetindo com certa freqüência, dependendo da taxa de transmissão do conteúdo. Para poder utilizar esta informação de maneira inequívoca, o Proxy mantém um contador auxiliar, com mais 16 bits (além dos 16 bits proveniente do cabeçalho do RTP), para “estender” o número de seqüência do RTP. Desta maneira, durante a transmissão, este contador é incrementado de um cada vez que o número de seqüência do cabeçalho RTP reinicia. O número de seqüência estendido, de 32 bits, é, então, utilizado nos metadados, para que os *caches* e clientes consigam determinar exatamente quantos pacotes foram transmitidos desde o início do fluxo, mesmo no caso de conteúdos com alta resolução – HD – e conseqüente alta taxa de transmissão. Possuindo 32 bits, este número de seqüência estendido consegue representar univocamente cerca de 4 bilhões de pacotes RTP. Como cada pacote RTP transporta, tipicamente, mais de 1 *kilobyte* de dados, este número de seqüência consegue representar os pacotes de conteúdos com até 4 *terabytes*, que é, atualmente, mais que suficiente para qualquer transmissão de conteúdo em um sistema de IPTV.

O *Proxy* finalmente publica estes arquivos de metadados em um repositório (e.g., servidor FTP), para que os *caches* e os clientes possam obtê-los. Estes arquivos são necessários tanto no processo de alinhamento dos dados nos *caches* quanto no processo de verificação de integridade realizado pelos *caches* e clientes (conforme descrito na seção 5.1.3).

5.1.2 Alinhamento dos Dados Armazenados nos *Caches*

Como descrito no capítulo 4 referente à “Arquitetura de um Sistema IPTV com Apresentação Deslocada no Tempo”, o conteúdo que é armazenado pelos *caches* (através da recepção dos dados do fluxo RTP *multicast* pelo JMF) pode ter sofrido perdas na transmissão, tanto através de perdas de pacotes na rede como também de perda do início do fluxo devido a atrasos no ingresso do grupo *multicast*. Sendo assim, um mecanismo de alinhamento dos dados recebidos por todos os *caches* se faz necessário, para possibilitar a recuperação dos pedaços perdidos nos casos descritos acima e, para que, posteriormente, os *caches* possam compartilhar o

conteúdo de maneira colaborativa (ou seja, que os dados consistentes possam ser obtidos paralelamente de diversos *caches*).

A Figura 5.1 ilustra estes problemas com relação às perdas de pacotes na transmissão e ao atraso no ingresso ao grupo *multicast*.

Figura 5.1 – Problemas de sincronismo dos dados armazenados no *cache*

Uma vez que o fluxo de vídeo é encapsulado em RTP (*Real Time Protocol*), cada *cache* consegue verificar a **perda de pacotes no meio** da transmissão através da utilização do número de sequência do cabeçalho RTP (informação obtida através do JMF). No caso de pacotes perdidos é deixado um espaço em branco no arquivo de *cache* de tamanho correspondente ao número de pacotes perdidos (o tamanho do campo de dados do RTP utilizado na transmissão é fixo) para possibilitar a obtenção destes futuramente. Isto soluciona a questão de perdas de pacotes, porém ainda existe o problema da **perda no início** do fluxo, pois neste caso não há como o *cache* saber qual foi o primeiro número de sequência utilizado na transmissão do conteúdo.

Para isso, o módulo *Proxy* é também responsável por armazenar localmente o conteúdo dos pacotes RTP do fluxo *multicast*, já que é a origem do conteúdo sob o ponto de vista da rede de acesso. Neste caso, gera os metadados com o número de sequência inicial utilizado na transmissão, o que possibilita aos *caches*, após

obtenção destes metadados (de um repositório, que no protótipo é um servidor FTP), descobrir quantos pacotes foram perdidos no início do fluxo. O processo de ingestão do conteúdo pelo *Proxy*, incluindo a geração dos metadados, é descrito em detalhes na seção 5.1.1.

Baseado no número de seqüência inicial contido nos metadados, cada nó de *cache* alinha seu próprio arquivo, deslocando o conteúdo armazenado para coincidir com o arquivo armazenado pelo *Proxy*. Este processo de deslocamento dos dados é ilustrado na Figura 5.2. O nó de *cache* precisaria reservar um espaço no começo do arquivo para receber, futuramente, os pedaços faltantes de dados através da rede P2P. Para evitar o uso de uma quantidade excessiva de memória, ao invés de se armazenar em *buffer* cada bloco de vídeo antes de escrevê-lo no disco, verifica-se quantos dados foram perdidos no início da transmissão e deslocam-se os dados armazenados para liberar exatamente o espaço no disco requerido para gravar o pedaço faltante posteriormente.

Figura 5.2 - Alinhamento dos dados armazenados no *cache*

Para deslocar os dados é importante notar que o processo de *cache* deve continuar ativo, com os dados sendo escritos no final do arquivo. Assim, os devidos cuidados

precisam ser tomados para evitar perda de dados durante o processo. A idéia é avançar a posição de gravação exatamente a quantidade de dados perdidos no início, deixando o processo de *cache* continuar salvando os dados no final do arquivo, ao mesmo tempo em que se cria um espaço em branco do tamanho dos dados perdidos, assim como mostrado na Figura 5.2b.

Após isto, inicia-se o processo de mover os dados imediatamente “à esquerda” do espaço em branco para a direita (Figura 5.2d), deslocando-se os dados desta maneira sucessivamente até que o espaço em branco atinja o começo do arquivo (Figura 5.2e). Note, novamente, que o fluxo de vídeo continua a ser recebido e armazenado simultaneamente a este processo. Como não ocorre tentativa de escrita simultânea na mesma parte do arquivo, o processo de alinhamento funciona sem a necessidade de qualquer tipo de bloqueio explícito na escrita em disco, tornando o arquivo pronto para recuperar, de outros *peers*, os dados iniciais perdidos e para o compartilhamento com elementos da rede P2P (i.e., clientes e *caches*).

Note que na arquitetura proposta é possível iniciar o processo de armazenamento do conteúdo em qualquer instante durante a transmissão. Assim sendo, como o número de seqüência “estendido” (descrito no processo de ingestão em 5.1.1) não é transmitido em momento algum, o *Proxy* armazena o número de seqüência inicial (estendido) de cada bloco de vídeo (nos respectivos metadados), e não somente do início da transmissão. Caso contrário, não seria possível ao *cache* definir qual a quantidade de dados transmitida desde o início do fluxo, visto que o número de seqüência do cabeçalho RTP pode repetir durante a transmissão de um conteúdo (pois possui somente 16 bits).

5.1.3 Verificação de Integridade do Conteúdo

Após realizar o processo de alinhamento descrito na seção anterior, os nós de *cache* verificam a integridade dos dados armazenados, calculando os *hashes* de cada pedaço de vídeo e comparando com os gerados pelo *Proxy*. Note que os *hashes* gerados pelo *Proxy* (conforme descrito em 5.1.1) fazem parte dos metadados no formato **torrent** publicados no repositório pelo *Proxy* e que são obtidos pelos *caches*

no processo de alinhamento dos dados, já estando, portanto, disponíveis localmente.

Os pedaços cujo *hash* calculado não coincide com o publicado pelo *Proxy* são descartados e serão obtidos futuramente via rede P2P, recuperando-se não somente as perdas ocorridas no início da recepção como também as perdas ocorridas no meio da transmissão.

Após esta verificação de integridade, os *caches* começam a compartilhar o bloco de vídeo, aguardando solicitações de pedaços por outros *peers*, quer sejam outros *caches* tentando recuperar perdas de dados ou clientes requisitando conteúdos para apresentação deslocada no tempo.

Os clientes, também, realizam a verificação de integridade do conteúdo para assegurar que os pedaços obtidos de outros *peers* (quer sejam estes outros clientes, *caches*, ou mesmo o *Proxy*) estejam íntegros e correspondam aos dados do conteúdo original. Para tanto, os clientes obtêm, do repositório, o arquivo de metadados (*torrent*) do bloco em questão publicado pelo *Proxy* e, conforme recebem cada pedaço de outro *peer*, calculam o *hash* do mesmo e verificam a sua integridade, comparando com o *hash* publicado pelo *Proxy* (contido nos metadados) e descartando-o caso não haja coincidência entre o calculado e o publicado.

5.1.4 Operação do Módulo Cliente

O módulo **Cliente** utiliza o *jVLC* para exibir os conteúdos recebidos através do protocolo P2P ou via *multicast*. Para transmissões lineares, o **Cliente** utiliza o *jVLC* para receber e exibir o fluxo *multicast*, enquanto que para apresentação deslocada no tempo o *jVLC* é utilizado para exibir o conteúdo obtido através do protocolo P2P.

Para obter um conteúdo para apresentação deslocada no tempo, o **Cliente** utiliza a API do *jBitTorrent*. Sempre que um novo conteúdo é selecionado para ser exibido, o módulo cliente inicia a obtenção do conteúdo de outros *peers* (*proxy*, *caches* e outros clientes), gerenciando o *jVLC* em situações críticas. Tal gerenciamento inclui funções de verificação do *buffer*, analisando se o *buffer* tem dados suficientes para continuar a exibição do conteúdo (pausando caso contrário e continuando a exibição após obter os dados necessários); atualização da interface gráfica (barras de

progresso referentes à obtenção do conteúdo e à posição da exibição); e atualização de uma base de dados com os conteúdos e blocos de conteúdos que o **Cliente** já obteve.

Para cada bloco de vídeo que se deseja exibir, a aplicação primeiramente verifica se o bloco já existe. Caso o bloco ainda não tenha sido obtido, a API do jBitTorrent é utilizada para obtê-lo. Para tanto, a lista de *peers* que possuem o conteúdo (ou parte do mesmo) é obtida através da DHT, sendo que alguns desses *peers* são utilizados de fato na obtenção dos dados. A seleção dos *peers* dentre todos os informados pela DHT, para os quais serão solicitados dados, é realizada utilizando-se informações de localidade, conforme descrito na seção 5.1.5, ao invés de aleatoriamente como no caso do protocolo original do BitTorrent. Com este conjunto de *peers* selecionado, inicia-se uma comunicação direta com cada um, aguardando, então, a permissão (mensagem de *unchoke*) para iniciar a requisição dos pedaços do bloco de vídeo. O fluxo de mensagens do BitTorrent, referente à comunicação entre *peers*, segue o protocolo original, conforme descrito na seção 3.4.1 e ilustrado na Figura 3.18.

