Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Um protocolo *Cross-Layer* para Distribuição de Mídias Ao Vivo pelo Compartilhamento de Fluxos de Dados em Redes P2P Formada por Roteadores

Leandro Melo de Sales

Tese de Doutorado submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências, domínio da Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Angelo Perkusich e Hyggo Almeida
(Orientadores)

Campina Grande, Paraíba, Brasil

Resumo

O transporte de conteúdos multimídia em tempo real pela Internet é primordial em aplicações como voz sobre IP (VoIP), videoconferência, jogos e WebTV. Ao longo dos anos, observa-se um intenso crescimento de utilização das aplicações cujo cenário é caracterizado por um único nó transmissor e milhares de nós receptores. Por exemplo, o Youtube Live transmitiu os jogos da copa américa 2011 para 100 milhões de usuários. Um ponto crítico nessas aplicações é a sobrecarga de utilização de recursos computacionais nos servidores e canais de transmissão. Isto tem demandado investimentos exorbitantes na construção de Redes de Distribuição de Conteúdos por empresas como Google, Netflix etc. Porém, as aplicações continuam à mercê de protocolos de transportes tradicionais (TCP, UDP, DCCP e SCTP), que não foram projetados para transmitir dados multimídia em larga escala. Para suprir as limitações desses protocolos, os desenvolvedores implementam mecanismo para utilizar os recursos de rede de forma mais eficiente, destacando-se o uso do modelo de serviço P2P (Peer-to-Peer). Todavia, estas soluções são paleativas porque são disseminadas em forma de sistemas ou protocolos de aplicação, sendo impossível evitar a pulverização e fragmentação das mesmas, aumentando-se a complexidade de implantação em larga escala na Internet. Neste trabalho propõe-se um protocolo de transporte multimídia denominado *Global Media Transmission Protocol* (GMTP). O GMTP constrói dinamicamente uma rede de sobreposição entre os nós de uma transmissão (P2P), sem a influência direta da aplicação e com suporte à transmissão multicast e controle de congestionamento. Resultados preliminares apontam que o GMTP utiliza de forma eficiente os recursos de redes e melhora a escalabilidade das aplicações multimídia, evitando-se o retrabalho de desenvolvimento ao concentrar os principais mecanismos em um único protocolo de transporte.

Abstract

The transport of multimedia content in real time over the Internet is essential in applications such as voice over IP (VoIP), video conferencing, games and WebTV. In the last years, there has been an increasing number of applications with a single transmitting node and thousands of receiving nodes. For example, YouTube Live broadcasted games of the American Cup 2011 to 100 million users. A critical aspect in these applications is the overhead of using computing resources on servers and transmission channels. This has demanded exorbitant investment in building Content Delivery Networks (CDNs) by companies like Google, Netflix etc. However, the applications are still at the mercy of traditional transport protocols (TCP, UDP, SCTP and DCCP), which were not designed to transmit multimedia data in a large scale. To address the limitations of these protocols, developers implement a mechanism to use network resources more efficiently, specially using P2P (Peer to Peer) architectures. However, these solutions are palliative because they are disseminated in the form of systems or application protocols, where it is impossible to avoid scattering and fragmentation of them, increasing the complexity of large-scale deployment in the Internet. In this work it is proposed a multimedia transport protocol called Global Media Transmission Protocol (GMTP). The GMTP dynamically builds an overlay network between nodes in a P2P fashion, without the direct influence of the application and supporting multicast transmission and congestion control. Preliminary results indicate that the GMTP efficiently utilizes network resources and improves scalability of multimedia applications, avoiding the rework development by concentrating the main mechanisms in a single transport protocol.

Conteúdo

Capítulo 1

Trabalhos Relacionados

Neste capítulo, apresenta-se uma avaliação crítica acerca de um conjunto de trabalhos sobre sistemas e protocolos para distribuição de mídias ao vivo. Nesse contexto, inclui-se também os trabalhos que, apesar de não apresentarem uma proposta completa para distribuição de mídias ao vivo, apresentam-se como parte de um todo, com similaridades relevantes se comparado ao GMTP.

A compilação dos trabalhos a seguir foi realizada com base em informações obtidas e adaptadas de publicações encontradas em diversas fontes (revistas, conferências, livros, teses e dissertações) disponíveis na literatura. Na Seção 1.1, apresenta-se uma breve discussão acerca das aplicações e protocolos de distribuição de mídias ao vivo, enfatizando às propostas de protocolos que utilizam uma abordagem P2P e/ou P2P/CDN. Na Seção 1.2, apresenta-se os trabalhos relevantes para distribuição de mídias ao vivo em redes centradas no conteúdo. E, por fim, na Seção 1.3, apresenta-se uma breve discussão comparativa sobre os principais protocolos discutidos e o GMTP.

1.1 Aplicações e Protocolos para Distribuição de Mídias ao Vivo

Nos últimos anos, os pesquisadores vinculados à IETF propuseram a especificação de protocolos para transmissão e/ou distribuição de mídias ao vivo, declarando-os como padrões públicos. Em geral, executam-se os protocolos dessa categoria na camada de aplicação, os

quais permitem a criação, encerramento e controle de sessões de transmissão de mídias ao vivo, como videoconferência e TV através da Internet. Contudo, o esforço de padronização não tem sido suficiente, abrindo oportunidades para protocolos proprietários de empresas e da comunidade acadêmica para este mesmo fim. As soluções propostas englobam produtos de software bem como trabalhos acadêmicos que nunca foram postos em funcionamento em larga escala, mas que seu legado contribuiu para a evolução do estado da arte. Nos últimos 15 anos (pelo menos), a maior parte das soluções propostas para disseminação de mídias ao vivo são apresentadas como sistemas ou *middlewares* para o fim que se discute, salvas raras excessões.

Nessa perspectiva, a seguir, destacam-se as principais soluções com foco nos protocolos empregados (parte ou todo) utilizados para distribuir mídias ao vivo, geradas por um nó transmissor e transmitidas através da rede para milhares de nós receptores. Além de permear as soluções mais proeminentes, objetiva-se também mostrar ao leitor o panorama atual com vistas à pulverização de diversas soluções para o mesmo fim, como discutiu-se na Seção **??**.

1.1.1 H.323, SIP, RTP e RTSP

O processo de desenvolvimento de padrões abertos para distribuição de mídias ao vivo em larga escala teve início com um grupo de estudos da ITU-T (*International Telecommunication Union – Telecommunication Section*), em 1996. O ITU-T especificou o padrão H.323 [1], que estabelece uma arquitetura de comunicação destinada ao controle de conferências de voz, vídeo e dados sobre redes TCP/IP. O H.323 se tornou largamente utilizado, uma vez que, a época da especificação de sua segunda versão, em 1998, não havia qualquer padrão aberto e aceito pelo mercado capaz de atender às aplicações multimídia. Embora as transmissões multimídia em tempo real por *multicast* já estivessem sendo realizadas no Mbone [2] há algum tempo, não havia ainda qualquer padrão aberto para controle de conferências multimídia e utilizar o Mbone exigia muita intervenção humana e empresarial.

No período equivalente a maturação do H.323, a IETF iniciou o desenvolvimento de uma arquitetura de comunicação mais flexível e poderosa que o H.323. Alicerçada sobre o protocolo SIP (*Session Initiation Protocol*) [3], a arquitetura SIP teve logo potencial reconhecido, uma vez que supria todos os pontos fracos da arquitetura H.323, como a demora no estabelecimento de conexão (canais H.225 e H.245) e a complexidade de operação. Apesar

do SIP ser muito utilizado atualmente, trata-se de um protocolo de negociação de sessões focado para chamadas telefônica sobre uma rede IP, apesar de também oferecer suporte para negociação de vídeo.

Devido à natureza das redes de datagramas cujo paradigma é o do "melhor esforço" e às limitações do H.323 e SIP, sem qualquer garantia sobre os requisitos de tempo de entrega, novos protocolos foram propostos para atender as necessidades de transmissão de dados em tempo real. O protocolo em destaque é o RTP (*Real Time Protocol*) [4], um protocolo que permite às aplicações informarem o instante exato em que um determinado conteúdo deve ser reproduzido pelo sistema final receptor. O RTP permite transportar essa marcação de tempo no cabeçalho que envolve os dados referentes a mídia sendo transmitida, e permite que às aplicações tenham controle sobre número de sequência de cada pacote. O RTP atualmente constitui a base das transmissões multimídias ao vivo na Internet, sendo o padrão adotado em praticamente todas as soluções multimídia baseadas em IP.

Em paralelo ao RTP, as aplicações multimídia utilizam o *Real Time Streaming Protocol* (RTSP) [5]. O RTSP é um protocolo a nível de aplicação desenvolvido pela IETF para controle na transferência de dados com propriedades de tempo real. Tal protocolo torna possível a transferência, sob demanda, de dados em tempo real como áudio e vídeo. O RTSP serve para estabelecer e controlar vários fluxos sincronizados de mídias contínuas pertencentes a um evento (como o vídeo e o áudio). Seus idealizadores propõem que o uso do RTSP seja feito combinando-se outros protocolos, como o UDP e o RTP. NO RTSP, o conjunto de fluxos a ser controlado é definido por uma descrição de apresentação, normalmente um arquivo, que pode ser obtido por um cliente usando HTTP ou outros meios, armazenado em um local diferente do servidor de mídia. Uma descrição de apresentação pode conter informações sobre um ou mais fluxos que compõe a apresentação, como endereços de rede e informações sobre o conteúdo da apresentação, além de parâmetros que tornam possível ao cliente escolher a combinação mais apropriada das mídias.

O problema do H.323 foi sua grande complexidade de uso, dificultando a integração com às aplicações, dando brechas a outras propostas com propósitos similares, como foi o caso do SIP, RTP e do RTSP. Estes protocolos se limitam à fronteira entre a sinalização de uma sessão multimídia e todos os outros aspectos relacionados a efetiva disseminação de conteúdos ao vivo, que atualmente envolve a cooperação de nós, controle e troca de mapa de

buffer, adaptação de fluxo e controle de congestionamento. Isto implica na necessidade de mais componentes que permitam a distribuição de mídias ao vivo, ampliando a complexidade na construção de um sistema para este fim.

1.1.2 *HLS – HTTP Live Streaming*, *HDS – HTTP Dynamic Streaming*, *DASH – Dynamic Adaptive Streaming over HTTP* e *HSS – HTTP Smooth Streaming*

Os protocolos HLS (*HTTP Live Streaming*) [6], HDS (*HTTP Dynamic Streaming*) [7], o MPEG-DASH *Dynamic Adaptive Streaming over HTTP* [8, 9], e o Smooth Streaming [7], são quatro protocolos independentes propostos pela Apple, Adobe, MPEG e Microsoft, respectivamente. Em geral, propõe-se que as transmissões de fluxos de mídias ao vivo sejam realizadas por servidores web, sendo que tais protocolos incrementam as funções do protocolo HTTP, oferecendo-lhe suporte a descrição de uma mídia, marcação de tempo, segurança e principalmente adaptação do fluxo de dados em uma sessão ao vivo de transmissão multimídia. Sendo assim, pode-se afirmar que os protocolos HLS, HDS, MPEG-DASH, têm propósitos similares aos protocolos H.323, SIP, RTP e RTSP, porém com base no protocolo HTTP.

Uma característica comum entre os protocolos HLS, HDS, HSS e MPEG-DASH é que a transferência dos fluxos multimídia ocorre sem a manutenção de uma conexão, os nós clientes podem escolher entre diferentes modos de codificação multimídia e suporte à criptografia. Nessas soluções que exploram a convergência dos serviços multimídia com base na web, como ilustra-se na Figura 1.1, especifica-se a fonte da mídia através de uma URI (*Uniform Resource Identifier*) [10], que aponta para um arquivo contendo uma lista ordenada das URIs para as mídias a serem reproduzidas. Para reproduzir um determinado conteúdo, um nó cliente primeiro obtém o arquivo contendo a lista de URIs e então obtém e reproduz cada segmento de mídia especificado na lista. Periodicamente, o nó cliente recarrega o arquivo contendo a lista de URIs a fim de descobrir os próximos segmentos a serem reproduzidos. Este período corresponde ao tempo em que um segmento é consumido para ser reproduzido ao usuário final. Considerando a teoria de distribuição de conteúdos multimídia ao vivo discutido no Capítulo **??**, pode-se afirmar que um segmento contém apontadores para as par-

tes da mídia que acabara de ser gerada, portanto é equivalente ao mapa de buffer em uma arquitetura P2P para distribuição de conteúdos multimídia ao vivo.

