

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

**GMTP: Distribuição de Mídias Ao Vivo através de  
uma Rede de Favores Constituída entre 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  
©Leandro Melo de Sales, 03/03/2014

## **Resumo**

TBD

## **Abstract**

TBD

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Delimitação e Argumentos . . . . .	4
1.2	Descrição do Problema . . . . .	9
1.3	Hipótese . . . . .	17
1.4	Objetivo . . . . .	17
1.4.1	Objetivos Específicos . . . . .	18
1.5	Relevância do Tema e da Tese . . . . .	19
1.6	Resumo das Contribuições . . . . .	20
1.7	Publicações . . . . .	21
1.8	Estrutura do Documento . . . . .	22
<b>2</b>	<b>Fundamentação</b>	<b>24</b>
2.1	Protocolos para Gerenciamento e Sincronização de Mídias Ao Vivo . . . . .	25
2.1.1	H.323, SIP, RTP e RTSP . . . . .	25
2.2	Estruturas para Distribuição de Mídia ao Vivo em P2P . . . . .	26
2.2.1	Estrutura Baseada em Árvore . . . . .	27
2.2.2	Estrutura Baseada em Malha . . . . .	30
2.2.3	Estrutura Híbrida . . . . .	34
2.3	Sistemas de Transmissão ao Vivo em P2P . . . . .	37
2.3.1	Geração do Conteúdo da Transmissão . . . . .	40
2.3.2	Armazenamento e Consumo de Dados . . . . .	41
2.3.3	Estratégia de Seleção de <i>Chunks</i> . . . . .	42
2.3.4	Realização de Parcerias e Obtenção de <i>Chunks</i> . . . . .	44
2.4	Criptografia de Hash e Assinatura Digital . . . . .	45

2.4.1	Criptografia de Hash . . . . .	46
2.4.2	Criptografia Assimétrica . . . . .	46
2.4.3	Assinatura e Certificação Digital . . . . .	46
2.5	Sumário do Capítulo . . . . .	49
<b>3</b>	<b>Trabalhos Relacionados</b>	<b>51</b>
3.1	Visão Geral das Propostas para Distribuição de Mídias ao Vivo . . . . .	52
3.2	Aplicações e Protocolos para Distribuição de Mídias ao Vivo . . . . .	55
3.2.1	<i>HLS – HTTP Live Streaming, HDS – HTTP Dynamic Streaming, DASH – Dynamic Adaptive Streaming over HTTP e HSS – HTTP Smooth Streaming</i> . . . . .	55
3.2.2	<i>PDT – Peer Distributed Transfer Protocol</i> . . . . .	60
3.2.3	<i>CPM – Cooperative Peer Assists and Multicast</i> . . . . .	62
3.2.4	<i>HySAC – Hybrid Delivery System with Adaptive Content Management for IPTV Networks</i> . . . . .	64
3.2.5	<i>PPSP/Swift – P2P Streaming Protocol / The Generic Multiparty Transport Protocol</i> . . . . .	66
3.2.6	<i>DONet/CoolStreaming</i> . . . . .	69
3.2.7	<i>CoolStreaming/Denacast</i> . . . . .	75
3.2.8	Outras propostas . . . . .	77
3.3	Redes Centradas na Informação . . . . .	78
3.3.1	Redes de Dados Nomeados (NDN) . . . . .	79
3.4	Sumário do Capítulo . . . . .	88
<b>4</b>	<b>Global Media Transmission Protocol (GMTP)</b>	<b>89</b>
4.1	Visão Geral . . . . .	92
4.1.1	Tipos de Pacotes . . . . .	97
4.1.2	Resumo das Principais Funcionalidades . . . . .	99
4.2	Definições, Relações e Restrições . . . . .	101
4.3	Constituição da Rede de Favores $\eta$ . . . . .	104
4.3.1	Registro de Participação de $r_d$ em $\eta$ . . . . .	105
4.3.2	Tabela de Recepção de Fluxos de Dados . . . . .	110

4.3.3	Formação de Parcerias . . . . .	112
4.4	Transmissão de $p_x \in P$ através de $\eta$ . . . . .	119
4.4.1	Indexação de Conteúdo . . . . .	119
4.4.2	Estabelecimento de Conexão entre $c_f$ e $s_a$ para Obter $P$ . . . . .	124
4.4.3	Fase 1: Primeira Requisição a um Fluxo de Dados $P$ . . . . .	125
4.4.4	Fase 2: Próximas Requisições para Obter $P$ . . . . .	129
4.4.5	Fase 3: Busca por Mais Parceiros $r_q$ para Obter $P$ . . . . .	130
4.4.6	Envio e Recebimento de $p_x \in P$ em $\eta$ . . . . .	134
4.5	Controle de Congestionamento em $\eta$ . . . . .	137
4.5.1	Controle de Congestionamento Unicast . . . . .	138
4.5.2	Controle de Congestionamento Multicast . . . . .	150
4.6	Autenticidade de $P$ . . . . .	153
4.6.1	Transmissão e Assinatura de Autenticidade de $p_x \in P$ . . . . .	154
4.6.2	Verificação de Autenticidade de $p_x \in P$ . . . . .	155
4.6.3	Habilitar / Desabilitar a Validação de Pacotes $p_x \in P$ . . . . .	156
4.6.4	Obtenção da Chave Pública $K_{s_a}^+$ de $s_a$ . . . . .	157
4.7	Outras Considerações . . . . .	158
4.7.1	Canais de Comunicação . . . . .	158
4.7.2	Procedimentos para Desconexão de nós $c_f$ , $l_w$ e $r_d$ . . . . .	159
4.7.3	Eleição de nós $l_w$ . . . . .	161
4.8	Sumário do Capítulo . . . . .	161

# Capítulo 1

## Introdução

A Internet é um sucesso tecnológico e sua infra-estrutura global conecta bilhões de hóspedes, transmitindo-se uma volumosa quantidade de dados digitais diariamente. Para se ter uma ideia, estima-se que em 2016 ocorrerão aproximadamente 18,9 bilhões de conexões de rede – quase 2,5 conexões para cada pessoa no planeta – em comparação aos 10,3 bilhões registrados em 2011 [1].

O número crescente de conexões de banda larga à Internet têm provocado o aumento do consumo e da exigência de qualidade de serviço, que eleva os custos com largura de banda dos distribuidores de conteúdo. Por exemplo, o YouTube<sup>TM</sup> registra um custo operacional com largura de banda da ordem de US\$1 milhão por mês para atender cerca de 20 milhões de usuários por dia, para transmitir o equivalente a 60 mil anos de vídeos se fossem reproduzidos sequencialmente [2]. Estima-se que em 2014 o tráfego de vídeo será maior do que foi o tráfego de redes entre pares (P2P) para compartilhamento de arquivos em 2009, correspondendo a 39 % do tráfego de dados total na Internet. Além disso, estima-se que o tráfego de VoIP, vídeo e jogos na Internet atingirá a marca de 40 exabytes por mês, quase 50 % do tráfego de dados total na Internet previsto para 2014 [3]. Segundo um estudo da empresa Cisco<sup>TM</sup>, estima-se que em 2016 cerca de 1,2 milhão de minutos serão transmitidos pela Internet a cada segundo – o equivalente a 833 dias ou mais de dois anos [1].

Com a evolução da WWW (*World Wide Web*) para a Web 2.0 [4,5], os usuários passaram a ter um papel de destaque no processo de prover alguns serviços. Um exemplo notório é o serviço de distribuição de conteúdos multimídia, sejam serviços onde as empresas disponibilizam conteúdos armazenados, como o YouTube e Netflix<sup>TM</sup>, ou principalmente os casos

de transmissões empresariais e residenciais ao vivo, como o CoolStreaming™, PPLive™ e o UStream.tv™. Nesses últimos casos, os sistemas permitem a transmissão de conteúdos ao vivo gerados a partir do computador de um usuário para milhares de outros usuários conectados à Internet [6]. O UStream.tv recebe mais de 50 milhões de acessos e transmite 1,5 milhão de horas de vídeo ao vivo por mês, com mais de 2 milhões de usuários cadastrados e um dos canais<sup>1</sup> com mais de 284 milhões de acessos entre 2010 e fevereiro de 2014, com um pico de 100 milhões de acessos em uma semana. Outro caso é o da NASA.TV<sup>2</sup> (*National Aeronautics and Space Administration Television*), que em 2011 migrou todos seus canais ao vivo na Internet para o UStream.tv, com mais de 25 milhões de acessos em fevereiro de 2014.

Ao observar essas e outras estatísticas [6–11], o que preocupa é o crescimento acentuado do consumo de recursos computacionais de rede resultante principalmente das estratégias e protocolos de rede adotados para realizar a distribuição dos conteúdos multimídia, cada um com suas próprias soluções [12–17]. No final do primeiro semestre de 2011, um artigo publicado no *Financial Post* chamou atenção ao “*anunciar momentos de grandes congestionamentos na Internet nos próximos anos*” [18,19]. Destacou-se a Netflix como sendo a maior consumidora de banda de rede da América do Norte, respondendo por 24,71 % de todos os dados transferidos durante o horário de pico de uso da Internet por norte-americanos, ultrapassando até mesmo a rede BitTorrent™ de compartilhamento de arquivos, como ilustra-se na Figura 1.1(a). “*A questão é que se não fosse a Netflix, teria sido a Amazon™, se não fosse a Amazon teria sido o Google™, mas o verdadeiro e iminente problema de congestionamento da rede começará quando todos passarem a fazer isto.*”, afirmou Colin Gillis, analista sênior de tecnologia da BGC Partners<sup>3</sup>.

Essa discussão se generaliza em um relatório publicado pela empresa de consultoria Sandvine, onde se menciona que “*com o rápido sucesso da Netflix, YouTube, UStream e outros, os provedores de Internet em todo o mundo devem se preparar para um futuro em que serviços de multimídia ao vivo será disponibilizado em grandes proporções, podendo ser responsável pela maior fatia do tráfego de dados na Internet*” [19]. De acordo com o que foi

<sup>1</sup><http://www.ustream.tv/decoraheagles>. O nome do canal é *Decorah Eagles*, um acen-tamento de águias, localizado em Decorah, Iowa.

<sup>2</sup><http://www.ustream.tv/nasa>

<sup>3</sup>BGC Partners: <http://www.bgcpartners.com/>

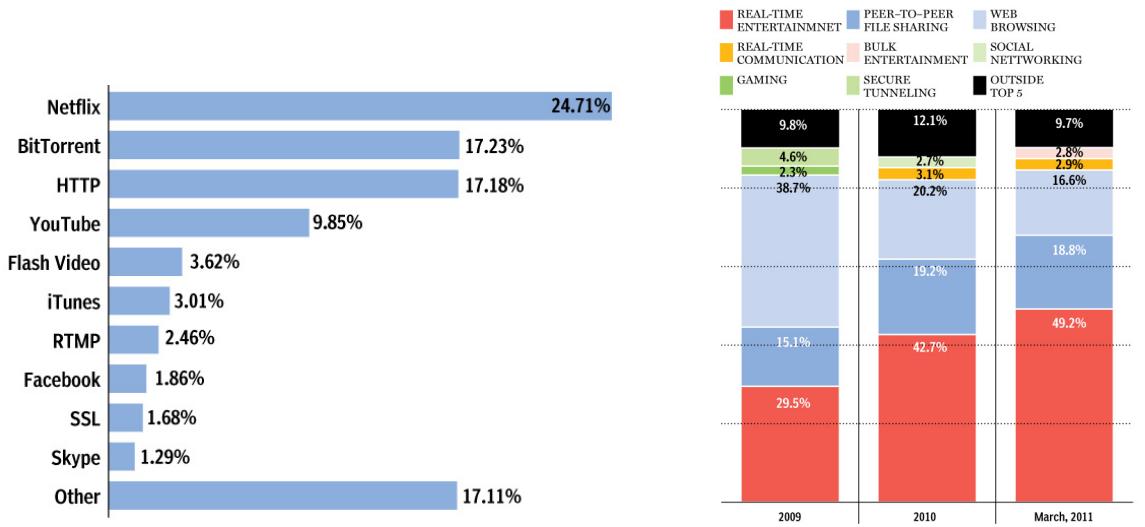


Figura 1.1: Perfil de tráfego de rede da América do Norte durante o horário de pico (considerando-se redes fixas) [18].

publicado nesse relatório, serviços ao vivo de entretenimento foram responsáveis por quase metade (49,2 %) de todo o tráfego de Internet na América do Norte no primeiro trimestre de 2011, como ilustrado-se na Figura 1.1(b). Este número alcançou 60 % no final de 2011 e “*o fato é que o volume de tráfego na Internet cresce exponencialmente e o congestionamento da rede tende a piorar. As pessoas sabem o que está acontecendo, mas está acontecendo mais rápido do que elas esperavam.*”, comentou Tom Donnelly, vice-presidente da Sandvine.

Nesse contexto, nota-se que o panorama atual dos serviços de distribuição de conteúdos multimídia ao vivo se apresenta em fase de grande expansão, principalmente no que diz respeito ao interesse dos usuários e ao super consumo dos recursos de rede. Por isto, há uma grande motivação para estudar e propor novas soluções para utilizar eficientemente as redes de computadores a fim de melhorar os serviços de distribuição multimídia na Internet [20, 21].

## 1.1 Delimitação e Argumentos

O ponto de partida no contexto deste trabalho são os protocolos de rede utilizados nos sistemas baseados em uma arquitetura de rede híbrida P2P/CDN, ou seja, par-a-par (*Peer to Peer - P2P*) [22–25] e cliente-servidor com suporte de uma rede de distribuição de conteúdos (*Content Delivery Network - CDNs*) [26–28] (Figura 1.2). Isto porque há evidências contundentes [29–38] de que se trata da principal escolha dos sistemas mais robustos, conseguindo-se escalabilidade do número de usuários e redução de custos com infra-estrutura de rede, por meio das redes P2P; ao passo que se consegue facilidade no gerenciamento e maior estabilidade de disponibilização dos serviços, por meio das CDNs.

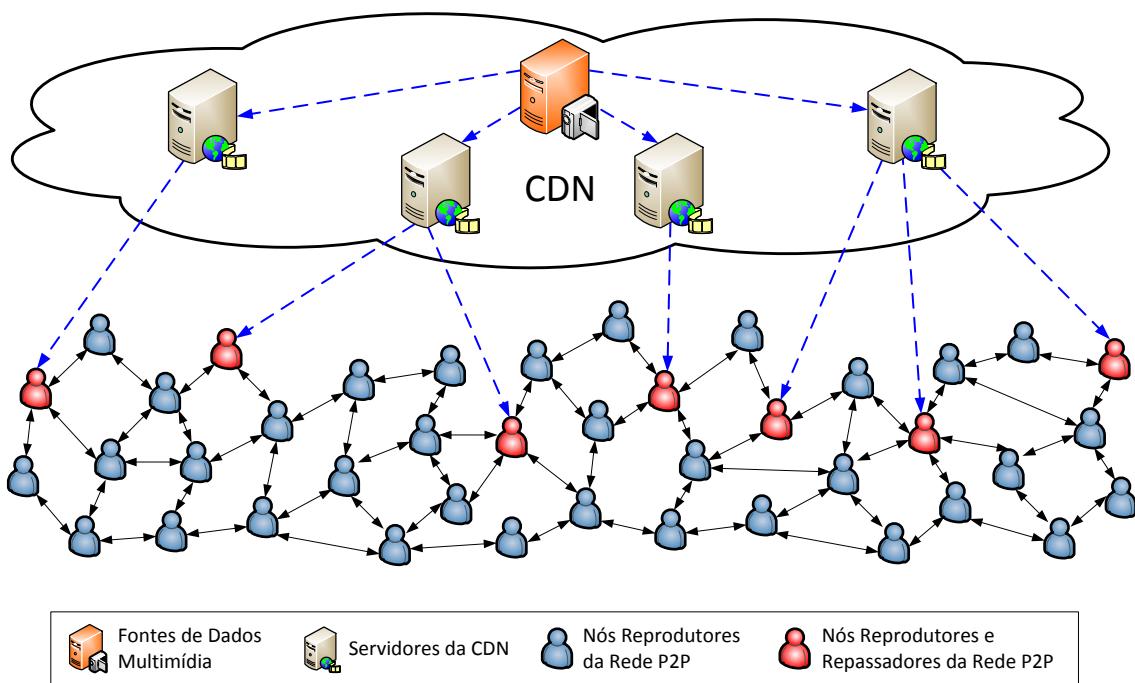


Figura 1.2: Estrutura de rede de uma arquitetura híbrida P2P/CDN para distribuição de conteúdo multimídia.

Nesse tipo de arquitetura de rede, os servidores da CDN atuam como super nós para a rede P2P, ao passo que os nós da rede P2P cooperam entre si a fim de disseminar mais rapidamente os datagramas, reproduzindo-os também localmente. Os datagramas são transmitidos após a captura dos quadros de um vídeo e uma aplicação os comprime, empacota-os e imediatamente os transmite para os sistemas remotos interessados. Do ponto de vista de protocolos de rede, utiliza-se conexões TCP (*Transmission Control Protocol*) [39] para se-

lecionar os nós parceiros e para troca de mapas de *buffers*, que servem para indexar quais nós possuem quais partes da mídia; ao passo que utiliza-se o UDP (*User Datagram Protocol*) [40] para transmitir datagramas contendo as partes da mídia e exibi-las ao usuário final.

Um aspecto importante para esses sistemas é definir de que forma os nós devem escalar-nar o uso de seus recursos de forma que a alocação seja a mais eficiente possível, evitando-se sobrecarga nos servidores da CDN e nos canais de comunicação. Em sistemas dessa natureza, os nós estão sempre acessando e disponibilizando recursos uns dos outros, ao mesmo tempo que os servidores da CDN organizam os nós da rede P2P e mantém os serviços de distribuição mais estáveis, tornando o sistema menos vulnerável ao dinamismo de participação dos nós de uma redes P2P, também conhecido por *churn* [41, 42].

Isto posto, os cenários de aplicação considerados neste trabalho são os que apresentam um nó de rede gerador de um conteúdo multimídia e milhares de nós receptores, estabele-cendo portanto uma relação  $1 \rightarrow n$ . Tais aplicações possuem uma característica em comum: a existência de muitos usuários interessados por um mesmo conteúdo e pouco ou nenhum dado individualizado, ou seja, que precisa ser transmitido apenas para um usuário ou um grupo restrito deles. Esta característica leva as seguintes peculiaridades:

1. o usuário de um potencial nó contribuidor tem que expressar interesse em um deter-minado evento no instante da sua ocorrência e não quando já possui o conteúdo a ser compartilhado, como nos sistemas P2P de compartilhamento de arquivo. Além disso, as parcerias são realizadas entre os nós com interesses comuns por um único conteúdo e não por múltiplos conteúdos ao mesmo tempo;
2. quando o usuário de um potencial nó contribuidor não tem interesse em um conteúdo, a aplicação, na maioria das vezes, não é executada e portanto a dependência pelo oportunismo é mais crítica do que nos sistemas de compartilhamento de arquivos. Isto torna a rede mais dinâmica e consequentemente os serviços mais instáveis – há um impacto direto nos parâmetros que determinam a experiência do usuário ao reproduzir um conteúdo multimídia devido ao aumento do *churn*;
3. o estado do mapa de *buffer* de reprodução de cada nó da rede é semelhante, pois não existe a possibilidade de um nó ter um mapa de *buffer* com muito mais dados para re-

produzir do que outros nós. Da mesma forma, não faz sentido manter por muito tempo dados já reproduzidos no *buffer* de reprodução. Os datagramas expiram<sup>4</sup> rapidamente e, nestes casos, devem ser descartados pelas aplicações dos nós que os recebem. Assim, o tamanho necessário para o *buffer* de reprodução deve ser o suficiente para armazenar alguns segundos<sup>5</sup> da mídia e repassá-la aos seus nós parceiros, requerendo baixa<sup>6</sup> capacidade de armazenamento dos processadores de rede (roteadores, pontos de acesso etc.), incluindo dos nós finais (*desktops*, celulares, *tablets* etc.);

4. o caminho dos fluxos de dados transmitidos por um mesmo servidor da CDN  $s_1$  para um conjunto de nós  $c_1..c_n$ , localizados em redes distintas, são mais previsíveis. Isto possibilita parcerias de melhor qualidade ao levar em consideração que é possível determinar pontos de intersecção das rotas desses fluxos de dados, uma vez que estes convergem para um mesmo servidor  $s_1$ . Nestes casos, é possível manter apenas um fluxo de dados e **replica-lo** no ponto de intersecção. Isto permite o fomento à cooperação e o melhor agrupamento entre os nós.

Estas peculiaridades viabilizam uma abordagem muito importante adotada neste trabalho: a participação mais efetiva dos roteadores no processo de distribuição do conteúdo multimídia. Isto ocorre ao permitir que os roteadores: (i) constituam uma rede de sobreposição P2P; (ii) selezionem nós parceiros (outros roteadores); (iii) sirvam como pontes de acesso aos servidores da CDN; e (iv) ajudem na execução do serviço de controle de congestionamento. Isto porque conjectura-se que os roteadores são elementos de rede estáveis com relação a sua disponibilidade se comparados aos sistemas finais, portanto atenuando o *churn* da rede P2P e assim permitir que as aplicações forneçam serviços mais estáveis aos usuários, mesmo quando um usuário não tem interesse explícito por um determinado conteúdo. Esta decisão também se baseia no fato de que se existir um nó  $c$  interessado por um fluxo de dados  $P$ , transmitido por um servidor  $s$  através de um roteador  $r$ , os pacotes de dados de  $P$  que deverão ser entregues a  $c$  passarão obrigatoriamente por  $r$ . Os resultados de medições realizadas no contexto de trabalhos anteriores, como as apresentadas em [43], corroboram com

<sup>4</sup>Em aplicações de transmissão multimídia ao vivo, estima-se *jitter* no máximo de 180 ms

<sup>5</sup>No máximo, 5 segundos

<sup>6</sup>Utilizando uma compressão MPEG-4, utiliza-se alguns kilobits por segundo até dezenas de megabits por segundo.

tal abordagem ao discutir questões relacionadas à localidade de tráfego e o conhecimento que a rede possui sobre os fluxos de dados sendo transmitido.

De forma mais extensiva, pode-se afirmar que o uso da abordagem supracitada é justificado por uma tendência em utilizar os roteadores para auxiliar e otimizar os serviços das camadas TCP/IP mais acima, como os serviços de controle de congestionamento, NAT dinâmico (*Dynamic Network Address Translation*) através de UPnP (*Universal Plug and Play*) [44], DNS dinâmico [45], dentre outros. Essa tendência é também sustentada pelas Redes Centradas na Informação (*Information-Centric Networks – ICN*) [46–49]. O princípio é que uma rede de comunicação deve viabilizar o acesso ao conteúdo de interesse de um usuário (*content-centric approach*) independente do local físico onde tal conteúdo está armazenado, enfraquecendo a abordagem restrita de organização dos protocolos em camadas funcionais (*host-centric approach*). Em linhas gerais, abstrai-se o esquema de acesso a um recurso pelo endereço IP passando a acessá-lo pelo seu nome, independente da localização do nó gerador do conteúdo de interesse. Isto porque se considera que entregar o conteúdo é mais importante do que a forma de obtê-lo, pois 90 % do tráfego na Internet  consistem de dados disseminados a partir de uma fonte para diversos destinos [50].

O fato é que manter informação de estado sobre a taxa de transmissão por fluxo de dados e definir a nova taxa de transmissão de acordo com eventos de perda não é a melhor abordagem [51–53], prática comum nos algoritmos de controle de congestionamento tradicionais, como os utilizados no TCP. Em vez disso, deve-se utilizar protocolos que permitem a ocupação máxima da capacidade de repasse do roteador ao sinalizar para os fluxos de dados competindo pelo uso do canal qual a taxa de transmissão a ser utilizada, definida de acordo com diversos critérios, tais como a ocupação da fila de roteamento, variação do RTT, etc [54,55]. Assim, reduz-se drasticamente as perdas de dados, porque as filas de roteamento não atingem 100 % da sua capacidade; promove-se o compartilhamento equânime do canal e descarta-se a necessidade de procedimentos de inicialização que subutilizam os canais de transmissão, como a fase de partida lenta do TCP [39,56].

Nesse contexto, baseando-se em resultados de pesquisas publicados recentemente, fazer uso dos algoritmos de controle de congestionamento assistidos pela rede é a melhor abordagem para realizar controle de congestionamento em transmissões de dados na Internet [51]. Sendo assim, convém também utilizar estes algoritmos para distribuição de mídias ao vivo.