Vale notar que o **Cliente** começa a compartilhar pedaços deste bloco logo após o início do *download*, auxiliando na sua distribuição. Para isso informa a sua condição de *peer* para este bloco à DHT. Além disso, é importante observar que a API do jBitTorrent foi modificada para permitir o *download* de todos os blocos de vídeo de um mesmo conteúdo diretamente em um único arquivo, apesar de serem recebidos múltiplos arquivos **torrent** (um por bloco de vídeo). Isto é feito para evitar sobrecarga de processamento e duplicação dos dados no sistema de arquivos do **Cliente**.

Sob o ponto de vista do usuário, o módulo **Cliente** possui uma interface gráfica com duas janelas: a principal, na qual os conteúdos são exibidos permitindo-se o controle do usuário sobre a sua exibição, e outra com o EPG (*Electronic Program Guide*). Por meio deste EPG, o usuário consegue verificar a programação dos conteúdos que estão sendo transmitidos no momento, que foram transmitidos no passado, e que serão transmitidos no futuro, de qualquer canal de TV.

A Figura 5.3 mostra a janela do EPG, na qual conteúdos em verde (“*Movie 1*” e “*Movie 5*”) foram transmitidos no passado, estando disponíveis somente através do

serviço de apresentação deslocada no tempo. Conteúdos em rosa (“*Movie 2*” e “*Movie 6*”) estão sendo transmitidos no momento, sendo acessíveis via *multicast* (da posição de transmissão corrente) ou através da rede P2P, pelo serviço de apresentação deslocada no tempo (do começo ou de qualquer parte já transmitida). Finalmente, conteúdos em azul (“*Movie 3*” e “*Movie 7*”) serão transmitidos no futuro, não estando disponíveis ainda.

Figura 5.3 - Janela do EPG (*Electronic Program Guide*)

O usuário pode avançar e retroceder no tempo, clicando em conteúdos (para apresentação deslocada) ou em canais (para exibição da transmissão linear) para assisti-los. Uma descrição do conteúdo ou do canal é exibida quando o usuário passa o *mouse* sobre o mesmo. E finalmente, o usuário pode chavear entre a janela do EPG e a principal, sendo que o chaveamento pode ocorrer de maneira automática, quando o usuário seleciona um conteúdo ou canal, ou manual, através do acionamento de um botão.

A Figura 5.4 apresenta a janela principal, com um conteúdo sendo exibido. O estado da obtenção do conteúdo, com informações sobre quanto do conteúdo já foi obtido, pode ser visto na barra de progresso da obtenção no canto inferior direito, e o progresso da exibição pode ser acompanhado através da outra barra de progresso, sendo que esta pode sofrer intervenção do usuário quando o mesmo deseja avançar ou retroceder a posição de exibição do conteúdo. O usuário pode ainda pausar ou

parar a exibição deslocada no tempo, bem como chavear entre os canais de transmissão linear.

Figura 5.4 - Janela principal do Cliente

Quando o usuário altera a posição de exibição, a aplicação verifica se a parte correspondente do conteúdo já foi obtida, obtendo-a, caso necessário, via protocolo P2P, e informa o jVLC sobre a nova posição a ser exibida. Esta posição é uma aproximação, uma vez que o vídeo pode possuir taxa (*bit rate*) variável e o codec utilizado pode não possibilitar a configuração de uma posição de tempo exata através do jVLC.

5.1.5 Mecanismo de Seleção de *Peers*

Muitos trabalhos atuais, que avaliam a escalabilidade das redes P2P, trazem conclusões importantes sobre a escalabilidade destas antes desconsideradas por muitas arquiteturas (BINDAL ET AL., 2006; CHEN ET AL., 2007; XIE ET AL., 2007). Por exemplo, o tradicional modelo de simulação de redes P2P que considera a Internet como uma nuvem²⁷, esconde possíveis problemas de rede que estas

²⁷ Este modelo de Internet, como uma nuvem ("*Internet as a cloud*"), desconsidera as conexões físicas de rede entre os *peers*, considerando simplesmente que todos os *peers* conseguem se comunicar entre si na velocidade

soluções podem ocasionar (e.g., congestionamento na rede e altas taxas de tráfego inter-provedor desnecessariamente), resultando em uma falsa conclusão de escalabilidade e eficiência. Conforme apresentado por Chen et al. (2007), a utilização deste tipo de modelo pode super-estimar os benefícios do protocolo P2P em três vezes ou mais. Portanto, é importante observar que se certos cuidados não forem tomados, o uso do protocolo P2P pode acarretar em um aumento significativo de tráfego em enlaces internos da rede, congestionando e possivelmente degradando a mesma.

Para se evitar estes problemas, é fundamental observar que a seleção aleatória de *peers*, utilizada pelos protocolos P2P tradicionais, não leva em conta características da rede física. Neste caso, pode-se, por exemplo, escolher um *peer* muito distante apesar de outro *peer*, presente na mesma sub-rede, também possuir o conteúdo desejado, e poder fornecê-lo, evitando-se, desta maneira, o aumento de tráfego em enlaces internos.

É justamente para tratar problemas deste tipo que o mecanismo de seleção de *peers* utilizado neste trabalho foi desenvolvido. Este utiliza informações de localidade ao invés de, como no protocolo original do BitTorrent, efetuar a seleção de forma aleatória. No protótipo implementado como prova de conceito, a idéia é, sempre que possível, dar preferência aos *peers* pertencentes à mesma sub-rede. Quando isto não é possível (e.g., quando não existem *peers* na mesma rede que possuem o conteúdo), os *peers* são ordenados pela proximidade dos endereços IPs, ou seja, comparam-se quantos bits, da esquerda para a direita, são coincidentes entre o endereço IP do *peer* solicitante e o endereço IP de cada *peer* disponível.

Este mecanismo, apesar de simples, já pode trazer grande economia na utilização de banda dos enlaces internos da rede, como mostram os resultados apresentados na seção 6.4.6. Porém, parâmetros muito mais complexos podem ser utilizados como métrica da distância entre *peers*, tais como: informação sobre posição geográfica, tempo de resposta e número de roteadores que as mensagens precisam atravessar (CASTRO, M. ET AL., 2002; FREEDMAN ET AL., 2005).

5.2 Limitações

A implementação da prova de conceito da arquitetura proposta apresenta algumas limitações, em parte devido às limitações inerentes das APIs empregadas. Nesta seção, são discutidas estas limitações, suas implicações e como elas foram contornadas ou corrigidas.

Porém, vale notar que estas limitações são específicas da implementação do protótipo, e não representam limitações da arquitetura proposta neste trabalho, mas somente da prova de conceito desenvolvida.

5.2.1 OpenChord DHT

A API utilizada para criar a Tabela de *Hash* Distribuída (OpenChord) é uma implementação da DHT Chord, possuindo as operações básicas de criação de uma DHT, ingresso (*join*) e egresso (*leave*) da DHT, e inserção, obtenção e remoção de dados. Além disso, existe um mecanismo de auto-recuperação (*healing*), que trata a questão do ingresso e egresso arbitrário freqüente de nós na rede, lidando com os problemas relacionados ao dinamismo inerente de ambientes P2P (e.g., mantendo certa taxa de replicação dos dados para garantir que quando participantes deixam o sistema, dados não sejam perdidos). Porém, uma vez que as informações contidas na DHT não expiram, permanecendo armazenadas até que alguém as remova explicitamente, existe o problema de informações desatualizadas permanecerem na DHT no caso de *peers* encerrarem sua atividade abruptamente, sem remover suas entradas da DHT. Este problema é contornado no protótipo, encerrando-se as aplicações da maneira correta (i.e., fechando-se a aplicação cliente ao invés de encerrar o processo).

Outro problema relacionado à DHT neste cenário ocorre quando muitos *peers* compartilham um mesmo conteúdo. Neste caso, os nós da DHT precisam armazenar um grande volume de informação e, principalmente, os interessados neste conteúdo recebem uma lista muito grande com todos os *peers* disponíveis, o que impõe uma sobrecarga desnecessária ao sistema, uma vez que cada interessado vai utilizar no máximo algumas dezenas de *peers* para obter o conteúdo desejado. Modificar a

DHT para responder às requisições com somente um subconjunto de todos os *peers* para um conteúdo (preferivelmente os “melhores”²⁸ para atender à requisição em questão) e utilizar o conceito de DSHT (*Distributed Sloppy Hash Table*) poderiam ser alternativas para evitar esta sobrecarga, aumentando a eficiência e a escalabilidade do sistema (FREEDMAN; MAZIÈRES, 2003; FREEDMAN; FREUDENTHAL; MAZIÈRES, 2004). Porém, como a DHT não é o foco deste trabalho, optou-se por manter esta limitação do protótipo, possivelmente limitando a sua escalabilidade, o que não representa uma limitação da arquitetura proposta. Além disso, o esforço para realizar as modificações necessárias não traria benefícios visíveis à prova de conceito, visto que o ambiente de testes do sistema não é grande o suficiente para que esta limitação cause variações sensíveis nos resultados dos testes.

5.2.2 Java BitTorrent

A API do jBitTorrent foi desenvolvida com a finalidade de ser simples e fácil de utilizar, simplificando, para os desenvolvedores Java, a criação de aplicações que fazem uso do protocolo BitTorrent. Contudo, esta API não possui otimizações de código e contém pequenos erros de implementação, que impactam a construção de um sistema escalável. Esta API não possui nenhum mecanismo de fila que poderia ser empregado para compartilhar diversos arquivos simultaneamente. Ao invés disso, cria *threads* independentes para lidar com cada arquivo compartilhado e abre uma porta de escuta (*listening port*) para cada um destes arquivos ao invés de utilizar o campo “*file_id*” da mensagem de *hand-shake* do protocolo BitTorrent²⁹. Além disso, uma porta é aberta para cada *peer* que possui o conteúdo. Como consequência um número muito alto de portas é alocado quando muitos conteúdos são compartilhados ou quando existem muitos *peers* para um conteúdo (uma porta é utilizada para cada *peer* e para cada conteúdo). Como se pode notar, no cenário de IPTV com milhares de usuários que possuem um conteúdo, e cada usuário possuindo vários conteúdos, o número de portas utilizadas pela API pode ultrapassar facilmente o limite de 64k portas do TCP/IP.