Por exemplo, na Figura 1.2, ilustra-se um cenário de transmissão de um fluxo de dados multimídia entre um servidor e um cliente MPEG-DASH, proposto como padrão a ser publicado pela ISO (ISO/IEC 23009-1). O conteúdo da mídia é capturado e armazenado no servidor web, que são transmitidos para os clientes via HTTP. O servidor web MPEG-DASH armazena o conteúdo em duas partes. A primeira parte é um arquivo que descreve o conteúdo disponível, como os endereços dos servidores fontes, tempo de duração, formatos da mídia e resoluções, largura de banda máxima e mínima, aspectos de direitos autorais (DRM), dentre outras informações. A segunda parte são os segmentos, que são arquivos que contém, de fato, os bits de dados da mídia.

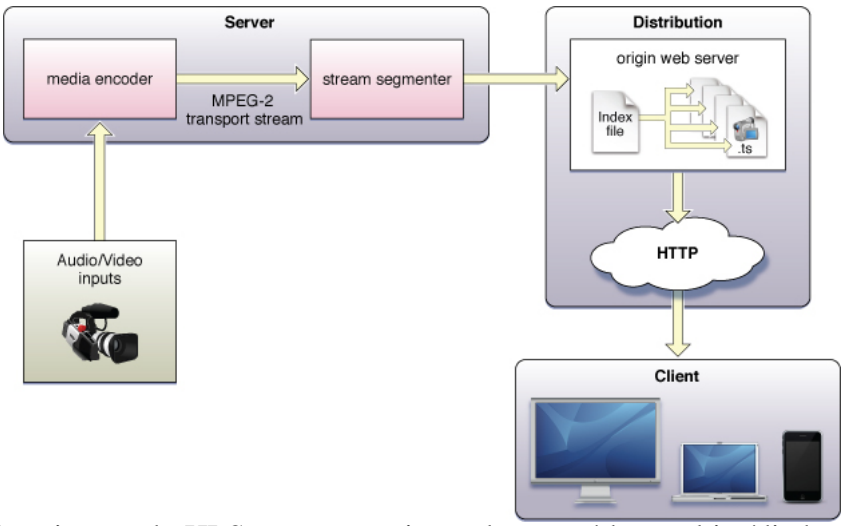


Figura 1.1: Arquitetura do HLS para transmissão de conteúdos multimídia baseado no protocolo HTTP. Figura extraída e adaptada de [6].

Uma aplicação baseada no protocolo HLS transmite, obrigatoriamente, fluxos de dados no formato MPEG2-TS (parte 1) e funciona apenas para o navegador Safari^{TM} e os dispositivos que suportam o sistema operacional iOS. Já as aplicações baseadas no protocolo HDS transmite, obrigatoriamente, um formato específico criado pela Adobe conhecido por fMP4 que, na prática consiste em diversos fragmentos de um arquivo codificado no formato MPEG-4 (partes 12 e 14). Os protocolos HDS, HSS e o MPEG-DASH suportam funções de adaptação de fluxos de dados multimídia por oferecerem suporte ao formato MPEG-4.

A principal estratégia das soluções de transmissão de vídeo baseadas em HTTP é adaptar

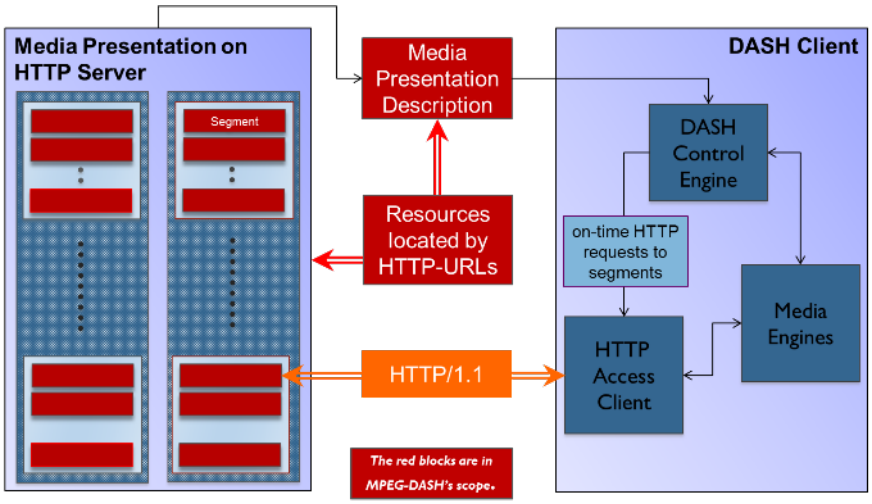


Figura 1.2: Arquitetura do MPEG-DASH para transmissão de conteúdos multimídia utilizando o protocolo HTTP. Os formatos e as funcionalidades dos blocos vermelhos são definidos pela especificação do MPEG-DASH. Adaptado de [8].

os fluxo de dados de acordo com diferentes taxas de transmissão e transcodificar o conteúdo para diferentes formatos suportados pelas aplicações em execução nos nós clientes. Isto permite que inúmeros clientes, que suportam diferentes formatos e conectados através de redes com diferentes capacidades de recepção, possam reproduzir o conteúdo de forma adaquada. Para que isso seja possível, os nós clientes devem monitorar sua capacidade de recepção de dados e solicitar o conteúdo multimídia de acordo com a taxa de bit corresponde e anunciada pelo servidor.

Considerações sobre os trabalhos

Os protocolos supracitadoss são mais um exemplo da pulverização das aplicações multimídia discutidas no Capítulo **??**. Com base na discussão anteriormente sobre a distribuição de mídias ao vivo através de servidores web com suporte aos protocolos supracitados, deve-se considerar os seguintes pontos:

- Os protocolos supracitados não oferecem efetivamente mecanismos para distribuição em larga escala. Para que isto ocorra, deve-se implementar a distribuição do conteúdo com o uso das Redes de Distribuição de Conteúdo (CDN). Isto aumenta os custos de uma solução devido à necessidade de distribuir estrategicamente servidores ao redor do mundo e melhorar os canais de transmissão, pois, em essencia, trata-se de requisições

HTTP. No caso do GMTP, a camada de aplicação o utiliza de forma similar ao TCP, mas o transporte dos dados entre os servidores da CDN ocorrer com o suporte de uma rede de favores formada pelos roteadores de rede, onde o conteúdo pode ser entregue pelo servidor diretamente ao roteador do nó cliente interessado em receber o fluxo de dados ou indiretamente, entregue por um roteador parceiro ao roteador do do cliente interessado pelo fluxo. O roteador parceiro por receber o fluxo do servidor ou por outro roteador parceiro, e assim sucessivamente. Sendo assim, o MPEG-DASH pode fazer uso do GMTP, em vez do TCP;

- Os servidores web executam o protocolo HTTP na camada de aplicação e, na camada de transporte, dependem do protocolo TCP. Sendo assim, realiza-se a transmissão das partes da mídias através de um protocolo orientado à conexão, com garantia de entrega e ordenação, podendo gerar atrasos na entrega dos segmentos devido às retransmissões quando há perda de segmentos devido ao congestionamento da rede. Além disso, pode ocorre o problema da tragédia dos bens comuns, discutido no Capítulo **??**;

- As soluções baseadas em HTTP são baseadas no método *pull*. Isto significa que os nós clientes que desejam reproduzir um conteúdo ao vivo devem solicitar periodicamente os próximos segmentos que devem ser reproduzidos. Em se tratando de transmissões de mídias ao vivo em larga escala, pode-se aumentar sobremaneira o número de requisições dos arquivos de lista, aumentando-se o tráfego na rede com dados considerados de controle. Apesar dos inúmeros esforços para melhorar os mecanismos de escalonamento baseados em *pull*, o uso do método *pull* sempre acarretará em um *trade-off* entre a sobrecarga de controle e o atraso. Ou seja, os protocolos sempre terão uma alto sobrecarga na troca de pacotes de controle ou um alto atraso na recepção dos dados [11–13]. No caso do GMTP, utiliza o método híbrido *push/pull*, onde o método *push* é utilizado como padrão e o *pull* em casos especiais, como quando um nó está prestes a reproduzir um determinado *chunk* e este ainda não está disponível. Além disso, apesar do método *pull* não necessitar conexão e o HTTP é interesse para isso, no GMTP, reduz-se o número de conexões ao permitir que os roteadores de rede interceptem os pedidos de múltiplas conexões para um mesmo conteúdo transmitidas em direção ao nó servidor;

- O TCP implementa um mecanismo tradicional de controle de congestionamento, adaptando sua taxa de transmissão de acordo com a perda dos segmentos. Por outro lado, os clientes web precisam monitorar sua capacidade de recepção e solicitar ao servidor web os segmentos correspondente. Isto significa que o conteúdo é adaptado de acordo com o estado da rede percebida pelo cliente TCP [14]. Em canais assimétricos de transmissão, não é simples disponibilizar soluções para medir a capacidade de transmissão quando se utiliza o método *pull*. Isto porque o nó transmissor perceberá uma taxa de transmissão diferente da taxa de transmissão do receptor, que não envia dados com conteúdo. Além disso, de acordo com o estado da arte, existem soluções mais sofisticadas para fazer melhor uso dos canais de transmissão, expondo às aplicações informações mais precisas sobre o estado da rede com relação a sua capacidade de transmissão. Isto ocorre com o uso dos algoritmos de controle de congestionamento assistidos pela rede. Como consequência, tais soluções reduzem o congestionamento nos pontos de interconexão, melhorando a qualidade dos fluxos de dados multimídia reproduzidos aos usuários finais. No caso do GMTP, utiliza-se uma versão adaptada do RCP [15];

- Soluções baseadas em HTTP, devem considerar aspectos relacionados à segurança do HTTP, descritos na Seção 15 da referência [16]. Os principais aspectos a serem considerados são a disponibilização de informações pessoais e privadas, comprometimento dos segmentos devido aos ataques de interceptação implementados com servidores de cache e os arquivos de lista contém URIs, utilizadas pelos clientes para requisitarem os segmentos a servidores arbitrários e que podem estar comprometimento. No GMTP, utiliza-se um mecanismo de segurança imbutido no próprio protocolo que permite os roteadores validarem o conteúdo através de assinatura digital e o conteúdo pode ser criptografado utilizando métodos tradicionais, como criptografias assimétricas, com suporte a diferentes método, como RSA e Curvas Elípticas.

1.1.3 PDTP – *Peer Distributed Transfer Protocol*

O protocolo *Peer Distributed Transfer Protocol* (PDTP) [17] surgiu em 2002 com a promessa de prover um método para transferência de arquivos e mídia em tempo real similar ao

BitTorrent. O uso do protocolo foi perdendo força e no final de 2007 foi descontinuado. Sua implementação de referência era conhecida pelo nome de DistribuStream [1].

O PDTP previa o uso de servidores para gerenciamento automático de diretórios de conteúdo, fazendo-o similar a protocolos como o HTTP e o FTP. Além disso, na proposta do PDTP previa suporte a meta descrição e validação de integridade de conteúdo através do uso de assinatura digital. A *Internet Assigned Numbers Authority* (IANA) alocou a porta 6086 para o uso do protocolo em aplicações multimídia. Suporta um mecanismo de *tracker* similar ao PPSP/Swift e utiliza o protocolo UDP para transmissão de dados.

O PDTP especifica um conjunto de nós chamados de *hubs*, que tem como responsabilidade prover o mapa da rede, listagem de diretórios e serviço de arquivos. O serviço de arquivo é similar ao esquema de *seed* do BitTorrent, com a diferença do uso de outro conjunto de nós chamados de *Piece Proxies* (PP). Os PPs fazem download e cache de pedaços de arquivos armazenados nos nós hubs e então servem estes pedaços na rede sob demanda, reduzindo o consumo de banda dos *hubs*. Segundo os autores, o BitTorrent resolve esse problema com o uso de multiplos *seeds*, porém se não existir nenhum *seed* disponível para um *torrent* o conteúdo fica inacessível. Na Figura 1.3 ilustra-se a organização geral dos nós PDTP.