No escopo deste trabalho tal abordagem é adotada. Para isto, destaca-se o protocolo *Rate Control Protocol* (RCP) [53,57,58], utilizado de forma adaptada para atender às necessidades do protocolo proposto neste trabalho. Outros protocolos desta mesma classe também deve ser mencionado, como o *eXplicit Control Protocol* (XCP) [52, 59, 60], o *Variable-Structure Congestion Control Protocol* (VCP) [61–65] e o *Congestion Exposure* (ConEx) [66–68]. Apesar de citar o ConEx, este não é considerado um protocolo para controle de congestionamento propriamente dito, mas pode ser útil em auxiliar uma fonte transmissora de um fluxo de dados a expor para a rede informações sobre o congestionamento sofrido pelos pacotes anteriormente transmitidos no contexto de um fluxo de dados (por exemplo, a quantidade de pacotes perdidos). Em seguida, com base nas informações de congestionamento, um algoritmo de congestionamento pode tomar decisões para regular a taxa de transmissão mais apropriada para os sistemas finais.

Apesar dos avanços apresentados nos trabalhos citados anteriormente, tais soluções ainda não foram aplicadas em cenários de distribuição de multimídia que consideram uma arquitetura P2P/CDN, combinadas com a utilização dos roteadores no processo de distribuição de conteúdos multimídia ao vivo. Tal estratégia faz parte do escopo deste trabalho.

Por fim, argumenta-se neste trabalho que ao mover a execução de alguns serviços para o núcleo da rede, ou pelo menos utilizar a rede para auxiliar neste processo, facilita-se o uso de alguns recursos antes difíceis de serem utilizados, tais como transmissões de fluxos de dados em modo *multicast*. No caso do *multicast*, embora esteja disponível há mais de 20 anos, utilizar tal abordagem largamente na Internet não se mostrou uma opção viável e poucas soluções o adotaram largamente na Internet [69, 70]. Isto ocorre devido às barreiras administrativas que dificultam a manipulação de rotas *multicast*, principalmente por conta da dependência do administrador da rede em ter que habilitar explicitamente este tipo de tráfego de dados para um cenário específico de aplicação. Contudo, é possível utilizá-lo quando controlado por um protocolo capaz para habilitá-lo automaticamente de acordo com a demanda das aplicações, onde os fluxos poderão ser compartilhados, mantendo-se o princípio da independência dos serviços de rede, ou seja, sem que a aplicação precise se preocupar como isto é feito.

Com esse norte, delimitou-se algumas métricas que determinam a qualidade de serviço para permitir que os usuários assistam conteúdo ao vivo na Internet com base na referê-

cia [37]. Tais métricas são divididas em três grupos, detalhados a seguir:

1. *Qualidade da experiência do usuário*: avalia-se o tempo em que os nós levam para começar a reproduzir os primeiros datagramas; o tempo que os nós levam para voltar a reproduzir o conteúdo, quando seus nós parceiros se desconectam da rede (índice de continuidade); e a qualidade do conteúdo recebido em comparação ao conteúdo original.
2. *Escalabilidade dos sistemas com relação a quantidade de nós conectados*: avalia-se a quantidade de nós simultâneos conectados, a contribuição da rede CDN e da rede P2P na transmissão de um fluxo de dados, bem como o nível de duplicação desse fluxos.
3. *Execução dos protocolos*: avalia-se a sobrecarga de controle gerados pelos protocolos a fim de executar suas funções.

É no contexto de utilização de conceitos e desenvolvimento de protocolos de redes de computadores, para aumentar a eficiência das transmissões de conteúdos multimídia da classe de aplicações de rede  $1 \rightarrow n$ , que se insere esse trabalho, motivado pelo grande interesse de pesquisa, indústria e mercado em evoluir o estado da arte das soluções para transporte de conteúdos multimídia em larga escala, especialmente na Internet. Para viabilizar isto, deve-se abordar problemas relacionados à comunicação multi-ponto e escalabilidade quanto ao número de nós, utilizando-se de forma eficaz a infra-estrutura de rede.

## 1.2 Descrição do Problema

Nos últimos anos, diversos esforços acadêmicos e da indústria foram feitos para disponibilizar sistemas de distribuição de conteúdo ao vivo na Internet [71–85]. A disseminação de diferentes sistemas e protocolos de rede com este propósito tem levado à pulverização de soluções para transporte de dados, gerando uma falta de padronização na forma como tais sistemas transportam seus dados na Internet, principalmente por tais soluções serem completamente implementadas na camada de aplicação [86–88]. Isto não seria um problema para a infra-estrutura de rede e consequentemente para os sistemas finais se essa pulverização não potencializasse duplicações de fluxos de dados da transmissão de um evento destinado a um

sub-conjunto de redes contendo nós receptores interessados em tal evento [89–97]. Como consequência, exponencia-se o consumo desnecessário de recursos computacionais e de rede.

Para entender o panorama atual nesse contexto, considere o seguinte cenário do sistema de TV tradicional (teledifusão). Quando uma pessoa está assistindo um canal e troca para outro, o receptor de sinal de TV consegue interpretar o novo conteúdo mesmo este sendo transmitido por outra emissora de TV. Para que isto funcione, existem padrões que descrevem o formato do vídeo/áudio (NTSC, PAL-M, TVD etc.), a forma como os sinais devem ser transmitidos, em diferentes frequências e como devem ser decodificados, de acordo com cada emissora de TV, etc. Isto é possível porque todas as TVs funcionam simplesmente porque seguem padrões, independente da sua marca e modelo, caso contrário somente conseguiriam interpretar o sinal transmitido apenas por um conjunto restrito de emissoras de TV.

Considerando a analogia anterior, os nós servidores dos atuais sistemas de distribuição multimídia na Internet são as emissoras de TV, ao passo que as aplicações clientes, os aparelhos de TV. A diferença é que os sistemas não seguem um padrão para transmitir e receber os dados e consequentemente uma aplicação cliente de um sistema não consegue reproduzir o conteúdo transmitido pelo servidor de outro sistema. É como se cada TV funcionasse apenas para um determinado conjunto de canais que seguem um determinado padrão, sendo incapaz de reproduzir o sinal de vídeo oriundo de outras emissoras de TV que funcionam com base em outros padrões estabelecidos ou soluções proprietárias.

Apesar de existirem protocolos para a Internet que descrevem o conteúdo da mídia transmitida [98–100], atualmente não existe um protocolo de transporte de mídia ao vivo que considere toda a complexidade existente em se utilizar as redes orientadas a datagramas, ponderando-se as recentes soluções empregadas nos sistemas mais modernos e estudados neste trabalho. Em vez disso, cada sistema implementa suas próprias formas de transporte de dados, com a execução de ações complexas implementadas na camada de aplicação, com predominância e variações de uso dos protocolos RTP/RTSP combinado com o UDP.

A transmissão de múltiplos fluxos de dados com o mesmo conteúdo por parte de diferentes sistemas, gera um consumo desnecessário de recursos de rede e, principalmente, a impossibilidade de que dois ou mais nós conectados em sistemas distintos possam cooperar entre si, compartilhando o mesmo fluxo de dados transmitido por um servidor, quando há interesse das partes envolvidas. Por exemplo, é comum encontrar na Internet diversos

sistemas de distribuição de conteúdo, cada um com milhares de usuários conectados, porém recebendo fluxos independentes do mesmo jogo de futebol, da corrida de fórmula 1, etc [80,83,101–106]. A duplicação pode ser percebida até em cenários mais simples, quando dois ou mais nós conectados à Internet pela mesma rede estão utilizando diferentes sistemas para assistir ao mesmo evento ao vivo. Idealmente, os nós deveriam compartilhar o mesmo fluxo de dados e cada um renderizar o conteúdo da forma definida pela aplicação, desacoplando a forma de transportar os dados da forma como estes são apresentados pela aplicação (princípio da independência dos serviços) aos seus respectivos usuários, como no caso do serviço Web.

O grande sucesso do serviço Web, no ponto de vista de protocolo de comunicação, ocorre devido à independência da forma que se implementa desacopla o conteúdo a ser transmitido e a comunicação entre o cliente (navegador) e o servidor. Independente do modelo e versão do navegador e do servidor web, todos transportam dados utilizando o protocolo TCP, evitando-se assim a pulverização de diferentes formas de transportar os dados de tal serviço – fica apenas a cargo do desenvolvedor de cada aplicação decidir como exibir as informações recebidas. Contudo, percebe-se a inexistência de um protocolo equivalente ao TCP para os sistemas de distribuição de conteúdos multimídia, apesar da grande demanda e sucesso de utilização dos mais variados sistemas de transmissão multimídia disponíveis na Internet. Um protocolo similar ao TCP para o cenário de distribuição de conteúdos multimídia ao vivo poderia evitar as duplicações de transmissão por meio de estratégias colaborativas entre os nós interessados por um mesmo conteúdo, uma vez que não há a necessidade de individualização dos dados transmitidos. Como consequência, seria possível fazer melhor uso dos canais de transmissão ao enviar fluxos de dados apenas quando necessário, restando para as aplicações a responsabilidade de renderizá-los ao seus usuários, como ocorre no serviço Web. Nesse contexto, a arquitetura de serviço P2P [107] combinada com o *Application Layer Multicast - ALM* [108] é uma abordagem interessante que explora os aspectos de compartilhamento de fluxo de dados, mas os desenvolvedores ainda continuam obrigados a imbutir a implementação das diretrizes do P2P/ALMA nas próprias aplicações, em vez de utilizá-las como os desenvolvedores de navegadores web ou de serviços web fazem: delegar ao TCP algumas importantes responsabilidades. O TCP abstrai toda complexidade de algumas responsabilidades da camada de transporte, tal como garantia de entrega de dados, ordenação

de segmentos, controle de congestionamento etc. Algumas funções implementadas em soluções P2P/ALMA estão consolidadas e bastante utilizadas nos sistemas de distribuição de conteúdos multimídia ao vivo, cada um a sua maneira e, mesmo que sejam similares, ainda são incompatíveis.

Como a visão supracitada atualmente (HTTP + TCP) não é empregada na Internet para o caso de distribuição de conteúdos multimídia ao vivo, em essência devido às limitações dos protocolos de transporte que não oferecem recursos para auxiliar a execução dos sistemas de distribuição de conteúdos multimídia ao vivo, passou-se a utilizar arquiteturas de sistemas baseadas na combinação P2P/CDN + UDP. Nesse contexto e considerando as diversas soluções disponíveis, os desenvolvedores fazem uso de *middlewares* que implementam as principais funções utilizadas nos sistemas de distribuição de conteúdos multimídia ao vivo. Devido a existência de diferentes opções, cada sistema faz uso de um *middleware* específico (quando o fazem), resultando novamente na pulverização dos sistemas devido a incompatibilidade entre os *middlewares* utilizados.

Em geral, nos middlewares para distribuição de conteúdos multimídia ao vivo, implementam-se mecanismos para conexão multi-ponto e topologias [12, 23, 97, 109–113], seleção de nós [114–118], tolerância à desconexões e de outros problemas causados pelo *churn* [42, 109, 113, 119, 120], disponibilização de informação de contexto sobre rede para dar suporte à execução dos serviços das aplicações, como localização da rede, custos entre redes em uma granularidade configurável e outras propriedades dos sistemas finais [114, 121–123], estratégias para incentivos à cooperação entre nós [118, 124–129], algoritmos para inibir a participação de nós *free-riders* [127, 130–132], adaptação de fluxo baseado na capacidade de recepção dos nós [133–136] e segurança [129, 137–140]. Temas como estes foram ganhando importância ao longo dos anos porque os serviços de transmissão de mídias ao vivo foi se tornando uma tendência e hoje é uma realidade. Apesar da arquitetura da Internet ser modularizada em camadas funcionais, ao longo dos anos as soluções promovidas no contexto dos referidos temas não foram disponibilizadas estrategicamente nas camadas corretas, mas sim apenas na camada de aplicação. Isto se explica pelo fato de que é muito mais fácil implementar uma solução na camada de aplicação do que em outras camadas mais abaixo, não apenas por aspecto de facilidade de desenvolvimento, mas também de implantação em larga escala no mercado, pois todas as outras camadas estão implementadas do sistema operacional para

baixo.

De forma alternativa, uma maneira eficiente de se estruturar e enviar um fluxo de vídeo a um grupo de usuários na Internet seria a utilização de *multicast* no nível de IP [96]. Em uma sessão de *multicast* IP, uma estrutura de árvore é formada. A fonte de vídeo se torna a raiz desta árvore *multicast* e os clientes recebem o fluxo de vídeo através dos vários nós desta árvore, formada pelos roteadores que suportam *multicast* em nível de IP. Apesar de melhorar a escalabilidade no número de receptores do mesmo conteúdo ao vivo, tal estratégia não é suficiente devido a complexidade de habilitar, principalmente em grande escala, os canais de comunicação *multicast*, pois, para isso, precisa-se de intervenção humana. Para contornar a dificuldade em habilitar o modo *multicast* em nível de IP, a arquitetura ALM, equivalente ao IP *multicast*, porém implementada a nível da camada de aplicação. A diferença entre o *multicast* IP e o ALM é que no primeiro os roteadores duplicam e repassam os datagramas para os próximos roteadores até alcançar os sistemas finais, ao passo que no segundo os sistemas finais executam as ações citadas anteriormente, mas ainda dependem de funções de rede de mais baixo nível, o que pode aumentar o tempo de processamento e a complexidade da aplicação. Existem diversos *middlewares* baseados em ALM e estes implementam suas próprias interfaces e uso e funcionalidades, incompatíveis entre si [85, 141–147]. Apesar de todos esses esforços, questão da pulverização de aplicações mencionada no ínicio dessa seção vem à tona mais uma vez: cada sistema adota suas estratégias de formação da árvore *multicast*, portanto incompatíveis entre si.

Em vez disso, pode-se combinar as duas formas supracitadas em uma solução de protocolo disponível para qualquer aplicação de rede interessada em transmitir e/ou receber um fluxo de dados ao vivo. Ou seja, um protocolo que constitui uma rede de sobreposição formada por roteadores de rede, com suporte automático ao uso de *multicast*, através da qual as aplicações possam trocar dados entre si de forma transparente. Isto significa mover alguns serviços consolidados na camada de aplicação para camadas mais abaixo, abstraindo-se a grande complexidade de se constituir redes P2P, principalmente compatíveis entre si. Nas Seções 2.2 e 2.3, estende-se essa discussão no sentido de entender o funcionamento e os aspectos arquiteturais das aplicações de distribuição de conteúdos multimídia ao vivo, em redes P2P, bem como os principais problemas com base em alguns cenários de uso.

Por exemplo, os protocolos de transporte poderiam absorver parte da complexidade dos



serviços de distribuição multimídia, mas não os fazem porque tanto o UDP quanto o TCP não foram projetados para atender as necessidades desse tipo de serviço e alterá-los requer um grande esforço, principalmente por serem de domínio público e utilizados abundantemente por serviços com esta finalidade. Este fato acontece primeiro por um motivo histórico, pois na época que tais protocolos foram propostos não se discutia sobre este tipo de serviço, tampouco sobre sistemas P2P e muito menos sobre redes CDN. Além disso, um outro motivo é o fato de que os sistemas foram sendo disponibilizados levando em consideração toda a complexidade na formação das redes P2P, uma vez que os protocolos de transporte não eram eficazes para a distribuição de datagramas IP em larga escala na Internet, cuja principal forma era utilizar *multicast* (seja na camada de rede ou na de aplicação).

Além do TCP e do UDP, na segunda metade dos anos 2000, outros protocolos mais modernos passaram a apresentar avanços significativos para permitir a transmissão de conteúdos multimídia, como é o caso do DCCP (*Datagram Congestion Control Protocol*) [148, 149]. Apesar de sua eficiência em alguns cenários de transmissão de dados multimídia na Internet, o DCCP possui falhas críticas quando utilizado em larga escala em cenários  $1 \rightarrow n$  [150–152]. O fato é que o DCCP é um protocolo orientado à conexão e portanto para cada novo usuário interessado em receber um fluxo multimídia transmitido, uma nova conexão se faz necessária, não havendo a possibilidade de compartilhamento entre diferentes nós interessados pelo mesmo conteúdo, pelo menos de forma automática e transparente para as aplicações de rede. Qualquer protocolo que exija estabelecimento de conexão e realize controle de congestionamento por fluxo de dados apresenta tal limitação, o que gera dois problemas:

1. *excessivo consumo de recursos computacionais*: para cada nova conexão, o nó transmissor deve alocar recursos computacionais (memória e processamento) para tratar cada nova conexão. Em cenários de transmissão multimídia  $1 \rightarrow n$ , se muitos nós estão conectados em um único servidor, então isto elevará sobremaneira o consumo de recursos computacionais do nó transmissor proporcionalmente à quantidade de nós receptores interessados pelo fluxo multimídia transmitido. Além disso, embora o conteúdo transmitido por um nó seja de interesse de muitos outros nós, os fluxos são mantidos independentemente uns dos outros, o que gera duplicações desnecessárias e consequentemente desperdício de recursos de rede. O trabalho mais notável onde

discute-se as consequências deste problema em aplicações de rede é o encontrado na referência [96].

2. *a taxa de transmissão de fluxos DCCP individualmente tenderá a 0 (zero)*: no caso do DCCP, utiliza-se uma equação matemática para definir a taxa de transmissão durante uma conexão. À medida que mais nós se conectam a um nó transmissor, menor será a taxa de transmissão do nó transmissor para cada um dos nós receptores conectados a ele. Para a rede, esta estratégia é equânime e evita que a mesma entre em colapso de congestionamento, mas para cada fluxo de dados isto é ruim. Este problema tem uma relação estreita com o dilema observado por Garrett Hardin em 1968 e denominado de Tragédia dos Bens Comuns (*Tragedy of the Commons*) [153], presente em diferentes áreas do conhecimento. No caso de protocolos como o DCCP, a tragédia dos bens comuns ocorre porque, à medida que novos fluxos são transmitidos na rede, menor é a taxa de transmissão alocada individual para cada fluxo, a qual pode se tornar insuficiente para a recepção de um fluxo multimídia e, por consequência, nenhum nó receptor reproduzirá o fluxo transmitido pelo nó transmissor, embora todos os fluxos utilizarão o canal de forma equânime.

Desta forma, apesar de os algoritmos de controle de congestionamento buscarem convergir para o caso do melhor global (equidade para com todos os fluxos e assim evitar congestionamento da rede), isto provoca o efeito do caso do pior local (redução da taxa de recepção de cada nó da rede). Este fato pode ser explicado analiticamente utilizando como base a Equação 1.1, que define cada taxa de transmissão  $X_i$  para realizar o controle de congestionamento em cada fluxo [154]. Para efeito de estudo, esta equação do protocolo DCCP/CCID3 foi a escolhida, por ser utilizada em vários protocolos de rede, porém equações similares de outros algoritmos para controle de congestionamento poderiam ter sido utilizadas. Nesta equação, define-se:

- $X_i$  é a taxa de transmissão em bytes/segundo;
- $s$  é o tamanho do pacote em bytes;
- $R$  é o RTT (*Round Trip Time*) em segundos;
- $p$  é a taxa de ocorrência de perdas, entre 0 e 1;

- $RTO$  (*Retransmission TimeOut*) é o valor do temporizador de retransmissão do TCP em segundos; e
- $b$  é igual a 1 e representa o número máximo de pacotes confirmados por um único ACK.

Considerando o problema descrito anteriormente, o uso total da capacidade do canal por  $N$  fluxos DCCP pode ser definido por  $C = \sum_{i=1}^N X_i$ . Em condições severas de congestionamento na rede, o valor de  $C$  é equivalente à largura de banda do canal de transmissão. Quando isto ocorre, tem-se que  $N$  atingiu um valor maior do que a rede suporta, fazendo com que os *buffers* de recepção dos roteadores alcancem seus limites e portanto os valores de  $p$  e  $R$  na Equação 1.1 também aumentam, resultando que o  $\lim_{N \rightarrow \infty} \frac{C}{N} = 0$ , logo,  $X_i$  se aproxima de 0 (zero).

$$X_i = \frac{s}{R \times \sqrt{2 \times b \times \frac{p}{3}} + (RTO \times 3\sqrt{3 \times b \times \frac{p}{8}} \times p \times (1 + 32 \times p^2))} \quad (1.1)$$

Embora esta seja uma discussão teórica, simulações foram realizadas em busca de evidências mais contundentes de que este fato pode ocorrer na prática. Os resultados e discussão sobre estas simulações são apresentados em [150, 151, 155]. Em resumo, de acordo com os resultados obtidos, apenas fluxos DCCP foram suficientes para causar o problema em questão, sequer fluxos de outros protocolos foram utilizados. Em situações mais realistas, o problema se agrava porque o protocolo DCCP disputará o canal de transmissão não apenas com fluxos DCCP, mas também com fluxos de protocolos como o TCP, UDP, SCTP [156, 157], dentre outros.

Apesar de o protocolo DCCP ter sido utilizado para evidenciar o problema que acabara de ser apresentado, esta discussão pode ser generalizada para qualquer outro protocolo de rede que seja orientado à conexão e que realize controle de congestionamento de fluxos individualmente. A fim de aumentar a representatividade de protocolos nesse contexto, o leitor pode consultar outros trabalhos referenciados em [158–177] e no Capítulo 3 deste documento.

Para contornar os problemas discutidos até aqui, torna-se imprescindível projetar um protocolo de rede com vistas a reduzir a pulverização dos sistemas de distribuição de conteúdo,

inspirando-se em casos consistentes ao longo dos anos, como o serviço Web, e que considere a evolução do estado da arte como aliado nesse processo, tais como as funções consolidadas das redes P2P, os novos conceitos das redes centradas no conteúdo e os algoritmos de controle de congestionamento assistidos pela rede, que são mais eficazes se comparados aos algoritmos baseados em perda de pacotes, mudança no atraso e relatórios enviados pelos receptores aos transmissores [178].

Diante do exposto, pretende-se responder a seguinte questão de pesquisa: *como disseminar conteúdos multimídia ao vivo na Internet evitando o fenômeno da tragédia dos bens comuns dos protocolos de transporte, considerando o princípio da independência dos serviços de rede?*

### 1.3 Hipótese

Para o problema em questão é adotada a hipótese de que *a constituição de uma rede de favores entre roteadores que compartilham fluxos de dados transmitidos por servidores de uma CDN, através do uso transparente (para a aplicação) de multicast negociado no processo de estabelecimento de conexão entre os nós transmissores e receptores, com o auxílio de um algoritmo de controle de congestionamento assistido pela rede, possibilita a disseminação em larga escala de conteúdos multimídia ao vivo.*

Diante desta hipótese, remete-se esta pesquisa ao uso de técnicas e mudanças na perspectiva da execução de serviços em camadas inferiores a da aplicação, visto que argumenta-se neste trabalho que a infra-estrutura de rede deve fornecer um suporte mais efetivo no processo de disseminação de conteúdos multimídia ao vivo. Propõe-se que isto ocorra evitando-se que os desenvolvedores de aplicações implementem funcionalidades intrínsecas ao processo de distribuição de conteúdos na rede, com escalabilidade do número de nós e autenticidade do conteúdo distribuído através da rede de favores.

### 1.4 Objetivo

Neste trabalho, o objetivo é a concepção e a avaliação de um protocolo de rede denominado *Global Media Transmission Protocol* (GMTP), para uso em sistemas que consideram uma

arquitetura híbrida P2P e CDN para distribuição de conteúdos multimídia ao vivo. Mais especificamente, propõe-se um protocolo que explora a relação entre o escalonamento de recursos computacionais em redes P2P, a estabilidade das redes CDN e os mecanismos que possam mitigar a complexidade no processo de construção desses sistemas ao mover diversas funções atualmente implementadas nesses sistemas para as camadas de transporte e rede, a fim de utilizar eficientemente os recursos de rede e consequentemente melhorar a qualidade da experiência do usuário ao reproduzir um conteúdo ao vivo na Internet.