²⁸ “Melhores” pode referir-se a diversas métricas, tais como *peers* que ofereçam menor latência, menor custo ao provedor do serviço, maior disponibilidade de recursos ociosos, etc.

²⁹ Ver tópico 3.4.1 para descrição do protocolo BitTorrent e Apêndice C para detalhamento dos campos das mensagens.

Para contornar esta limitação, inicialmente alterou-se o módulo **Cliente** para compartilhar o número máximo de 15 blocos de vídeo, deixando de compartilhar os blocos mais antigos caso este número seja ultrapassado. Porém, para permitir que realmente os clientes auxiliem na distribuição de conteúdos para apresentação deslocada no tempo, compartilhando qualquer bloco de vídeo que possua quando solicitado, a API foi alterada para evitar o desperdício de portas, mantendo-se uma única porta de escuta aberta para todos os conteúdos sendo compartilhados e tratando o campo “*file_id*” de maneira adequada para fornecer os dados que o outro *peer* deseja. Com estas alterações, portanto, esta limitação da API original foi completamente eliminada, possibilitando a escalabilidade da solução no que tange ao compartilhamento em redes P2P.

5.2.3 Java Bindings for VideoLan Client (jVLC)

A API do jVLC é uma interface Java para o VLC (*VideoLan Client*). O VLC pode ser utilizado, no modo servidor, na transmissão de fluxos de conteúdos; e, no modo cliente, na decodificação e apresentação de conteúdos. O jVLC, por sua vez, executa chamadas nativas de sistema às bibliotecas do VLC, utilizando JNDI (*Java Naming and Directory Interface*). Possui, portanto, praticamente as mesmas funcionalidades que o próprio VLC, em relação tanto à inúmera quantidade de formatos e *codecs* de vídeo e áudio que trata, como também ao tratamento de fluxos contínuos (*streaming*), provendo funções de servidor de vídeo, através do empacotamento (e.g., sobre RTP) e transmissão do vídeo via *unicast* ou *multicast*.

Esta flexibilidade foi a razão principal da escolha do VLC como tocador e servidor de vídeo do protótipo. Contudo, devido à natureza intrínseca dos tocadores de vídeo, quando tocando arquivos de conteúdo diretamente a partir do sistema de arquivos, estes não estão preparados para suportar arquivos variando de tamanho durante a sua exibição. Esta restrição é de fato um problema para o sistema implementado, uma vez que a exibição do vídeo ocorre em paralelo com a sua obtenção.

O VLC, no entanto, comporta-se relativamente bem em relação a esta restrição, em comparação com outros tocadores que travam o arquivo (lock), não permitindo a escrita por outro processo e impossibilitando a obtenção de pedaços do conteúdo

durante a sua exibição. Contudo, a posição relativa da exibição fica desatualizada quando o tamanho do arquivo está sendo modificado (o VLC somente atualiza o tamanho do arquivo no início da exibição). Esta informação é necessária para a funcionalidade de gerenciamento do esvaziamento do *buffer*, responsável pela pausa e retomada da exibição do vídeo após a obtenção dos dados necessários, não sendo possível utilizar a posição relativa informada pela API para identificar e tratar a situação de falta de dados no *buffer*.

Sendo assim, foi construído um método aproximado para verificar se os dados necessários para continuar a exibição estão disponíveis, baseando-se na verificação da disponibilidade do bloco de vídeo **completo** antes da exibição do mesmo. Isto implica em uma latência maior no início da exibição e possíveis interrupções desnecessárias por falta de dados, porém evita interrupções abruptas e descontinuidade na exibição do conteúdo.

5.2.4 Configurações Estáticas dos Módulos no Protótipo

Outra limitação, relacionada puramente à implementação do protótipo como prova de conceito, está no fato de toda a configuração do sistema ser realizada de forma estática, na fase de inicialização dos módulos. Os módulos são configurados através de arquivos de propriedades, que definem os diretórios onde os arquivos de vídeo e de metadados serão armazenados, e através de um arquivo XML (*eXtensible Markup Language*), que contém informações referentes à programação dos canais de TV.

Os módulos lêem estes arquivos de configuração na fase de inicialização, não sendo, portanto, possível a alteração dos diretórios onde serão salvos os dados ou da programação dos canais de TV após a inicialização do sistema. Além disso, o arquivo XML precisa ser fornecido a cada um dos componentes do sistema manualmente. Um exemplo deste arquivo XML, mostrando a estrutura do mesmo, é apresentado no Apêndice D.

O módulo *Proxy* utiliza informações deste XML para saber os horários em que as transmissões dos conteúdos dos canais devem ser realizadas, bem como de onde obter estes conteúdos. Além disso, o *Proxy* utiliza a informação do intervalo entre

blocos de vídeo contido no XML para a definição do tamanho dos blocos de vídeo, podendo ser diferente para cada conteúdo, e a informação do endereço (URL) do repositório onde os arquivos de metadados devem ser publicados.

O módulo de *cache* utiliza informações deste XML para saber os horários em que as transmissões dos conteúdos serão realizadas para iniciar o processo de *cache* de cada conteúdo no momento correto. Adicionalmente, o *Cache* também utiliza a informação do intervalo entre blocos de vídeo, para identificar a periodicidade com que os metadados serão gerados e publicados pelo *Proxy*, obtendo esses metadados do repositório (cujo endereço se encontra no XML) tão logo sejam publicados pelo *Proxy*.

Finalmente, o módulo Cliente utiliza, também, as informações contidas no XML para saber os horários em que as transmissões dos conteúdos serão realizadas, para montar e apresentar ao usuário o Guia de Programação Eletrônica (EPG – *Electronic Program Guide*). Além disso, a informação do endereço do repositório é utilizada quando o usuário solicita a apresentação deslocada no tempo de um conteúdo, para obter os arquivos de metadados do mesmo.

Esta limitação impossibilita o uso do protótipo comercialmente, porém não ocasiona impacto algum na utilização deste como prova de conceito da arquitetura de IPTV para oferta do serviço de apresentação deslocada no tempo, dos conteúdos transmitidos linearmente, através da combinação de *multicast*, armazenamento distribuído e redes de distribuição P2P.

5.3 Considerações Finais

Com a implementação do protótipo como prova de conceito da arquitetura proposta comprovou-se a viabilidade em se utilizar técnicas *peer-to-peer* para prover o serviço de apresentação deslocada no tempo aproveitando os recursos ociosos dos equipamentos dos usuários. Os equipamentos dos usuários são, freqüentemente, equipamentos cedidos em comodato pelo provedor do serviço.

Devido à utilização do protocolo P2P, consegue-se aproveitar o espaço de armazenamento e banda disponível no sistema como um todo, de maneira mais

eficiente, possibilitando a oferta de uma quantidade maior de conteúdos. Além disso, possibilita a oferta do serviço de apresentação deslocada no tempo, atendendo às expectativas do usuário no que se refere à diversidade de conteúdos e serviços ofertados. São minimizados, também, os custos de infra-estrutura devido à necessidade de um número menor de servidores na borda da rede para atender às requisições dos usuários, uma vez que os próprios usuários contribuem na distribuição dos conteúdos. E finalmente, a transmissão *multicast* é aproveitada para a realização de *cache* transparente sem a necessidade de retransmissão posterior por parte do provedor de conteúdo, quando ocorre a solicitação deslocada no tempo deste mesmo conteúdo por parte dos usuários, uma vez que o conteúdo já se encontra armazenado de maneira distribuída em diversos pontos da rede.

Uma desvantagem referente à utilização de P2P em comparação às soluções existentes baseadas em *unicast* decorre de uma maior latência para o início da exibição de um vídeo devido ao tempo gasto na troca de mensagens necessária para descobrir outros *peers* que possuam o conteúdo desejado. Porém, os resultados dos testes, descritos no capítulo seguinte, mostram que a sobrecarga gerada por esta troca de informações de controle é relativamente baixa, e que é possível obter baixa latência para o início da exibição utilizando-se blocos de vídeo pequenos.

6 DESCRIÇÃO E ANÁLISE DOS RESULTADOS

Neste capítulo, são descritos os resultados obtidos nos testes do protótipo do sistema IPTV proposto, que emprega as funcionalidades de *multicast* da rede de distribuição para o caso de transmissões lineares, e as funcionalidades de armazenamento (*cache*) distribuído e de sistemas P2P suportadas pela mesma rede de distribuição para o caso de apresentações deslocadas no tempo.

Tais testes são descritos apresentando-se: a topologia e os cenários de teste aos quais o protótipo desenvolvido como prova de conceito do sistema foi submetido. Além disso, são descritos o método empregado na execução destes testes, bem como os resultados obtidos.

Durante os procedimentos de testes e de avaliação do desempenho do sistema IPTV proposto, foi possível alterar e refinar partes da arquitetura de modo a se aperfeiçoar o sistema proposto para os cenários de aplicação considerados neste trabalho. Assim sendo, foram avaliados:

1. O impacto do tamanho do bloco de vídeo em:
 - a) Latência para o início da exibição do vídeo selecionado no modo de apresentação deslocada no tempo;
 - b) *Overhead* gerado pelas mensagens de sinalização e controle necessárias para a operação correta do sistema;
 - c) Tempo total de obtenção de um conteúdo selecionado.
2. O impacto da capacidade de processamento do equipamento do usuário na latência para o início da exibição do vídeo;
3. O impacto da replicação do conteúdo em *caches* e outros clientes na latência para o início da exibição;
4. O impacto do uso de informações de localidade no processo de escolha dos *peers* em:
 - a) Latência para o início da exibição do vídeo selecionado no modo de apresentação deslocada no tempo;
 - b) Taxa de utilização dos enlaces internos da rede.