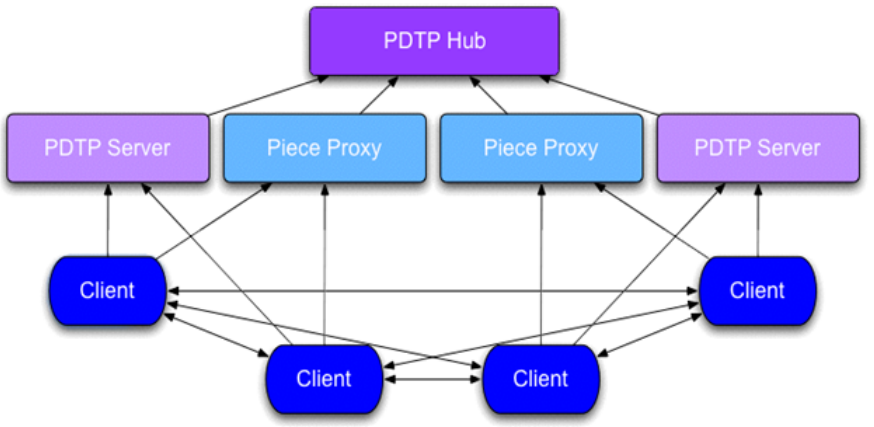


Figura 1.3: Organização dos Nós PDTP. Figura extraída de **??**.

[1] DistribuStream: `http://freecode.com/projects/distribustream`

Considerações sobre o trabalho

A proposta do protocolo PDTP tem um ponto positivo porque organiza os nós interessados por um mesmo conteúdo de forma hierárquica e o conjunto de comandos disponíveis do protocolo é similar a protocolos tradicionais, como o HTTP e o FTP.

Embora os autores mencionem a possibilidade de utilizar o PDTP em transmissões de mídia em tempo real, nenhum referência disponível menciona detalhes sobre tal capacidade. O uso do protocolo UDP caracteriza um protocolo com os problemas problemas já discutidos ao longo deste trabalho e presente nos outros trabalhos apresentados neste capítulo.

1.1.4 CPM – *Cooperative Peer Assists and Multicast*

No *Cooperative Peer Assists and Multicast* (CPM) [18] propõe-se uma abordagem unificada para prover suporte eficiente de transmissão de vídeos sob de demanda para ser utilizada por provedores de serviços. O CPM é um protocolo de aplicação que suporta transmissão em modo multicast, cache de dados nos nós clientes, compartilhamento de dados entre os cliente, onde o servidor utiliza modo de transmissão unicast.

Na Figura 1.4 ilustra-se a visão geral do funcionamento do CPM através de um diagrama de sequência. Primeiramente o cliente conecta o servidor para saber sobre a existência de algum grupo multicast, e então passa a receber o conteúdo em modo multicast. Caso não exista um grupo multicast para o conteúdo de interesse, o cliente solicita, através de um servidor de diretórios a lista de nós que detém o conteúdo de interesse e então inicia a transferência. Caso não exista nenhum nó com o conteúdo requisitado, o cliente requisita o conteúdo diretamente para o servidor.

Na arquitetura do protocolo CPM existem três componentes principais: (1) o modelo de dados do vídeo; (2) um protocolo para descoberta e transferência de conteúdo e (3) um escalonador inteligente no lado do servidor.

O modelo de dados divide o vídeo em pedaços (*chunks*) de tamanhos fixos. Cada pedaço é identificado por um GUID (*Globally Unique Identifier*), onde um segmento consiste em uma sequência de pedaços e uma sequência de segmentos constitui um vídeo.

O protocolo de transferência assume que o vídeo deve está completamente armazenado no servidor para permitir que os nós façam cache dos pedaços e redistribuí-los *a posteriori*.

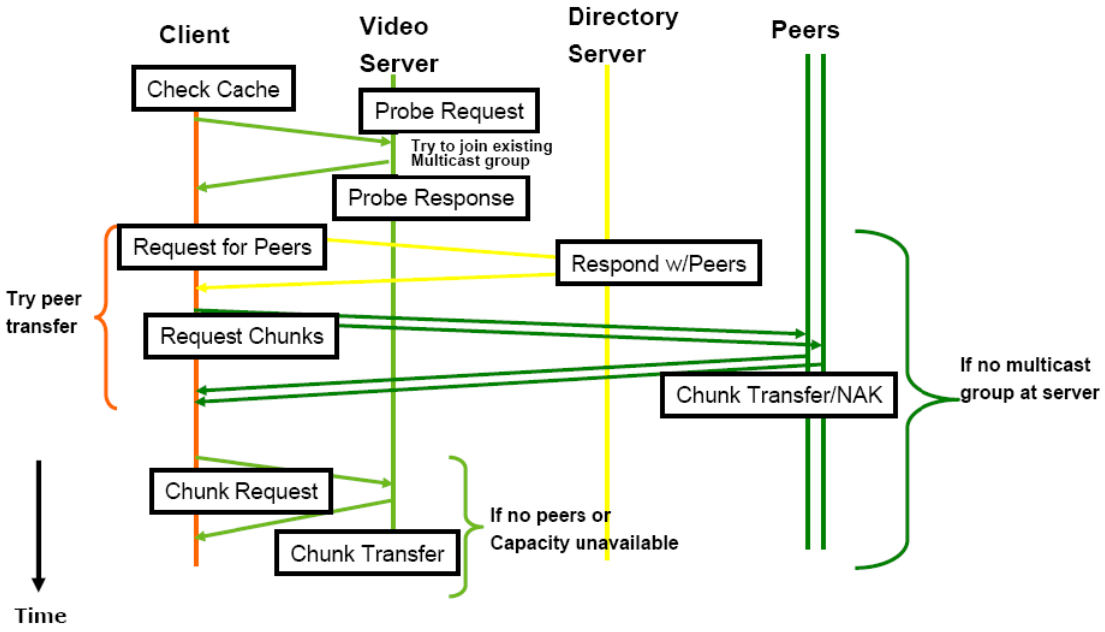


Figura 1.4: Diagrama de sequência do CPM (*Cooperative Peer Assists and Multicast*). Figura extraída de **??**.

Quando um cliente envia um pedido de reprodução de vídeo, o servidor mapeia o conteúdo do vídeo requisitado, que é então formatado em sequências de pedaços e transmitidos para o cliente. Este procedimento é executado em paralelo com outros nós da rede. O modo de transmissão multicast é ativado pelo servidor e só ocorre quando múltiplos clientes tem interesse pelo mesmo conteúdo.

Considerações sobre o trabalho

A capacidade para transmitir o conteúdo em modo híbrido é um aspecto positivo para o CPM. A seguir enumeram-se os pontos fracos identicados.

1. Para utilizar o modo de transmissão multicast, o cliente tem que ter rota multicast diretamente para o servidor, pois apenas este pode iniciar o processo de transmissão utilizando este modo. No GMTP esse mecanismo é segmentado e qualquer nó pode transmitir em modo multicast.

2. O mecanismo de transmissão de conteúdo quebra os segmentos em pedaços, o que torna o gerenciamento mais complexo devido ao espalhamento dos pedaços entre os nós participantes da transmissão. Isto pode gerar atrasos na reprodução do conteúdo

no cliente devido a necessidade de localizar cada pedaço individualmente, embora o uso dessa abordagem pode aumenta a velocidade de download. No GMTP a existe a idéia de quebrar os segmentos em pedaços menores. Tal abordagem é deixada a cargo da aplicação.

3. O uso de servidor de diretórios para consultar a lista de nós que mantém o conteúdo multimídia desejado não faz muito sentido para sistemas de transmissão de mídia ao vivo. No GMTP não utiliza-se este tipo de solução.

4. Como o CPM foi desenvolvido com foco em transmissão de vídeo sob demanda, os nós só podem começar a repassar o conteúdo se previamente o este já tenha reproduzido o vídeo no passado. No caso de sistemas transmissão de vídeo em tempo real esta abordagem é completamente inútil. No GMTP qualquer nó é capaz de realizar o repasse de conteúdo, inclusive em modo multicast.

1.1.5 HySAC – *Hybrid Delivery System with Adaptive Content Management for IPTV Networks*

No *Hybrid Delivery System with Adaptive Content Management for IPTV Networks* (HySAC) [19], propõe-se uma nova arquitetura e um sistema adaptativo e híbrido que utiliza um esquema chamado de pre-população para distribuição de vídeos sob demanda em redes IPTV. Os autores do HySAC criticam o protocolo CPM ao afirmarem que tal abordagem não utiliza os recursos de rede de forma otimizada, uma vez que o conteúdo de mídia não é armazenado de modo pré-planejado de acordo com a demanda dos nós clientes, o que eleva o consumo de recursos de rede e provê uma baixa qualidade na transmissão do conteúdo multimídia ao usuário final.

Diferente do CPM, o HySAC provê a arquitetura ilustrada na Figura 1.5. Para evitar que a grande quantidade de usuários concorrents sobrecarregue os servidores de mídia, os autores do HySAC propõem um sistema adaptativo de transmissão de mídia que otimiza o processo de entrega de dados baseado na popularidade do conteúdo e nos recursos de rede disponíveis. O conteúdo é categorizado em diferentes classes e o modo de entrega do conteúdo é baseado na popularidade do mesmo. A popularidade de um conteúdo é computada baseando-se no interesse dos usuários e no número de requisições que chegam ao servidor. Os servidores

HySAC categorizam os conteúdos e os servidores de indexação e descoberta utilizam Tabelas Dinâmicas de Hash (DHT) para encontrar os servidores que armazenam o conteúdo. Quando a localização de um conteúdo muda, os servidores de indexação são atualizados e quando um novo conteúdo é adicionado, os servidores cuidam da replicação do mesmo de acordo com a sua popularidade.

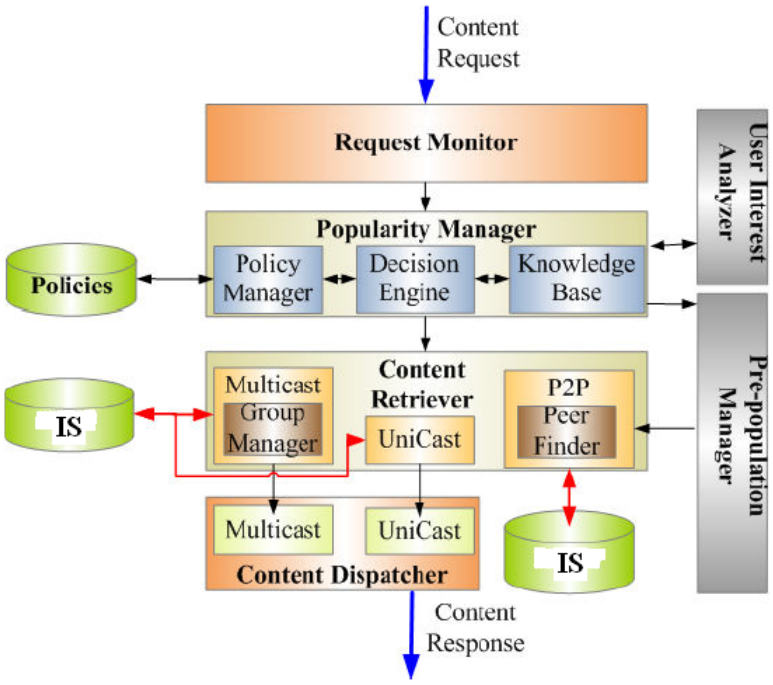


Figura 1.5: Arquitetura do HySAC (*Hybrid Delivery System with Adaptive Content Management for IPTV Networks*). Figura extraída de **??**.

O HySAC utiliza três modo de transmissão: unicast, multicast e P2P. Inicialmente o HySAC entrega o conteúdo baseado em informações estáticas sobre a popularidade do conteúdo. O HySAC gerencia um *rank* de pupularidade do vídeo e dependendo de um determinado limiar de popularidade o vídeo é selecionado para ser transmitido em modo multicast. Quanto mais alto for a popularidade do vídeo, maior é a chance dele ser transmitido em modo multicast. Se a popularidade do vídeo for intermediária, o vídeo será transmitido em modo P2P e se for baixa o vídeo será transmitido do servidor diretamente para o cliente em modo unicast.