### 1.4.1 Objetivos Específicos

Pode-se dividir o objetivo principal deste trabalho nos seguintes objetivos específicos:

1. compreender os sistemas e protocolos para distribuição de conteúdos multimídia baseados em arquiteturas P2P/CDN, com suporte a controle de congestionamento e transmissão *multicast* na camada de aplicação. Além disso, entender quais outras funções estão presentes nesse tipo de sistema;
2. definir um protocolo de rede que opera nas camadas de transporte e rede da pilha TCP/IP para distribuição de conteúdos multimídia. O protocolo deve permitir que os sistemas finais sejam capazes de transmitir e receber dados considerando o uso de mecanismos consolidados como boas práticas em sistemas de distribuição de conteúdos ao vivo na Internet, bem como fazer uso de novos algoritmos propostos no contexto deste trabalho;
3. implementar o protocolo proposto em um ambiente que proporcione a obtenção de valores para as métricas avaliadas, permitindo-se estudar o protocolo por meio de diferentes configurações de rede, entender seus limites e os impactos que seus recursos podem gerar tanto sobre os nós quanto sobre a rede; e
4. avaliar e discutir os métodos e técnicas empregados no protocolo proposto em comparação a um sistema disponível no estado da arte, discutindo-se sobre às melhorias na qualidade da experiência do usuário, suas vantagens e desvantagens;

## 1.5 Relevância do Tema e da Tese

Transporte de fluxos de dados multimídia ao vivo para distribuição de conteúdos multimídia é um tema relevante no contexto de redes de computadores devido aos recentes interesses dos usuários por parte desse tipo de serviço, em particular, na Internet, como discutido na Seção 1.1.

Como propõe-se neste trabalho, combinar tópicos específicos do tema estudado em uma solução para melhorar a experiência do usuário ao reproduzir um conteúdo ao vivo é um grande desafio, principalmente devido ao surgimento de novas técnicas, métodos e/ou paradigmas, como as redes centrada no conteúdo e os algoritmos para controle de congestionamento assistidos pela rede. A distribuição de conteúdos multimídia em larga escala é cada vez mais relevante devido às características inerentes à classe de aplicações e cenários que têm sido considerados na Internet, tais como seleção de nós, tolerância à desconexão, segurança e controle de congestionamento no envio de datagramas. Estas características, aliadas à transparência da solução na perspectiva do desenvolvedor da aplicação, tornam o tema ainda mais relevante para o contexto de boas práticas na forma de como os serviços de distribuição de conteúdos multimídia são implementados atualmente.

Nesse contexto, apesar de as soluções existentes resolverem parte do problema em discussão, principalmente quando se utiliza redes de distribuição de conteúdo, argumenta-se que ainda é possível obter melhores resultados quanto ao desempenho dos sistemas com a finalidade que se discute neste trabalho, não somente na perspectiva do consumo de recursos de rede, mas também do estado da prática no que diz respeito à forma que tais sistemas são desenvolvidos. Como a demanda por serviços multimídia em redes de computadores tem aumentado dia após dia, o estudo desenvolvido e os artefatos de software produzidos no contexto desta tese podem contribuir sobremaneira para o desenvolvimento de aplicações multimídia mais eficientes e compatíveis entre si, tal como ocorre no serviço web, além de ajudar na tomada de decisões sobre futuros desenvolvimentos desse tipo de pesquisa. Isto é possível através das contribuições específicas desenvolvidas no contexto deste trabalho, summarizadas na Seção 1.6.

No que diz respeito à relevância do trabalho na perspectiva de engenharia de software para sistemas distribuídos, o autor considera indispensáveis três principais requisitos que

estão sendo contemplados e que reforçam a relevância da tese. Estes requisitos servem como motivação para a realização das atividades que foram desenvolvidas ao longo deste trabalho.

O primeiro deles é a consistência teórica. O protocolo de rede proposto foi concebido a partir de evidências sólidas com base nos trabalhos anteriormente publicados e em simulações de rede, com o problema-chave apresentado e discutido por meio de fundamentos matemáticos e provas contundentes obtidas com o uso de um consagrado simulador de rede. Propõe-se a descrição do protocolo de forma rigorosa e não-ambígua, permitindo um melhor entendimento e futuros investimentos no protocolo teórico proposto.

O segundo requisito é a contribuição científica. Diversos trabalhos relacionados foram estudados antes da concepção do protocolo proposto. A partir deste estudo, identificou-se o problema anunciado anteriormente e foram elencadas as possíveis soluções para o problema, o que culminou com a definição deste trabalho. Até o momento da escrita deste documento, não foram encontrados trabalhos com as características aqui propostas, o que reforça o caráter de originalidade e contribuição científica, a qual já vem sendo respaldada pela comunidade através da publicação de artigos em veículos relevantes da área e pelo interesse industrial, como apresentado na Seção 1.7.

O terceiro requisito é o potencial prático. A implementação do protocolo de rede, assim como o conjunto de ferramentas desenvolvidas e que serão mantidas, tem como objetivo demonstrar que a abordagem é viável e praticável. Um protocolo de rede simplesmente especificado sem nenhuma implementação real tornaria as reais contribuições deste trabalho apenas suposições. O compromisso com a utilização dos conceitos para construir soluções que possam ser aplicadas na indústria, torna o trabalho relevante em termos práticos, sobretudo em escala global na Internet.

## 1.6 Resumo das Contribuições

No contexto desta tese, enumera-se as grandes contribuições de forma resumida.

1. a concepção e descrição detalhada do protocolo introduzido. Neste caso, apresenta-se a elaboração do projeto de um protocolo para distribuição de conteúdos multimídia ao vivo, que considera a comunicação mais efetiva entre as duas camadas mais importantes da pilha TCP/IP, a de transporte e a de rede. Além disso, traz-se à tona os recentes

avanços da área de estudo, testando-se o uso dos novos conceitos das redes centradas no conteúdo aplicados ao processo de distribuição de conteúdos e combinados com os recentes avanços dos algoritmos de controle de congestionamento assistido pela rede. O uso do protocolo introduzido contribuirá para a padronização da forma como os desenvolvedores implementam as principais funcionalidades dessas aplicações e, sobretudo, permitindo que estas sejam implementadas de tal forma a promover o reúso em soluções existentes, utilizando-se dos recursos de rede de forma mais eficiente.

2. a implementação de artefatos de software que permitem a reprodução do ambiente de estudo, tais como implementações no simulador de rede OMNet++ e NS2 [179], bem como um protótipo de implementação no núcleo do sistema operacional Linux;
3. os resultados e discussões aprofundadas acerca do protocolo introduzido frente ao CoolStreaming/Denacast [36] e a arquitetura de rede NDN [50]. O primeiro é um sistema de distribuição de mídias ao vivo baseado no CoolStreaming com suporte a uma rede de distribuição de conteúdos (P2P/CDN), ao passo que o segundo é um novo modelo de serviço onde o acesso aos conteúdos ocorre através dos seus nomes e não mais através de sua localização (endereço IP); e
4. os encaminhamentos da pesquisa sobre futuros avanços do estado da arte, tais como as limitações do protocolo proposto e uma breve discussão de como estas poderão ser aprimoradas.

Ademais, o autor destaca a importância do presente trabalho por ser o primeiro a trazer à tona um problema e uma proposta de solução do uso de um protocolo de transporte e rede exclusivamente projetado para a distribuição de conteúdos multimídia para a Internet, antes realizada apenas com os protocolos TCP, UDP, DCCP ou outros particulares, em geral, disponíveis na camada de aplicação.

## 1.7 Publicações

1. *GMTP: A Crossing-layer Optimized Protocol for Large Scale Distribution of Live Multimedia Content over the Internet.* A ser submetido para IEEE Transactions on Broadcasting (JCR 2.087) ou IEEE Network (JCR 2.853).

2. *Designing GMTP: The Global Media Transmission Protocol for Massive Multimedia Distribution over the Internet.* A ser submetido para IEEE Software (JCR 1.62).
3. *Generalized Connection and Incentives for Supporting CE Devices in P2P/CDN Live Streaming Systems.* IEEE Transaction on Consumers Eletronics (JCR 1.087).
4. *About Encouraging Residential Users to Share Upload Bandwidth with CDN/P2P Live Streaming Systems.* 31st IEEE International Conference on Consumer Electronics 2013. Qualis A-2 (Computação). 2013. Referência [180].
5. *Multi(Unicast) DCCP for Live Content Distribution with P2P Support.* 9th IEEE Wireless Communications and Networking Conference (WCNC 2012). Qualis A-1 (Computação). 2012. Referência [150].
6. *Multimedia Content Distribution of Real Time Controlled and Non-reliable Datagrams Between Peers.* 29th IEEE Globecom 2011. 2nd IEEE Workshop on Multimedia Communications & Services. Qualis A-1 (Computação). 2011. Referência [151].
7. *Distribuição de Conteúdo Multimídia em Tempo Real com Transporte de Fluxos Controlados e Não Confiáveis entre Pares.* Simpósio Brasileiro de Redes de Computadores 2011. VII Workshop de Redes Dinâmicas e Sistemas P2P. 2011. Qualis A-N (Computação). Referência [155].
8. *On the Performance of TCP, UDP and DCCP over 802.11g Networks.* 23rd ACM Symposium on Applied Computing. Qualis A-1 (Computação). 2008. Referência [181].

## 1.8 Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

- No Capítulo 2, apresentam-se os principais conceitos relacionados aos sistemas de distribuição de mídias ao vivo em arquiteturas P2P.

- No Capítulo 3, apresentam-se os trabalhos relacionados a esta pesquisa, destacando-se os principais avanços científicos no que diz respeito aos protocolos e sistemas de distribuição de mídias ao vivo.
- No Capítulo 4, apresenta-se uma visão geral do *Global Media Transmission Protocol* (GMTP).
- No Capítulo ??, apresentam-se os resultados sobre o desempenho do protocolo GMTP quando confrontado com duas outras proeminentes soluções disponíveis na literatura.
- Por fim, no Capítulo ??, apresentam-se as considerações finais, discutindo-se os principais tópicos elencados neste trabalho, conclusões e trabalhos futuros.

# Capítulo 2

## Fundamentação

A concepção de protocolos de rede para sistemas de distribuição de conteúdos multimídia em tempo real trás à tona uma série de conceitos necessários para o entendimento da solução proposta neste trabalho.

Neste capítulo, apresentam-se os protocolos existentes para gerenciamento e sincronização de mídias ao vivo na Internet, bem como as principais arquiteturas e modelos de serviços empregados nos sistemas de distribuição de mídia ao vivo. Para finalizar, discute-se brevemente sobre conceitos básicos de criptografia e assinatura digital. Caso o leitor se considera familiarizado com esses assuntos, recomenda-se, pelo menos, a leitura do sumário deste capítulo, disponível na Seção 2.5. Caso necessário, sugere-se as seguintes referências complementares:

- [107, 182, 183], para conceitos básicos sobre comunicação e redes de computadores, modos de transmissão, redes de distribuição de conteúdo e protocolos de rede;
- [152], para funcionamento de protocolos de transporte mais modernos, como o DCCP e o SCTP. Além disso, discussões sobre a aplicação do DCCP em cenários reais de transmissão de mídias;
- [48, 50, 68, 184, 185], para conceitos sobre Redes Centradas no Conteúdo; e
- [96, 137], para mais detalhes sobre os conceitos apresentados neste capítulo.

## 2.1 Protocolos para Gerenciamento e Sincronização de Mídias Ao Vivo

### 2.1.1 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 [186], 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 [70] há algum tempo, não havia ainda qualquer padrão aberto para controle de conferências multimídia e utilizar o Mbone exigia muita intervenção humana e empresarial [69].

No período equivalente à 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*) [187], 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*) [99], 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) [100]. 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.

## 2.2 Estruturas para Distribuição de Mídia ao Vivo em P2P

Nesta seção, apresenta-se uma revisão dos esforços prévios no sentido de estruturar e organizar os sistemas de transmissão ao vivo em arquiteturas P2P.

A transmissão de vídeos na Internet pode se classificar em duas grandes categorias: vídeos pré-armazenados, enviados sob demanda (*on-demand*) e vídeos ao vivo (*live*). Os usuários de vídeos assistidos sob demanda têm a flexibilidade de assistir um conteúdo previa-

mente armazenado, a qualquer momento. Por outro lado, um conteúdo ao vivo é transmitido no mesmo momento em que o fluxo é gerado. Logo, todos os usuários devem estar sincronizados e devem assistir o fluxo de vídeo ao mesmo tempo. Isto significa que esses sistemas têm requisitos críticos de tempo e a retransmissão baseada no controle de erro não é adequado em cenários de altos atrasos. Por isso, tais sistemas utilizam suas próprias funções para tolerar perda de pacotes, como mascarar erros e adaptação de conteúdo, em transmissões de fluxos de dados sem garantia de entrega, utilizando UDP.

A solução básica para o envio do fluxo de vídeo na Internet é a utilização do modelo cliente-servidor. Nesse modelo, um cliente cria uma conexão com um servidor de vídeo e o conteúdo é enviado para o cliente diretamente do servidor. Existem algumas variantes deste modelo, mas as soluções baseadas em cliente-servidor demandam uma larga capacidade de transmissão nos canais de comunicação utilizados pelo servidor, o que gera um alto custo operacional [96].

Recentemente, vários sistemas P2P foram desenvolvidos para prover conteúdo de vídeo ao vivo e sob-demanda na Internet, com baixo custo operacional [83, 85, 102, 103, 188–192]. As redes entre pares (P2P) emergiram como um novo paradigma para construir aplicações distribuídas. Neste tipo de aplicação, os usuários são encorajados a atuarem como clientes e servidores. Em uma rede P2P, os nós participantes, além de obterem serviços da rede, também os provêem. Assim, a banda de rede dos usuários finais é utilizada para reduzir a grande demanda por banda de rede, outrora necessária aos servidores.

Os sistemas de envio de vídeo que utilizam arquitetura P2P podem ser classificados em duas categorias quanto a sua estrutura: os baseados em uma estrutura de árvore ou em malha. As seções a seguir são dedicadas a descrever funcionamento de cada uma dessas estruturas.

### 2.2.1 Estrutura Baseada em Árvore

Os sistemas baseados em árvore têm uma estrutura sobreposta bem organizada e, tipicamente, distribuem o fluxo de vídeo enviando dos nós para seus filhos. Um dos maiores problemas desta abordagem é que são vulneráveis à entrada e abandono dos nós participantes da rede (*churns*) [193, 194]. Assim, quando um participante deixa a rede, a estrutura de árvore se rompe, e parte do sistema sofre, temporariamente, uma ruptura no fluxo do vídeo.

Uma maneira eficiente de se estruturar e enviar um fluxo de vídeo a um grupo de usuários

na Internet seria a utilização de *multicast* no nível de IP [96]. Em uma sessão de *multicast* IP uma estrutura de árvore é formada. A fonte de vídeo se torna a raiz desta árvore *multicast*, e os clientes recebem o fluxo de vídeo através dos vários nós desta árvore, formado pelos roteadores que suportam o *multicast* em nível de IP. Para contornar a falta de suporte de *multicast* em nível de IP, a função equivalente tem sido implementada no nível da camada de aplicação. Os servidores de vídeo e os usuários formam uma rede sobreposta à rede real e, assim, organizam-se para distribuir o fluxo de vídeo. De maneira similar ao *multicast* IP, formado por uma árvore de roteadores no nível de rede, os nós participantes da sessão de vídeo formam uma árvore na camada de aplicação, cuja origem é o servidor de vídeo.

Cada nó do sistema se conecta à árvore em um certo nível. Em seguida, cada nó recebe o vídeo de seus pais, no nível superior, e reenvia o conteúdo aos seus filhos, no nível mais baixo. Algumas aplicações, como Overcast [145], utilizam esta abordagem. Na Figura 2.1, ilustra-se um sistema com quinze nós participantes.

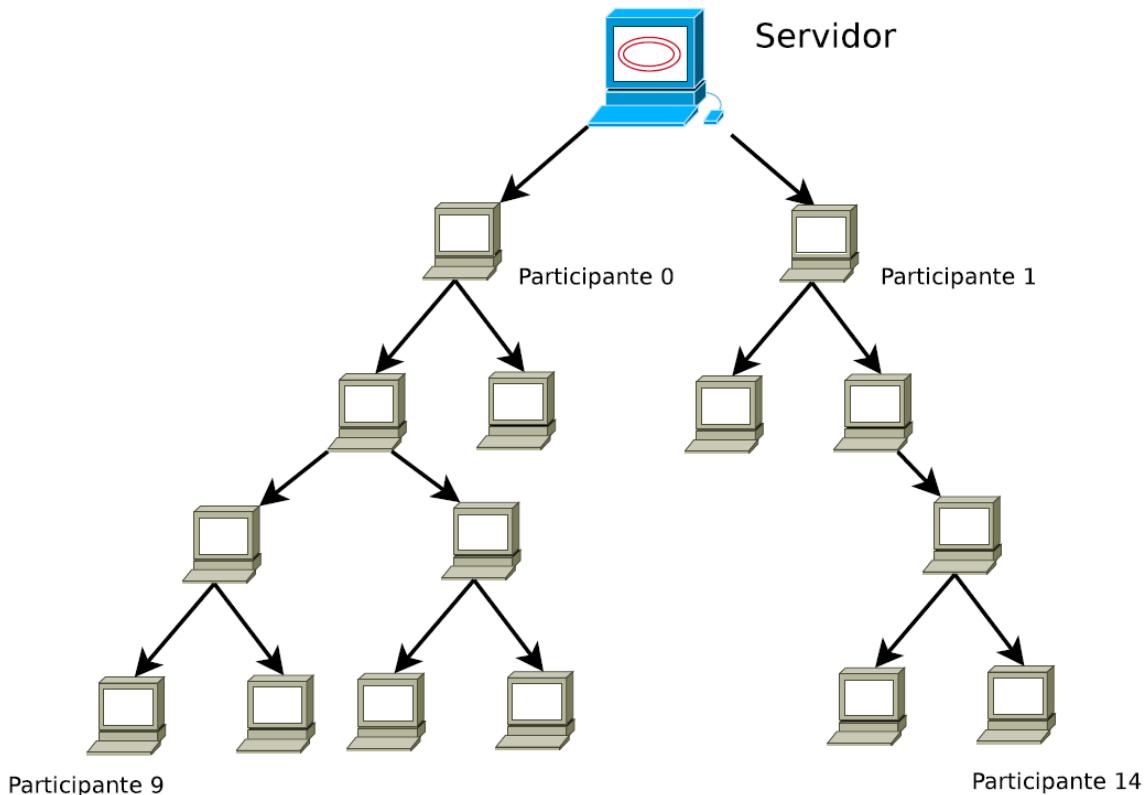


Figura 2.1: Árvore de *multicast* em nível da camada de aplicação.

Existem várias maneiras possíveis de se construir a árvore para o envio de fluxo de vídeo. Deve-se considerar a altura da árvore e a quantidade de filhos de cada nó da árvore. Os nós

em níveis inferiores da árvore recebem o fluxo de vídeo após tal fluxo percorrer vários outros nós, e isto pode induzir a grandes latências. Para reduzir esse problema, deve-se preferir uma árvore com o mínimo de níveis possível, o que pode requerer a participação de nós com grande largura de banda, retransmitindo para vários filhos.

Tão importante quanto a construção da árvore é a manutenção da sua estrutura. Os usuários de uma aplicação de vídeo em sistemas P2P podem ser muito dinâmicos, entrando e deixando a rede de forma muito imprevisível. Quando um nó abandona a aplicação de transmissão de fluxo contínuo em P2P, interrompe-se a transmissão e todos os seus descendentes ficam sem uma fonte do fluxo de vídeo. Para reduzir essas interrupções, a árvore de envio de fluxo de vídeo deve ser reconstruída o mais rapidamente possível. Na Figura 2.2, ilustra-se um cenário em que um nó deixa o sistema de vídeo e a árvore de *multicast* ao nível de aplicação, que deve ser reconstruída.

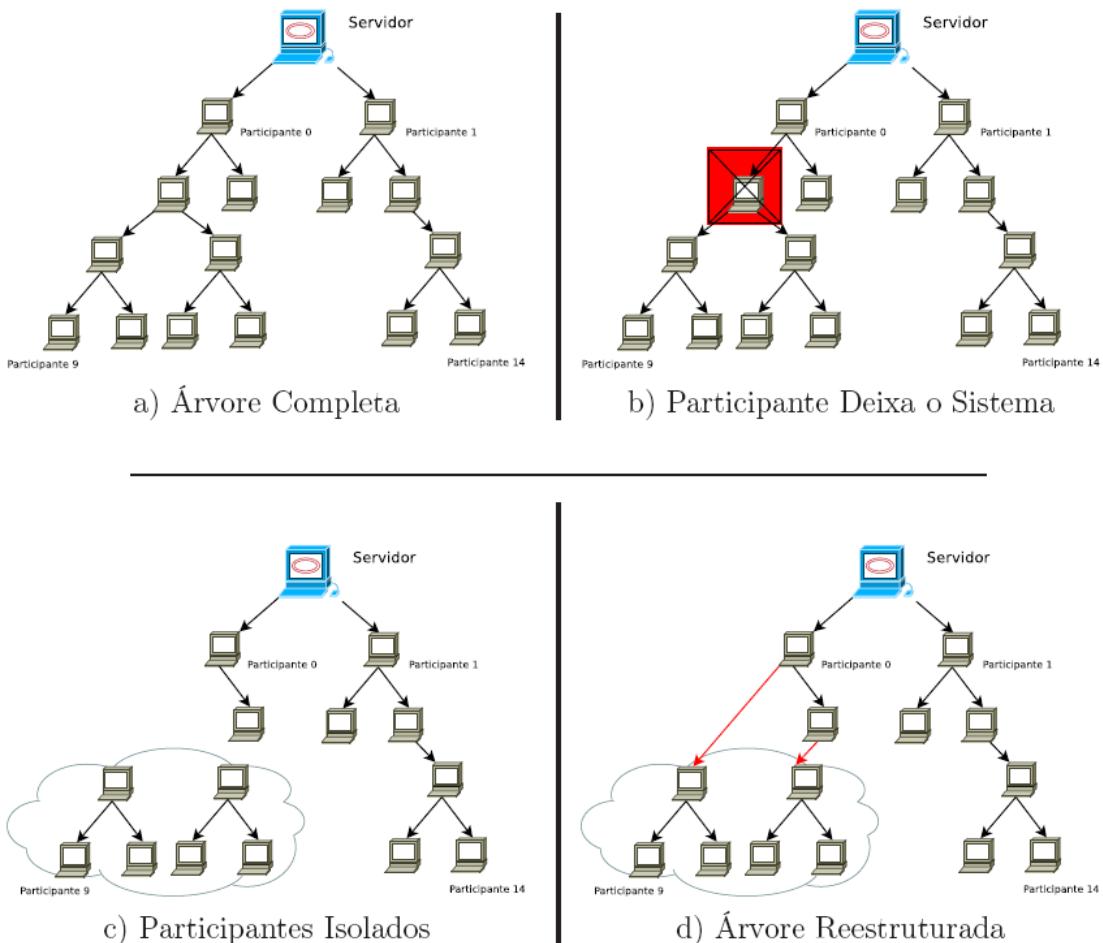


Figura 2.2: Manutenção da árvore de *multicast* em nível da camada de aplicação.

A construção e manutenção da árvore de envio de fluxo P2P pode ser realizada de maneira centralizada ou descentralizada. Em uma abordagem centralizada, um servidor controla a construção da árvore e sua recuperação. Para grandes sistemas de envio de vídeo, uma abordagem centralizada pode se tornar um gargalo e um ponto de falha [96]. Vários algoritmos distribuídos abordam e tratam o problema de manutenção e construção da árvore de maneira distribuída [106]. Mesmo assim, uma abordagem baseada em árvore não consegue se recuperar de maneira rápida o suficiente para lidar com a dinâmica dos nós participantes, pois a constante interrupção do fluxo e a reconstrução da árvore de envio de fluxo contínuo podem causar uma sensação de baixa qualidade no serviço oferecido [96, 193, 194].

Outro problema encontrado ao se usar uma árvore simples é que os nós, que estão na folha da árvore, acabam por não contribuir com o sistema. Assim a utilização de banda não é totalmente aproveitada. Uma vez que existe um grande número de nós folhas, a capacidade da árvore se torna subestimada. Para lidar com esse problema, foram propostas abordagens baseadas em múltiplas árvores como em [85]. Nesta abordagem, um servidor divide o fluxo de vídeo em vários subfluxos e para cada um destes, uma árvore *multicast* ao nível de aplicação é construída. Cada participante deve se conectar a todas as árvores criadas, para obter um fluxo de vídeo completo. Preferencialmente, os nós participantes se conectam em lugares diferentes nos vários níveis existentes. Assim, os nós folhas de uma árvore podem se tornar nós internos em outra, fazendo melhor uso da capacidade disponível. A Figura 2.3 ilustra uma aplicação de envio de fluxo de vídeo com duas árvores.