Com relação aos equipamentos utilizados, o *Proxy* e os *Caches* foram instalados em servidores HP Proliant DL145 OPT 2.2 1MB com processadores Intel Xeon(R) Dual Core CPI 3050 2.13 GHz e 4 GB de memória RAM, com sistema operacional Ubuntu 7.10. Os roteadores são na verdade servidores (HP Proliant DL140 G2 com processador Intel Xeon Dual Core 2.80 GHz e 1 GB de memória RAM) rodando o software XORP (*eXtensible Open Router Platform*)³⁰ sobre uma distribuição de Linux (Fedora 6), que implementa diversos protocolos de roteamento incluindo protocolos de roteamento *multicast* (e.g., IGMP e PIM-SM utilizados neste trabalho); os switches são VLANs de um único *switch* físico (um Linksys modelo DGS 3048); e finalmente, os clientes rodam em *barebone PCs*, com processador Intel Celeron 2.2 GHz e 1 GB de memória RAM.

6.2 Caracterização do Conteúdo Utilizado nos Testes

Para poder executar os testes múltiplas vezes sob as mesmas condições além de poder comparar os resultados dos testes com diferentes parâmetros (e.g., tamanho do bloco de vídeo, *peers* com o conteúdo disponível para *upload*, número de servidores de *cache* e capacidade de processamento do equipamento do usuário) o mesmo conteúdo foi utilizado nos diversos testes, e em todas as repetições destes.

Utilizou-se um conteúdo o mais próximo possível daqueles empregados em sistemas de IPTV, com taxa de bits variável, codificado em H.264, com GoP (*Group of Picture*) de 25 quadros (i.e., um I-frame a cada 25 quadros), 25 quadros por segundo, taxa de transmissão média 2,2 Mbps, mínima de 0,9 Mbps e máxima de 5,9 Mbps. A taxa de bits do conteúdo é apresentada na Figura 6.2. Além disso, para possibilitar a execução de cada um dos testes múltiplas vezes, o conteúdo utilizado tinha duração de 5 minutos (300 segundos).

³⁰ <http://www.xorp.org/>

Figura 6.2 - Taxa de bits do conteúdo

6.3 Método de Testes

Para propiciar a obtenção de medidas mais precisas, cada um dos experimentos foi repetido por 10 vezes, sendo então calculada a média e o desvio padrão das medidas. Antes de iniciar cada teste todo o sistema é “limpo” e reiniciado, para evitar qualquer tipo de resíduos que poderiam impactar o resultado dos testes impossibilitando comparações entre os diversos cenários.

Após a definição dos casos de teste, um *script* foi escrito para automatizar a execução dos testes, a coleta dos dados e a organização das informações coletadas em formato de tabela para facilitar a criação de gráficos e a interpretação dos resultados obtidos. Os cenários testados e os resultados obtidos seguindo este método são apresentados a seguir.

6.4 Cenários de Testes e Resultados Obtidos

Com o intuito de identificar os componentes da latência para o início da exibição, e mensurar esta latência e o *overhead* de sinalização e controle do sistema com relação ao tamanho do bloco de vídeo utilizado no sistema foram definidos alguns cenários de teste apresentados a seguir, juntamente com os resultados obtidos. Além disso, foram montados cenários de teste para mensurar o impacto de diversos parâmetros na latência inicial, no tempo de obtenção do conteúdo como um todo e no tráfego nos enlaces intermediários da rede. Tais parâmetros incluem: a capacidade de processamento do cliente, o número de *peers* que provêem conteúdo na rede P2P e a utilização de informações de localidade na escolha dos *peers*.

6.4.1 Componentes da Latência para o Início da Exibição

Para se identificar todos os componentes da latência para o início da exibição de um conteúdo deslocado no tempo, desde o instante em que o usuário seleciona determinado conteúdo até a exibição do primeiro quadro de vídeo na tela, todas as etapas deste processo foram analisadas. Na Figura 6.3 estes componentes são apresentados.

Figura 6.3 - Componentes da latência para o início da exibição

Após esta análise inicial, todas as mudanças de estado e eventos deste processo começaram a ser registradas, juntamente com o instante de ocorrência destas mudanças, sendo possível mensurar a duração de todas as suas etapas. A Figura

6.4 apresenta a parcela correspondente de cada componente (sendo que os componentes estão agrupados para facilitar a exibição e compreensão) para diversos tamanhos de bloco de vídeo. Observe que o tempo para início da exibição, após a obtenção do bloco, é constante. Isso se deve ao fato de que este tempo está diretamente relacionado à codificação e ao tamanho do GoP (*Group of Picture*) utilizado no conteúdo, não dependendo do tamanho do bloco de vídeo definido no sistema. Observe, ainda, que o tempo gasto com a obtenção dos dados da DHT e a troca de informações de controle do BitTorrent são ínfimos. Por outro lado, o *download* dos pedaços tem impacto significativo na latência inicial, e pode ser bastante reduzido com a utilização de blocos de vídeo menores.

Figura 6.4 - Parcela da latência correspondente a cada componente

6.4.2 **Overhead de Sinalização e Controle**

A utilização de blocos menores, para diminuir a latência para o início da exibição, acarreta, no entanto, em aumento do *overhead* de controle. Porém, como pode ser observado na Figura 6.5, o *overhead* de controle do BitTorrent em si, que corresponde a maior parte do *overhead* total, não se altera muito, uma vez que o tamanho total do conteúdo continua o mesmo e portanto o número de pedaços que são trocados via P2P é praticamente o mesmo. Por outro lado o *overhead* na obtenção dos metadados e da DHT aumenta, pois são replicadas informações nos metadados e mais listas de *peers* são armazenadas e obtidas da DHT.

Figura 6.5 - *Overhead* de sinalização e controle

Observe que apesar do aumento relativamente grande do *overhead* (33% de aumento) quando o tamanho do bloco é reduzido de 10 para 2 segundos, ainda assim tem-se um *overhead* total relativamente baixo (3%). Para este conteúdo, devido aos componentes constantes da latência inicial apresentados anteriormente, apesar da redução na componente relativa ao *download* dos dados ter sido de 75% (de aproximadamente 1,6 para 0,4 segundo), a latência inicial total foi reduzida em apenas 32% (de 3,7 para 2,5 segundos). Ainda assim, esta redução da latência é significativa e positiva, e o aumento no *overhead* pode ser considerado aceitável.

6.4.3 Tempo Total de Obtenção do Conteúdo

Por outro lado, a partir da análise do tempo total para a obtenção do conteúdo todo, percebe-se que apesar do *overhead* não aumentar muito com relação à quantidade de dados de sinalização e controle trafegados na rede, o tempo de troca de informações causa grande aumento no tempo total de *download* quando o tamanho do bloco é pequeno demais. A redução do tamanho do bloco de 10 para 2 segundos, por exemplo, causa um aumento significativo no tempo total de *download*, como mostra a Figura 6.6.

Figura 6.6 - Tempo total de obtenção do conteúdo

Observe que ainda assim, com tamanho de bloco de 2 segundos, o tempo total de obtenção do conteúdo é de aproximadamente 150 segundos, sendo suficiente para a não interrupção da exibição do conteúdo utilizado nos testes (a taxa média de obtenção ainda assim é o dobro do *bit rate* médio do conteúdo, uma vez que 300 segundos de conteúdo são obtidos em 150 segundos). Porém devido ao aumento de 127% (de 66 para 151 segundos) no tempo total de obtenção do conteúdo, a utilização de blocos muito pequenos deve ser considerada com cautela, sendo que o tamanho ideal do bloco de vídeo a ser utilizado depende das características do conteúdo em questão.

6.4.4 Capacidade de Processamento do Cliente

Como um *set-top box* possui capacidade de processamento muito limitada, e o cálculo dos *hashes* para verificação de integridade é uma operação que exige certa capacidade de processamento, comparou-se o desempenho de dois tipos diferentes de clientes: um computador pessoal convencional (processador Intel Pentium 4 3.2 GHz Dual Core) e um *barebone PC*, com capacidade limitada de processamento (Intel Celeron 2.2 GHz). Para realizar o teste sem nenhuma interferência de fatores externos, configurou-se o *Proxy* com os conteúdos de teste e os clientes de modo a terem o *Proxy* como única fonte do conteúdo deslocado no tempo. Cada cliente foi testado separadamente, sendo reiniciado todo o sistema entre testes para assegurar

que o sistema estava exatamente no mesmo estado no início de cada teste. A Figura 6.7 mostra a diferença na latência para o início da exibição do conteúdo em função da capacidade de processamento do cliente.

Figura 6.7 - Impacto da capacidade de processamento do cliente na latência para o início da exibição

Observe que para blocos de vídeo pequenos, a diferença de latência para o início da exibição do conteúdo se reduz e praticamente independe da capacidade de processamento do cliente. Sendo assim, pode-se dizer que a utilização de blocos de tamanho pequeno ameniza também os problemas de se utilizar dispositivos com capacidade de processamento limitada como o equipamento do usuário final (e.g., *set-top box*).

6.4.5 Replicação do Conteúdo em *Caches* e Outros Clientes

Nesta seção objetiva-se analisar o impacto do número de *peers* que possuem o conteúdo desejado, na latência para o início da sua exibição. Para isso, configurou-se o sistema de maneira que além do *Proxy*, dois nós de *cache* armazenaram o conteúdo durante a transmissão ao vivo do mesmo e três clientes obtiveram este mesmo conteúdo previamente e serviram como fonte de dados para o cliente solicitante. Neste cenário, foram obtidas as latências para os vários tamanhos de

bloco de vídeo e comparadas com os resultados obtidos no cenário anterior, onde somente o *Proxy* possuía e provia o conteúdo ao cliente. A Figura 6.8 apresenta os resultados deste teste, mostrando que o simples fato de ter mais *peers* no sistema não implica redução significativa da latência, uma vez que um *peer* sozinho consegue servir outro quase na velocidade máxima que este consegue receber.

Figura 6.8 - Impacto do número de *peers* servindo o conteúdo na latência para o início da exibição

Observe ainda que o *overhead* de sinalização e controle aumenta em torno de 15% quando o conteúdo é recebido de 6 *peers* ao invés de somente 1, conforme apresentado na Figura 6.9. Portanto, é desejável manter baixo o número de *peers* que fornecem dados para determinado usuário num determinado instante, uma vez que o *overhead* aumenta com o aumento do número de *peers* enquanto que a redução na latência inicial é irrisória.