Considerações sobre o trabalho

A capacidade de transmitir o conteúdo em modo unicast, multicast e P2P é um aspecto positivo para o HySAC. A seguir enumeram-se os pontos fracos identicados.

1. A decisão do modo de transmissão é baseado na popularidade do vídeo. Considerando-se transmissões de mídia em tempo real, a forma o como HySAC implementa o mecanismo de classificar o conteúdo requer um tempo de convergência, o que pode consumir recurso de rede desnecessariamente. No GMTP utiliza-se multicast sempre que possível, possibilitando que outros clientes recebam o conteúdo até mesmo sem precisar contactar o servidor, utilizando-se o modo de conexão rápida.

2. Da mesma forma que outras soluções, o HySAC utiliza o protocolo UDP para transmissão de dados da aplicação multimídia. Não foi encontrado nenhuma menção a respeito do uso de algoritmos para controle de congestionamento, ao contrário do GMTP.

3. Trata-se de um sistema de transmissão e não de um protocolo de rede propriamente dito. Isto significa que a proposta do HySAC servirá apenas para clientes que seguem sua especificação.

1.1.6 PPETP – *Peer-to-Peer Epi-Transport Protocol*

O *Peer-to-Peer Epi-Transport Protocol* (PPETP) é um protocolo distribuído que utiliza uma abordagem P2P para transmissão de mídias em tempo real, sendo proposto para operar em redes com nós heterogêneos [20].

O PPETP é um protocolo que constrói uma rede de sobreposição com suporte a transmissão em modo multicast. Em tal protocolo, propõe-se uma solução de fácil integração às aplicações multimídia existentes por meio de uma biblioteca de programação similar, porém não integrada, à *Socket* BSD. O PPETP tem como principal aplicação os sistemas de transmissão de vídeo executados por usuários residenciais, geralmente conectados através de uma tecnologia xDSL, fornecendo uma visão de um protocolo de transporte multicast ao desenvolvedor da aplicação, embora o PPETP é executado na camada de aplicação.

O PPETP utiliza uma abordagem de transmissão do tipo *push* onde os nós iniciam e finalizam as conexões utilizando pacotes de controle e trocam dados em modo unicast. Ao

desejar receber um fluxo de dados, um nó B envia um pedido de conexão ao servidor PPETP (Figura 1.6). Em seguida, o servidor responde com uma informação que determina qual nó o cliente B deve solicitar os dados da transmissão, além de solicitar que o nó B obtenha um arquivo de configuração dos parâmetros de conexão em um servidor de configuração chamado de *starting point*. O servidor de configuração pode ser um nó diferente do servidor gerador do fluxo de dados multimídia. O nó B então requisita os dados ao nó determinado pelo servidor. No PPETP utiliza-se o protocolo UDP para transmissão de dados por padrão, embora permite-se o uso de outros protocolos, como o TCP e possivelmente o DCCP.

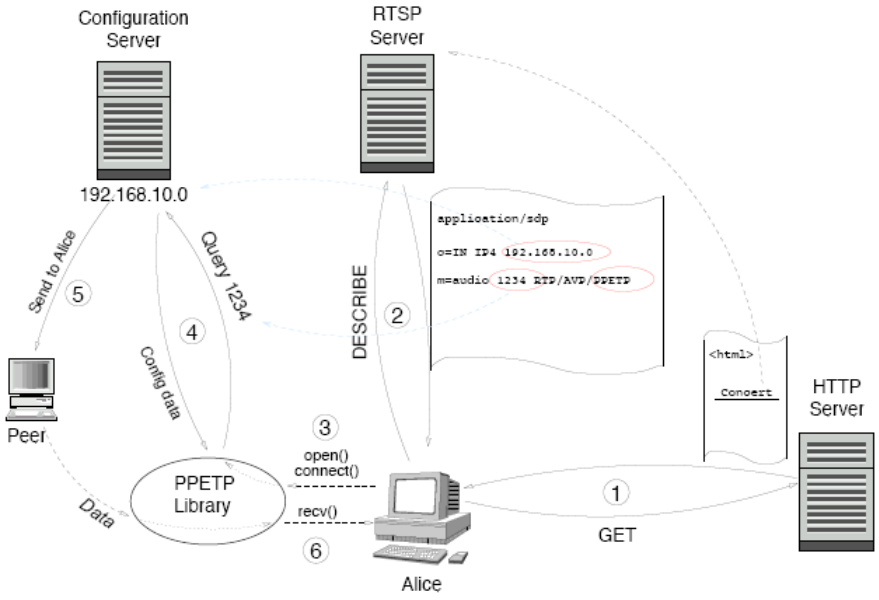


Figura 1.6: Arquitetura e funcionamento do protocolo PPETP.

Uma característica fundamental do PPETP é conhecida pelo nome de procedimento de redução (*reduction procedure*). Considerando-se o princípio de que um fluxo de dados é uma sequência de pacotes, o procedimento de redução do PPETP evita que todos os nós do sistema tenham sempre que repassar todos os pacotes desse fluxo de dados, permitindo-se que mesmos os nós conectados na Internet através de canais com largura de banda limitada consigam contribuir com o sistema.

No PPETP utiliza-se uma função de redução do tamanho do pacote que é parametrizável e diversas funções podem ser utilizadas através de uma arquitetura de componentes. Atualmente existem duas funções, a de *Vandermonde* e a básica (sem redução). A função de *Vandermonde* reduzido cada pacote por um fator R e então o pacote é repassado para outros

nós. Cada nó recebe um conjunto de pacotes reduzidos, reconstrói o conteúdo do pacote, entrega-o para a aplicação e repete o procedimento de redução, repassando-o para outros nós interessados pelo conteúdo.

Um aspecto importante do PPETP é sua capacidade de tolerar perdas de pacotes. Como o mecanismo de redução de pacotes permite a reprodução do conteúdo sem que todos os pacotes reduzidos alcancem o receptor, isto torna o protocolo resiliente a perdas de dados. Os autores prometem que em uma rede com N nós e um fator de redução R, a reconstrução de um pacote pode acontecer mesmo se $N - R$ nós se desconectarem.

Considerações sobre o trabalho

O aspecto positivo do PPETP é sua capacidade de funcionar com nós heterogêneos no ponto de vista dos recursos de rede disponíveis por cada um deles. O esquema de *procedimento de redução* dos tamanhos dos pacotes parecer ser bastante promissor, porém ao que pôde-se constatar é complexo de ser implementado e só funciona com a participação de muitos nós.

Os pontos fracos do PPETP são vários e enumerados a seguir.

1. O PPETP é um protocolo na camada de aplicação e considerado pelos autores de ser um protocolo de pseudo-transporte, pois abstrai da aplicação diversas funcionalidades dos sistemas de transmissão de mídia em tempo real que utiliza a abordagem P2P. Neste sentido, a disponibilização e a efetiva utilização do PPETP por parte das aplicações pode ser dificultada por ser um protocolo de aplicação e não genuinamente de transporte. Isto significa que o PPETP não tem uma separação explícita de responsabilidade no ponto de vista de transporte de dados, misturando responsabilidades da camada de aplicação e da camada de transporte. No GMTP, essa separação é explícita por se tratar de um protocolo disponibilizado na camada de transporte sem qualquer influência da aplicação, delegando para a mesma apenas responsabilidades de sinalização e descrição do conteúdo a ser transportado.

2. O fluxo de dados de controle é centralizado no servidor. Isto significa que para que um nó A comece a receber um fluxo de dados de um outro nó, primeiro o nó A precisa solicitar ao servidor o acesso ao conteúdo para em seguida efetivamente começar a

recebê-lo. No GMTP isto não acontece, pois qualquer nó pode funcionar como servidor, o que ocorre de forma transparente para a aplicação.

3. O PPETP atualmente utiliza o protocolo UDP que, como já discutido, possui diversas desvantagens para a aplicação e para a rede, principalmente em situação de congestionamento na rede. O GMTP é um protocolo de transporte e portanto não necessita de nenhum outro protocolo da sua própria camada. Além disso, o GMTP possui um arcabouço para adicionar novos algoritmos de controle de congestionamento, tanto para as transmissões em modo unicast quanto para as transmissões em modo multicast.

4. O PPETP não suporte transmissão de dados em modo multicast. No PPETP os dados são transmitidos entre os nós em modo unicast, apesar dos autores mencionarem que o protocolo funciona em modo multicast, pelo menos no ponto de vista da aplicação. O termo multicast empregado nesse contexto é apenas para dar a idéia que poucos fluxos são transmitidos a partir do nó transmissão, mas que todos os nós receptores os recebem através da rede sobreposição criada pelo PPETP. No GMTP utiliza-se um mecanismo híbrido de transmissão: sempre que possível usa-se o modo multicast, caso contrário usa-se o modo unicast.

5. No PPETP alguns mecanismos bastante utilizados em sistemas de transmissão de mídias em tempo real, como o de descoberta de nós, deve ser implementado na camada de aplicação. Apesar dessa abordagem do PPETP funcionar por ser flexível para a camada de aplicação, a mesma limita o uso desses mecanismos à própria aplicação, impedindo que outras aplicações façam uso dos mesmos. No GMTP procurou-se adicionar tal funcionalidade dentro do próprio protocolo, permitindo-se o reúso desses mecanismos em diferentes aplicações. Desta forma, é possível que um algoritmo para descoberta de nós seja implementado no GMTP, em forma de componente, e qualquer outra aplicação reutilizar tal mecanismo. Com isto, o GMTP permite a interioperabilidade entre diferentes aplicações a nível de camada de transporte.

6. Para que o PPETP funcione efetivamente nas aplicações serão necessárias diversas alterações em protocolos da camada de aplicação, como no RTSP e no SDP (*Session Description Protocol*). Todas as modificações necessárias estão listadas no documento

disponível na referência [20].

1.1.7 PPSP/Swift – *P2P Streaming Protocol / The Generic Multiparty Transport Protocol*

O *Peer-to-Peer Streaming Protocol* (PPSP) é um protocolo para sinalização e controle para sistemas de transmissão de fluxos de dados em tempo real. Dentro do PPSP existe o Swift, um protocolo cujo objetivo é disseminar o conteúdo para um conjunto de nós interessados por um mesmo conteúdo.

O PPSP define *peers* e *trackers* como dois tipos de nós para um sistema de transmissão de mídia baseado em P2P. Os *peers* são nós que enviam e recebem conteúdos multimídia e os *trackers* são nós conhecidos com conexão estável que mantêm meta informações sobre os conteúdos transmitidos e uma lista dinâmica de *peers*. Os *trackers* podem ser organizados de forma centralizada ou distribuída. No PPSP propõe-se dois protocolos base. O protocolo dos *trackers*, que tratam as trocas de meta informações entre os *trackers* e os *peers*, tais como a lista dos peers e informações sobre os conteúdos. E o protocolo dos *peers*, que controla os anúncios e informações sobre a disponibilidade de dados da mídia entre os *peers*.

O funcionamento básico do PPSP ocorre da seguinte forma 1.7. Um nó transmissor *Peer-P* notifica ao tracker a transmissão realizada por ele (passo 1). O *tracker* então transmite uma mensagem para os *Peer-M* e *Peer-D* interessados em receber o conteúdo multimídia para se juntarem ao grupo (passos 2 e 4). O *Peer-P* transmite o conteúdo para o *Peer-M*, que repassa para o *Peer-D*. Em seguida, outros *peers* se registram como clientes interessado o *tracker*

O processo descrito anteriormente é governado pelo protocolo PPSP. Porém, como o PPSP não é um protocolo de transporte, seus idealizadores criaram o *The Generic Multiparty Transport Protocol* (Swift). A responsabilidade do Swift no processo descrito é cuidar do transporte de dados entre os *peers Peer-M*, *Peer-D* e quaisquer outros participantes da transmissão. O Swift especifica o conteúdo de um stream como pedaços chamados de *chunks*. O Swift transmite os dados entre os *peers* utilizando o protocolo de transporte UDP com suporte de controle de congestionamento chamado de LEDBAT [21, 22], o mesmo adotado no BitTorrent.