### 2.2.2 Estrutura Baseada em Malha

Em uma estrutura baseada em malha (*mesh-based*), os nós participantes não se organizam em uma topologia estática. As relações são estabelecidas baseando-se nos recursos disponíveis momentaneamente. Um nó participante se conecta a um subconjunto de outros nós participantes do sistema e, periodicamente, todos trocam informações entre si. Os dados são buscados nos nós participantes que já os têm. Como um nó participante tem múltiplos vizinhos ao mesmo tempo, a organização em malha é robusta à dinâmica dos nós. Entretanto, essa relação dinâmica faz com que a distribuição de vídeo se torne imprevisível.

Diversos trabalhos recentes na área de fluxo contínuo P2P adotam uma estrutura baseada em malha [82, 103, 192, 195]. Em um sistema desse tipo não existe uma topologia fixa da

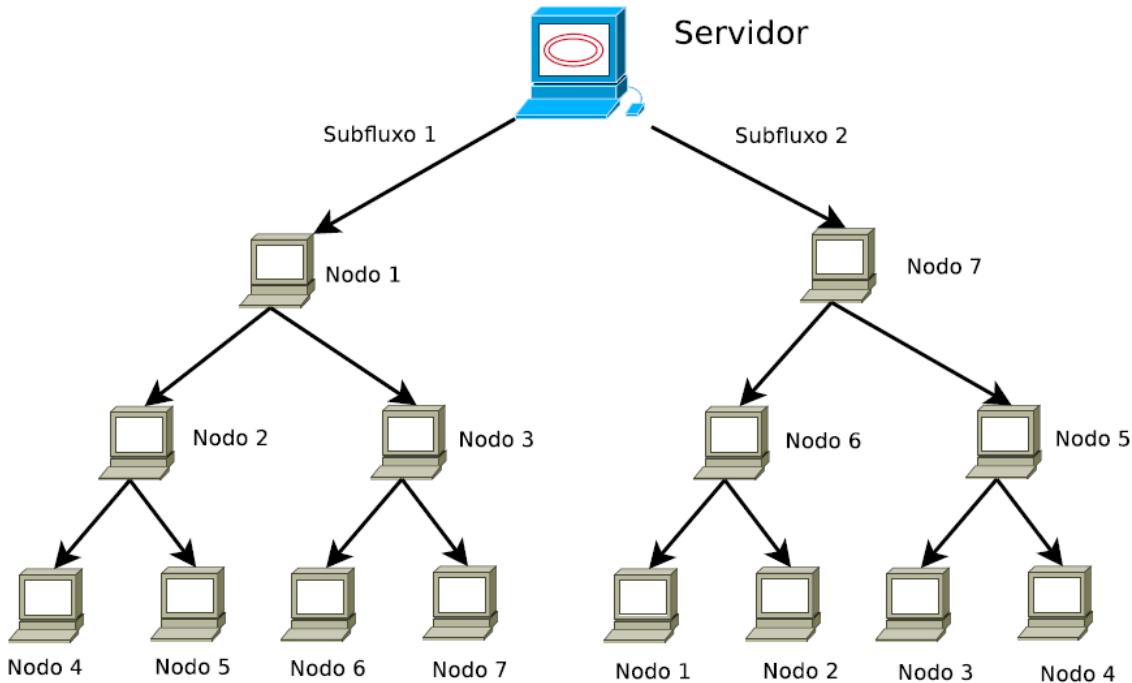


Figura 2.3: Sistema baseado em múltiplas árvores com dois subfluxos.

rede P2P. Os nós estabelecem suas conexões dinamicamente, de acordo com seus interesses. Os nós participantes sempre mantêm parcerias com vários outros vizinhos, podendo fazer envio ou recepção de dados de múltiplos nós parceiros e, se um nó participante deixar o sistema, seus vizinhos continuam recebendo o conteúdo desejado dos demais nós parceiros, com os quais eles mantêm contato. Caso seja do interesse de um nó participante, este pode encontrar novos nós parceiros para manter um nível de conectividade alto. Um alto grau de conectividade faz com que a estrutura em malha se torne robusta à dinâmica dos nós participantes do sistema. Trabalhos recentes, como o disponível na referência [194], mostram que uma estrutura baseada em malha tem um desempenho superior a uma estrutura baseada em árvores.

De maneira similar ao que acontece a um dos sistemas de compartilhamento de arquivos mais populares, o BitTorrent™, uma estrutura em malha, tem um servidor centralizado. Esse servidor mantém uma lista dos nós participantes ativos na sessão de vídeo. Quando um usuário utiliza o sistema cliente de distribuição de mídia contínua ao vivo pela primeira vez, tal usuário realiza um cadastro no servidor. O servidor de *bootstrap*, *rendezvous* ou *tracker*, como costuma ser chamado, retorna à aplicação cliente do usuário uma lista com informação de um subconjunto aleatório de outros usuários da sessão de vídeo.

Após receber a lista com os possíveis nós parceiros, o novo nó participante tenta realizar as parcerias. Se a parceria é aceita pelo nó contatado, o novo nó participante irá adicioná-lo a sua lista de vizinhos. Depois de obter alguns vizinhos, o novo nó participante começa a trocar pedaços de vídeo com seus nós parceiros. A Figura 2.4 mostra o processo inicial de cadastro no sistema e realização das parcerias iniciais.

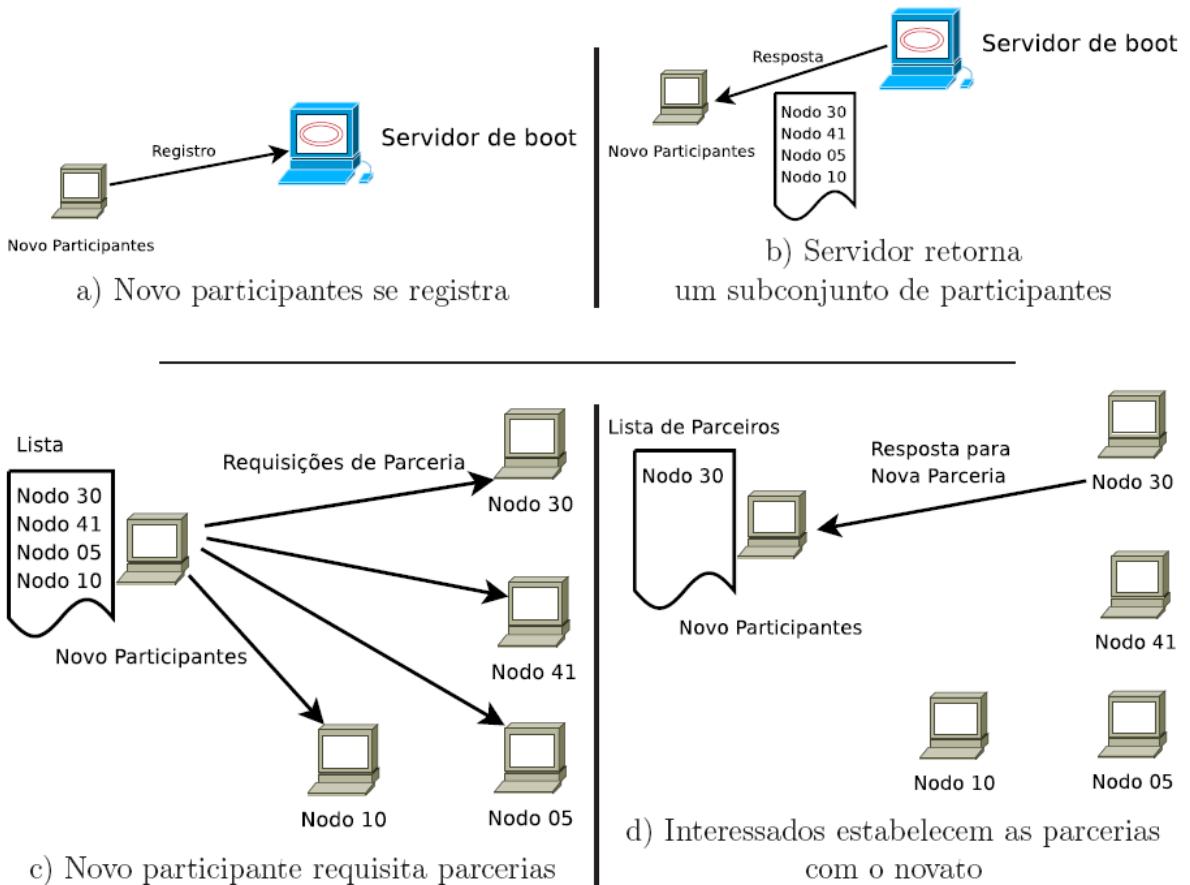


Figura 2.4: Atividade inicial de um novato - rede P2P baseada em malha.

Os nós participantes do sistema trocam regularmente mensagens de informação de vida (*keep-live messages* ou *ping*). Caso um vizinho não responda às mensagens de vida, seu nó parceiro o remove da lista e, possivelmente, tenta obter novos nós parceiros para manter sua conectividade [192]. Uma parceria é estabelecida por um acordo mútuo entre os nós participantes. Os diferentes sistemas existentes possuem estratégias variadas para estabelecimento destes acordos. Por exemplo, o número de vizinhos que os participantes possuem, a banda de rede disponível, a dinâmica dos seus vizinhos e a qualidade percebida do fluxo de vídeo [96]. Com base nesses critérios, um nó participante se conecta a um novo vizinho e

também procura por novas parcerias.

Em uma estrutura baseada em árvore, o fluxo de vídeo é transmitido a partir de uma fonte geradora para todos os nós participantes do sistema, seguindo a estrutura lógica da árvore formada. Em uma estrutura baseada em malha, não existe um fluxo contínuo transmitido nestes mesmos moldes. Nesses sistemas, a fonte do vídeo (servidor) faz a codificação e a divisão do vídeo, criando os pequenos pedaços chamados *chunks*. Cada *chunk* contém dado para um pequeno intervalo de tempo de visualização. Por exemplo, as aplicações atuais transmitem dados a uma taxa aproximada de 6 *chunks* por segundo de vídeo [96]. Esses *chunks* são numerados em uma sequência temporal, para que os nós participantes possam identificar e executar o vídeo correspondente de forma apropriada. Os pedaços do fluxo são disseminados a partir do servidor para diversos participantes da rede, que os disseminam para seus companheiros, e assim por diante. Como os *chunks* tomam diferentes caminhos para atingir os diversos pontos da rede, estes chegam a um usuário fora de ordem e, para uma execução contínua do vídeo, os nós participantes guardam os *chunks* em um armazenamento temporário de memória, onde são ordenados antes de sua apresentação. Dependendo do tipo de aplicação, o armazenamento pode variar de segundos a minutos. Em uma sessão de vídeo ao vivo, que é o período em que um fluxo de mídia é transmitido, a sequência de identificação dos *chunks* cresce enquanto o vídeo é disseminado.

Os dados são trocados principalmente através de duas estratégias: requisitando ou enviando (*pull* e *push*). Em um sistema do tipo *mesh-push* (malha e requisição), um nó envia os dados que recebe aos seus vizinhos que provavelmente ainda não os obtiveram. Não há uma relação clara de pai-filho neste esquema e o envio dos dados é estabelecido por interações passadas entre os nós participantes, onde indicam quais são os dados desejados. Um nó participante pode estabelecer parcerias com diversos outros nós e anunciar a necessidade por dados a todos estes. Por consequência, pode existir envio de dados redundantes na rede, pois mais de um dos nós parceiros pode responder por um pedido. Para tratar esse problema deve existir um planejamento entre os nós participantes do sistema, com escalonamento das transferências dos dados [192].

Caso seja usado um sistema *mesh-pull*, os nós participantes, periodicamente, trocam entre si um mapa de *chunks*. Este mapa tem informações dos *chunks* disponíveis localmente por um participante. Além disso, no mapa de *chunks* contém também informações sobre os

dados faltantes. Ao obter os mapas de seus vizinhos, um participante decide como escalarar o pedido de *chunks* (e a qual vizinho enviar o pedido). As transmissões redundantes são evitadas, uma vez que os participantes solicitam *chunks* a um único parceiro. Porém, as frequentes trocas de mapas de *chunks* e mensagens por pedidos aumentam a sobrecarga do protocolo e podem introduzir novos atrasos ao sistema.

Na Figura 2.5, ilustra-se a troca de *chunks* em uma aplicação com estrutura baseada em malha. Por esta figura, o nó 2 gera seu mapa, indicando quais *chunks* tem disponível em seu armazenamento temporário. O nó 2 então troca este mapa com o nó participante 1 e, como resposta, o nó 1 envia também o seu mapa. Observe que o nó 1 possui uma lista com os diversos mapas de seus parceiros. Os pedaços de vídeo faltantes no nó 2 serão requisitados ao nó 1. Finalmente, o nó 1 responde às requisições pelo nó 2.

### 2.2.3 Estrutura Híbrida

Uma estrutura híbrida para transmissões ao vivo em P2P pode ser caracterizada de duas formas. Na primeira, a arquitetura da rede é um misto entre uma arquitetura baseada em árvores e uma arquitetura baseada em malhas. Na segunda, o método de transmissão de dados entre os nós participantes é um misto entre um sistema P2P, orientado por pedidos explícitos por dados, e um encaminhamento automático dos dados da mídia. Em ambos os casos, há uma tentativa de se obter os benefícios de cada uma das propostas e isolar os pontos fracos das mesmas.

#### Híbrido de Árvore-Malha:

Em uma rede sobreposta P2P baseada em árvore, os nós participantes da rede são organizados de forma hierárquica. Assim, a transmissão ao vivo flui dos níveis mais altos na hierarquia (do nó participante que está codificando o vídeo) para os níveis mais baixos. Os nós participantes mais próximos à fonte apresentam menores latências no vídeo assistido e menos problemas com relação a rupturas na hierarquia da árvore.

As estruturas baseadas em malha contornam o problema de rupturas na árvore. Nesse modelo, os nós participantes do sistema realizam parcerias e trocam dados entre si. Não há hierarquias, entretanto, a recepção dos dados da transmissão ao vivo está sujeita a atrasos e

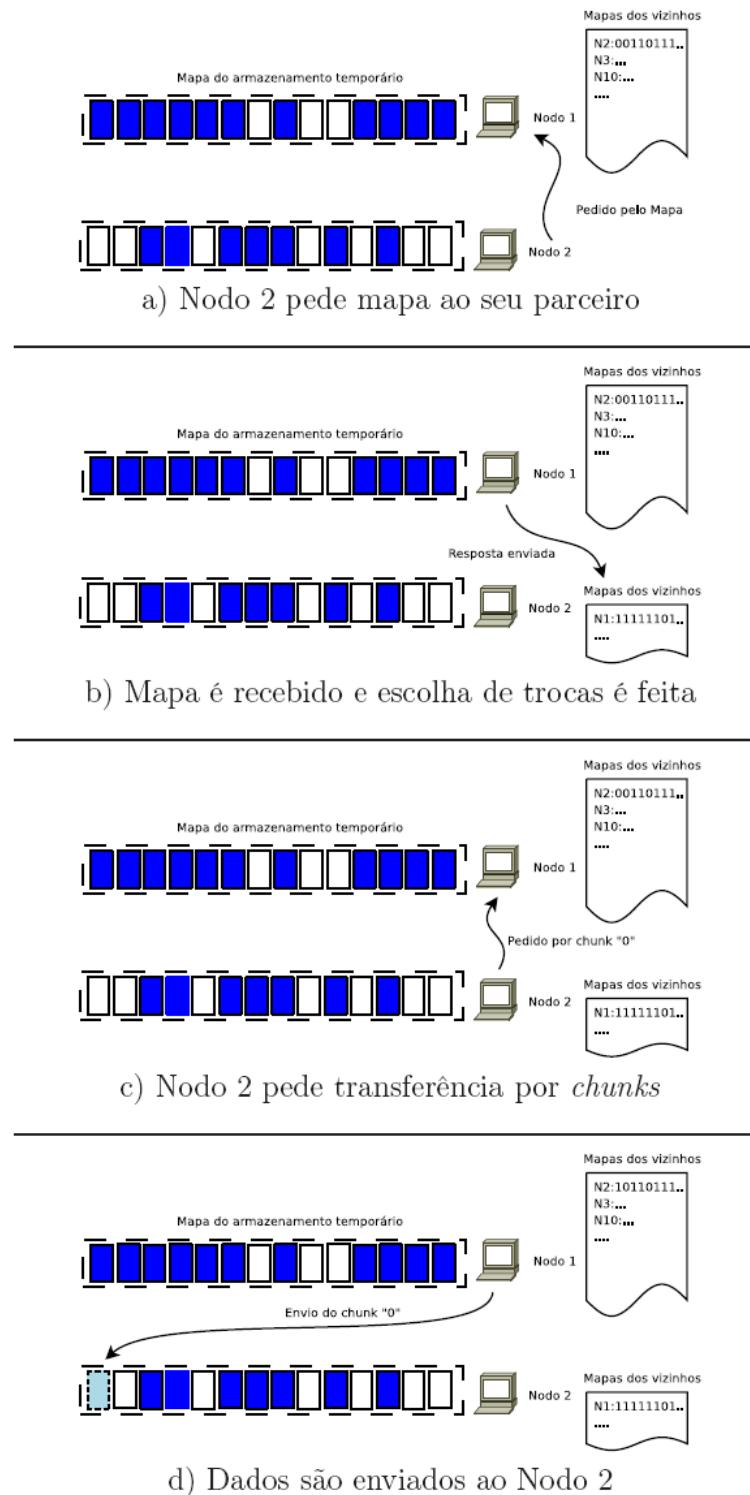


Figura 2.5: Troca de dados na aplicação baseada em malha.

imprevisibilidade [81].

Uma abordagem de construção híbrida da rede sobreposta adota partes da rede como uma árvore, e outras partes como uma rede em malha. Os nós participantes do sistema podem

participar de ambas as estruturas. Sistemas como o Anysee2 [80] adotam estratégias de alocação dos nós participantes na árvore e, na rede em malha formada, adotam estratégias para otimizar o agendamento de entrega de dados. Alguns dos critérios utilizados para alocar os nós participantes na árvore são estabilidade do nó participante na rede e a proximidade entre os nós participantes na rede física.

Mais precisamente, no Anysee2, estrutura-se os nós participantes em uma rede de controle e em uma rede de troca de dados. A rede de controle é baseada em uma árvore, enquanto a rede de troca de dados é baseado em uma malha.

### **Híbrido por encaminhamento automático / pedidos explícitos (*Push-Pull*):**

O método híbrido para obtenção de dados utiliza duas formas em conjunto para encaminhar/receber a mídia transmitida: o encaminhamento automático da mídia (utilizado em uma estrutura de árvores) e pedidos explícitos pelos dados (utilizado em uma estrutura em malha). Nesse caso, abordagens *Push* (encaminhamento automático) e *Pull* (pedido explícita), são utilizadas em uma rede P2P não estruturada. Dessa forma, esses sistemas quase sempre apresentam um protocolo/estrutura simples, sem a necessidade de coordenação e hierarquia entre os nós participantes. Isso torna o sistema naturalmente resistente à dinâmica dos nós participantes e a outros imprevistos.

Existem alguns mecanismos propostos com a combinação do “*push-pull*” [77, 196]. Esses mecanismos usam o “*push*” para espalhar os dados rapidamente e o “*pull*” para preencher as lacunas dos dados recebidos. Nesses dois trabalhos supracitados, ambos os mecanismos coexistem, não havendo uma alternância entre ambos.

O protocolo proposto em [197] alterna as operações de “*push*” e “*pull*”. Cada nó participante é autônomo e independente, sem a necessidade de sincronia com outros nós participantes. Durante a operação de “*push*”, o nó participante envia dados alguns de seus nós parceiros. Na operação de “*pull*”, o nó participante busca por dados necessários localmente.

A utilização do mecanismo de “*push-pull*”, como discutido no trabalho disponível através da referência [198], pode levar a uma redução da sobrecarga do tráfego da rede. Os resultados nesse trabalho mostram que, em comparação com um sistema do tipo “*mesh-pull*” e com o GridMedia [199], houve uma redução da sobrecarga de rede de 33 % e 37 %, respectivamente. Além disso, o sistema com a abordagem híbrida alcançou resultados com latência e

taxa de execução do vídeo melhores que os sistemas comparados.

## 2.3 Sistemas de Transmissão ao Vivo em P2P

Nesta seção, apresentam-se a descrição e o funcionamento de uma aplicação de envio de mídia ao vivo em P2P. A descrição apresentada utiliza como base as principais aplicações existentes atualmente, como a Sopcast [102], o PPLive [104], o GridMedia [192, 199], Octoshape, e o CoolStreaming [94, 103]. Na Figura 2.6, ilustra-se um cenário exemplo de um sistema de transmissão ao vivo em P2P que será detalhado nessa seção.

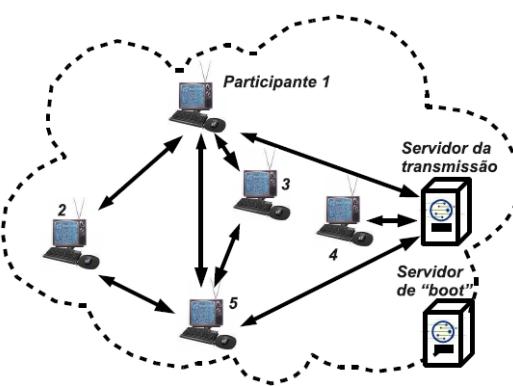


Figura 2.6: Modelo de sistema utilizado.

As principais entidades envolvidas nesses sistemas são as seguintes:

- **servidor da transmissão ao vivo:** o servidor de transmissão é um nó especial do sistema P2P. Este nó captura e codifica o vídeo que será transmitido pela rede. O servidor é a fonte inicial dos dados de vídeo da rede;
- **servidor de boot:** o servidor de *boot* (ou *bootstrap*) é uma entidade centralizadora por meio do qual os demais nós do sistema encontram seus parceiros iniciais para entrar na rede P2P. Todo nó se registra nesse servidor para fazer parte da lista dos nós do sistema. Quando um novo nó se registra, o servidor de *boot* envia ao nó requisitante uma lista com alguns nós parceiros candidatos. Quando um nó antigo deseja realizar mais parcerias, este pede ao servidor de *boot* uma lista com alguns nós para tentar novos contatos.

- **participantes (clientes/nós/peers):** são os usuários do sistema P2P de transmissão ao vivo. Cada nó participante está em contato com um subconjunto de todos os nós participantes do sistema. Não há hierarquia entre esses participantes, e qualquer um pode servir dados de vídeo e também responder a pedidos por dados oriundos de seus parceiros.

Os sistemas de envio de mídia contínua ao vivo em P2P são sistemas compostos por nós que colaboram entre si para a disseminação do conteúdo gerado por um servidor. Esses nós se organizam em uma rede virtual, sobreposta à rede real de computadores. A organização dessa rede se baseia, geralmente, em duas estruturas, a de árvore e a de malha, como discutiu-se na Seção 2.2. Nesses sistemas de transmissão ao vivo, um servidor  $S$  gera todo o conteúdo a ser disseminado pela rede e os demais nós do sistema recebem a mídia gerada em  $S$ , dividida em diversas partes, conhecidas por *chunks*, reproduzindo-as e repassando-as para seus nós parceiros.

Mais detalhadamente, os sistemas P2P para envio de mídia contínua ao vivo usam recursos do conjunto  $P = p_1, p_2, \dots, p_n$  de seus nós participantes para repassar o conteúdo que é transmitido pelo servidor  $S$ . Cada nó participante  $p_i$  é livre para entrar e sair do sistema a qualquer momento. Tal comportamento diferencia a aplicação de transmissão ao vivo em P2P de aplicações baseadas em IP *multicast*, pois, em IP *multicast* a estrutura formada é pouco dinâmica.

Os sistemas mais populares de envio de mídia ao vivo em P2P utilizam uma rede sobreposta baseada em malha. Mais ainda, no modelo de malha, a rede não é estruturada de forma rígida e as parcerias no sistema são formadas aleatoriamente. As interações e trocas de dados entre os nós participantes  $p_i$  e  $p_j$ , com  $i \neq j$ , são normalmente orientadas pelos pedidos de dados e informações entre  $p_i$  e  $p_j$ . Assim, esse tipo de rede sobreposta do tipo malha é utilizada para aliviar os efeitos de entrada e saída dos nós participantes na rede [101, 199].