Figura 6.9 – Impacto do número de *peers* no *overhead* de sinalização e controle

No entanto, é importante notar que, apesar de não perceptível sob o ponto de vista do usuário, a replicação do conteúdo é importante para diminuir a carga no *Proxy*, permitindo que a solução escale para um número grande de usuários.

6.4.6 Impacto do Uso de Informações de Localidade

Uma grande preocupação das operadoras de rede com relação ao tráfego P2P deve-se ao fato de que este pode gerar um grande volume de tráfego inter-provedor muitas vezes desnecessário, uma vez que os pedaços poderiam ser obtidos de *peers* dentro do mesmo ISP (*Internet Service Provider*), mas acabam sendo requisitados a *peers* pertencentes a outro domínio devido à característica de aleatoriedade na seleção dos *peers* do protocolo original da rede P2P. Portanto, faz-se necessário utilizar informações, como por exemplo, sobre a topologia física da rede, para que a distribuição P2P de conteúdo seja o mais eficiente possível e realmente benéfica para o provedor em termos da quantidade de recursos utilizados na distribuição deste conteúdo (CHEN ET AL., 2007). A Figura 6.10 mostra o impacto da utilização de informações de localidade (conforme mecanismo de seleção de *peers* descrito na seção 5.1.5) na latência para o início da exibição do conteúdo selecionado, comparado ao caso em que não é utilizada tal informação, para dois cenários: no primeiro – com localidade no Cliente 1 – este cliente obtém o

conteúdo somente do outro *peer* (Cliente 2) que está na mesma sub-rede que ele, enquanto que no segundo – com localidade no Cliente 3 – este obtém o conteúdo tanto de outro cliente que está na mesma sub-rede que ele (Cliente 4), quanto de um servidor de *cache* (Cache 2), que também está na mesma sub-rede. A diferença observada na latência deve-se ao fato de o *cache* conseguir servir o cliente mais rápido que um cliente que possui recursos bem mais limitados.

Figura 6.10 – Impacto do uso de informações de localidade na escolha de *peers* na latência para o início da exibição

Note, no entanto, que a diferença na latência inicial fica muito pequena quando blocos menores que 10 segundos são utilizados, e que com as informações de localidade pode-se reduzir imensamente a utilização dos enlaces de rede. Observe na Figura 6.11 que o tráfego em todos os enlaces internos é reduzido para praticamente zero no caso do Cliente 1 obter o conteúdo utilizando informações de localidade. No caso do Cliente 3 obter o conteúdo utilizando informações de localidade, somente o enlace do servidor de *cache* da mesma sub-rede é utilizado, sendo que o tráfego nos demais enlaces também se reduz praticamente a zero. Sendo assim, com o uso das informações de localidade os enlaces somente seriam utilizados se não existissem *peers* locais suficientes.

Figura 6.11 - Redução do tráfego nos enlaces de rede decorrentes do uso de localidade

6.5 Considerações Finais

Neste capítulo, foram apresentados os testes realizados com o protótipo do sistema de IPTV e os resultados obtidos. Um dos principais problemas detectados nesta fase de testes foi a alta latência para início de apresentação deslocada no tempo. Assim sendo, logo no início desta fase, os componentes da latência para o início da exibição do conteúdo no serviço de apresentação deslocada no tempo foram identificados, possibilitando o aperfeiçoamento do protótipo. Com isto, eliminaram-se atrasos desnecessários devido à implementação não otimizada do protótipo, reduzindo a latência inicial.

Parte desta latência, como mostrado na Figura 6.3, é dependente do tocador de vídeo utilizado, parte deve-se ao protocolo P2P, incluindo o controle e a obtenção do conteúdo via rede P2P, e parte deve-se ao tipo de codificação do conteúdo utilizado.

Após esta identificação inicial dos componentes da latência para o início da exibição do conteúdo no serviço de apresentação deslocada no tempo, iniciou-se a fase de avaliação de desempenho do protótipo, medindo-se tal latência e o *overhead* de controle da operação deste serviço. Conforme o esperado, observou-se, durante os testes realizados, que a latência para o início da exibição do vídeo diminui quando utilizado blocos de vídeo menores. Porém, uma observação importante foi que, com

a utilização de blocos de vídeo menores, o *overhead* permaneceu praticamente constante (até blocos de 10 segundos), crescendo somente quando utilizados blocos de vídeo muito pequenos (e.g., blocos de 2 segundos), sendo que, mesmo neste caso, o *overhead* continua relativamente baixo (aproximadamente 3,5%). Além disso, pode-se observar que o *overhead* continua baixo mesmo quando o conteúdo encontra-se replicado e é distribuído por diversos *peers* simultaneamente. Esta característica é importante quando se pensa no sistema como um todo, no qual se deseja replicar o conteúdo para aliviar a carga dos servidores (e.g., o *Proxy*) que possuem tal conteúdo.

Por meio dos testes realizados empregando-se equipamentos de usuário com diferentes capacidades de processamento foi possível analisar o impacto desta capacidade na latência para o início da exibição do conteúdo no serviço de apresentação deslocada no tempo. Os resultados dos testes mostram que, com a utilização de blocos de vídeo pequenos, o impacto da capacidade de processamento destes equipamentos se torna menos relevante, não influenciando consideravelmente tal latência.

Adicionalmente, o impacto da utilização de informações de localidade dos *peers* no mecanismo de seleção destes foi testado, mostrando que a utilização de tais informações pode ser extremamente benéfica ao sistema, proporcionando redução na carga imposta à rede pela distribuição de conteúdos para apresentação deslocada no tempo.

7 CONSIDERAÇÕES FINAIS

Neste trabalho, uma arquitetura híbrida de IPTV, que combina transmissão *multicast*, armazenamento distribuído dos conteúdos e distribuição P2P, para oferta do serviço de apresentação deslocada no tempo, foi desenvolvida. O protótipo implementado funciona como prova de conceito, mostrando a viabilidade desta arquitetura de IPTV para oferta deste serviço. Apesar do protótipo apresentar algumas limitações específicas da implementação realizada para a prova de conceito, estas limitações não são vinculadas à arquitetura propriamente dita, mas na maioria dos casos às APIs empregadas na sua implementação conforme foi discutido no capítulo 5.

A partir do protótipo desenvolvido, testes foram realizados para se avaliar o desempenho do sistema, sendo que os resultados apresentados no capítulo 6 indicam um desempenho que satisfaz os requisitos de latência inicial e *overhead*. Com relação à escalabilidade do sistema, pode-se dizer que a escalabilidade da arquitetura está diretamente relacionada com a escalabilidade do BitTorrent, sendo, portanto, escalável. Porém, para se medir mais precisamente o quão escalável a solução proposta é, bem como qual a redução propiciada nos custos de infraestrutura, faz-se necessário um trabalho de simulação desta arquitetura, que pode ser desenvolvido como um trabalho futuro.

Com isso, mostrou-se a possibilidade de ofertar apresentação defasada dos conteúdos para muitos usuários, com: a distribuição eficiente destes conteúdos utilizando o paradigma P2P; o aproveitamento da infra-estrutura ociosa do cliente para auxiliar na distribuição e evitar a necessidade de maior infra-estrutura instalada na rede; e o armazenamento dos conteúdos aproveitando o fato dos mesmos estarem sendo transmitidos linearmente nos canais de TV, para evitar que estes precisem ser retransmitidos por toda a rede, desde o provedor de conteúdo até o usuário final no instante da requisição do usuário, e minimizar situações de congestionamento dos enlaces desnecessárias.

7.1 Contribuições e Inovações da Dissertação

Entre as contribuições e inovações deste trabalho valem destacar a arquitetura em si desenvolvida no escopo de um projeto de pesquisa e, mais especificamente, o processo de ingestão de conteúdos em uma rede P2P aproveitando-se da transmissão *multicast* destes, patenteado em (GALLO; CARVALHO; ET AL., 2008).

Além disso, outra patente resultante deste trabalho está em processo de submissão, referente a um mecanismo de prioridade, baseado no gerenciamento do *buffer* de vídeo, para ser utilizado como substituto do atual mecanismo de *unchoke* do BitTorrent no cenário de IPTV.

7.2 Trabalhos Futuros

Diversos trabalhos relacionados a partes específicas da arquitetura podem ser realizados derivando deste trabalho. Alguns destes são listados e exemplificados a seguir.

- Extensivo trabalho de simulação da arquitetura proposta para definir-se: a quantidade ideal de elementos de *cache* no sistema, dependendo do número de usuários e do padrão de comportamento destes, e a redução efetiva dos custos comparativamente à solução tradicional baseada em Redes de Distribuição de Conteúdo (CDNs).
- Generalização da solução para distribuição de conteúdos produzidos pelos usuários, na qual conteúdos (incluindo fluxos ao vivo) poderiam ser adicionados ao sistema não somente pelo provedor, mas também pelos usuários do sistema;
- Oferta de vídeo conferência com armazenamento distribuído da mesma para possibilitar que usuários assistam qualquer pedaço da conferência, a qualquer momento, inclusive podendo ingressar durante a conferência e assistir o início da mesma;
- Utilização de técnicas de proteção de direitos autorais, para garantir que os conteúdos do sistema não sejam utilizados de maneira imprópria (e.g.,

retransmitido ou compartilhado por usuários do sistema com pessoas sem autorização de receber tal conteúdo);

- Verificação de autenticidade, uma vez que a distribuição se baseia na transmissão de pedaços dos conteúdos por usuários, facilitando a possibilidade de transmissão de conteúdo adulterado;
 - Atualmente isso é parcialmente realizado pela verificação dos *hashes* do arquivo de *torrent*, provenientes de um repositório confiável. Porém a verificação de integridade e autenticidade do vídeo sem a necessidade dos *hashes* do BitTorrent seria interessante, como trabalho futuro, para garantir de fato a integridade e autenticidade dos dados.
- Utilização da topologia de rede e de informações de localidade dos *peers* para evitar tráfego de dados nos enlaces internos da rede desnecessariamente.