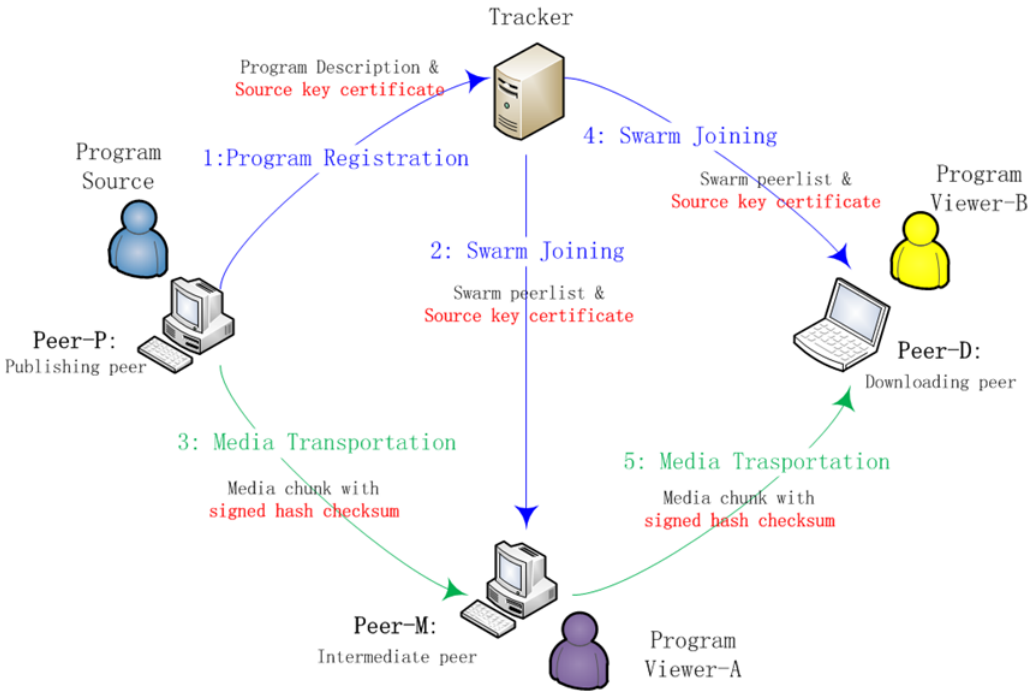


Figura 1.7: Arquitetura e funcionamento do protocolo PPSP/Swift.

Considerações sobre o trabalho

Os pontos positivos do PPSP/Swift são dois. O primeiro é a separação do mecanismo de sinalização e descrição da mídia da parte de transporte. O segundo ponto é o mecanismo do Swift de distribuição de conteúdo, baseado em enxames.

Os pontos fracos do PPSP/Swift são enumerados a seguir.

1. Ausência de suporte para extensão para recursos da aplicação, como por exemplo, descoberta, controle de congestionamento e tolerância à falhas. O GMTP é extensível nesse aspecto.

2. Não suporta compartilhamento de conexão com suporte a transmissão em modo multicast.

3. Menciona o uso futuro do algoritmo para controle de congestionamento TFRC, porém não possui suporte a controle de congestionamento em grupo. No GMTP isso é feito utilizando o algoritmo MCC, apresentado nas Seções **??** e **??**.

4. A transmissão de conteúdo com o transporte de *chunks* é interessante em aplicações para compartilhamento de arquivos, onde o tempo de resposta não é requisito funda-

mental para a qualidade de serviço no ponto de vista do usuário que o utiliza. O uso dessa abordagem em aplicações de transmissão de mídia em tempo real não é uma estratégia interessante devido a complexidade de indexar e remontar os pacotes de dados de acordo com cada *chunk*. Esses procedimentos podem onerar o tempo em que um pacote de dados é entregue para a camada de aplicação, gerando-se um atraso no fluxo contínuo de dados para a camada de aplicação.

5. Uso do protocolo UDP, apesar de fornecer mecanismo para controle de congestionamento. Esta prática quebra a idéia da organização dos protocolos em camadas funcionais, onde uma camada fornece serviços para a camada superior e, obviamente, usufrui de serviços da camada inferior. Mecanismos para controle de congestionamento devem ser implementados na camada de transporte e não na camada de aplicação. Isso limita o uso dos recursos implementados no Swift apenas para aplicações que utilizam sua implementação, a *libswift*, disponível em forma de biblioteca de software. O GMTP é um protocolo de transporte e portanto independente de qualquer outro, além de implementar e suportar à adição de seus próprios algoritmos para controle de congestionamento.

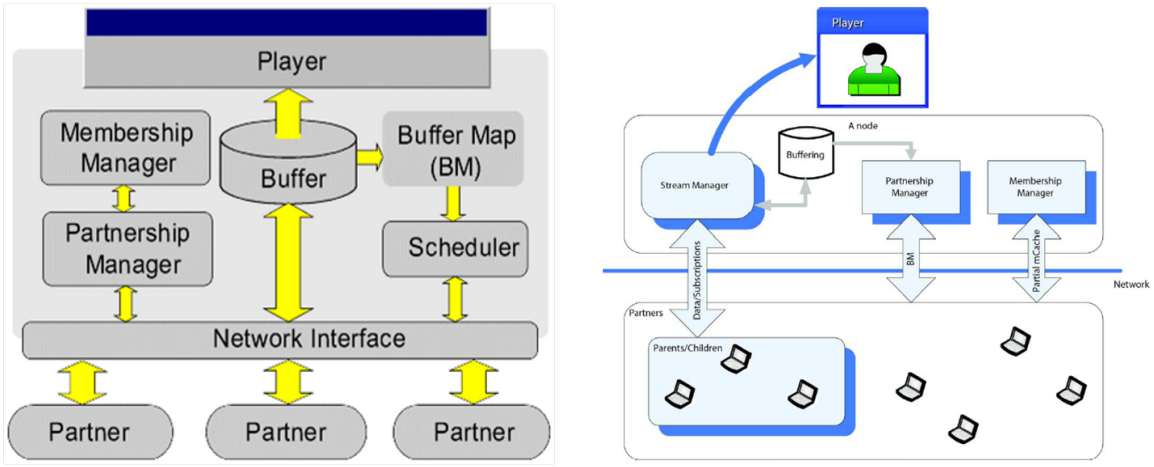
1.1.8 DONet/CoolStreaming

O CoolStreaming é um sistema para distribuição de mídias ao vivo baseado em uma arquitetura P2P [13, 23, 24]. A primeira versão do Coolstreaming foi lançada em 2004 e aprimorada ao longo dos anos, tornando-se o sistema mais conhecido e robusto para transmissão de mídias ao vivo em larga escala. Atualmente (2014), o sistema CoolStreaming é uma das principais referências no contexto de distribuição de mídias ao vivo, tanto no contexto acadêmico quanto comercial. O sistema CoolStreaming está disponível para uso no mercado, servindo a milhares de nós ao redor do mundo e, por ter sido descrito e estudado exaustivamente, também está disponível na rede PlanetLab e nos principais simuladores de redes P2P, com o OMNet++/Oversim[2] e o OMNet++/OSSim.

A sigla DONet significa *Data-drive Overlay Network* e CoolStreaming significa *Coope-*

[2]OMNet++/Oversim: de fato, o CoolStreaming está disponível através do projeto Denacast, desenvolvido com base no OMNet++/Oversim. Na Seção 1.1.9, detalha-se o Denacast.

rative Overlay Streaming, sendo a implementação de referência da DONet. Desde a primeira versão, seus autores propuseram que partes da mídia fossem distribuídas sobre uma rede de sobreposição onde os nós sempre repassavam as partes de uma mídia para outros nós interessados pelo mesmo conteúdo, sem nenhuma regra pre-definida, como nó pai ou filho; nó interno ou externo; capacidade de upload e download, etc. Com essa visão centrada no dado, os autores decidiram que a disponibilidade do dado era o critério para guiar as estratégias de escalonamento dos fluxos de dados transmitidos através da rede de sobreposição, adaptando-se melhor às dinâmicas dos nós em uma rede P2P. Cada nó periodicamente troca informações sobre a sua disponibilidade de *chunk* transmitindo seu mapa de buffer com um conjunto de nós parceiros, obtendo os *chunks* ausentes a partir dos nós que os anunciam como disponíveis.



(a) CoolStreaming 1.0. Extraído de [24].

(b) CoolStreaming 2.0. Extraído de [23].

Figura 1.8: Arquitetura genérica de blocos funcionais do CoolStreaming 1.0 e do 2.0.

Conforme ilustra-se na Figura 1.8, o sistema CoolStreaming teve duas versões de produção. Na Figura1.8a, ilustra-se o diagrama genérico do sistema para um nó na rede DONet, quando se utilizava apenas o método *pull* para obtenção de conteúdo, estratégia bastante similar ao BitTorrent, com seleção aleatória de nós parceiros. Já na Figura1.8a, ilustra-se o diagrama genérico da versão aprimorada do CoolStreaming, onde diversos aspectos foram aprimorados e o mecanismo de obtenção das partes da mídia foi remodelado para uma abordagem híbrida *pull/push* de obtenção dos blocos de vídeo (*chunks*). Na versão mais atual do CoolStreaming, organiza-se um fluxo de dados multimídia em sub-fluxos que carregam

blocos de dados contendo partes da mídia para ser reproduzida na aplicação em execução no nó receptor. Na primeira versão, os blocos de dados eram obtidos sempre utilizando o método *pull*, o que acarretava o aumento no atraso para obter o conteúdo em 1 RTT. Na segunda versão, a estratégia híbrida *pull/push* mudou para a seguinte forma. Quando um nó C_1, interessados em obter um determinado bloco de vídeo, recebe de um nó parceiro C_2 o mapa de buffer contendo a disponibilidade dos blocos do vídeo, C_1 envia uma requisiçao à C_2 solicitando os blocos de vídeo desejados (*pull*). Em seguida, o nó C_2 transmite os blocos de vídeo para C_1 e, a partir desse momento, todo novo bloco de vídeo que C_2 receber, repassa ao nó C_1 (*push*).

Os outros componentes do sistema CoolStreaming são:

- *Membership manager*, que permitem os nós do sistema manterem uma visão parcial da rede de sobreposição, sendo adicionado na versão 2 um componente chamado *mCache*, que registra uma lista parcial dos atuais nós ativos na rede;

- *Partnership manager*, que estabelece e mantém as parcerias com os outros nós e também é responsável por trocar o mapa de *buffer*. O algoritmos padrão para seleção de nós é baseado em uma escolha aleatória entre os nós disponíveis na lista *mCache*;

- *Stream Manager*, que efetivamente transmite os fluxos de dados representados nos blocos de vídeo. Este componente é responsável por escalonar o momento exato de enviar cada *chunk* correspondente a um vídeo sendo compartilhado com outros nós do sistema;

- *Buffer Map*, que representa o estado atual do buffer para um determinado vídeo. Como discutido no Capítulo **??**, um nó pode sinalizar os blocos de vídeo que estão disponíveis ou os blocos de vídeo que estão ausentes e portanto necessários.

Além da mudança para uma abordagem híbrida *pull/push*, os autores do CoolStreaming 2.0 propuseram o conceito de sub-fluxo. O nó transmissor divide o vídeo em blocos de tamanhos iguais, e a cada bloco é assinalado um número de sequência para permitir a reprodução do conteúdo em ordem. Dessa forma, cada nó pode obter diferentes blocos de vídeo a partir de diferentes nós parceiros. Com isto, espera-se reduzir o impacto causado na rede devido à saída ou falha de comunicação em um nó transmissor.

Considerações sobre o trabalho

A robustez do CoolStreaming em tratar a dinâmica da rede, bem como a possibilidade de realizar as ações apresentadas anteriormente sem requerer qualquer suporte da rede física, são aspectos positivos do CoolStreaming. Suas características possibilitaram a aplicação do CoolStreaming em cenários reais.

Por outro lado, os pontos negativos do sistema CoolStreaming são apresentados a seguir.