O funcionamento desse tipo de sistema acontece da seguinte forma: inicialmente, um nó  $p_i$  conecta-se a um servidor centralizado de inicialização, denominado de *bootstrap*  $B$  ou rastreador. Na inicialização de  $p_i$ , o servidor  $B$  envia um subconjunto, dos nós do sistema para o nó  $p_i$ . Esse subconjunto é definido por  $LPC_i$  ( $LPC_i \subseteq P$  e  $LPC_i \neq \emptyset$ ) e é a lista inicial dos nós candidatos a parceiros do nó  $p_i$ . Além de se registrar e obter uma lista de candidatos a parceiros, o novo nó sincroniza a posição atual da mídia ao vivo com a posição

informada pelo servidor  $B$  [101]. Assim,  $p_i$  tem uma referência do ponto da mídia ao vivo que está sendo gerada pelo servidor  $S$  e saberá a partir de qual ponto deverá solicitar os dados para reproduzir a mídia ao vivo.

O novo nó  $p_i$  seleciona, aleatoriamente, uma quantidade  $n$  de nós de  $LPC_i$  como parceiros candidatos. Estes nós candidatos formarão o conjunto de parceiros de  $p_i$ , denominado  $LP_i$  ( $LP_i \subseteq LPC_i$ ). Os conjuntos  $LPC_i$  e  $LP_i$  são dinâmicos, pois cada nó  $p_j$  está livre para abandonar o sistema. Quando  $p_j \in LP_i$  e o nó  $p_i$  detecta a inatividade deste parceiro,  $p_i$  remove  $p_j$  de  $LPC_i$  e  $LP_i$  e seleciona um novo elemento de  $LPC_i$  para criar uma nova parceria.

O nó  $p_i$  sempre tenta manter sua  $LPC_i$  com um número de candidatos acima de um limiar  $L_i$ . O valor de  $L_i$  pode ser dado pela capacidade de recurso de cada nó, como banda de rede ou número de conexões disponível. Assim, quando  $|LPC_i| < L_i$ , então  $p_i$  recorre ao servidor  $B$  para obter novos elementos para  $LPC_i$ .

Cada nó  $p_i$  também contém um mapa de partes da mídia de tamanho  $m$ , representado por  $cm_i$ . Esse mapa sinaliza os *chunks* da mídia que o nó  $p_i$  contém ou necessita. Ou seja, o mapa  $cm_i$  representa um trecho contínuo da mídia transmitida ao vivo pelo sistema P2P que será reproduzida pelo nó  $p_i$ .

Inicialmente, cada posição do mapa é marcada como “desejada”, ou seja,  $cm_i[x] = desejada$ , onde  $x = [0..m]$ . Periodicamente,  $p_i$  requisita  $cm_j$  a cada um de seus parceiros  $p_j$ , com ( $p_j \in LP_i$ ). Dessa forma,  $p_i$  verifica quais parceiros podem satisfazer a sua necessidade por determinado *chunk*  $c_t$ . Quando  $p_i$  recebe um *chunk*  $x$  qualquer,  $p_i$  marca  $cm_i[x] = disponivel$ .

Periodicamente,  $p_i$  verifica quais *chunks*  $c_t$  são necessários ( $cm_i[c_t] = desejada$ ) e verifica entre seus parceiros  $p_j$ , com  $p_j \in LP_i$ , quais possuem o *chunk*  $c_t$  com  $cm_j[c_t] = disponivel$ . O parceiro  $p_j$  que contém o *chunk*  $c_t$  e que possui maior disponibilidade de recursos é escolhido por  $p_i$  para a realização do pedido de *chunk*. Quando  $p_i$  recebe o *chunk*  $c_t$  de  $p_j$ ,  $p_i$  marca  $cm_i[c_t] = disponivel$ . Os processos de escolha do *chunk* a ser requisitado e a escolha do parceiro serão detalhados nas seções seguintes. Na Tabela 2.1, apresenta-se um resumo dos elementos utilizados para descrever um sistema P2P de transmissão ao vivo.

Tabela 2.1: Resumo dos elementos de um sistema P2P de transmissão ao vivo.

Elemento	Descrição
$S$	Servidor de mídia contínua.
$B$	<i>Bootstrap</i> ou rastreador do sistema.
$P = p_1, p_2, \dots, p_n$	Conjunto dos nós participantes do sistema.
$p_i$	Nó participante $i$ do sistema
$LP_i$	Conjunto de nós parceiros de $i$ .
$LPC_i$	Conjunto de nós parceiros candidatos de $i$ .
$L_i$	Número mínimo de nós candidatos $p_i$ .
$cm_i$	Mapa de <i>chunks</i> de $p_i$ .
$cm_i[x] = \text{desejada}$ , onde $x = [0..m]$	Conteúdo inicial do mapa de <i>chunks</i> de $p_i$ .

### 2.3.1 Geração do Conteúdo da Transmissão

Na aplicação de envio de mídia contínua ao vivo em P2P, o servidor  $S$  é responsável pela aquisição e codificação da mídia em um formato apropriado para a transmissão. O servidor  $S$  gera o conteúdo da transmissão e o divide em *chunks*. Cada novo *chunk*  $c_i$  é armazenado na área de memória apropriada de  $S$  (*buffer*). Assim,  $S$  marca  $cm_s[c_i]$  como disponível no seu mapa de *chunks*, ou seja,  $cm_s[c_t] = \text{disponível}$ .

Os parceiros do servidor  $S$  atualizarão as informações sobre o buffer do servidor  $S$  a partir da troca dos mapas de *chunks* e perceberão a existência de novos dados. Os nós fazem requisições ao servidor  $S$  por *chunks* produzidos e então  $S$  começará a disseminar os dados da mídia. Outros nós do sistema irão encontrar os novos dados produzidos por  $S$  quando algum de seus parceiros os receberem, seja por um pedido direto a  $S$  ou por outro nó do sistema.

Nas aplicações mais populares, o servidor  $S$  gera os dados da mídia contínua ao vivo a uma taxa aproximada de 6 *chunks* por segundo. Normalmente, um vídeo é codificado a aproximadamente 300  $Kbps$  e assim, cada *chunk* tem cerca de 6  $KB$  de dados.

Na Tabela 2.2, apresenta-se um resumo dos elementos utilizados para descrever a geração de conteúdo da transmissão ao vivo em P2P.

Tabela 2.2: Resumo dos elementos utilizados na geração de conteúdo de mídia ao vivo em sistemas P2P.

Elemento	Descrição
$S$	Servidor que gera o conteúdo da transmissão.
$cm_i$	Mapa de <i>chunks</i> de $p_i$ .
$cm_i[x] = \text{desejada}$ , onde $x = [0..m]$	Conteúdo inicial do mapa de <i>chunks</i> de $p_i$ .

### 2.3.2 Armazenamento e Consumo de Dados

Os nós do sistema P2P de transmissão ao vivo devem conseguir obter a mídia da rede a uma taxa apropriada. Caso não o consigam, a execução da mídia terá falhas e a experiência do usuário com relação a qualidade do conteúdo multimídia recebido poderá ser ruim. Assim, os nós devem obter os dados da mídia o mais rápido possível, uma vez que a aplicação de transmissão ao vivo exige uma baixa diferença de tempo entre a criação do trecho de mídia e a sua exibição nos nós do sistema P2P.

Caso a latência seja alta, os nós irão experimentar um atraso indesejável e, no pior dos casos, a informação será exibida muito tempo depois do fato ocorrido. Por exemplo, um gol em uma partida de futebol poderá ser comemorado por um usuário de um nó e somente muito tempo depois o usuário de um nó vizinho com recepção em atraso visualizará a mesma cena em sua aplicação. Por esse motivo, os nós devem obter a mídia a uma taxa de  $c$  *chunks* por segundo. Essa é a mesma taxa de produção *chunks* pelo servidor  $S$ .

Cada nó do sistema apresenta uma área de armazenamento temporário  $B_i$  (*Buffer*), onde são armazenados os dados da mídia recebidos da rede P2P. Esse *buffer* pode guardar uma quantidade pré-determinada de *chunks*. Inicialmente, cada posição  $j$  de  $B_i$  ( $B_i[j]$ ) é inicializada com conteúdo vazio e, à medida que  $p_i$  recebe os *chunks* de seus parceiros, cada posição vai sendo preenchida. Nesse esquema de armazenamento temporário, o nó  $p_i$  tem por objetivo manter o *buffer*  $B_i$  preenchido de forma que se garanta uma contínua exibição da mídia ao vivo, mesmo se este perder conexão temporariamente com seus parceiros.

Inicialmente, o *buffer*  $B_i$  pode ser representado por uma estrutura do tipo “*buffer circular*”. Dois processos trabalham em conjunto para manter o *buffer*  $B_i$  preenchido e a execução

da mídia constante (produtor/consumidor). Assim, a posição de  $B_i$  a ser consumida é  $B_i[0]$  e a posição mais recentemente a ser preenchida será  $B_i[b]$  ( $b$  é a última posição de um *buffer* com pelo menos  $b + 1$  posições).

Para manter uma visualização constante e sem interrupções, a aplicação deve obter alguns dados à frente do momento de exibição. Então, caso ocorra um problema temporário na rede P2P, há alguns dados armazenados no *buffer*. A cada intervalo de tempo, o nó  $p_i$  verifica quais *chunks* devem ser consumidos em uma janela de tempo futura. Os *chunks* pertencentes a essa janela de interesse deverão ser recolhidos da rede enquanto  $p_i$  executa os dados relativos aos *chunks* anteriores a essa janela.

O tamanho da janela de interesse deve ser pequeno para não possibilitar a exibição da mídia com atraso demasiado (espera longa para recolher os dados da rede). Porém, janelas de interesse muito curtas podem gerar uma série de perdas na exibição, pois pode ocorrer que um determinado *chunk* não tenha sido obtido e o momento de sua exibição tenha chegado.

Na Figura 2.7, exemplifica-se o mecanismo de consumo da mídia por um nó do sistema. Nessa figura, considera-se que a taxa de criação de *chunks* é de uma unidade por intervalo de tempo. Desta forma, a cada unidade de tempo, o nó deve tentar obter o próximo *chunk* criado. Assim, a cada intervalo de tempo, o nó desloca a sua janela de interesse para o próximo *chunk* indicado em seu mapa.

No primeiro momento da Figura 2.7, esse nó participante irá consumir o *chunk* à esquerda da janela de interesse. No segundo momento, a janela de interesse é deslocada para a direita e esse nó participante não tem o respectivo *chunk* para consumo (poderá haver uma falha na exibição). Neste mesmo instante, o nó participante consegue obter o *chunk* mais recente de seu interesse. Finalmente, o processo continua e o nó participante consome o próximo *chunk*.

### 2.3.3 Estratégia de Seleção de *Chunks*

A estratégia de seleção do *chunk* pode influenciar no desempenho da aplicação que reproduz o conteúdo multimídia. As estratégias existentes determinam qual dos *chunks*, entre os vários necessários, deve ser requisitado em um determinado momento. Essas estratégias de seleção tentam manter a continuidade da exibição da mídia ao vivo para um determinado nó do sistema, difundindo os *chunks* que acabaram de ser gerados o mais rápido possível para os

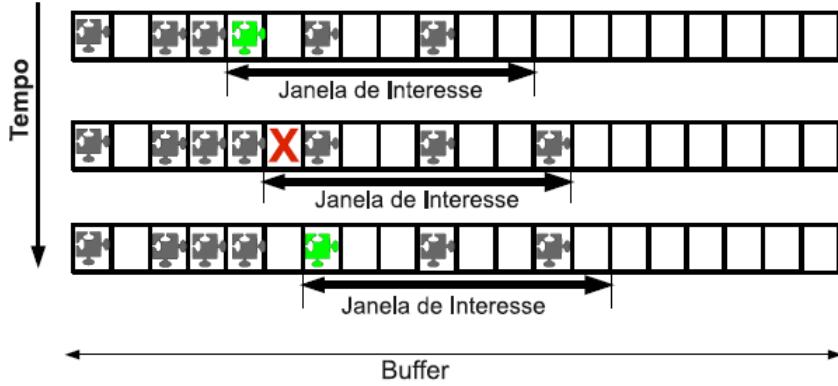


Figura 2.7: Mecanismo de consumo da mídia ao vivo.

demais nós do sistema.

Existem duas estratégias de seleção de chunks comumente utilizadas em aplicações P2P. A primeira é chamada de “Mais Raro Primeiro”, que é adotada em protocolos de aplicações de compartilhamento de arquivos em P2P, como o BitTorrent, e em envio de mídia ao vivo P2P, como o CoolStreaming [103]. A segunda estratégia é denominada “Gulosa”, onde os nós participantes privilegiam a escolha de *chunks* que estão próximos ao fim de suas janelas de visualização.

#### Estratégia “Mais Raro Primeiro”:

Na estratégia “Mais Raro Primeiro”, um nó  $p_i$  irá requisitar o *chunk* que está menos replicado pelo sistema de transmissão.

Para exemplificar essa estratégia, considere o *buffer*  $B_i$  do nó participante  $p_i$ . A posição  $B_i[0]$  será a mais rara e está vazia, pois acabara de ser criada pelo servidor  $S$ . A probabilidade de encontrar um parceiro que tenha esse dado disponível cresce com o tempo. Assim, no próximo intervalo de tempo, o *chunk* da posição  $B_i[0]$  irá para a posição  $B_i[1]$  que, no intervalo consecutivo, será movido para a posição  $B_i[2]$  e assim por diante. Dessa forma, percebe-se claramente que o *chunk* mais raro a ser buscado é o que acabara de ser criado, ou seja,  $B_i[0]$ . Portanto, a estratégia “Mais Raro Primeiro” seleciona os *chunks* em ordem crescente.

### Estratégia “Gulosa”:

Por outro lado, a estratégia “Gulosa” tem como objetivo preencher os espaços do *buffer* que estão próximos de seu prazo final de visualização. Assim, um nó  $p_i$  selecionará o *chunk* mais próximo à posição de visualização no seu *buffer*  $B_i$ . Por motivos de simplificação, pode-se considerar o *chunk* final do *buffer* ( $B_i[b]$ ) como sendo o elemento de próximo prazo final para visualização. Sendo assim, o nó  $p_i$  selecionará o *chunk*  $B_i[b]$  caso este não esteja preenchido em seu *buffer*, depois o *chunk*  $B_i[n - 1]$  e assim por diante. Desse modo, os nós do sistema tendem a ter armazenado em seus *buffers* os dados mais antigos produzidos pelo servidor  $S$ .

#### 2.3.4 Realização de Parcerias e Obtenção de *Chunks*

Um nó  $p_i$  realiza parcerias logo após seu primeiro contato com o servidor de *bootstrap*  $B$ . A partir do estabelecimento das parcerias iniciais, o nó efetivamente começa a participar do sistema de envio de mídia contínua ao vivo em P2P. Além deste momento inicial de estabelecimento de parcerias, um nó pode ser contatado por um outro nó participante  $p_j$ , requisitando sua parceria ou  $p_i$  pode tentar novas parcerias para aumentar sua conectividade.

Tanto no estabelecimento inicial de parcerias, quanto na descoberta de novos nós parceiros para aumento da conectividade, o nó  $p_i$  recorre à lista de nós parceiros candidatos  $LPC_i$  para selecionar um nó, ao qual envia uma requisição de parceria. Vários critérios podem ser utilizados para a escolha do candidato  $p_k$ , como uma escolha aleatória entre os nós pertencentes a  $LPC_i$ , com base nos recursos disponíveis em  $p_k$ , proximidade temporal ou até mesmo proximidade geográfica, informados por  $B$  ou por comunicação direta entre os parceiros.

Uma vez selecionado o nó candidato,  $p_i$  envia uma mensagem de pedido de parceria ao nó candidato  $p_k$ , selecionado de  $LPC_i$ . Caso  $p_k$  tenha recursos disponíveis (por exemplo, banda de rede, conexões disponíveis na aplicação etc.), este adiciona  $p_i$  à  $LPC_k$  e responde a solicitação de  $p_i$ . Quando  $p_i$  recebe a resposta de  $p_k$ , este adiciona  $p_k$  à  $LPC_i$  e a nova parceria é de fato estabelecida. No momento que um nó  $p_i$  adiciona um novo nó parceiro  $p_k$  à  $LPC_i$ , este inicia um temporizador  $t_{ik}$ , o qual é acionado em intervalos de tempo  $tp_i$ . A cada expiração do temporizador  $t_{ik}$ , o nó  $p_i$  envia uma mensagem ao nó parceiro  $p_k$  para verificar seu estado na aplicação. O objetivo desta mensagem é verificar se a parceria está

ativa, além de haver troca de informações, como os mapas de *chunks* de ambos os nós.

A partir da seleção de  $p_k$ , o nó  $p_i$  envia uma requisição pelo *chunk* de interesse  $c_j$ . Quando  $p_k$  recebe um pedido por dados vindo de um parceiro ativo  $p_i \in LP_k$ , este cria uma mensagem de resposta contendo o *chunk*  $c_j$  requisitado e o envia ao nó  $p_i$ . Caso o pedido seja enviado por um nó não parceiro, ou seja,  $p_i \notin LP_k$ ,  $p_k$  ignora o pedido por  $c_j$ , mas adiciona  $p_i$  à lista de nós parceiros candidatos  $LPC_k$ . O nó  $p_i$  espera a resposta de  $p_k$  acerca do pedido de parceria por um intervalo de tempo  $tp_i$ . Caso  $p_k$  não responda no intervalo de tempo  $tp_i$ ,  $p_i$  remove  $p_k$  de sua  $LP_i$ . Caso  $p_i$  receba a resposta de  $p_k$ ,  $p_i$  atualiza os dados relativos a  $p_k$ , como por exemplo, o mapa de *chunks*  $cm_k$ . Se o pedido pelo *chunk*  $c_j$  for respondido durante o intervalo de tempo esperado,  $p_i$  coloca o novo dado em seu *buffer* de reprodução e o assinala como disponível em seu mapa de *chunks*, ou seja,  $cm_i[c_j] = disponivel$ . Fazendo-se dessa forma,  $p_i$  passa a ter a capacidade de compartilhar o *chunk* recebido com seus nós parceiros. Esse processo pode ser repetido até que o *chunk*  $c_i$  seja recebido corretamente, ou que não faça mais sentido obter aquele determinado *chunk*, por exemplo, por já ter passado o seu tempo de visualização.

## 2.4 Criptografia de Hash e Assinatura Digital

Mudando de assunto, nesta seção, discute-se sobre os aspectos de criptografia baseada em chaves assimétricas, funções de *hash* e assinatura digital. Os conceitos apresentados aqui, adaptado de [200], são base para o entendimento de como, no protocolo proposta neste trabalho, verifica-se a autenticidade de um conjunto de pacotes de dados, que corresponde a um fluxo de dados multimídia de um evento ao vivo, transmitido por um servidor para um ou mais clientes. Com relação a certificação digital, seu uso permite que os nós clientes em uma transmissão possam confiar no conteúdo recebido de outros sistemas remotos, através da validação do conteúdo transportado dentro de um pacote de dados, ou seja, se tal conteúdo foi realmente emitido pelo nó gerador do fluxo de dados de interesse do nó cliente.

Especificamente, os conceitos apresentados a seguir são base para o entendimento do mecanismo de verificação de autenticidade de um fluxo de dados multimídia transmitido por um sistema final servidor e recebido por um sistema final cliente, através da utilização do protocolo GMTP. Este assunto é discutido no Capítulo 4, Seção 4.6.

### 2.4.1 Criptografia de Hash

Uma função de *hash* [201]  $H(m)$  permite transformar uma mensagem  $m$  de qualquer tamanho em um identificador digital  $m'$  de tamanho fixo, chamado de valor de *hash*. Uma vez obtido  $m'$  para uma mensagem  $m$ , é computacionalmente impossível fazer o processo inverso, ou seja, encontrar o valor  $m$  tal que  $H(m) = m'$ . Desta forma, uma função de *hash* tem a propriedade de ser uma função “*one way*”. Além disso, este tipo de função deve oferecer as seguintes características:

- $H(m)$  é relativamente fácil de ser computado, para qualquer valor de  $m$ ;
- $H(m)$  é livre de colisão. Ou seja, para quaisquer duas mensagens  $m_1$  e  $m_2$ , com  $m_1 \neq m_2$ , deve-se sempre obter  $H(m_1) \neq H(m_2)$ .

Algoritmos de *hash* são amplamente utilizados em diversos contextos dos sistemas de informação, tais como em indexação em tabelas de *hash*, detecção de dados duplicados, arquivos corrompidos etc. Os algoritmos de *hash* também podem ser utilizados em processos de autenticação, tais como assinaturas digitais, verificações de senhas e cadeias de *hash* (*One-Way chains*) [202].

### 2.4.2 Criptografia Assimétrica

Como ilustra-se na Figura 2.8, em criptografia assimétrica um sistema final transmissor de uma mensagem criptografa o conteúdo a ser transmitido utizando a chave pública do sistema final receptor. Por sua vez, o sistema final receptor, ao receber a mensagem criptografada, utiliza sua chave privada, combinada com sua chave pública, para decifrar o conteúdo. Sendo assim, apenas o sistema final receptor pode decifrar o conteúdo da mensagem, uma vez que apenas tal sistema conhece a chave privada. Nesse contexto, a chave privada é representada por  $K_A^-$ , ao passo que a chave pública é representada por  $K_A^+$ .

### 2.4.3 Assinatura e Certificação Digital

Uma assinatura digital [203] é um modelo de autenticação de informações digitais tipicamente análoga à assinatura física em papel. Através de uma assinatura digital, pode-se provar

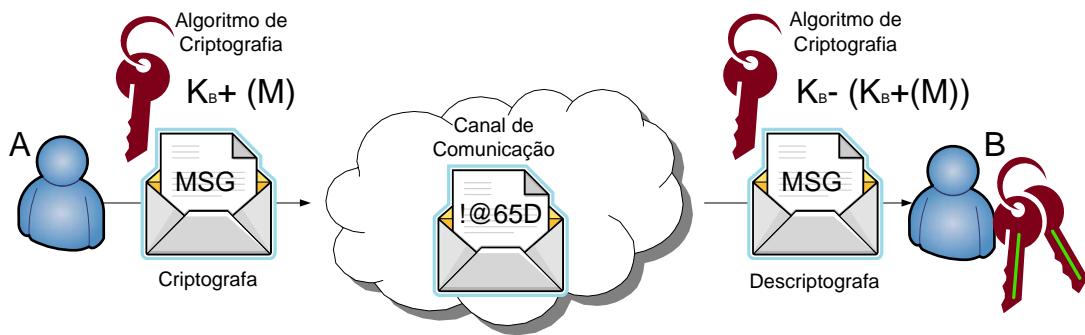


Figura 2.8: Modelo de criptografia assimétrica, chave pública representada por  $K^+$  e chave privada presentada por  $K^-$ .

que uma mensagem foi realmente enviada pelo emissor, o receptor pode confirmar que a assinatura foi feita pelo emissor (autenticidade) e que qualquer alteração da mensagem faz com que a assinatura digital não corresponda mais à mensagem (integridade). Basicamente, uma assinatura digital é construída a partir da combinação das funções de *hash* e dos algoritmos de criptografia assimétrica.

Como ilustra-se na Figura 2.9, para que um sistema final  $A$  envie uma mensagem  $m$  assinada digitalmente ao sistema final  $B$ , o sistema final  $A$  aplica uma função de *hash* à mensagem original  $m$ , gerando  $m'$  e, em seguida, cifra  $m'$  utilizando sua chave privada, resultando na assinatura digital. A mensagem e a assinatura digital são enviadas ao receptor através de um canal de comunicação.

Quando o sistema final  $B$  recebe a mensagem  $m$  transmitida pelo sistema final  $A$ , o sistema final  $B$  comprova a autenticidade da mensagem realizando dois procedimentos:

- calcula-se o valor de *hash* da mensagem  $m$ , resultando em  $m'_{recebida}$ ); e
- decifra-se a assinatura digital com a chave pública do sistema final  $A$ , que deve resultar em  $m'_{enviada}$ . Em seguida, conclui-se o seguinte:
  - Se  $m'_{recebida} = m'_{enviada}$ , a mensagem  $m$  está íntegra, ou seja, a mensagem  $m$  não sofreu qualquer alteração. Sendo assim, a assinatura está correta, pois foi gerada pela chave privada correspondente à chave pública utilizada na verificação; ou
  - Se  $m'_{recebida} \neq m'_{enviada}$ , a mensagem  $m$  foi alterada, ou seja, a assinatura digital emitida pelo sistema final  $A$  não corresponde à mensagem  $m$ .