REFERÊNCIAS

ABRAM-PROFETA, E. L.; SHIN, K. G. Scheduling video programs in near video-on-demand systems. In: *Proceedings of the 5th ACM International Conference on Multimedia*. p. 359-369. Seattle, Washington, United States: ACM, 1997.

AKAMAI TECHNOLOGIES. **Akamai Solution: Akamai Media Delivery**. Disponível em: http://www.akamai.com/dl/brochures/akamai_media_delivery_sb.pdf. Acesso em: 8 jul. 2008.

AKAMAI TECHNOLOGIES. **Akamai Solution: Stream OS**. Disponível em: http://www.akamai.com/dl/brochures/STREAM_OS_BROCHURE.pdf. Acesso em: 8 jul. 2008.

AKAMAI TECHNOLOGIES. **Akamai Case Study: Globo.com**. Disponível em: http://www.akamai.com/dl/casestudy/Akamai_CaseStudy_globo.pdf. Acesso em: 8 jul. 2008.

BINDAL, R.; CAO, P.; CHAN, W.; ET AL. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In: *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. Lisboa, Portugal: IEEE Computer Society, 2006.

BYERS, J. W.; LUBY, M.; MITZENMACHER, M.; REGE, A. A digital fountain approach to reliable distribution of bulk data. **SIGCOMM Computer Communication Review**, v. 28, n. 4, p. 56-67, 1998.

CAIN, B.; DEERING, S.; KOUVELAS, I.; FENNER, B.; THYAGARAJAN, A. RFC 3376 - Internet Group Management Protocol, Version 3. **Internet Engineering Task Force**, 2002.

CASTRO, M.; DRUSCHEL, P.; HU, Y. C.; ROWSTRON, A. **Exploiting network proximity in peer-to-peer overlay networks**. Technical report MSR-TR-2002-82, 2002. Disponível em: <http://research.microsoft.com/en-us/um/people/antr/OLD/PAST/location.pdf>. Acesso em: 29 jan. 2009.

CASTRO, M.; DRUSCHEL, P.; KERMARREC, A.; ET AL. SplitStream: high-bandwidth multicast in cooperative environments. In: *Proceedings of the 19th ACM Symposium on Operating Systems Principles*. p. 298-313. Bolton Landing, NY, USA: ACM, 2003.

CEBALLOS, M.; GORRICHIO, J. P2P file sharing analysis for a better performance. In: *Proceedings of the 28th International Conference on Software Engineering*. p. 941-944. Shanghai, China: ACM, 2006.

CHEN, Y.; HUANG, Y.; JANA, R.; ET AL. When is P2P technology beneficial for IPTV services? In: *Proceedings of the 17th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'07)*. Illinois, USA : ACM, 2007.

CHU, Y.; RAO, S. G.; SESHAN, S.; ZHANG, H. A case for end system multicast. **IEEE Journal on Selected Areas in Communications**, v. 20, n. 8, p. 1456-1471, 2002.

COHEN, B. Incentives Build Robustness in BitTorrent. In: *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*. v. 6. Berkeley, USA, 2003.

CRANOR, C. D.; GREEN, M.; KALMANEK, C.; ET AL. Enhanced Streaming Services in a Content Distribution Network. **IEEE Internet Computing**, v. 5, n. 4, p. 66-75, 2001.

DESHPANDE, H.; BAWA, M.; GARCIA-MOLINA, H. Streaming live media over peers. **Work at CS-Stanford**, 2002.

DIOT, C.; LEVINE, B. N.; LYLES, B.; KASSEM, H.; BALENSIEFEN, D. Deployment issues for the IP multicast service and architecture. **IEEE Network**, v. 14, n. 1, p. 78-88, 2000.

ELLACOYA NETWORKS. Ellacoya Data Shows Web Traffic Overtakes Peer-to-Peer (P2P) as Largest Percentage of Bandwidth on the Network. . Disponível em: www.businesswire.com/news/google/20070618005912/en. Acesso em: 15 jul. 2008.

EL-SAYED, A.; ROCA, V.; MATHY, L. A survey of proposals for an alternative group communication service. **IEEE Network**, v. 17, n. 1, p. 46-51, 2003.

ESTRIN, D.; FARINACCI, D.; HELMY, A.; ET AL. RFC 2362 - Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. **Internet Engineering Task Force**, 1998.

FENNER, W. RFC 2236 - Internet Group Management Protocol, Version 2. **Internet Engineering Task Force**, 1997.

FLOYD, S.; HANDLEY, M.; PADHYE, J.; WIDMER, J. Equation-based congestion control for unicast applications. In: *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. p. 43-56. Stockholm, Sweden: ACM, 2000.

FREEDMAN, M.; MAZIÈRES, D. Sloppy Hashing and Self-Organizing Clusters. In: **Peer-to-Peer Systems II**. p. 45-55, 2003.

FREEDMAN, M.; RAO, S.; SRIPANIDKULCHAI, K. Considering priority in overlay multicast protocols under heterogeneous environments. **IEEE INFOCOM**, 2006.

FREEDMAN, M. J.; FREUDENTHAL, E.; MAZIÈRES, D. Democratizing content publication with Coral. In: *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI'04)*. San Francisco, CA, USA, 2004.

FREEDMAN, M. J.; VUTUKURU, M.; FEAMSTER, N.; BALAKRISHNAN, H. Geographic locality of IP prefixes. In: *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. p. 153-158. Berkeley, CA: USENIX Association, 2005.

GALLO, D.; CARVALHO, T.; DAMOLA, A.; SOUZA, V. **A Method for Ingesting Multicast Content in a Peer-to-Peer Network**. PCT/EP2008/057885, 20 jun. 2008.

GALLO, D.; COROAMA, V.; ET AL. Time-Shift Based on P2P: Exploiting Network Resources for a Hybrid IPTV Architecture. In: *Proceedings of the 7th International Information and Telecommunication Technologies Symposium*. Foz do Iguaçu, Paraná, Brazil, 2008.

GALLO, D.; MIERS, C.; COROAMA, V.; ET AL. A Multimedia Delivery Architecture for IPTV with P2P-based Time-Shift Support. In: *Proceedings of the 6th Annual IEEE Consumer Communications & Networking Conference (CCNC 2009)*. Las Vegas, NV, USA, 2009.

GANESH, A. J.; KERMARREC, A.; MASSOULIÉ, L. Peer-to-Peer Membership Management for Gossip-Based Protocols. **IEEE Transactions on Computers**, v. 52, n. 2, p. 139-149, 2003.

GOYAL, V. K. Multiple description coding: compression meets the network. **IEEE Signal Processing Magazine**, v. 18, n. 5, p. 74-93, 2001.

GRAHAM-ROWE, D. The future of television. **The New Scientist**, v. 197, n. 2647, p. 35-39, 2008.

HUANG, Y.; CHEN, Y.; JANA, R.; ET AL. Challenges of P2P Streaming Technologies for IPTV Services. In: *Proceedings of IPTV Workshop International World Wide Web Conference*. Edinburgh, Scotland, United Kingdom, 2006.

JANARDHAN, V.; SCHULZRINNE, H. Peer assisted VoD for set-top box based IP network. In: *Proceedings of the 2007 Workshop on Peer-to-Peer Streaming and IP-TV*. p. 335-339. Kyoto, Japan: ACM, 2007.

KIM, C.; LEE, S. Multiple description motion coding algorithm for robust video transmission. In: *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems (ISCAS'00)*. v. 4, p. 717-720. Geneva, 2000.

KOSTIĆ, D.; RODRIGUEZ, A.; ALBRECHT, J.; VAHDAT, A. Bullet: high bandwidth data dissemination using an overlay mesh. **ACM SIGOPS Operating Systems Review**, v. 37, n. 5, p. 282-297, 2003.

LAO, L.; CUI, J.; GERLA, M.; MAGGIORINI, D. A comparative study of multicast protocols: top, bottom, or in the middle? In: *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*. v. 4, p. 2809-2814, 2005.

LEE, J. Y. B.; LEUNG, R. W. T. Study of a server-less architecture for video-on-demand applications. In: *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '02)*. v. 1, p. 233-236, 2002.

LEE, J.; LEE, G.; SEOK, S.; CHUNG, B. Advanced Scheme to Reduce IPTV Channel Zapping Time. In: **Managing Next Generation Networks and Services**. p. 235-243, 2007.

LEIBOWITZ, N.; RIPEANU, M.; WIERZBICKI, A. Deconstructing the Kazaa network. In: *Proceedings of the 3rd IEEE Workshop on Internet Applications (WIAPP'03)*. p. 112-120, 2003.

LIAO, X.; JIN, H.; LIU, Y.; NI, L. M.; DENG, D. AnySee: Peer-to-Peer Live Streaming. In: *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM'06)*. v. 6, 2006.

LIU, J.; RAO, S. G.; LI, B.; ZHANG, H. Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. **Proceedings of the IEEE**, v. 96, n. 1, p. 11-24, 2008.

LUBY, M. LT codes. In: *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*. p. 271-280, 2002.

LUBY, M. G.; MITZENMACHER, M.; SHOKROLLAHI, M. A.; SPIELMAN, D. A.; STEMANN, V. Practical loss-resilient codes. In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. p. 150-159. El Paso, Texas, United States: ACM, 1997.

PADMANABHAN, V. N.; WANG, H. J.; CHOU, P. A.; SRIPANIDKULCHAI, K. Distributing streaming media content using cooperative networking. In: *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'02)*. p. 177-186. Miami, Florida, USA: ACM, 2002.

PAI, V.; KUMAR, K.; TAMILMANI, K.; SAMBAMURTHY, V.; MOHR, A. Chainsaw: Eliminating Trees from Overlay Multicast. In: **Peer-to-Peer Systems IV**. p. 127-140, 2005.

PALLIS, G.; VAKALI, A. Insight and perspectives for content delivery networks. **Communications of the ACM**, v. 49, n. 1, p. 101-106, 2006.

DE PINHO, L. B.; DE AMORIM, C. L. Assessing the efficiency of stream reuse techniques in P2P video-on-demand systems. **Journal of Network and Computer Applications**, v. 29, n. 1, p. 25-45, 2006.