1. O conceito de sub-fluxo adiciona complexidade à solução sem necessariamente resultado em ótimos resultados. Em [23], os autores do CoolStreaming discutem que aumentar a quantidade de número de sub-fluxos não melhora proporcionalmente algumas métricas, como o índice de continuidade e utilização da capacidade de upload dos nós transmissores (em média). Os autores executaram simulações com 40 mil nós e 24 servidores auxiliares (que funcionaram apenas como nós transmissores) e observaram que a partir de 8 sub-fluxos, as duas métricas citadas anteriormente não melhora e, em alguns casos, até pioram (quando se utiliza nós com capacidade heterogêneas de transmissão). No GMTP, utiliza-se sempre o método *push* após um nó estabelecer uma conexão e os roteadores no caminho entre o nó servidor e o nó cliente podem interceptar os pedidos de conexão transmitidos por outros nós clientes. Essa estratégia reduzir a quantidade de requisições ao servidor e o tempo para iniciar a reprodução de um vídeo ao usuário final (apenas o primeiro usuário perceberá um atraso maior do que os demais). Com isso, se no caminho entre o nó cliente e o nó servidor não ocorrer nenhuma interceptação, a requisição alcançará o nó servidor, e a troca de dados ocorre. Porém, os próximos nós clientes que transmitirem seu pedido de conexão através de pelo menos parte do mesmo caminho já utilizado anteriormente, um nó servidor instruirá um roteador nesse caminho a replicar o fluxo para o novo nó cliente, em vez de responder com a aceitação do pedido de conexão.

2. No sistema CoolStreaming, a rede de sobreposição é centrada no dado. Os nós realizam parcerias considerando quais parceiros possuem as partes da mídia de interesse e a troca de parcerias ocorre ao longo da transmissão. Isso gera instabilidades na transmissão, impactando diretamente em métricas como o índice de continuidade. Além disso, as parcerias são efetivadas independente da posição lógica do nó na rede, levando-se

em consideração apenas o nó que detém um determinado contéudo de interesse e sua capacidade de upload, o que pode gerar sobrecarga na troca de informações de controle. No GMTP, a rede de sobreposição é centrada na conexão e a constituição de tal rede ocorre de forma transparente à aplicação. A formação da rede acontece no processo de pedido de conexão, onde os nós intermediários (roteadores), localizados entre o nó interessado pela mídia (cliente) o nó transmissor (servidor) são autorizados a interceptar o pedido de conexão e responder ao nó cliente como se fosse o servidor original. Somente depois dessa fase, os nós roteadores GMTP iniciam um processo de expansão de parcerias, onde podem realizar parcerias com outros nós que não estejam, necessariamente, conectados em um mesmo servidor da CDN.

3. Os nós da rede de sobreposição do sistema CoolStreaming são os sistemas finais, que executam aplicações de rede. No GMTP, constitui-se uam rede de sobreposição entre os roteadores da rede e não entre os sistemas finais. Dessa forma, a rede se torna estável com relação a dinâmica de entradas e saídas das redes, sendo possível continuar utilizando os recursos de um roteador (em geral, o da borda de uma rede), mesmo quando seus nós clientes desistem de continuar obtendo a mídia de interesse, por exemplo, se o usuário fechar o aplicativo. Sendo assim, os nós roteadores podem continuar transmitindo o fluxo de vídeo para outros nós, pelo menos momentaneamente, enquanto seus parceiros realizam outras parcerias.

4. Um nó recém integrado à rede DONet, pode levar muito tempo (em alguns casos 20 segundos) para obter os primeiros blocos de vídeo e assim iniciar a reprodução do conteúdo ao usuário final. Isto porque, ao se juntar à rede, um nó solicita o mapa de *buffer* a um conjunto de nós parceiros informados por um servidor de *boostrap*. Porém, o desafio é definir a partir de qual ponto do *buffer* um nó deve começar a solicitar os blocos de vídeo. Por exemplo, se o novo nó requisitar um bloco de vídeo muito antigo, este podem não mais estar disponíveis, um vez que após sua reprodução, tais blocos são removidos. Por outro lado, se o nó requisitar um bloco de vídeo muito recente, pode ser que nenhum de seus nós parceiros tenha disponível. No GMTP, situações como essas não ocorrem porque se utiliza, por padrão o método *push* e o conteúdo poderá já está disponível no próprio roteador de borda do nó solicitante.

Outras decisão nesse sentido foram tomadas no GMTP, as quais serão elucidadas no Capítulo **??**.

5. A seleção de nós sistema CoolStreaming ocorre com base na escolha aleatória de um sub-conjunto de nós disponíveis em uma lista de parceiros. Após realizar parcerias com um sub-conjunto de nós, um nó começa a receber os blocos de vídeo, ao mesmo tempo que monitora o status de recepção dos sub-fluxos, transmitidos por diferentes nós parceiros. Quando um nó percebe que a taxa de recepção não está satisfatória, inicia-se um processo para selecionar novos nós parceiros. A grande questão é definir quando, de fato, a taxa de recepção não está sendo suficiente. No CoolStreaming isso é feito monitorando o *buffer* de recepção dado um sub-fluxo j transmitido por um nó C_1 ao nó C_2 observando as inequações 1.1 e 1.2, onde:

   - T_s, T_p são duas métricas para especificar a capacidade de upload de um nó C_1. T_s e T_p são números de sequência de blocos para um sub-fluxos qualquer j no nó C_2;
   - T_s, é o limite do máximo número de sequência permitido entre os últimos blocos de vídeo recebidos qualquer dois sub-fluxos no nó C_2;
   - T_p, é o limite do máximo número de sequência dos últimos blocos de vídeo recebidos entre os nós parceiros de C_2 e os nós pais de C_2. A diferença entre os nós perceiros de um nó C e os nós pais de um nó C é a seguinte: as parcerias são estabelecidas entre dois nós que trocaram mapas de *buffer* com informações de disponibilidade de blocos de vídeo, ao passo que a relação pai e filho é estabelecida quando um nó (filho) está, de fato, recebendo o conteúdo de vídeo de um outro nó (pai);
   - $H_{S_{i,C_2}}$, é o número de sequência do último bloco de vídeo de um sub-fluxo S_i no nó C_2;
   - K, é o número de sub-fluxos gerados pelo nó transmissor que origina o conteúdo de vídeo.

$$max\{|H_{S_{i,C_2}} - H_{S_{j,C_1}}| : i \leq K\} < T_s \tag{1.1}$$

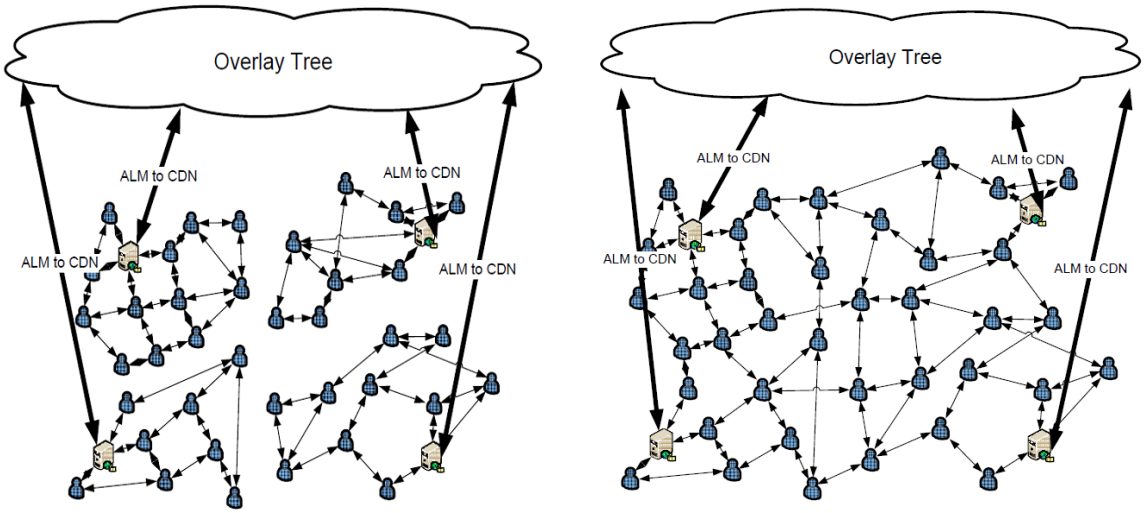
$$max\{H_{S_{i,q}} : i \leq K, q \in partners\} - H_{S_{j,C_1}} < T_s \quad (1.2)$$

A inequação 1.1 é utilizada para monitorar o *status* do *buffer* do nó C_2. Se a inequação 1.1 for falsa, significa que pelo menos um sub-fluxo está atrasado com relação ao limite estabelecido T_s. Isto indica que o nó C_2 deve selecionar outro nó para receber o fluxo de dados, pois o seu nó pai atual não tem capacidade de upload suficiente. A inequação 1.2 é utilizada para monitorar o status do buffer dos nós pais do nó C_2. Seja o conjunto $parents$, definido pelos nós pais do nó C_2 e o conjunto $partners$, definido pelos nós parceiros de C_2. O nó C_2 compara o status do *buffer* dos nós em $parents$ com relação aos status do *buffer* dos nós em $partners$. Se a inequação 1.2 for falsa para algum caso, implica que o nó pai C_1 está atrasado com relação ao número de blocos de vídeos comparado com pelo menos um dos nós em $partners$. Isto fará com que o nó C_2 finalize a comunicação com o nó C_1 atrasado e selecione um novo nó C_1 do cojunto $partners$.

Essa estratégia de seleção de nós aplicado no CoolStreaming é muito complexa porque exige o monitoramente constante dos buffers dos sub-fluxos, o que implica em exaustivas trocas de mapa de *buffer*, dependendo da quantidade de nós no conjunto $parents$ e $partners$ para um dado nó C_2. A consequência disso é um aumento significativo na qualidade de dados de controle trocados entre os nós parceiros e nós pais, além da troca de dados correspondentes ao vídeo. No GMTP, as parcerias são formadas de uma forma bastante diferente, que basicamente consiste em envolver um algoritmo de controle de congestionamento assistido pela rede que compartilha a sua capacidade de transmissão em um determinado instante *t*. Dessa forma, tanto um nó receptor quanto o nó transmissor conhecem a capacidade máxima de transmissão no canal que separa ambos. Sendo assim, o nó transmissor ajusta sua taxa de transmissão em direção ao nó receptor de acordo com a taxa de transmissão disponível em qualquer pacote de dados. No caso do GMTP, os nós são os roteadores de rede, que simplesmente repassam para os clientes os fluxos de dados que contém partes da mídia, correspondentes aos blocos de vídeo do CoolStreaming.

1.1.9 Denacast

O Denacast é um sistema para distribuição de mídias ao vivo baseado em um arquitetura híbrida P2P/CDN [25]. A porção P2P do sistema é adaptada do sistema CoolStreaming de modo que suporte a porção CDN. O escalonador do Denacast distribuição de mídias é idêntico ao do CoolStreaming que, segundo os próprios autores, é considerado o "coração" do sistema.



(a) Cenário de uma rede P2P/CDN de malhas desconectadas.

(b) Cenário de uma rede P2P/CDN de malhas conectadas.

Figura 1.9: Arquiteturas CDN/P2P utilizadas no Denacast. Figuras extraídas de [25].

Como ilustra-se na Figura 1.9, no Denacast, adota-se duas arquiteturas CDN/P2P, que tem em comum nós servidores como nós folhas de uma rede CDN *multicast*, os quais funcionam como nós fontes da mídia transmitida para a rede P2P.

- P2P/CDN de malhas desconectadas (Figura 1.9a): constitui-se diferentes redes de malhas independentes coordenadas por um servidor da rede CDN. Nessa arquitetura, cada servidor da CDN funciona como um nó *tracker* da sua respectiva rede malha;
- P2P/CDN de malhas conectadas (Figura 1.9b): constitui-se uma única rede de malhas formada por todos os nós servidores da CDN e todos os nós da rede P2P. Além dos vários servidores da CDN, utiliza-se também um nó que desempenha o papel de *tracker*. A responsabilidade do *tracker* é construir as redes de malhas e conectá-las à rede CDN.