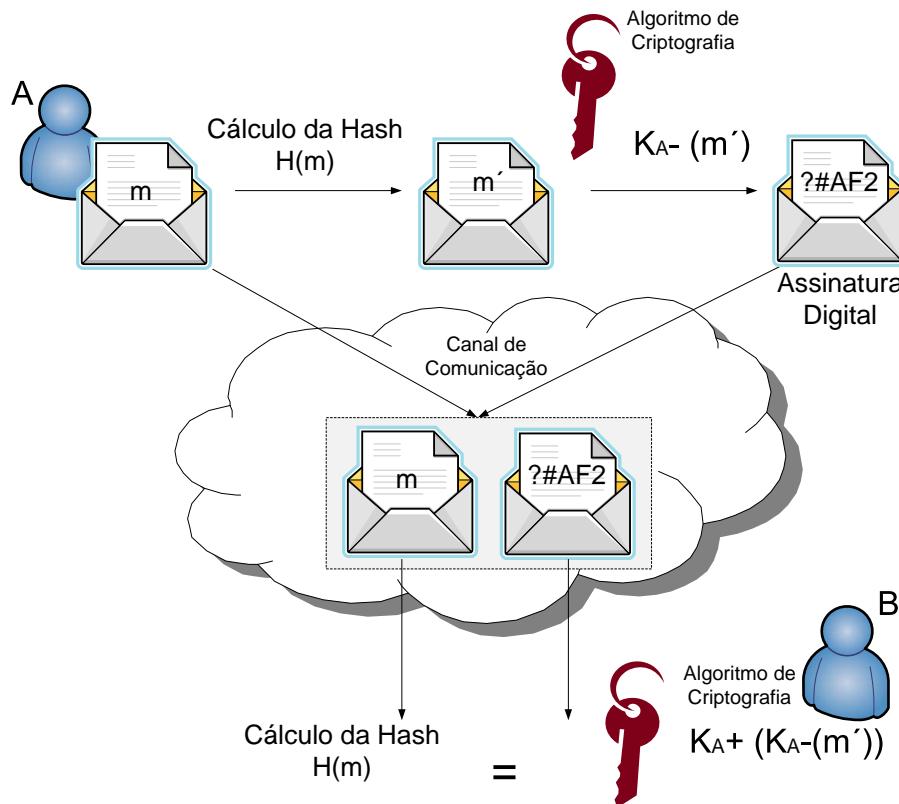


Figura 2.9: Geração de Assinaturas Digitais.

O uso das funções de *hash* pode contribuir para que o processamento da assinatura digital não seja comprometido devido à lentidão dos algoritmos de criptografia assimétrica, com rápida execução e uniforme, ou seja, independente do tamanho da mensagem a ser assinada. Ademais, as funções *hash* garantem a integridade da mensagem, visto que a alteração em um bit da mensagem irá gerar outro valor de *hash*. A criptografia assimétrica garante também a autenticidade e a não-repudiabilidade do emissor, pois apenas tal emissor é capaz de cifrar a mensagem com a chave privada correspondente à chave pública utilizada pelo receptor para checar a autenticidade da mensagem.

Os mecanismos oferecidos pela assinatura digital é uma forma eficaz de garantir a autoria de documentos eletrônicos. Em agosto de 2001, a Medida Provisória da República do Brasil, número 2.200 [204] (*Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil*), foi publicada para garantir a validade jurídica de documentos eletrônicos e a utilização de certificados digitais para atribuir autenticidade e integridade aos documentos, formalizando o conceito de assinatura digital.

## 2.5 Sumário do Capítulo

Neste capítulo, apresentou-se os principais conceitos de funcionamento dos sistemas de transmissão ao vivo P2P. Inicialmente, descreveu-se as estruturas de uma rede P2P constituídas por esses sistemas. Em seguida, apresentou-se o funcionamento básico de um sistema de transmissão e suas principais estratégias para geração e disseminação de *chunks*, bem como a seleção de nós parceiros.

Os cenários de transmissão de mídia ao vivo considerados são baseados em uma arquitetura em malha e sem organização rígida dos nós do sistema. Os nós podem entrar e sair a qualquer instante e realizam parcerias com um subconjunto de outros nós. Os parceiros trocam informações entre si para colaborar uns com os outros na obtenção de uma mídia transmitida ao vivo.

Para participar de um sistema P2P de transmissão ao vivo, a aplicação do usuário se registra em um servidor chamado de *bootstrap*. Esse servidor armazena as informações de todos os nós ativos do sistema. O novo nó recebe uma lista com outros nós do sistema e essa lista é utilizada para a tentativa inicial de estabelecimento de parcerias. Entre os nós do sistema há um especial: o servidor de mídia ao vivo. Este servidor captura o vídeo a ser transmitido, codifica-o em um formato apropriado e disponibiliza-o para toda a rede P2P.

Os nós têm um armazenamento local, onde guardam os dados do vídeo para uma execução contínua. Estes nós devem verificar quais pedaços de dados são necessários, havendo maneiras apropriadas de selecioná-los e requisitá-los. Para isso, apresentou-se duas abordagens denominadas “Gulosa” e “Mais Raro Primeiro”. Essas estratégias mantém o fluxo da execução sem interrupções e dissemina o conteúdo rapidamente pela rede P2P, respectivamente.

Caso mais de um parceiro possa contribuir com o dado necessário, um nó deve escolher a qual fará a solicitação. O mecanismo adotado se baseia na disponibilidade de recursos de cada parceiro, que será escolhido de acordo com a maior quantidade de recursos disponíveis, como por exemplo, banda de rede, processamento, memória etc.

Por fim, apresentou-se alguns conceitos básicos sobre criptografia de chave pública e assinatura digital. Esses conceitos são aplicados nesse trabalho no contexto de criptografar os dados transmitidos por um sistema de transmissão ao vivo e assim impedir ataques de

---

poluição.

# **Capítulo 3**

## **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 disponíveis na literatura (revistas, conferências, livros, teses e dissertações).

Para uma leitura linear deste capítulo, na próxima seção, apresenta-se uma breve discussão sobre os principais marcos na área de distribuição de mídias ao vivo, posicionando o GMTP em um cenário macro do estado da arte. Em seguida, na Seção 3.2, apresentam-se detalhes dos principais sistemas e protocolos de distribuição de mídias ao vivo, enfatizando suas respectivas arquiteturas e modelos de serviço. Já na Seção 3.3, apresenta-se uma nova vertente de pesquisa sobre distribuição de dados na Internet, conhecida por Redes Centradas na Informação, enfatizando suas principais características e os trabalhos relacionados à distribuição de mídias ao vivo. E, por fim, na Seção 3.4, apresenta-se um breve resumo sobre este capítulo. Caso o leitor deseje realizar uma leitura com foco nos trabalhos mais relevantes e fortemente relacionados ao GMTP, recomenda-se a leitura da próxima seção, bem como da Seção 3.2.6, onde se discute sobre o projeto CoolStreaming; da Seção 3.2.7, onde se discute sobre o projeto Denacast e, por fim, de toda a Seção 3.3.

## 3.1 Visão Geral das Propostas para Distribuição de Mídias ao Vivo

Diversas propostas e tecnologias destinadas à distribuição de mídias ao vivo foram desenvolvidas ao longo dos anos, disponibilizando-se um extenso ferramental e sistemas para este fim. Como resultado, aperfeiçoou-se a forma de comunicação entre redes visando, sobretudo, melhorar a qualidade de serviço e, consequentemente, a experiência do usuário final ao assistir um evento ao vivo através da Internet.

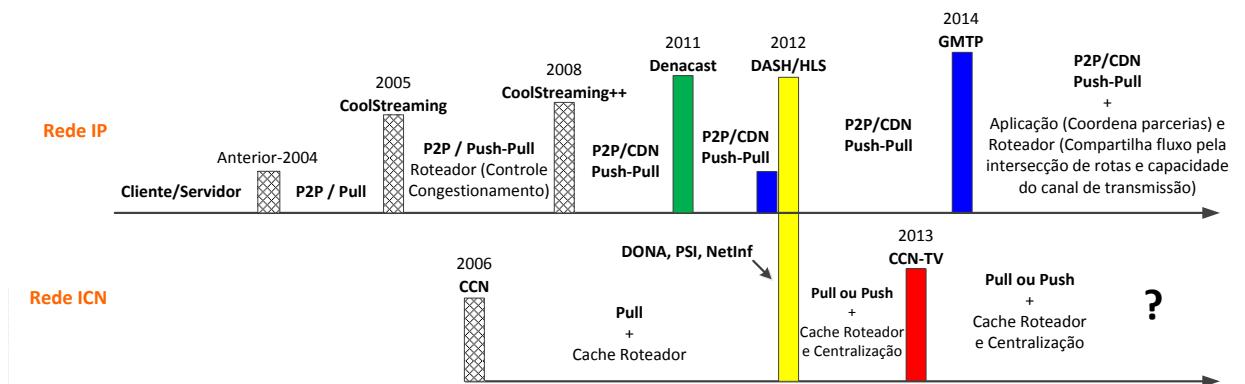


Figura 3.1: Representantes e marcos importantes no contexto de distribuição de mídias ao vivo. Evolução das arquiteturas e modelos de serviços adotados na Internet (incluindo o GMTP). Antes do GMTP, todas as soluções tem uma forte dependência dos sistemas finais para executar suas funções. No GMTP, os sistemas finais têm um papel coadjuvante na execução das funções, realizando-se parcerias entre os roteadores da rede.

Na Figura 3.1, ilustram-se os principais representantes e marcos no contexto de distribuição de mídias ao vivo, bem como a evolução das arquiteturas e modelos de serviços adotados na Internet (incluindo o GMTP). Como pode-se observar, atualmente existem duas vertentes para transmissão de dados na Internet: as Redes IP e as Redes Centradas na Informação. Para as soluções disponíveis nas redes IP, destacam-se o sistema DONet/CoolStreaming [82], o Denacast [36] e o protocolo DASH [136], detalhados na Seção 3.2. Já para as Redes ICN, destacam-se o CCN/NDN [50] e o sistema CCN-TV [205]. Apesar do foco ser nesses trabalhos, outros trabalhos relevantes serão discutidos ao longo deste capítulo.

Até 2004, predominaram soluções de transmissão de mídias ao vivo baseado no modelo de serviço cliente/servidor, com os sistemas sendo capazes de suportar algumas dezenas de

usuários por sessão. Em seguida, com base nas pesquisas sobre compartilhamento de arquivos considerando o modelo de serviço P2P, estenderam-se tal modelo de serviço para distribuição de mídias ao vivo. Naquela época (2005), destacou-se o projeto CoolStreaming, sendo o primeiro capaz de distribuir um mesmo evento para aproximadamente 40 mil nós clientes. Em 2008, os autores do CoolStreaming propuseram melhorias arquiteturais, propondo-se a mudança do método *pull* para um o método *push-pull* de *download* das partes de uma mídia.

Em paralelo, entre 2002 e 2005, publicaram-se as primeiras soluções de um protocolo de controle de congestionamento assistido pela rede chamado XCP [52]. Nesse contexto, os roteadores passaram a informar aos sistemas finais qual deveria ser a sua taxa de transmissão em um certo instante. As pesquisas evoluíram nesse sentido e em 2008 propuseram a primeira versão dos protocolos RCP [57] e do VCP [61], após a identificação de algumas limitações do XCP [55]. Em 2010, publicou-se o ConEx [67, 68], tendo como principal objetivo permitir que os sistemas finais compartilhem (com os roteadores) informações sobre o congestionamento experimentado pelos seus pacotes transmitidos. Desde então, não sugeriram novas propostas relevantes nesse contexto, exceto o ALTO (*Application-Layer Traffic Optimization*) [206, 207], um serviço que provê informações sobre as redes (por exemplo, localização, estrutura e rotas preferenciais), com o objetivo de modificar os padrões de consumo de recursos de rede e melhorar o desempenho das aplicações. O uso do ALTO em sistemas de distribuição de mídias ao vivo ocorre quando um nó cliente envia uma lista de possíveis nós parceiros e recebe uma lista ordenada dos melhores parceiros baseados em um determinado critério (por exemplo, capacidade de upload). Trata-se de uma solução centralizada em um modelo cliente/servidor e os próprios clientes são responsáveis por realizar as medições para determinar os valores dos critérios de seleção. Isto significa que um cliente pode prover informações incorretas sobre o seu estado atual, intencionalmente ou erroneamente devido à dinâmica da rede, pois é de responsabilidade da aplicação implementar o cliente ALTO para descobrir e conectar a um servidor ALTO. Além disso, algumas pesquisas apontam que o uso do ALTO não melhora o índice de continuidade de uma transmissão ao vivo, apesar de diminuir o tempo de inicialização para obter as primeiras partes de uma mídia [208]. Isto ocorre porque os nós clientes decidem com qual frequência atualiza seu estado no servidor (capacidade de transmissão, número de saltos entre o cliente e o servidor

etc.), que muda com frequência devido à dinâmica da rede. No GMTP, as medições são feitas pelos roteadores com base no protocolo RCP, decidindo-se quais serão os roteadores parceiros, de acordo com a capacidade de cada canal, auxiliado pelos nós servidores que transmitem um fluxo de dados multimídia.

Em 2011-2012, dentre as várias soluções para distribuição de mídias ao vivo (novas ou aprimoradas), várias se destacaram, como a Denacast, propondo-se um modelo híbrido P2P/CDN para distribuição de mídias ao vivo com base no CoolStreaming [36, 105, 209–212]. Nos dias atuais, existem diversas pesquisas para distribuição de mídias ao vivo através do uso do protocolo HTTP, permitindo-se que os nós clientes obtenham as partes da mídia que mais se adequar com a sua capacidade de recepção (adaptação dinâmica de fluxos de dados no lado do cliente). Nesse último caso, destacam-se o uso de CDNs comerciais em escala global, bem como soluções híbridas P2P/CDN. Até o presente momento, o esforço têm sido completamente em soluções na camada de aplicação.

Considerando a vertente das Redes Centradas na Informação o *Information Centric Networks* (ICN), em 2006, Van Jacobson introduziu o assunto em uma palestra proferida para um grupo de pesquisadores de várias partes do mundo, no GooglePlex, Montain View, Califórnia, EUA [213]. Esta palestra marcou o (re)início das pesquisas nesse área. Argumentou-se que o modelo de requisição origem/destino empregado na Internet não mais se sustenta, considerando o aumento significativo do tráfego de dados atual. A verdade é que alguns pesquisadores argumentam que essa vertente de pesquisa já existia desde os anos 1970, mas sem evolução até 2006 [48]. A partir de 2006, surgiram as seguintes propostas: DONA (*Data Oriented Network Architecture*), PSI (*Publish-Subscribe Internet Routing*), NetInf (*Network of Information*) e o CCN (*Content-Centric Networks*), atualmente conhecida pela sigla NDN (*Named-Data Networks*) [184]. Em 2013, propuseram-se o CCN-TV, o primeiro sistema para distribuição de mídias ao vivo em NDN e com suporte ao DASH. Na Seção 3.3, discute-se mais detalhes sobre as ICN, em especial a NDN, bem como sobre as propostas para distribuição de mídias ao vivo, como a CCN-TV.

Não obstante, neste trabalho, propõe-se o GMTP, apresentado em versão incipiente no final de 2011 e neste trabalho em sua versão aprimorada. No restante deste capítulo, confronta-se diversas propostas para distribuição de mídias ao vivo com o GMTP, ao passo que no Capítulo 4, detalha-se seu funcionamento. Dentre todas as soluções discutidas, considera-se

o CoolStreaming/Denacast e o CCN-TV as propostas fundamentalmente mais próximas ao GMTP. Por esse motivo, no Capítulo ??, compara-se o desempenho do GMTP, do CoolStreaming/Denacast e do CCN-TV em mais detalhes.

## 3.2 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 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.

### 3.2.1 *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*) [135], HDS (*HTTP Dynamic Streaming*) [214], o DASH *Dynamic Adaptive Streaming over HTTP* [136, 215], e o HSS (*Smooth Streaming*) [216], 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, incrementando o protocolo HTTP com funções para 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, HSS e DASH têm propósitos similares aos tradicionais

protocolos H.323, SIP, RTP e RTSP (Seção 2.1), porém com base no protocolo HTTP.

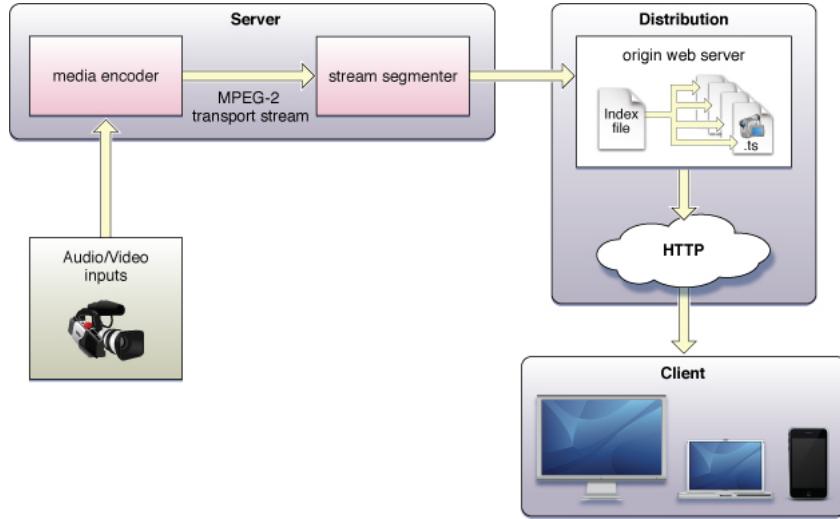


Figura 3.2: Arquitetura do HLS para transmissão de conteúdos multimídia baseado no protocolo HTTP. Figura extraída e adaptada de [135].

Uma característica comum entre os protocolos HLS, HDS, HSS e DASH é que a transferência dos fluxos multimídia ocorre sem a manutenção de uma conexão, já que o HTTP é um protocolo *stateless*. Além disso, os nós clientes podem escolher entre diferentes modos de codificação multimídia, o que naturalmente sugere uma solução para adaptação de mídias ao vivo de acordo com as condições da rede (em geral, capacidade de recepção de pacotes de dados observado pelo cliente). Nessas soluções que exploram a convergência dos serviços multimídia com base na web (Figura 3.2), especifica-se a fonte da mídia através de uma URI (*Uniform Resource Identifier*) [217], 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 2, pode-se afirmar que um segmento contém apontadores para as partes 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 3.3, ilustra-se um cenário de transmissão de um fluxo de dados

multimídia entre um servidor e um cliente 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 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.

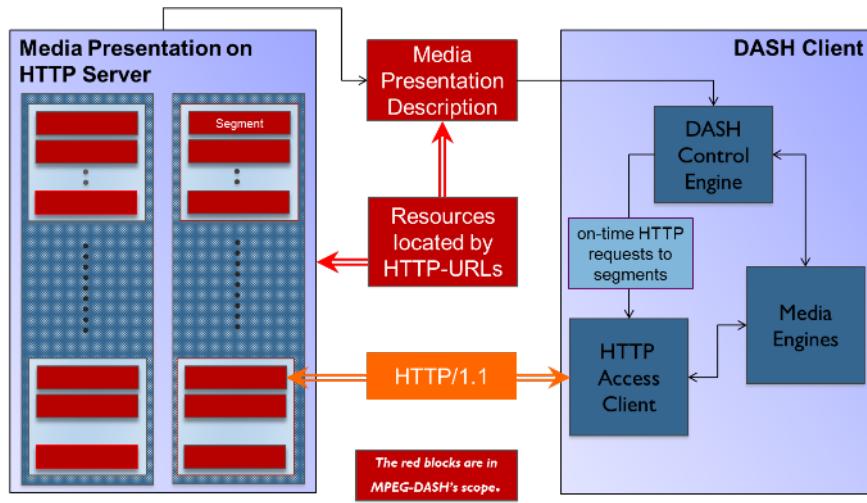


Figura 3.3: Arquitetura do DASH para transmissão de conteúdos multimídia utilizando o protocolo HTTP. Os formatos e as funcionalidades dos blocos vermelhos são definidos pela especificação do DASH. Adaptado de [136].

Uma aplicação baseada no protocolo HLS transmite, obrigatoriamente, fluxos de dados no formato MPEG2-TS (parte 1) e funciona apenas para o navegador Safari<sup>TM</sup> 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 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 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 per-

mite que inúmeros clientes, que suportam diferentes formatos e conectados através de redes com diferentes capacidades de recepção, possam reproduzir o conteúdo de forma adequada. Para que isso seja possível, os nós clientes devem monitorar sua capacidade de recepção de dados e selecionar o conteúdo multimídia com base em uma lista de diferentes taxas de bits anunciada pelo servidor. Por exemplo, se o cliente determinar que sua capacidade de recepção é de  $600\text{ Kbps}$  e o servidor anunciar a transmissão de vídeo em  $512\text{ Kbps}$ ,  $720\text{ Kbps}$ ,  $1\text{ Mbps}$  e  $4\text{ Mbps}$ , este deve selecionar receber o conteúdo a uma taxa de bits de  $512\text{ Kbps}$ .

### **Comparação entre as soluções HTTP e o GMTP para distribuição de mídias ao vivo:**

A proposta de distribuir conteúdos multimídia ao vivo utilizando HTTP é uma estratégia interessante pelo fato de se fazer uso de um protocolo bastante difundido, que praticamente está disponível em qualquer dispositivo e não é bloqueado em quase 100 % das redes que constituem a Internet. Isso pode ser considerado um ponto positivo em utilizar o HTTP para o fim que se discute, principalmente devido à facilidade de implantação da solução.

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 globalmente e melhorar os canais de transmissão, pois, em essência, 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 ocorre 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 cliente interessado pelo fluxo e assim sucessivamente. Sendo assim, o 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.

- 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* acarretará em maior sobrecarga de controle, e se diminuir a sobrecarga de controle, aumenta-se o tempo de atraso na recepção dos pacotes de dados [103, 218, 219]. 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 transmitidas em direção ao nó servidor para um mesmo conteúdo.
- 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 [220]. 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. No GMTP, utiliza-se um algoritmo de controle de congestionamento assistido pela rede [57] para **reduzem** o congestionamento nos pontos de interconexão, melhorando a qualidade dos fluxos de dados multimídia reproduzidos aos usuários

finais.

- Soluções baseadas em HTTP, devem considerar aspectos relacionados à segurança, descritos na Seção 15 da referência [221]. Os principais 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 comprometidos. No GMTP, utiliza-se um mecanismo de segurança que permite os roteadores validarem o conteúdo através de assinatura digital, podendo-se criptografá-lo com base em métodos tradicionais, como criptografias assimétricas, com suporte a diferentes métodos, tais como RSA e Curvas Elípticas.

### 3.2.2 PDTP – Peer Distributed Transfer Protocol

O protocolo *Peer Distributed Transfer Protocol* (PDTP) [222] surgiu 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<sup>1</sup>. Apesar de sua descontinuidade, ressalta-se alguns aspectos importantes sobre estratégias de distribuição de mídias ao vivo.

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-se 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 (discutido a seguir) 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

<sup>1</sup>DistribuStream: <http://freecode.com/projects/distribustream>

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 múltiplos *seeds*, porém se não existir nenhum *seed* disponível para um *torrent* o conteúdo fica inacessível. Na Figura 3.4, ilustra-se a organização geral dos nós PDTP.

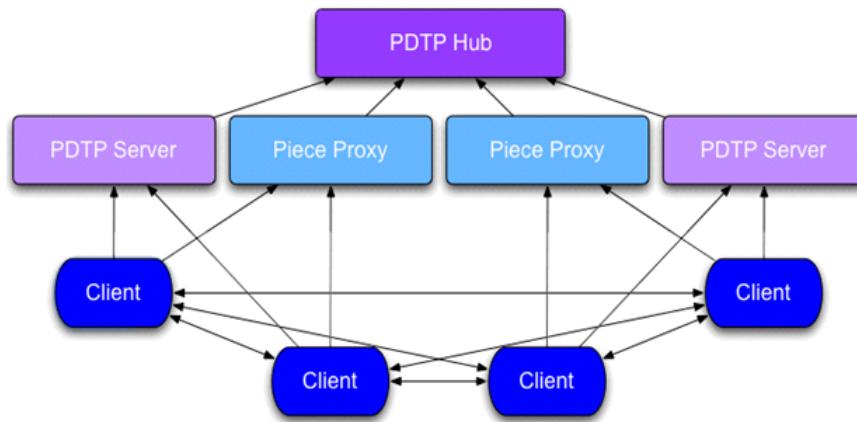


Figura 3.4: Organização dos nós PDTP. Adaptada de [222].