SAROIU, S.; GUMMADI, K. P.; GRIBBLE, S. D. Measuring and analyzing the characteristics of Napster and Gnutella hosts. **Multimedia Systems**, v. 9, n. 2, p. 170-184, 2003.

SCHULZE, H.; MOCHALSKI, K. **Internet Study 2007**. Disponível em: http://www.ipoque.com/userfiles/file/internet_study_2007.pdf. Acesso em: 8 jul. 2008.

SIMPSON, W.; GREENFIELD, H. **IPTV and Internet Video: Expanding the Reach of Television Broadcasting (NAB Executive Technology Briefings)**. p. 264. Focal Press, 2007.

SRIPANIDKULCHAI, K.; GANJAM, A.; MAGGS, B.; ZHANG, H. The feasibility of supporting large-scale live streaming applications with dynamic application endpoints. **ACM SIGCOMM Computer Communication Review**, v. 34, n. 4, p. 107-120, 2004.

TIAN, R.; ZHANG, Q.; XIANG, Z.; ET AL. Robust and efficient path diversity in application-layer multicast for video streaming. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 15, n. 8, p. 961-972, 2005.

VENKATARAMAN, V.; FRANCIS, P.; CALANDRINO, J. Chunkyspread: Multi-tree unstructured peer-to-peer multicast. In: *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006.

VENKATARAMAN, V.; YOSHIDA, K.; FRANCIS, P. Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast. In: *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP '06)*. p. 2-11. Washington, DC, USA: IEEE Computer Society, 2006.

WANG, F.; XIONG, Y.; LIU, J. mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast. In: *Proceedings of the 27th International Conference on Distributed Computing Systems*, 2007.

WAUTERS, T.; VAN DE MEERSSCHE, W.; DE TURCK, F.; ET AL. Management of Time-Shifted IPTV Services through Transparent Proxy Deployment. In: *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '06)*. p. 1-5, 2006.

XIAO, Y.; DU, X.; ZHANG, J.; HU, F.; GUIZANI, S. Internet Protocol Television (IPTV): The Killer Application for the Next-Generation Internet. **IEEE Communications Magazine**, v. 45, n. 11, p. 126-134, 2007.

XIE, H.; KRISHNAMURTHY, A.; SILBERSCHATZ, A.; YANG, Y. R. **P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers**. Disponível em: http://www.dcia.info/documents/P4P_Overview.pdf. Acesso em: 18 set. 2008.

ZHANG, X.; LIU, J.; LI, B.; YUM, T. P. CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In: *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*. v. 3, p. 2102-2111, 2005.

APÊNDICE A – Estrutura do Arquivo de Metadados do BitTorrent

Todos os dados nos arquivos de metadados do BitTorrent (arquivos de *torrent*) são codificados utilizando o padrão de codificação **bencoded**. Esta codificação é brevemente apresentada a seguir, sendo, na seqüência, mostrada a estrutura do arquivo de *torrent*.

Codificação bencoded: É uma maneira de se especificar e organizar dados em um formato conciso. Suporta *strings*, números inteiros, listas e dicionários, sendo estruturado da seguinte maneira:

- *strings*: São codificadas como <tamanho da *string* em base dez ASCII>:<dados da *string*>. Note que não existe um delimitador de início ou fim da *string*. Exemplo:
 - 4:casa representa a *string* “casa”.
- *números inteiros*: São codificados como i<inteiro em base dez ASCII>e. O “i” inicial e o “e” final são delimitadores. Números negativos são representados tal como i-3e. Números não podem ser prefixados com um zero, como em i04e, porém i0e é válido. Exemplo:
 - i3e representa o número inteiro 3.
- *listas*: São codificadas como l<valores *bencoded*>e. O “l” e o “e” são delimitadores inicial e final. Listas podem conter quaisquer tipos *bencoded*, incluindo números inteiros, *strings*, dicionários e outras listas. Exemplo:
 - l4:casaa5:carroe representa uma lista de duas *strings* [“casa”, “carro”].
- *dicionários*: São codificados como d<*string bencoded*><elemento *bencoded*>e. O “d” inicial e o “e” final são delimitadores. Note que as chaves precisam ser *strings* bencoded, e os valores podem ser quaisquer tipos bencoded, incluindo números inteiros, *strings*, listas e outros dicionários. Adicionalmente, as chaves devem aparecer em ordem. As *strings* podem ser comparadas utilizando comparação binária. Exemplos:

- **d4:casa7:moradia5:carro9:transportee** representa o dicionário {"casa" => "moradia", "carro" => "transporte"};
- **d5:nomesl1:a1:bee** representa {"nomes" => ["a", "b"]};
- **d4:nome5:diego7:webpage15:www.example.com5:local4:casae** representa {"nome" => "diego", "webpage" => "www.example.com", "local" => "casa"}.

Como dito anteriormente, todos os dados nos arquivos de metadados do BitTorrent são codificados utilizando o padrão de codificação *bencoded* apresentado. O conteúdo de um arquivo de *torrent* é um dicionário *bencoded*, contendo as chaves listadas a seguir. Todos os valores de *strings* de caracteres são codificados no formato UTF-8.

- *announce*: O endereço (URL) de anúncio do rastreador (*string*).
- *announce-list*: (opcional) Extensão da especificação original, utilizada para especificar listas de rastreadores reserva (de *backup*).
- *creation date*: (opcional) Data de criação do *torrent* no formato "UNIX epoch" (número inteiro de segundos desde 00:00:00 01/01/1970 UTC).
- *comment*: (opcional) Comentário textual do autor em formato livre (*string*).
- *created by*: (opcional) Nome e versão do programa utilizado para criar o *torrent* (*string*).
- *info*: Dicionário que descreve o(s) arquivo(s) do torrent. Existem duas formas possíveis: uma para o caso de *torrent* de um único arquivo e outra para o caso de múltiplos arquivos. As duas formas são apresentadas a seguir, sendo que algumas das chaves são comuns a ambas:
 - *piece length*: Tamanho (número de bytes) de cada pedaço em que o arquivo foi dividido (inteiro). Os pedaços possuem tamanho fixo, exceto pelo último pedaço que pode ser truncado. Este valor é comumente uma potência de 2, sendo os mais comuns 256k, 512k e 1M.
 - *pieces*: *String* que consiste da concatenação de todos os valores de *hash* SHA1 de 20 bytes de cada pedaço (*string*).

- Caso de torrent de arquivo único:
 - name: Nome do arquivo, puramente indicativo (*string*).
 - length: Tamanho do arquivo em bytes (inteiro).
- Caso de torrent de múltiplos arquivos:
 - name: Nome do diretório no qual os diversos arquivos serão armazenados, puramente indicativo (*string*).
 - files: Uma lista de dicionários, um para cada arquivo, contendo as seguintes chaves:
 - length: Tamanho do arquivo em bytes (inteiro).
 - path: Uma lista de *strings* correspondente ao caminho do arquivo, contendo subdiretórios e por último o nome do arquivo em si. Esta lista é codificada como uma lista de *strings bencoded*. Exemplo: “dir1/dir2/file.ext” é codificada como “l4:dir14:dir28:file.ext”.

APÊNDICE B – Parâmetros da Comunicação com o

Rastreador BitTorrent

A comunicação com o rastreador no BitTorrent é feita através do protocolo HTTP, sendo que o rastreador responde às requisições HTTP do tipo GET que os *peers* enviam. Essas requisições incluem métricas que auxiliam o rastreador a manter estatísticas gerais sobre o torrent. As respostas incluem uma lista de *peers* que ajuda o cliente a obter o conteúdo via P2P. Parâmetros são adicionados à URL do rastreador, obtida do arquivo de torrent, usando “?” depois da URL, seguido por seqüências de ‘parâmetro=valor’ separadas por “&”.

É importante notar que todos os dados binários precisam ser devidamente tratados antes de ser adicionado como parâmetro (especialmente o *info_hash* e o *peer_id*). Isto significa que qualquer byte que não na faixa de 0-9, a-z, A-Z, ‘.’, ‘-’, ‘_’ e ‘~’ precisa ser codificado utilizando o formato %nn, onde nn é o valor hexadecimal.³¹ Para um *hash* de 20 bytes igual a \x12\x34\x56\x78\x9a\xbc\xde\xfa\x12\x34\x56\x78\x9a, a correta codificação é %124Vx%9A%BC%DE%F1%23Eg%89%AB%CD%EF%124Vx%9A.

Levando isso em consideração, os parâmetros utilizados nas requisições dos clientes ao rastreador são apresentados a seguir:

- *info_hash*: Hash SHA1 de 20 bytes do campo info em formato *bencoded* do arquivo de metadados. Note que este é uma *substring* do arquivo de torrent.
- *peer_id*: *String* de 20 bytes utilizada como identificação do *peer*, sendo que cada *peer* gera seu próprio ID na inicialização aleatoriamente. Este ID deve ser único.
- *ip*: (opcional) Endereço IP (ou nome DNS) em que o *peer* se encontra. Em geral este parâmetro não é necessário uma vez que o IP do *peer* pode ser determinado pelo IP de origem da requisição HTTP. Este parâmetro só é necessário caso o IP de onde a requisição vem não seja o mesmo do *peer*, que acontece quando o cliente está se comunicando com o rastreador através de um *proxy* ou quando o cliente e o rastreador estão ambos sob NAT numa

³¹ Veja RFC 1738 para maiores detalhes.

mesma rede, caso no qual o *peer* deve fornecer seu endereço para acesso externo. É importante notar que diferentes implementações de rastreadores tratam este parâmetro de maneiras diferentes, sendo que alguns utilizam esta informação em determinados casos enquanto outros o ignoram completamente.