Para a construção da rede de malha, o nó *tracker* mantém uma lista dos nós da rede P2P que estao ativos. Um nó fonte da mídia se anuncia ao nó *tracker*. Cada nó da rede P2P requisita uma lista de possíveis nós parceiros ao nó *tracker* e informando o número de parcerias que deseja efetivar. A nó fonte codifica as partes da mídia à medida que se captura o evento ao vivo e as coloca em um *buffer* de transmissão. No mesmo instante, gera-se o mapa do *buffer* que é anunciado para os nós parceiros do nó fonte. Os nós parceiros do nó fonte solicita as partes da mídia de acordo com o mapa de *buffer* e em seguida a transmissão entre eles ocorre de forma similar ao CoolStreaming. Quando um novo nó receptor deseja se conectar à rede, este se conecta ao nó *tracker*, que seleciona a rede de malha com menos nós clientes e retorna a lista de candidatos a nós parceiros.

O diferencial do Denacast comparado ao CoolStreaming é seu mecanismo de conectar duas ou mais redes de malhas. A regra geral considerada pelo nó *tracker* é a seguinte: duas redes de malha se unirão através de dois nós receptores conectados em cada uma das redes de malha a ser unidas. Isto ocorre quanto a quantidade de nós ativos nas duas redes ultrapassa o valor do número de nós servidores da CDN disponíveis no sistema vezes o número limite de nós em cada rede de malha antes do início da transmissão. Quando uma rede de malha A é unida a uma rede de malha B, o nó *tracker* passa a sugerir nós da rede A para a rede B e vice-versa.

Considerações sobre o trabalho

No ponto de vista da rede P2P, as considerações sobre o sistema Denacast são similares ao caso do CoolStreaming. Com relação a arquitetura geral, o Denacast tem uma melhor organização da rede de sobreposição devido ao uso de servidores de uma rede CDN. Isto permite um melhor agrupamento dos nós em uma determinada região lógica da rede (delimitada pela localização do nó servidor da CDN). Nesse sentido, o Denacast escala melhor o número de nós e melhora as métricas de qualidade de serviço relacionadas à transmissão de uma mídia ao vivo se comparado ao CoolStreaming.

O Denacast é o sistema que mais se aproxima ao GMTP, devido a sua estratégia de unir diferentes redes de malha quando a quantidade de nós em uma determinada rede extrapola um determinado limite. Porém, como já foi discutido na Seção 1.1.8, em discussão sobre o CoolStreaming e que se estende ao Denacast, o GMTP oferece funções diferenciadas e deta-

lhadas no Capítulo **??**, ao passo que ambos os protocolos foram confrontados em simulações de rede e os resultados são apresentados no Capítulo **??**.

1.1.10 Outras propostas

A área de distribuição de mídias ao vivo tem sido explorada há pelo menos 15 anos. Existe uma vasta quantidade de propostas que permeiam diferentes abordagens, das mais simples, como as baseadas em arquiteturas cliente servidor, às mais complexas, como as baseadas em P2P e/ou P2P/CDN. Diante desse cenário, realizou-se uma exaustiva pesquisa sobre os trabalhos relacionados à proposta apresentada nessa tese de doutorado, destacando-se os principais, tal como foram apresentadas ao longo dessa seção. Apesar desse filtro, pode-se afirmar que as soluções propostas nessa área de pesquisa são apresentadas considerando os aspectos discutidos no Capítulo **??**. Em geral, decide-se sobre se a estrutura da rede de sobreposição será em árvore e/ou em malha; sobre se os nós realizam periodicamente requisições das partes da mídia (*pull-based*) e/ou se os nós transmissores enviarão as partes da mídia para o nó receptor após este último sinalizar interesse por tal conteúdo (*push-based*); e, sobre a arquitetura do serviço, se será P2P e/ou P2P/CDN (modelos tradicionais como cliente/servidor não tem sido mais consideradas).

Durante o levantamento bibliográfico realizado no contexto deste trabalho, catalogou-se uma série de outras propostas que não foram aqui detalhadas, mas que, ao menos, devem ser mencionadas, para se ter uma noção da pulverização de soluções para este fim. São elas (em ordem alfabética): AnySee [26], BEAM/Alliances [27], BitTorrentLIVE [28], DLNA-P2P [29], GridMedia [12, 30, 31], IV5S [32], Joost [33, 34], LayeredCast [35], LiveSky [36], Octoshape [37], OverCast [38], Pastry/SplitStream [39], PeerCast [40], PPLive [41], PRIME [42], PULSE [43], SAMP [44], SmoothCache [45], Sopcast [46], TURINstream [47] e ZIGZAG [48]. Além dessas, diversas outras propostas foram investigadas, mas não foram consideradas nessa lista porque se tratam de produtos de software proprietários, não sendo possível referenciá-las formalmente.

1.2 Redes Centradas no Conteúdo

COLOCAR ESSA CONVERSA DAQUI COMO SENDO: UMA SOLUÇÃO QUE USA CCN E NESSE TRABALHO COMPARA-SE O GMTP COM ELA. COMENTAR TODAS AS COISAS ANOTADAS NA SEÇÃO DE CONCLUSÃO, E ISTO SERÁ JUSTIFICATIVA PARA DIZER, NO CAPÍTULO DE RESULTADOS, PORQUE QUE O GMTP FOI MELHOR

ICN

Falar sobre o CCNx – *Content Centric Network Protocol*

Falar sobre o NDN – *Named Data Network*

*Supporting Mobile Applications with Information Centric Networking: the Case of P2P Live Adaptive Video Streaming - http://conferences.sigcomm.org/sigcomm/2013/papers/icn/p35.pdf PEGAR OS ARTIGOS PRINCIPAIS SOBRE CCNx + Video

Procurar mais, pegar aqueles publicados no NDN

6666771

1.2.1 Escolher outra solução e colocar aqui – VoCCN??

Preencher

1.2.2 Escolher uma solução e colocar aqui – CCNx + HLS

Preencher...

1.2.3 Escolher outra solução e colocar aqui – CCNx + MPEG-DASH

Preencher

Considerações sobre o trabalho

1.3 Sumário Comparativo

Nas seções anteriores, presentou-se as considerações sobre cada trabalho com comparações entre o respectivo trabalho e o protocolo GMTP. Na Tabela 1.1 apresenta-se um sumário das comparações apresentadas anteriormente. Os critérios de comparação são apresentados a seguir.

- Localizado na Camada de Transporte (CT)
- Suporte ao compartilhamento de conexão (CS)
- Suporte à transmissão em multicast e unicast (TMU)
- Suporte à descoberta de nós de forma distribuída (DND)
- Suporte a controle de congestionamento (CC)
- Suporte à adaptação de fluxo multimídia (AFM)
- Tolerância à desconexão (TD)
- Extensibilidade para novos algoritmos (ENA)
- Compatibilidade com API (*Application Programming Interface*) de *socket* BSD/POSIX (CAS)

Tabela 1.1: Tabela comparativa dos protocolos para transmissão de mídia em tempo real. **Legenda:** X = Suporta; $\frac{X}{2}$ = Suporta parcialmente, apenas para algoritmos de controle de congestionamento; S = Requer intervenção da aplicação; Ind. = Indefinido.

Protocolos	CT	CS	TMU	DND	CC	AFM	TD	ENA	CAS
UDP	X		S						X
TCP	X		S					$\frac{X}{2}$	X
DCCP	X							$\frac{X}{2}$	X
SCTP	X			X				$\frac{X}{2}$	X

Continuação na próxima página

Tabela 1.1 – continuação da página anterior

Protocolos	CT	CS	TMU	CC	DND	AFM	TD	ENA	CAS
PPETP				X	Ind.	X	X		S
PPSP/Swift				X	X		X		S
PDTP							X		S
CPM			X	X		X	X		
HySAC			X	X		X	X		
Danacast	X	X	X	X	X	X	X	X	X
GMTP	X	X	X	X	X	X	X	X	X

1.4 Sumário do Capítulo

Neste capítulo, apresentou-se uma avaliação crítica acerca de um conjunto de sistemas e protocolos para distribuição de conteúdos multimídia. Discutiu-se sobre trabalhos com propostas semalhantes ao protocolo GMTP, entendendo-se que os principais trabalhos disponíveis no estado da arte não contempla os recursos e a forma de proposta do GMTP. De fato, apenas o Denacast contempla uma proposta próxima ao protocolo GMTP e, por este motivo, no Capítulo **??**, discute-se em detalhes seu desempenho em comparação ao GMTP.

1.4.1 Preâmbulo ao GMTP

A definição do protocolo GMTP se baseou nos requisitos e estado da prática dos sistemas de distribuição de conteúdos (Capítulo **??**), bem como no levantamento das proposta disponíveis no estado da arte, como as discutidas nas seções anteriores deste capítulo. Como complemento, três questionamentos foram primordiais motivadores para o projeto de tal protocolo, são eles:

1. Quais as funcionalidades dos sistemas de distribuição de conteúdos ao vivo que poderiam que ser implementadas na camada de aplicação por falta de um protocolo de transporte de dados ideal para esse tipo de aplicação?

2. Quais dessas funcionalidades podem ser implementadas na camada de transporte a fim

de torná-las padronizadas para todas as aplicações existentes, evitando-se o retrabalho de desenvolvimento?

3. Como se pode, de forma eficiente, distribuir um mesmo fluxo de dados multimídia partindo de um servidor para múltiplos nós clientes conectados a diferentes redes, considerando o fato de que diferentes aplicações clientes possam ser utilizadas e com suporte a controle de congestionamento assistido pela rede?

Diante dessas questões, notou-se que ao tentar resolver problemas relacionados a cada um desses questionamentos, os desenvolvedores desse tipo de sistema enfrentam situações reincidentes, já experimentadas por outras equipes de desenvolvimento. No GMTP, propõe-se concentrar as principais funções dos sistemas de transmissão de mídias ao vivo em um único protocolo de rede, buscando-se contribuir com a evolução do estado da arte e da prática ao propor um protocolo que desacopla o processo de transporte de datagramas IP, os quais carregam conteúdos de mídias ao vivo, da forma como estes são exibidos ao usuário final através das aplicação de rede. Isto ocorre com o emprego de técnicas de engenharia de software para abstrair a complexidade no desenvolvimento de sistemas dessa natureza, ao passo que se propõe algoritmos que otimizam o consumo de recursos de rede, tudo isto visando melhorar a qualidade de experiência do usuário ao assistir um conteúdo ao vivo através da Internet.

Em termos arquiteturais, o GMTP está posicionado de tal forma que suas funções são disponibilizadas para diferentes processos de aplicação em execução em um sistema operacional, de modo que se torna independente de linguagem de programação, bibliotecas de funções e dos mais variados tipos de dispositivos de usuários finais (desktops, notebooks, tablets, smartphones, smartTVs, etc.), independente da mobilidade dos usuários e podendo ser embarcado no núcleo de qualquer sistema operacional moderno. Com isto, promove-se recursos funcionais para a camada de aplicação cada vez mais estáveis e eficientes, propagando-se as boas práticas (algoritmos) que passam a ser utilizadas e testadas por um conjunto cada vez maior de desenvolvedores, aprimorando-as com o passar do tempo sem causar um impacto direto à camada de aplicação, tal como tem ocorrido com o protocolo TCP ao longo dos seus 32 anos de existência, desde da sua primeira versão em 1981.

Com esta visão, realizou-se um estudo sobre as principais estratégias adotadas pelos desenvolvedores de sistemas de transmissão de conteúdos multimídia ao vivo, objetivando-se mapear o estado da prática e utilizar este mapeamento como artefato de tomada de decisão do projeto do GMTP. O resultado foi o seguinte:

1. constatou-se que os sistemas mais robustos de distribuição de conteúdos multimídia são baseados em arquiteturas híbridas P2P/CDN. Isto ocorre porque se obtém escalabilidade do número de usuários e redução de custos com infra-estrutura de rede por meio das redes P2P; e facilidade no gerenciamento e maior estabilidade de disponibilização dos serviços, por meio das CDNs [25, 49–56];

2. observou-se que nos sistemas desse tipo não se viabiliza o acesso a um conteúdo ao vivo por parte de um usuário levando-se em consideração apenas o seu interesse em assistir a um evento ao vivo (*abordagem centrada no conteúdo*). Nas soluções existentes, a forma de acesso ao conteúdo de um evento ao vivo é dependente do local físico (servidor) onde tal conteúdo está sendo transmitido (*abordagem centrada no hospedeiro*), sendo cruscial saber de qual nó fonte o evento está sendo recebido;

3. dado que na maioria dos sistemas de distribuição de conteúdos ao vivo se transmite fluxos de dados UDP, tais sistemas não executam mecanismos para controle de congestionamento. Desta forma, se isto for necessário em seus sistemas, os desenvolvedores são forçados a executar algoritmos de controle de congestionamento na camada de aplicação, sem qualquer padronização na forma como os fluxos de dados são controlados, com diferentes equipes de desenvolvimento implementando, das mais variadas formas, a mesma funcionalidade presente nesse tipo de aplicação, sem qualquer compartilhamento desse esforço [**?**, 7, 7, 7];

4. não há uma forma efetiva de centralizar e/ou disponibilizar as boas soluções e práticas (algoritmos) para as diferentes funcionalidades empregadas nesse tipo de sistema. De fato, existem diversos *middlewares* de desenvolvimento de sistemas de distribuição de conteúdo que tentam suprir as limitações existentes dos protocolos da camada de transporte, tratando-se de soluções intermediárias e fragmentadas para o transporte de dados multimídia nas redes de computadores, salvas suas devidas contribuições científicas e reais [7, 7, 7, 7].