### **Comparação entre o PDTP e o GMTP para distribuição de mídias ao vivo:**

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 já discutidos no Capítulo 1. O uso de um servidor centralizador de índices pode ser um problema para distribuição de mídias ao vivo porque pode desfavorecer os nós clientes que estão mais distantes. NO GMTP, os servidores de mídia de uma rede CDN são também os indexadores de conteúdo, bem como auxiliam os nós roteadores a efetivarem as parcerias entre si.

### 3.2.3 CPM – Cooperative Peer Assists and Multicast

No *Cooperative Peer Assists and Multicast* (CPM) [112], propõe-se uma abordagem unificada para prover suporte eficiente de transmissão de vídeos sob 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 3.5, 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* relacionado ao conteúdo de interesse. Em seguida, 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.

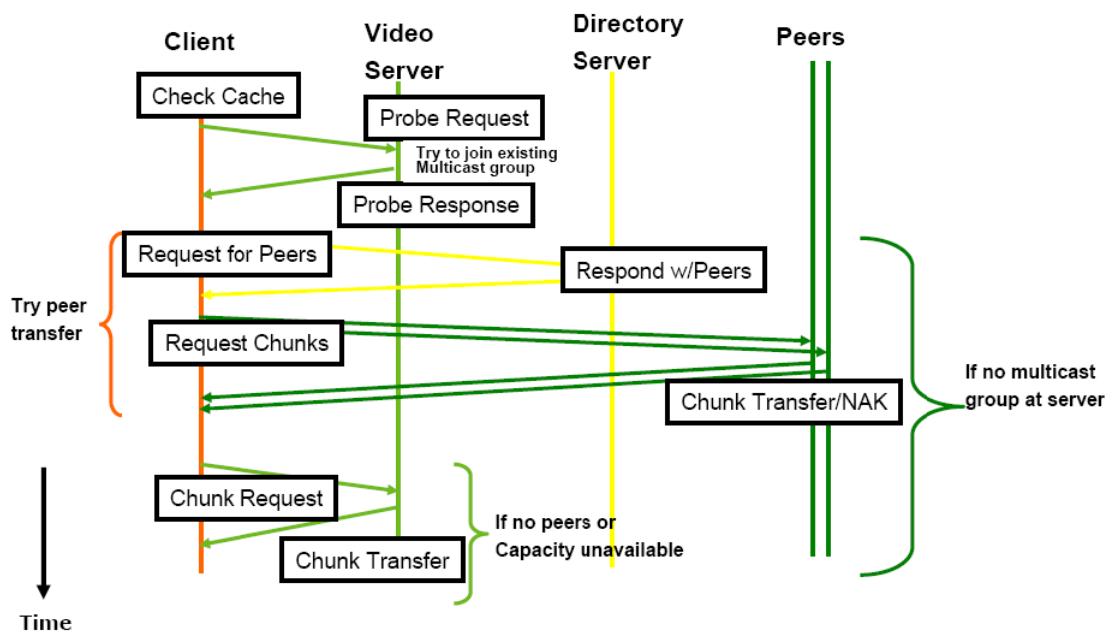


Figura 3.5: Diagrama de sequência do CPM (*Cooperative Peer Assists and Multicast*). Figura extraída de [112].

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 estar completamente armazenado no servidor para permitir que os nós façam cache dos pedaços e redistribuí-los *a posteriori*. 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 têm interesse pelo mesmo conteúdo.

### **Comparação entre o CPM e o GMTP para distribuição de mídias ao vivo:**

A capacidade para transmitir o conteúdo em modo híbrido (unicast/multicast) é um aspecto positivo para o CPM. A seguir enumeram-se os pontos fracos identificados.

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*. Além disso, o GMTP resolve a distribuição em *multicast* no próprio roteador e de forma dinâmica, sem uso de um servidor à parte.
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 aumentar a velocidade de *download*. No caso do GMTP, realiza-se uma duplicação controlada dos pedaços da mídia, realizada pelos próprios roteadores. Os nós clientes não têm influência sobre essa função, portanto não precisam gerenciar múltiplas fontes (outros nós clientes e servidores).
3. O uso de servidor de diretórios para consultar a lista de nós que mantêm o conteúdo multimídia desejado, não faz muito sentido para sistemas de transmissão de mídia ao vivo. No GMTP, não se utiliza essa abordagem.

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

No *Hybrid Delivery System with Adaptive Content Management for IPTV Networks* (HySAC) [223], 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.

Na Figura 3.6, ilustra-se a arquitetura do HySAC. Para evitar que a grande quantidade de usuários concorrentes sobrecarregue os servidores de mídia, propõe-se 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 utilizam Tabelas Dinâmicas de Hash (DHT) para encontrar os servidores que armazenam o conteúdo. Quando a localização de um conteúdo muda, informa-se aos servidores de indexação e quando um novo conteúdo é adicionado, os servidores replicam tal conteúdo de acordo com sua popularidade.

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 *ranking* de **popularidade** do vídeo e dependendo de um determinado limiar, o vídeo é selecionado para ser transmitido em modo *multicast*. Quanto mais alto for a popularidade do vídeo, maior é a chance deste de ser transmitido em modo *multicast*. Se a popularidade do vídeo for intermediária, o vídeo será transmitido em modo P2P e se for baixa, transmite-se o vídeo do servidor diretamente para o cliente em modo *unicast*.

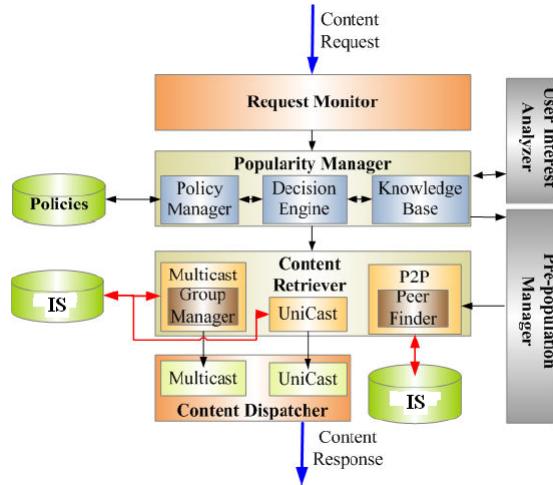


Figura 3.6: Arquitetura do HySAC (*Hybrid Delivery System with Adaptive Content Management for IPTV Networks*). Figura extraída de [223].

#### **Comparação entre o HySAC e o GMTP para distribuição de mídias ao vivo:**

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.

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, pois armazena-se as partes da mídia em cache no roteador.
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. No GMTP, em vez do critério de popularidade guiar as estratégias de distribuição das parte de uma mídia, utiliza-se um mecanismo de intersecção de rotas entre os nós clientes e o servidor.

### 3.2.5 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*.

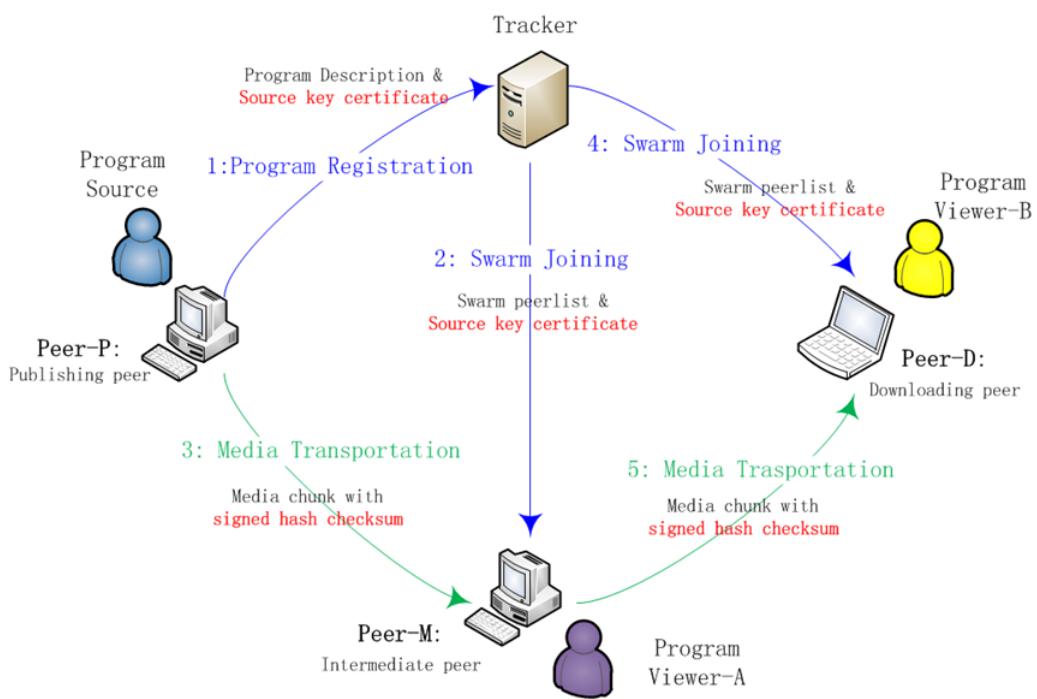


Figura 3.7: Arquitetura e funcionamento do protocolo PPSP/Swift.

O funcionamento básico do PPSP ocorre da seguinte forma (Figura 3.7). Um nó transmissor *Peer-P* notifica ao *tracker* que está transmitindo um certo fluxo de dados de mídia ao

vivo (Passo 1). Em seguida, o *tracker* transmite uma mensagem para os *Peer-M* e *Peer-D* interessados em receber o conteúdo multimídia para se juntarem ao grupo (Passos 2 a 4). O *Peer-P* transmite o conteúdo para o *Peer-M*, que repassa para o *Peer-D*. Em seguida, outros *peers* enviam uma mensagem ao nó *tracker* registrando interesse em receber o mesmo fluxo.

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 Multi-party Transport Protocol* (Swift). A responsabilidade do Swift no processo descrito é cuidar do transporte de dados entre os nós *Peer-M*, *Peer-D* e quaisquer outros nó interessado em receber um fluxo de dados multimídia. O Swift especifica o conteúdo de um *stream* contendo as partes da mídia, que não necessariamente constituem um conteúdo reproduzível. As partes da mídia são combinadas quando o cliente recebe o conteúdo a partir de diferentes fontes, quando então podem ser reproduzidos pelo nó cliente final. O Swift transmite os dados entre os nós utilizando o protocolo de transporte UDP com suporte de controle de congestionamento chamado de LEDBAT [167, 224], o mesmo empregado no BitTorrent.

#### **Comparação entre o PPSP/Swift e o GMTP para distribuição de mídias ao vivo:**

No PPSP/Swift, separa-se o mecanismo de sinalização e descrição da mídia da parte de transporte, levando-se em consideração uma proposta de distribuição de mídias através de uma abordagem de enxames. Comparado ao GMTP, os pontos fracos do PPSP/Swift são:

1. No PPSP/Swift, adota-se uma estratégia puramente P2P, onde os nós finais são responsáveis pela realização de parcerias e os servidores de mídias não realizam nenhum papel importante no processo de distribuição. No GMTP, adota-se uma abordagem híbrida P2P/CDN, onde os servidores tem a responsabilidade de instruir os roteadores de rede a realizarem parcerias entre si, ao passo que os nós clientes são coadjuvantes nesse processo, responsáveis apenas por ações simples no processo de distribuição de uma mídia ao vivo.
2. No PPSP/Swift, não há suporte nativo ao compartilhamento de conexão e transmissão em modo *multicast*. Se necessário, transmissões multicast devem ser disponibilizadas na camada de aplicação. No caso do GMTP, negocia-se dinamicamente os canais *multicast*, de acordo com as demandas locais de cada rede. Entre o servidor e o nó

cliente, transmite-se o fluxo de dados entre o servidor e o roteador do cliente em modo *unicast* e, no roteador, estabelece-se um canal *multicast* através do qual os nós clientes receberão as partes da mídia referentes a um evento a vivo. Nesse ínterim, roteadores intermediários entre o nó cliente e o nó servidor, podem interceptar o pacote de dados e também servir seus clientes locais (quando houver interesse), em um canal *multicast* negociado independente de outras redes.

3. 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 fundamental 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 na reprodução da mídia na camada de aplicação.
4. O uso do protocolo UDP e controle de congestionamento na camada de aplicação, enfraquece a ideia da organização dos protocolos em camadas funcionais, onde uma camada fornece serviços para a camada superior e, obviamente, usufrui de serviços da camada inferior. Os 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*. No GMTP, por ser um protocolo de transporte, abstrai-se as principais funções de distribuição de mídias ao vivo e considera o uso de *multicast* sempre que possível, com suporte a algoritmos de controle de congestionamento tanto no modo *unicast* quanto no modo *multicast*. Além disso, se duas aplicações independentes, em execução em dois nós distintos, utilizarem o protocolo GMTP, poderão compartilhar o conteúdo multimídia transmitido por um nó servidor, sendo tal recurso disponibilizado de forma nativa e transparente à qualquer aplicação.

### 3.2.6 DONet/CoolStreaming

O CoolStreaming é um sistema para distribuição de mídias ao vivo baseado em uma arquitetura P2P [82, 103, 225]. 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 está disponível para uso no mercado, servindo a milhares de nós globalmente 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<sup>2</sup> e o OMNet++/OSSim.

A sigla DONet significa *Data-drive Overlay Network* e CoolStreaming significa *Cooperative 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 repassam as partes de uma mídia para outros nós interessados pelo mesmo conteúdo, sem nenhuma regra pré-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 *chunks* transmitindo seu mapa de *buffer* com um conjunto de nós parceiros, obtendo os *chunks* ausentes a partir dos nós que os anunciam como disponíveis.

Conforme ilustra-se na Figura 3.8, o sistema CoolStreaming teve duas versões de produção. Na Figura 3.8(a), 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 Figura 3.8(b), ilustra-se o diagrama genérico da versão aprimorada do CoolStreaming, onde diversas funções foram corrigidas e o mecanismo de obtenção das partes da mídia foi remodelado para uma abordagem híbrida *push/pull* de obtenção dos blocos de vídeo (*chunks*). Na versão mais atual

<sup>2</sup>OMNet++/Oversim: de fato, o CoolStreaming está disponível através do projeto Denacast, desenvolvido com base no OMNet++/Oversim. Na Seção 3.2.7, detalha-se o Denacast.

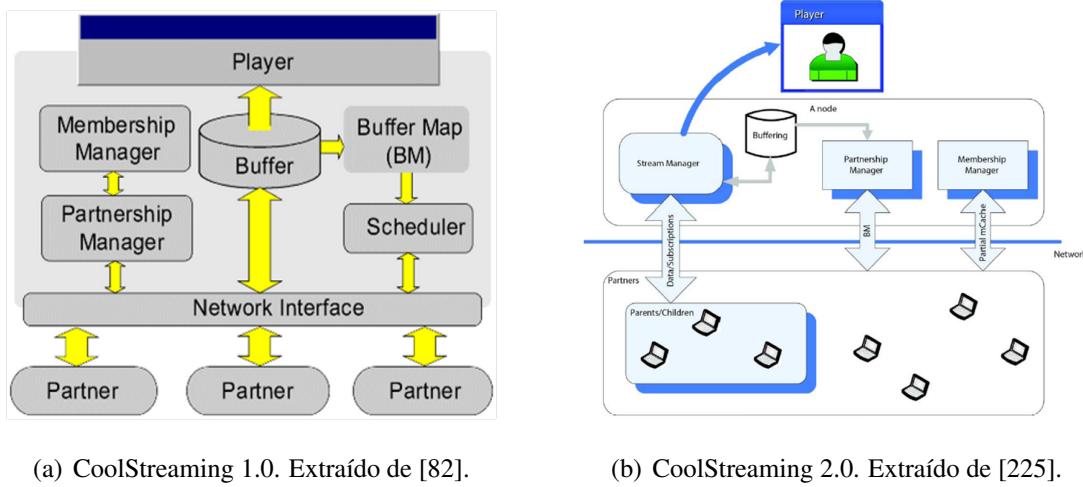


Figura 3.8: Arquitetura genérica de blocos funcionais do CoolStreaming 1.0 e do 2.0.

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 cliente. 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 *push/pull* mudou o mecanismo de obtenção de *chunks* para a seguinte forma. Quando um nó  $C_1$ , interessado em obter um determinado bloco da mídia, recebe de um nó parceiro  $C_2$  o mapa de buffer contendo a disponibilidade dos blocos do vídeo,  $C_1$  envia uma requisição à  $C_2$  solicitando os blocos desejados da mídia (ou seja, *pull*). Em seguida, o nó  $C_2$  transmite os blocos da mídia para  $C_1$  e, a partir desse momento, todo novo bloco da mídia 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 2.0 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 contendo os blocos da mídia. Este componente é responsável por escalar o momento

exato de enviar cada *chunk* correspondente de uma mídia, compartilhado-o com outros nós do sistema;

- *Buffer Map*, que representa o estado atual do *buffer* para um determinado vídeo. Como discutido no Capítulo 2, um nó pode sinalizar os blocos de vídeo que estão disponíveis ou os blocos da mídia que estão ausentes e portanto necessários.

Com relação ao conceito de sub-fluxo empregado no CoolStreaming, o nó transmissor divide o vídeo em blocos de tamanhos iguais e assinala-se um número de sequência em cada bloco, permitindo-se a reprodução do conteúdo em ordem. Dessa forma, cada nó pode obter diferentes blocos da mídia a partir de diferentes nós parceiros.

#### **Comparação entre o CoolStreaming e o GMTP para distribuição de mídias ao vivo:**

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 os principais aspectos positivos.

Com relação aos pontos negativos do sistema CoolStreaming, destacam-se:

1. O conceito de sub-fluxo adiciona complexidade à solução sem necessariamente resultado em ótimos resultados. Em [225], 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 evoluem positivamente, piorando em alguns casos (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 mudança 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 conteúdo 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 uma rede de sobreposição entre os roteadores e não entre os sistemas finais. Dessa forma, a rede se torna estável com relação a dinâmica de entradas e saídas de nós clientes, sendo possível continuar utilizando temporariamente os recursos de um roteador, mesmo quando seus nós clientes desistem de continuar obtendo a mídia de interesse (se o usuário fechar o aplicativo, por exemplo).
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 *bootstrap*. 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, pode ser que tal bloco de vídeo não esteja mais disponíveis, já que o nó cliente o remove após sua reprodução. 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*, além do mais, as últimas partes da mídia já podem estar disponíveis no roteador, quando há outros clientes anteriores recebendo o fluxo de dados correspondente. Outras decisões nesse sentido foram tomadas no GMTP, em destaque no Capítulo 4.

5. A seleção de nós no 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. Para isso, monitorase o *buffer* de recepção dado um sub-fluxo  $j$  transmitido por um nó  $C_1$  ao nó  $C_2$ , observando as inequações 3.1 e 3.2, 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 parceiros 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.

$$\max\{|H_{S_i, C_2} - H_{S_j, C_1}| : i \leq K\} < T_s \quad (3.1)$$

$$\max\{H_{S_i, q} : i \leq K, q \in \text{partners}\} - H_{S_j, C_1} < T_s \quad (3.2)$$

Um nó cliente utiliza a inequação 3.1 para monitorar o *status* do *buffer* do nó  $C_2$ . Se a inequação 3.1 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 3.2 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 3.2 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  $C_1$  a partir do conjunto *partners*.

Essa estratégia de seleção de nós aplicado no CoolStreaming é muito complexa porque exige o monitoramento 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 expressivo de pacotes de controle entre os nós parceiros e seus nós pais, além da transmissão de pacotes de dados contendo as partes da mídia. 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. Como já se sabe, no GMTP, os nós são os roteadores de rede, que simplesmente repassam para os clientes os fluxos de dados que contém partes da mídia. Esse assunto será retomado no próximo capítulo.

### 3.2.7 CoolStreaming/Denacast

O Denacast é um sistema para distribuição de mídias ao vivo baseado em um arquitetura híbrida P2P/CDN [36]. A porção P2P do sistema é adaptada do sistema CoolStreaming de modo que suporte a porção CDN. Segundo seus autores, o escalonador de distribuição de mídias do Denacast é idêntico ao do CoolStreaming, que é considerado o “coração” do sistema.

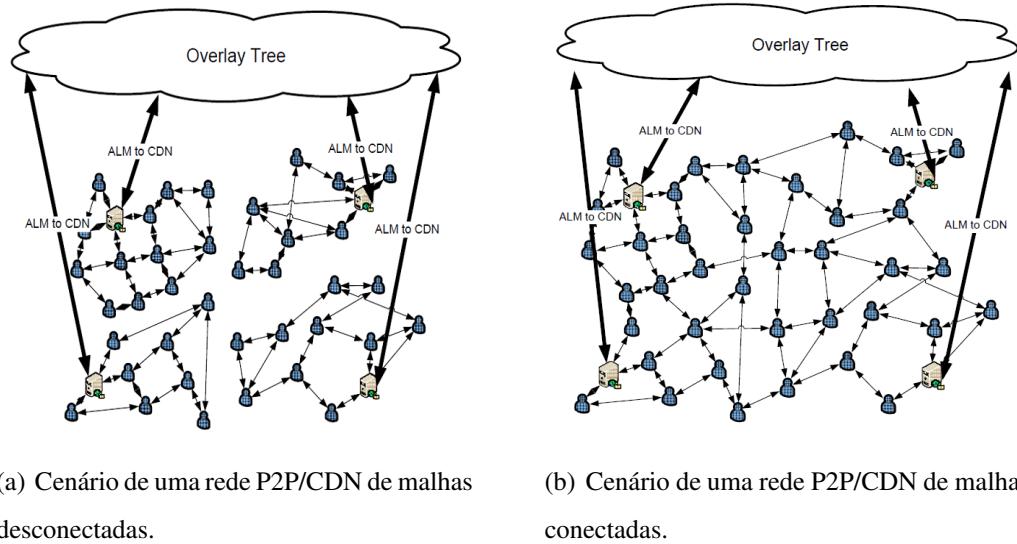


Figura 3.9: Arquiteturas CDN/P2P utilizadas no Denacast. Figuras extraídas de [36].

Como ilustra-se na Figura 3.9, 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. São elas:

- P2P/CDN de malhas desconectadas (Figura 3.9(a)): 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 3.9(b)): 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 estão 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*, informando-se 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.

#### **Comparação entre o Denacast e o GMTP para distribuição de mídias ao vivo:**

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 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 [36].

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 3.2.6, em discussão sobre o CoolStreaming e que se estende ao Denacast, o GMTP oferece funções diferenciadas e detalhadas no Capítulo 4. Por esse motivo, reserva-se um discussão mais detalhada sobre o Denacast em confronto com o GMTP para o Capítulo ??.

### 3.2.8 Outras propostas

Como discutiu-se no início deste capítulo (Seção 3.1), a área de distribuição de mídias ao vivo em larga escala tem sido explorada há pelo menos 10 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 anteriormente os principais. 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.

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 [80], BEAM/Alliances [25], BitTorrentLIVE [226], DLNA-P2P [227], GridMedia [101, 198, 219], IV5S [228], Joost [209, 210], LayeredCast [229], LiveSky [211], Octoshape [212], OverCast [230], Pastry/SplitStream [85], PeerCast [231], PPLive [104], PRIME [232], PROMISE [233], PULSE [105], SAMP [234], SmoothCache [235], Sopcast [102], TURINstream [236] e ZIGZAG [106]. 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.

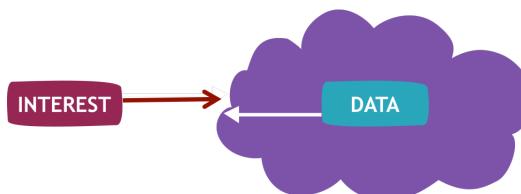
### 3.3 Redes Centradas na Informação

Desde da criação da Internet, diversos módulos complementares foram disponibilizados para atender às necessidades dos usuários, como o NAT, DNS com suporte a balanceamento de carga, etc. Entretanto, essas soluções criaram seus próprios problemas [237], que foram resolvidos por outras proposta [238] ou utilizadas de forma errada [239]. Em geral, essas mudanças têm levado a situações complexas de gerenciamento de rede, ao passo que tem sido disponibilizados avanços incipientes e lentos com foco no núcleo da rede. No entanto, como ilustra-se na Figura 3.10, uma nova vertente de pesquisa propõe a próxima arquitetura de serviço para a Internet considerando o princípio de que os nós da rede devem trocar dados com base no nome do conteúdo de interesse e não com base em caminhos (rede de telefonia), tampouco em sua localização (rede IP). Essa abordagem é conhecida por Redes Centradas na Informação (*Information Centric Networks – ICN*).