- *port*: Número da porta de escuta em que o *peer* está aguardando novas conexões. Tipicamente um *peer* tenta escutar na porta 6881 e se esta porta já estiver sendo utilizada tenta a 6882, depois a 6883, até a 6889, quando desiste.
- *uploaded*: Quantidade total de *upload* deste *peer* até o momento, codificada em base 10 em ASCII. Apesar de não ser explicitado na especificação original, um consenso é que o valor deste campo deve ser o número total de **bytes** que o *peer* forneceu.
- *downloaded*: Quantidade total de *download* deste *peer* até o momento, codificada em base 10 em ASCII. Também não é explicitado na especificação original a unidade deste campo, mas o consenso é que o valor deve ser em **bytes**.
- *left*: Quantidade de bytes que o *peer* ainda precisa obter para concluir o *download*, codificada em base 10 em ASCII.
- *event*: (opcional) Se especificado o valor deve ser '*started*', '*completed*' ou '*stopped*' (ou vazio que é equivalente a não estar presente). Este campo vazio ou não especificado significa que a requisição é uma das realizadas periodicamente.
 - *started*: A primeira requisição do *peer* ao rastreador deve conter este valor de evento.
 - *stopped*: Este valor deve ser enviado quando a aplicação cliente se encerra normalmente.
 - *completed*: Utilizado para informar o rastreador que o *download* foi concluído. No entanto, quando o *download* já está completo quando o cliente inicializa, não se deve enviar este valor.

- *compact**: Quando este campo possui valor 1 significa que o cliente aceita resposta no formato compacto. A lista de *peers* é substituída por uma *string* de *peers*, com 6 bytes por *peer* (quatro para o endereço IP e dois para porta). Alguns rastreadores somente suportam o formato compacto, para economia de banda, recusando requisições que não contém “*compact=1*” ou simplesmente enviando no formato compacto a não ser que a requisição contenha “*compact=0*” (recusando a requisição).
- *no_peer_id**: Indica que o rastreador pode omitir o campo *peer id* no dicionário de *peers* da resposta. Esta opção é ignorada quando modo compacto é utilizado.
- *numwant**: (opcional) Número de *peers* que o cliente gostaria de receber na resposta do rastreador. Pode ser zero e se omitido tipicamente o valor padrão é 50.
- *key**: (opcional) Informação adicional não compartilhada com outros clientes para permitir um cliente provar sua identidade no caso de mudança de IP.
- *trackerid**: (opcional) Se recebido um *tracker id* em uma resposta anterior do rastreador, este parâmetro deve ser enviado com o mesmo valor.

Os parâmetros indicados com um “*” não fazem parte da especificação original do BitTorrent, mas muitas das aplicações clientes e de rastreadores existentes fazem uso destes para otimizar a comunicação. Por este motivo optou-se por apresentá-los neste documento, sendo que o mesmo ocorre com os parâmetros da resposta do rastreador, que também estão indicados por “*” quando não fazem parte da especificação oficial. A resposta do rastreador após obter uma requisição é um documento em formato “*text/plain*” consistido de um dicionário *bencoded* com os seguintes parâmetros:

- *failure reason*: Contém uma mensagem de erro legível com o porquê da requisição ter falhado. Quando presente nenhum outro parâmetro deve estar presente.
- *warning message**: (opcional) Similar ao anterior, porém a resposta é processada normalmente.

- *interval*: Intervalo, em segundos, que o cliente deve esperar entre envios das requisições periódicas ao rastreador.
- *min interval**: (opcional) Intervalo mínimo de anúncio. Quando presente o cliente não pode enviar requisições mais freqüentemente que esse valor.
- *tracker id**: Uma *string* que o cliente deve enviar de volta nas requisições subseqüentes.
- *complete**: Quantidade de *peers* que contém o conteúdo completo, chamados de *seeders* (inteiro).
- *incomplete**: Quantidade de *peers* que não contém o conteúdo completo, chamados de *leechers* (inteiro).
- *peers*: Lista de dicionários correspondente aos *peers*, cada um contendo os parâmetros *peer id*, *ip*, *port*.
 - *peer id*: Número de identificação selecionado pelo próprio *peer*, como descrito anteriormente para requisição ao rastreador (*string*).
 - *ip*: endereço IP do *peer* ou nome DNS (*string*).
 - *port*: número da porta de escuta do *peer* (inteiro).

A lista de *peers* ao invés de utilizar uma lista de dicionário como a descrita acima pode ser em formato binário, sendo que o valor do parâmetro *peers* neste caso é uma *string* múltipla de 6 bytes, onde os primeiros 4 bytes são o endereço IP e os últimos 2 bytes são o número da porta.

APÊNDICE C – Detalhamento das Mensagens do BitTorrent

O protocolo de mensagens entre *peers* do BitTorrent consiste de um *handshake* seguido por um fluxo contínuo de mensagens de tamanho definido no prefixo das mesmas. A mensagem de *handshake* é mandatória e deve ser a primeira mensagem transmitida no início da comunicação com qualquer outro *peer*. Essa mensagem possui tamanho de 49 bytes mais o número de bytes utilizados na *string* de identificação do protocolo. A mensagem de *handshake* e os respectivos campos são apresentados a seguir:

Handshake: <psrlen><pstr><reserved><info_hash><peer_id>

- *psrlen*: Tamanho da *string* “pstr”, sendo que esta informação ocupa 1 byte.
- *pstr*: *String* identificadora do protocolo, de tamanho definido acima. Na versão 1.0 do protocolo *psrlen*=19 e *pstr*=“*BitTorrent protocol*”.
- *reserved*: 8 bytes reservados, que são zero nas implementações, podendo ser utilizados para extensão do protocolo.
- *info_hash*: Hash SHA1 de 20 bytes do valor do campo *info* dos metadados. É o mesmo valor utilizado na comunicação com o rastreador e serve para identificar o arquivo a ser compartilhado. Se ambos os lados não transmitirem o mesmo valor a conexão é encerrada.
- *peer_id*: 20 bytes de identificação do *peer*. O mesmo contido nas requisições ao rastreador e em listas de *peers* nas respostas do rastreador. Se o *peer_id* recebido não é o esperado (ou seja, o mesmo recebido do rastreador na lista de *peers*) a conexão é encerrada.

Todas as demais mensagens do protocolo possuem formato <length prefix><message ID><payload>, sendo que o length prefix ocupa 4 bytes e especifica o tamanho da mensagem, o message ID ocupa um byte, e o payload depende do tipo da mensagem.

Keep-alive: <len=0000>

- A mensagem de *keep-alive* é uma mensagem de zero bytes, especificada através do prefixo de tamanho da mensagem com valor igual a zero. Este tipo de mensagem não possui *message ID* nem *payload*. *Peers* podem encerrar a

conexão caso não recebam nenhuma mensagem em certo período de tempo, portanto esse tipo de mensagem é utilizado para evitar o encerramento da conexão, mantendo-a ativa caso nenhum comando seja executado em um dado intervalo de tempo (geralmente dois minutos).

Choke: <len=0001><id=0>

- Mensagem de tamanho fixo (1 byte), sem *payload*. Tem a função de informar a outro *peer* que ele foi colocado no estado de *choked*, devendo parar de solicitar pedaços até que receba uma mensagem de *unchoke*.

Unchoke: <len=0001><id=1>

- Mensagem de tamanho fixo (1 byte), sem *payload*. Informa que o *peer* foi colocado no estado de *unchoked* e pode iniciar a requisição de pedaços caso deseje.

Interested: <len=0001><id=2>

- Mensagem de tamanho fixo (1 byte), sem *payload*. Esta mensagem é enviada quando existe interesse em obter algum pedaço que o outro *peer* possui, informando este desejo ao mesmo e esperando o recebimento da mensagem de *unchoke*.

Not interested: <len=0001><id=3>

- Mensagem de tamanho fixo (1 byte), sem *payload*. Informa a outro *peer* que não existe mais o interesse em nenhum dos pedaços que aquele *peer* possui.

Have: <len=0005><id=4><piece index>

- Mensagem de tamanho fixo (5 bytes), contendo o índice do pedaço que o *peer* terminou de obter e verificar a integridade, para informar aos outros *peers* que agora ele possui aquele pedaço caso alguém necessite.

Bitfield: <len=0001+X><id=5><bitfield>

- Esta mensagem somente pode ser enviada logo após o *handshake*, antes de enviar qualquer outra mensagem, e informa o outro *peer* sobre quais pedaços ele possui. Não precisa ser enviada caso não possua nenhum pedaço. O tamanho do bitfield (X) depende do número de pedaços que o arquivo em

questão contém, sendo que cada bit indica a disponibilidade ou não de um pedaço específico, de índice igual a posição do bit.

Request: <len=0013><id=6><index><begin><length>

- Este tipo de mensagem possui tamanho fixo (13 bytes) e é utilizada para requisitar parte de um pedaço, especificando o índice (*index*) deste, o *offset* dentro do pedaço (*begin*) e a quantidade de dados desejada (*length*).

Piece: <len=0009+X><id=7><index><begin><piece>

- Mensagem de tamanho variável, onde X é o número de bytes da parte do pedaço sendo transmitida. Contém o índice do pedaço (*index*), o *offset* dentro deste (*begin*) e os dados propriamente ditos <piece>.

Cancel: <len=0013><id=8><index><begin><length>

- Esta mensagem pode ser utilizada para cancelar a requisição de pedaços. O *payload* é idêntico ao da mensagem de *request* e é tipicamente utilizada no final do *download*, quando um mesmo pedaço pode ser solicitado a diversos *peers* simultaneamente para acelerar a conclusão do *download* (“*end game mode*”).

APÊNDICE D – Exemplo do Arquivo XML de Configuração dos Módulos

A seguir é apresentada a estrutura do arquivo XML de configuração utilizado no protótipo, na inicialização dos módulos.

```
<root>
  <channel>
    <name>TV Globo</name>
    <description>Rede Globo de Televisão.</description>
    <multicastAddress value="224.123.111.101"/>
    <ttl value="5"/>
    <port value="1234"/>
    <contents>
      <content>
        <name>Casseta e Planeta Urgente</name>
        <description>Comédia</description>
        <uri value="/media/sda1/content.avi"/>
        <date>
          <startTime value="2008-11-18 22:00:00 -0300"/>
          <endTime value="2008-11-18 23:00:00 -0300"/>
        </date>
        <torrentProperties>
          <interval value="10"/>
          <torrentURI value="ftp://repositorio/">
        </torrentProperties>
        <RemovalDate value="2008-12-31 23:59:59 -0200"/>
      </content>
      <content>...</content>
      ...
    </contents>
  </channel>
  <channel>...</channel>
  ...
</root>
```