Bibliografia

[1] Hong Liu and P. Mouchtaris. Voice over IP Signaling: H.323 and Beyond. In *Communications Magazine, IEEE*, volume 28, pages 142 – 148, 10 2000.

[2] K.C. Almeroth. The evolution of Multicast: From the MBone to Interdomain Multicast to Internet2 Deployment. *Network, IEEE*, 14(1):10–20, Jan 2000.

[3] J. Rosenberg, H. Schulzrinne, and G. Camarillo. SIP: Session Initiation Protocol, 6 2002. `http://www.ietf.org/rfc/rfc3261.txt`. Último acesso: 5 de Fevereiro de 2014.

[4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, 7 2003. `http://www.ietf.org/rfc/rfc3550.txt`. Último acesso: 5 de Fevereiro de 2014.

[5] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp), 4 1998. `http://www.ietf.org/rfc/rfc2326.txt`. Último acesso: 5 de Fevereiro de 2014.

[6] R. Pantos and W. May. HTTP Live Streaming, 10 2013. `http://tools.ietf.org/html/draft-pantos-http-live-streaming-12`. Último acesso: 5 de Fevereiro de 2014.

[7] TBE. Tbe, 3 2008.

[8] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *MultiMedia, IEEE*, 18(4):62–67, April 2011.

[9] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann. Dynamic adaptive http streaming of live content. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–8, June 2011.

[10] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax, 1 2005. `http://www.ietf.org/rfc/rfc3986.txt`. Último acesso: 5 de Fevereiro de 2014.

[11] Chen Feng, Baochun Li, and Bo Li. Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits. In *INFOCOM 2009, IEEE*, pages 891–899, April 2009.

[12] Meng Zhang, Qian Zhang, Lifeng Sun, and Shiqiang Yang. Understanding the power of pull-based streaming protocol: Can we do better? *Selected Areas in Communications, IEEE Journal on*, 25(9):1678–1694, December 2007.

[13] Susu Xie, Bo Li, G.Y. Keung, and Xinyan Zhang. Coolstreaming: Design, theory, and practice. *IEEE Transactions on Multimedia*, 9(8):1661–1671, December 2007.

[14] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, pages 157–168, New York, NY, USA, 2011. ACM.

[15] Nandita Dukkipati. *Rate control protocol (rcp): congestion control to make flows complete quickly*. PhD thesis, Stanford, CA, USA, 2008. AAI3292347.

[16] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP, 6 1999. `http://www.ietf.org/rfc/rfc2616.txt`. Último acesso, Abril de 2008.

[17] A. Arcieri. Peer distributed transfer protocol, 7 2004. `ftp://ftp.good.net/dl/bd/convention.cdroms/defcon12/Arcieri/draft-arcieri-peer-distributed-transfer-protocol.txt`. Último acesso: 5 de Fevereiro de 2014.

[18] V. Gopalakrishnan, B. Bhattacharjee, K.K. Ramakrishnan, R. Jana, and D. Srivastava. Cpm: Adaptive video-on-demand with cooperative peer assists and multicast. In *INFOCOM 2009, IEEE*, pages 91 –99, april 2009.

[19] C. Kulatunga, G. Kandavanam, A.I. Rana, S. Balasubramaniam, and D. Botvich. Hysac: A hybrid delivery system with adaptive content management for iptv networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5, june 2011.

[20] Riccardo Bernardini, Roberto Rinaldo, and Roberto Fabbro. Peer-to-Peer Epi-Transport protocol, 7 2011. `http://tools.ietf.org/html/draft-bernardini-ppetp-03`. Último acesso: 5 de Fevereiro de 2014.

[21] D. Rossi, C. Testa, S. Valenti, and L. Muscariello. Ledbat: The new bittorrent congestion control protocol. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pages 1–6, Aug.

[22] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. Low extra delay background transport (LEDBAT), 10 2011. `http://tools.ietf.org/id/draft-ietf-ledbat-congestion-09.txt`. Último acesso: 5 de Fevereiro de 2014.

[23] Bo Li, Susu Xie, Yang Qu, G.Y. Keung, Chuang Lin, Jiangchuan Liu, and Xinyan Zhang. Inside the New Coolstreaming: Principles, Measurements and Performance Implications. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages –, 4 2008.

[24] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y. -S.P Yum. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2102– 2111 vol. 3. IEEE, March 2005.

[25] S. M Y Seyyedi and B. Akbari. Hybrid cdn-p2p architectures for live video streaming: Comparative study of connected and unconnected meshes. In *Computer Networks and*

Distributed Systems (CNDS), 2011 International Symposium on, pages 175–180, Feb 2011.

[26] Qi Huang, Hai Jin, Ke Liu, Xiaofei Liao, and Xuping Tu. Anysee2: an auto load balance P2P live streaming system with hybrid architecture. In *Proceedings of the 2nd international conference on Scalable information systems*, InfoScale '07, pages 30:1–30:2, Suzhou, China, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ACM ID: 1366843.

[27] Darshan Purandare and R. Guha. An Alliance Based Peering Scheme for P2P Live Media Streaming. *Multimedia, IEEE Transactions on*, 9(8):1633–1644, 2007.

[28] Qingchao Cai, Xiaolu Zhang, and Xuejie Zhang. Streaming live media over bittorrent. In *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, volume 3, pages 44–49, Jan 2009.

[29] Chin-Feng Lai, Yueh-Min Huang, and Han-Chieh Chao. Dlna-based multimedia sharing system for osgi framework with extension to p2p network. *Systems Journal, IEEE*, 4(2):262–270, June 2010.

[30] Li Zhao. GridMedia+: a P2P streaming system for live and On-Demand video. In *2009 6th IEEE Consumer Communications and Networking Conference*, pages 1–2, Las Vegas, Nevada, USA, January 2009.

[31] Zhenjiang Li, Yao Yu, Xiaojun Hei, and Danny H. K Tsang. Towards low-redundancy push-pull P2P live streaming. In *Proceedings of the 5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, QShine '08, pages 13:1–13:7, Hong Kong, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ACM ID: 1535589.

[32] Zhijia Chen, Chuang Lin, and Xiaogang Wei. Enabling on-demand internet video streaming services to multi-terminal users in large scale. *Consumer Electronics, IEEE Transactions on*, 55(4):1988–1996, November 2009.

[33] Majed Alhaisoni and Antonio Liotta. Characterization of signaling and traffic in Joost. *Peer-to-Peer Networking and Applications*, 2(1):75–83, 2009.

[34] J. Moreira, R. Antonello, S. Fernandes, C. Kamienski, and D. Sadok. A step towards understanding Joost IPTV. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 911–914, April 2008.

[35] M. Moshref, R. Motamedi, H.R. Rabiee, and M. Khansari. Layeredcast - a hybrid peer-to-peer live layered video streaming protocol. In *Telecommunications (IST), 2010 5th International Symposium on*, pages 663–668, Dec 2010.

[36] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Livesky: Enhancing cdn with p2p. *ACM Trans. Multimedia Comput. Commun. Appl.*, 6(3):16:1–16:19, August 2010.

[37] J. Peltotalo, J. Harju, A. Jantunen, M. Saukko, L. Vaatamoinen, I. Curcio, I. Bouazizi, and M. Hannuksela. Peer-to-peer streaming technology survey. In *Networking, 2008. ICN 2008. Seventh International Conference on*, pages 342–350, April 2008.

[38] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole, Jr. Overcast: Reliable multicasting with on overlay network. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation - Volume 4*, OSDI'00, pages 14–14, Berkeley, CA, USA, 2000. USENIX Association.

[39] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: high-bandwidth multicast in cooperative environments. *SIGOPS Oper. Syst. Rev.*, 37(5):298–313, October 2003. ACM ID: 945474.

[40] Jie Xiong and R.R. Choudhury. Peercast: Improving link layer multicast through cooperative relaying. In *INFOCOM, 2011 Proceedings IEEE*, pages 2939–2947, April 2011.

[41] X Hei, C Liang, J Liang, Y Liu, and KW Ross. Insights into PPLive: a measurement study of a Large-Scale P2P IPTV system. In *In Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[42] N. Magharei and R. Rejaie. Prime: Peer-to-peer receiver-driven mesh-based streaming. *Networking, IEEE/ACM Transactions on*, 17(4):1052–1065, Aug 2009.

[43] F. Pianese, J. Keller, and E.W. Biersack. Pulse, a flexible p2p live streaming system. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6, April 2006.

[44] Weizhan Zhang, Zhenyan Li, and Qinghua Zheng. Samp: supporting multi-source heterogeneity in mobile p2p iptv system. *Consumer Electronics, IEEE Transactions on*, 59(4):772–778, November 2013.

[45] Roberto Roverso, Sameh El-Ansary, and Seif Haridi. Smoothcache: Http-live streaming goes peer-to-peer. 7290:29–43, 2012.

[46] B. Fallica, Yue Lu, F. Kuipers, R. Kooij, and P. Van Mieghem. On the quality of experience of SopCast. In *The Second International Conference on Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08*, pages 501–506. IEEE, September 2008.

[47] A. Magnetto, R. Gaeta, M. Grangetto, and M. Sereno. TURINstream: A Totally pUsh, Robust, and effIcieNt P2P Video Streaming Architecture. *Multimedia, IEEE Transactions on*, 12(8):901–914, Dec 2010.

[48] D. A Tran, K. A Hua, and T. Do. ZIGZAG: an efficient peer-to-peer scheme for media streaming. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1283– 1292 vol.2. IEEE, April 2003.

[49] Dongyan Xu, Sunil Suresh Kulkarni, Catherine Rosenberg, and Heung keung Chai. A cdn-p2p hybrid architecture for cost-effective streaming media distribution. *Computer Networks*, 44:353–382, 2004.

[50] Dongyan Xu, SunilSuresh Kulkarni, Catherine Rosenberg, and Heung-Keung Chai. Analysis of a cdn-p2p hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11:383–399, 2006.

[51] Melika Meskovic, Himzo Bajric, and Mladen Kos. Content delivery architectures for live video streaming: Hybrid cdn-p2p as the best option. In *The Fifth International*

Conference on Communication Theory, Reliability, and Quality of Service, pages 26–32, 2012.

[52] R. Buyya, A.-M.K. Pathan, J. Broberg, and Z. Tari. A case for peering of content delivery networks. *Distributed Systems Online, IEEE*, 7(10):3–3, Oct.

[53] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Understanding hybrid cdn-p2p: why limelight needs its own red swoosh. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '08, pages 75–80, New York, NY, USA, 2008. ACM.

[54] Kideok Cho, Hakyung Jung, Munyoung Lee, Diko Ko, T.T. Kwon, and Yanghee Choi. How can an ISP merge with a CDN? *Communications Magazine, IEEE*, 49(10):156–162, Oct.

[55] Z. Chen, H. Yin, C. Lin, Y. Chen, and M. Feng. Towards a universal friendly peer-to-peer media streaming: metrics, analysis and explorations. *Communications, IET*, 3(12):1919–1933, December.

[56] Xuening Liu, Hao Yin, and Chuang Lin. A novel and high-quality measurement study of commercial cdn-p2p live streaming. In *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, volume 3, pages 325–329, Jan.