(a) Telefonia: nomeia o caminho.

(b) IP: nomeia os sistemas finais.



(c) ICN: nomeia o conteúdo.

Figura 3.10: Evolução das abstrações das redes de comunicação. Figuras de Van Jacobson.

Nas ICNs, objetiva-se facilitar a distribuição de conteúdo definindo rotas, funções de transporte e decisões de repasse com base no nome do conteúdo. Isto significa que um roteador pode fazer *cache* de um conteúdo bastante acessado e retorná-lo quando houver interesse por mais de um nó da rede. Atualmente (2014), todas as propostas para uma nova arquitetura para a Internet estão em estágio inicial de desenvolvimento, de modo que seus modelos de serviço e interfaces de transporte estão longes de uma versão final. Por isso, os

benefícios de utilizar tais arquiteturas para a distribuição de vídeo ao vivo são questionáveis, sem qualquer pesquisa que apresente resultados contundentes sobre seu uso em larga escala na Internet, principalmente no contexto de transmissão de mídias ao vivo [240]. Por exemplo, é impossível para um nó receptor adaptar o fluxo de dados de acordo com as características de uma rota de rede, pois as partes de um vídeo podem ser servidas por diferentes nós de *cache*. De fato, suportar a distribuição de mídias ao vivo usando ICN, fazem-se necessárias diversas melhorias no projeto arquitetural, como, desenvolver protocolos de transporte considerando o paradigma atual de transmissão fim-a-fim, tal como se propõe no GMTP.

Dentre as propostas de uma nova arquitetura para a Internet com base no princípio das ICNs, destaca-se a *Named-Data Networks (NDN)* [50, 185, 241] ou Redes de Dados Nomeados, ao passo que outras propostas ainda estão em estágio embrionário [46–48, 184, 242], como discutiu-se no início deste capítulo (Seção 3.1). Como a NDN apresenta diferenças substanciais em seu modelo de serviço para prover o conteúdo de interesse e funções disponíveis no núcleo da rede, a seguir, detalhe-se seu funcionamento e suas atuais aplicações no contexto de distribuição de mídias ao vivo.

### 3.3.1 Redes de Dados Nomeados (NDN)

A arquitetura NDN, originalmente conhecida por *Content-Centric Network (CCN)*, nomeia os pacotes de conteúdos e provê um modelo de serviço orientado a pedido e resposta, onde os nós obtêm os pacotes de dados a partir da rede, não necessariamente a partir dos nós finais. Como ilustra-se na Figura 3.11, os nós requisitam os pacotes de dados enviando uma requisição com o nome do conteúdo de interesse, que é único na rede. A definição dos nomes dos conteúdos seguem uma estrutura similar às URIs: os nomes são definidos em hierarquias, com componentes de tamanhos variados, por exemplo, */a/b/c.mp4*. Um nó cliente transmite tal requisição para obter um determinado conteúdo ao transmitir um pacote especial chamado de Pacotes de Interesse (*Interest Packets*). Em resposta ao pacote de interesse do cliente, algum nó da rede, em geral um roteador, transmite ao nó requisitante os pacotes de dados com o conteúdo requisitado (se o conteúdo estiver em *cache*). Toda interação ocorre sem o uso de endereçamento de origem/destino e, para cada envio de pacote de interesse, o nó requisitante recebe um pacote de dados, caracterizando uma relação estrita um-para-um entre os pacotes de interesse e os pacotes de dados.

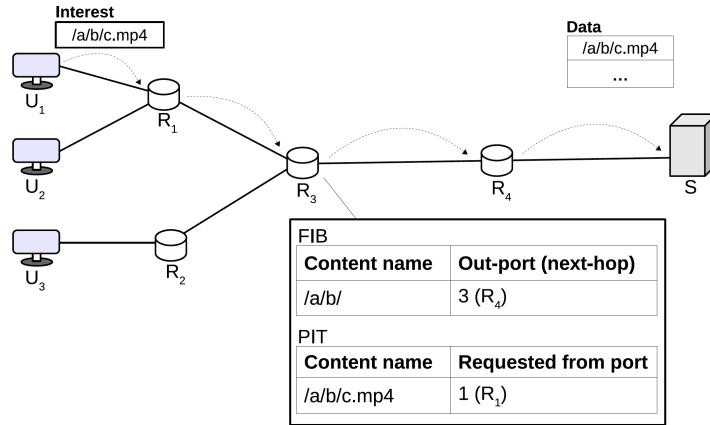


Figura 3.11: Um exemplo de interação da *Named-Data Network (NDN)*.  $U_1$  envia um pacote de interesse por  $/a/b/c.mp4$ , que está localizado em  $S$ . Com as setas, ilustra-se a direção de propagação do pacote de interesse. O pacote de dados segue o caminho inverso. Observa-se também as tabelas FIB e PIT para o roteador  $R_3$ . Figura extraída de [240].

Os roteadores propagam os pacotes de interesse e retornam os pacotes de dados utilizando o caminho inverso através do qual o pacote de interesse foi roteado. Isto acontece com o auxílio de três estruturas da dados, definidas por:

1. *Cache Store (CS)*: é um *cache* contendo os pacotes de dados que já foram repassados pelo respectivo roteador. Ao receber um pacote de interesse, o roteador verifica em seu *cache* a existência do pacote de dados correspondente. Se o pacote de dados estiver em *cache*, transmite-se imediatamente uma resposta de volta, através da interface que acabara de receber o pacote de interesse.
2. *Forwarding Information Base (FIB)*: é uma tabela de repasse baseada no nome do conteúdo. Ao receber um pacote de interesse, o roteador realiza uma verificação de correspondência mais longa (*Longest Prefix Match*) na FIB e então encaminha o pacote de interesse em direção ao nó fonte.
3. *Pending Interest Table (PIT)*: o roteador adiciona um pacote de interesse na tabela PIT antes de roteá-lo. Após roteá-lo, o roteador monitora suas interfaces de entrada verificando a recepção de pacote de dados correspondente ao pacote de interesse roteado anteriormente. Ao receber um pacote de dados, o roteador remove a entrada correspondente ao pacote de interesse e repassa o pacote de dados na direção inversa. Se o

roteador receber múltiplas requisições em diferentes interfaces, duplica-se o pacote de dados e o transmite através das interfaces correspondentes.

Além disso, os serviços de transporte de dados é orientado ao nó receptor, ao passo que os nós fontes não mantém qualquer estado de conexão, simplesmente respondem pacotes de dados à medida que recebem pacotes de interesse. Por exemplo, quando um nó cliente não recebe um pacote de dados em um determinando instante (*timeout interval*, similar ao TCP) ou se receber um pacote com erro verificado via mecanismo de *checksum*, o nó receptor simplesmente retransmite um pacote de interesse correspondente aos pacotes de dados perdidos. Os nós receptores também executam as funções de controle de fluxo e controle de congestionamento. Para isso, os nós receptores controlam a taxa em que os pacotes de interesse são transmitidos, utilizando-se um modelo baseado em janela deslizante, similar ao TCP [178].

### **Aplicações de transmissão de mídias ao vivo em NDN:**

O suporte de sistemas de distribuição de mídias ao vivo em NDN tem atraído o interesse da comunidade científica desde suas primeiras versões [185, 205, 243–247]. Em NDN, em vez de um nó servidor transmitir os datagramas diretamente para os sistemas finais receptores (UDP/IP), o servidor anuncia os datagramas na FIB e os nós receptores interessados em obtê-los requisita cada pacote de dados separadamente.

Um nó receptor inicialmente envia um pacote de interesse para o evento correspondente (nome do evento) e recebe um pacote de dados contendo metadados sobre o evento. O metadado mais importante é o esquema de nomes que o sistema utilizará para os próximos pacotes de dados correspondente à mídia transmitida, que também inclui o nome do último pacote de dados, permitindo o nó receptor construir os nomes para os próximos pacotes de dados e determinar em qual momento parar de enviar os pacotes de interesse. Após obter tais metadados, o nó interessado em obter o fluxo de dados começa a enviar pacotes de interesse e receber os pacotes de dados correspondentes.

Os sistemas de distribuição de mídias ao vivo em NDN precisam transmitir os dados da mídia à medida que estes são gerados. Porém, um nó receptor não pode transmitir um pacote de interesse para cada pacote de dado, pois receberia os pacotes de dados com atraso em pelo menos um RTT. Para minimizar essa limitação, a estratégia de transmitir mídias ao vivo em NDN consiste em permitir que o nó receptor envie pacotes de interesse para

pacotes de dados que ainda não existem. Por exemplo, considere um fluxo de dados de uma mídia ao vivo com o nome /a/b/c/live e que se transmite pacotes de dados há uma taxa de 100 pacotes por segundo (pps) e já se transmitiram 2000 pacotes de dados. Um nó receptor que desejar receber o referido fluxo de dados, estima o RTT (por exemplo, 100 ms) e decide transmitir pacotes de interesse há cada 2 s. Isto resulta em  $2 \times (100 \text{ pps} \times 100 \text{ ms}) = 20$  pacotes. Consequentemente, o nó receptor transmitirá 20 pacotes de interesse começando de /a/b/c/live/2001 até /a/b/c/live/2020 e assim sucessivamente. Quando os pacotes de interesse alcançarem o nó transmissor, aguarda-se até a geração dos pacotes de dados correspondentes e então os transmite imediatamente em direção ao nó receptor [240].

Atualmente, os trabalhos voltados para transmissão de mídias com base em NDN ainda são bastante primitivos. Nenhum trabalho existente na literatura aborda o assunto de distribuição de mídias ao vivo em larga escala, pelo contrário, os cenários considerados são bastante simples, compreendendo no máximo duas dezenas de nós clientes. O primeiro trabalho de transmissão de mídias ao vivo foi apresentado por Van Jacobson em uma aplicação chamada VoCCN [246]. De fato, não se discutiu nenhum aspecto relacionado a distribuição de conteúdos multimídia em larga escala, sendo o VoCCN apresentado apenas como uma prova de conceito para demonstrar a viabilidade do uso da CCN, como era chamada na época.

Em [243], os autores apresentam uma análise de uma solução experimental para transmissão de dados multimídia com adaptação de fluxo utilizando o protocolo HTTP, com suporte a uma versão modificada do DASH, discutido na Seção 3.2.1. Como metodologia do estudo, os autores avaliaram o desempenho e a sobrecarga de controle introduzida pela NDN em comparação do HTTP 1.0 e o 1.1 em rede TCP/IP. Com base nos resultados obtidos, concluiu-se que a NDN gera uma maior sobrecarga de dados de controle se comparado ao uso do protocolo HTTP 1.0 ou 1.1, apenas equiparando-se em alguns cenários ao protocolo HTTP 1.0, mas não ao HTTP 1.1. Em geral, observou-se que o impacto causado pelo aumento do atraso na rede no HTTP 1.1 é menor se comparado ao HTTP 1.0 e a NDN. Por exemplo, em um dos cenários experimentados, em uma rede com atraso de 150 ms, o HTTP 1.1 atingiu uma diferença de taxa de transmissão de 735 kbps (39 % maior) se comparado ao uso de DASH em NDN.

No trabalho [248], os autores propõem uma solução chamada de DASC (*Dynamic Adaptive Streaming over Content centric networking*), assemelhando-se ao anterior pelo fato de

utilizar o DASH em redes NDN, onde os nós transmitem pacotes de interesse para um determinado nível de qualidade do vídeo (*bit-rate*) de acordo com diferentes condições da rede. Como principal conclusão, os autores observaram o mesmo fenômeno observado no trabalho anterior, de que o modelo de pedido/resposta adotado pela CCN gera uma grande sobrecarga de controle na rede.

O trabalho [247] também se assemelha ao anterior no que diz respeito ao estudo de NDN combinada com o uso de DASH, com a diferença que se propõe o uso de uma rede P2P constituída entre nós móveis conectados em duas redes, uma celular (por exemplo, 3G) e a outra de proximidade (por exemplo, Wi-Fi Direct). Nesse contexto, os nós cooperaram entre si e sincronizam quais partes da mídia cada um irá obter via a rede celular, ao passo que os nós compartilham suas respectivas partes obtidas através da rede Wi-Fi e em modo *multicast*, considerando a função de roteamento baseado em nome empregada pela NDN.

Já em [185], os autores adotaram uma abordagem similar aos três primeiros trabalhos anteriores, porém, em vez de utilizar DASH, utilizou-se HLS, discutido também na Seção 3.2.1.

Diferentemente dos quatro trabalhos anteriores, em [205], os autores propuseram o CCN-TV, que é o único sistema de distribuição de mídias ao vivo existe, com base em uma rede CCN. Apesar de ainda incipiente, o CCN-TV foi o primeiro a sugerir o conceito de canal de transmissão, onde em cada canal se transmite um fluxo de dados de mídia ao vivo. Porém, por usar CCN, as partes da mídia continuam sendo requisitadas individualmente através do uso do pacote de interesse, como nos trabalhos anteriores. O CCN-TV identifica as partes de uma mídia por números progressivos e sua arquitetura é definida em três módulos principais: a fase inicial (*bootstrap*), estratégia para controle de congestionamento e o gerenciamento de retransmissão de pacotes de interesse. O módulo de fase inicial consiste em iniciar as sessões de mídia (canais) e envolve operações para encontrar uma rota para o provedor de canal mais próximo ao nó interessado em acessá-lo, além de localizar o primeiro identificador válido para uma mídia (por exemplo, o quadro I ao utilizar um codec MPEG) e assim poder iniciar a reprodução do conteúdo. Já o módulo de controle de congestionamento é baseado em uma abordagem de janela deslizante. Cada nó mantém uma janela deslizante de tamanho W para armazenar W partes pendentes da mídia. Um algoritmo monitora a janela removendo todas as partes da mídia que já não são mais úteis, ou seja, que já foram reproduzidas; retransmite os pacotes de interesse correspondentes a todas as partes da mídia que ainda não

chegaram após um dado tempo de expiração; e transmite um novo pacote de interesse para novas partes da mídia à medida que novos espaços se tornam disponíveis na janela. Por fim, o módulo de gerenciamento de pacotes de interesse permite a retransmissão de um pacote de interesse diretamente ao nó transmissor do pacote de dados, evitando que roteadores entre o nó interessado em obter o pacote de dados bloqueie o repasse do pacote de interesse por já está registrado em sua PIT.

Após a definição da arquitetura, os autores avaliaram o CCN-TV com vistas ao desempenho do sistema levando em consideração a quantidade de largura de banda necessária para executar o serviço de transmissão de um fluxo de dados ao vivo; o tempo de expiração utilizado no mecanismo de controle de congestionamento; o atraso de inicio de reprodução e quão efetivo é a política de *cache* adotada no CCN para disseminação das partes de uma mídia. Como resultado, concluiu-se que os recursos de *cache* empregado pela CCN pouco contribui para uma melhor disseminação das partes de uma mídias ao vivo, pois estas são transitentes. Entretanto, os autores observaram que a política de gerenciamento da PIT modificada, alinhada ao modelo de controle de congestionamento baseado em janela deslizante, pode reduzir a sobrecarga de controle gerada por excessivas transmissões de pacotes de interesse.

### **Comparação entre as redes ICN (em especial a NDN) e o GMTP para distribuição de mídias ao vivo:**

Com relação a arquitetura de rede ICN, em especial a NDN, e o GMTP, enumeram-se as seguintes comparações:

1. Como discutiu-se no Capítulo 1, as aplicações de transmissão de mídias ao vivo possuem algumas peculiaridades que precisam ser tratadas nas camadas abaixo da aplicação, sendo praticamente impossível generalizar uma infra-estrutura que sirva para fluxos de dados elásticos e inelásticos. Em geral, as redes ICN estão tendendo a considerar sua arquitetura mais genérica possível para permitir diferentes padrões de tráfego, mas isso não é trivial. Por outro lado, no GMTP, decidiu-se partir dos requisitos específicos dos sistemas de distribuição de mídias ao vivo e questionar sobre quais funções que se utilizam nesses sistemas que podem ser generalizadas pela rede. Sendo assim, o GMTP adiciona à rede IP, originalmente proposta para transportar fluxos de

dados em sua maioria elásticos, a capacidade de prover funções comuns a todas as aplicações que transmitem fluxos de dados inelásticos.

2. Como consequência do item anterior, a rede NDN transmite mais pacotes de controle do que o GMTP por ter uma proposta de modelo de serviço mais genérico, baseado em *pulling*. No GMTP, propõe-se um modelo de serviço híbrido *push-pull*. Por ter um modelo de serviço mais específico para sistemas de distribuição de mídias ao vivo, no GMTP o mecanismo de *cache* das partes de uma mídia é mais simples de se implementar, pois utiliza-se uma estrutura de *buffer* circular, substituindo-se as partes da mídia à medida que se recebe as subsequentes, sem a necessidade do roteador transmitir, a todo instante, o equivalente ao que seria um pacote de interesse – o conceito de pacote de interesse da NDN não existe para o GMTP. Em CCN, um dos grandes desafios é definir quanto tempo um dado deve permanecer em *cache*. Um outro desafio é determinar se o dado em *cache*, principalmente em aplicações elásticas, ainda é válido. Todas essa funções, aumentam a sobrecarga de controle nas redes ICN e, como já se sabe, em redes IP essas questões não é um problema. O GMTP apenas instrui os roteadores a realizar uma política mais sofisticada de repasse quando um pacote de dados é alocado na fila de roteamento, aproveitando a oportunidade de repassar o mesmo pacote em rotas adicionais, de acordo com a demanda.
3. Para executar as aplicações de rede existentes na Internet sobre uma rede ICN, serão necessárias alterações mais específicas nos sistemas de transmissão de mídias ao vivo. Em ICN, faz-se necessário que todas as aplicações mudem a lógica de requisição das partes da mídia, sendo necessário também alterar o esquema de identificação do conteúdo de interesse, que passa a ser por nome e não mais por endereço IP e número de porta. Além disso, há uma inversão na forma que uma aplicação obtém os dados quando se utiliza NDN. Em NDN, o nó cliente controla praticamente todos as funções de transporte de dados, como controle de perda/erro, controle de congestionamento e fluxo etc. Entretanto, atualmente, um nó cliente para obter um conteúdo multimídia, estabelece uma conexão informando implicitamente que deseja receber aquele fluxo, pois o endereço IP e porta já define indiretamente qual é o conteúdo. Ao perceber essa relação implícita, no GMTP, praticamente não será necessário alterar a aplicação,

apenas mudar o valor do parâmetro do protocolo de transporte atual (em geral, UDP), pelo valor do protocolo GMTP. No GMTP, extrapola-se os limites da constituição de uma rede P2P na camada de aplicação. Em vez das aplicações realizarem tal função, os próprios roteadores realizam as parcerias entre si, ocorrendo no contexto de execução do GMTP. Com isso, não se delega à aplicação qualquer atividade específica sobre obtenção do conteúdo de interesse, seleção de nós, controle de congestionamento e definição de canais multicast. O esquema de requisição continua o mesmo, ou seja, os sistemas finais solicitam o conteúdo por endereço IP e porta, com a diferença que o GMTP internamente gera um nome (*hash*) combinando esses dois parâmetros, utilizando-o para implementar funções de compartilhamento de um mesmo fluxo de dados ao vivo e instruir os roteadores que realizem parcerias determinadas parcerias.

4. Nas redes ICN, o conceito de comunicação fim-a-fim foi abolido e sequer utiliza-se o mecanismo de endereçamento IP. Isto pode gerar um conflito sobre até quando um pacote de interesse circulará até encontrar um roteador que possua o pacote de dados correspondente. No caso do GMTP, uma requisição feita por um nó cliente para obter um fluxo de dados ao vivo continua sendo transmitida em direção ao servidor de origem. Se nenhum roteador interceptar o pedido, todo o processo ocorre como em uma conexão fim-a-fim tradicional. No entanto, se qualquer roteador presente na rota entre o nó cliente e o servidor já estiver repassando o pacote de dados por aquele caminho ou para outra interface, intercepta-se o pedido de conexão e responder como se fosse o nó servidor.
5. No ICN, o mecanismo que determina a interceptação de um pacote de interesse é limitado a verificação local se o fluxo de dados já está sendo recebido ou não, sem qualquer auxílio de qualquer outro nó. Por exemplo, no NDN, o nó servidor tem um papel co-adjuvante, apenas responde pacotes de dados quando recebe um pacote de interesse, sem manter estado de conexão. No GMTP, o nó servidor tem um papel importante no processo de distribuição ao auxiliar a rede a decidir qual é a melhor rota para obter um fluxo de dados com base na interceptação de duas ou mais rotas conhecidas. De fato, há uma troca de serviços, pois o roteador também auxilia o servidor informando qual deve ser sua taxa ideal de transmissão. Já o auxílio do servidor à rede ocorre porque

no procedimento de conexão do GMTP, registra-se no pacote de pedido de conexão, os identificadores de cada roteador. Esse identificador é único em toda rede e gerado pela combinação dos endereços MAC de todas as interfaces de rede. Ao receber um pacote de pedido de conexão, o nó servidor conhece por quais roteadores o pacote passou e então sugere a intersecção de outros caminhos já conhecidos, se possível, caso contrário, responde normalmente ao nó cliente.

6. Nas redes ICN, não se tem discutido claramente sobre o uso de soluções mais avançadas de controle de congestionamento, sendo as propostas focadas em algoritmos tradicionais baseado em janelas deslizantes ou similares. Em diversas pesquisas recentes, demonstrou-se que utilizar os roteadores de rede para auxiliarem os sistemas finais no processo de controle de congestionamento, obtêm-se um melhor desempenho da rede, reduzindo-se o atraso e finalizando-se mais rapidamente os fluxos de curta duração, bem como permitindo o compartilhamento mais equânime dos recursos de rede. No GMTP, optou-se por utilizar tal abordagem, ainda estendendo-a ao permitir que o nó servidor sugira aos roteadores quais parcerias devem ser efetivadas, baseando-se na capacidade atual de transmissão dos canais conhecidos. Já nas ICNs, com pôde-se constatar, as pesquisas do estado da arte estão concentradas em fazer com o que o nó servidor disponibilize múltiplos fluxos de dados codificados em diferentes *bit-rates*, ao passo que os nós clientes devem monitorar sua capacidade de recepção e requisitar o fluxo apropriado. Porém, monitorar a capacidade de transmissão dos sistemas finais baseado apenas nas perdas de dados e atraso não é uma solução efetiva devido à dinâmica de entrada e saída dos nós da rede (*churn*).
7. Em NDN, requisitar cada pacote de dados pode introduzir uma significativa sobrecarga à rede, devido ao uso de largura de banda para transmitir a quantidade de pacotes de interesse igual à quantidade de pacotes de dados, aumentando a ocupação dos *buffers* dos roteadores. Como consequência, a rede NDN pode apresentar problemas de desempenho e escalabilidade, podendo apresentar significativos níveis de congestionamento. Para sistemas de distribuição de mídias ao vivo, isto pode ser ainda mais grave, pois estão sujeitos às condições de rede do canal de *upload* (do nó receptor ao nó transmissor, já que o nó cliente controla os serviços de transporte). Sendo assim, em

caso de congestionamento no canal de *upload*, os pacotes de interesse podem sofrer atrasos ou descartes. Consequentemente, a rede não será capaz de transmitir os pacotes de dados correspondentes em tempo hábil ou nunca transmitir, caso o pacote de dado correspondente expire no cache de todos os roteadores, levantando questões sobre qual é o melhor momento de desistir de requisitá-lo e requisitar o próximo. Sendo assim, mesmo em cenários onde o canal de *download* esteja livre, os nós receptores podem experimentar uma baixa qualidade de serviço devido às perdas de pacotes de interesse, não necessariamente de pacotes de dados. Esses cenários são típicos em serviços de conexões residenciais, onde os *enlaces* são geralmente assimétricos em termos de largura de banda, por exemplo, ADSL. Atualmente, existe um estudo sobre introduzir uma função chamada de agregação de pacotes de interesse [240]. Nesse caso, um único pacote de interesse agregaria a requisição de múltiplos pacotes de dados. Entretanto, essa proposta ainda não é oficial e acarreta em outro problema. Por exemplo, a perda de um pacote de interesse agregado resultará na perda de múltiplos pacotes de dados, impactando diretamente na qualidade de serviço de uma aplicação de transmissão de mídia ao vivo. No ponto de vista prático, o usuário experimentará uma rajada de perda de pacotes de dados e a reprodução contínua será comprometida. Além disso, os pacotes de dados em NDN tem MTU de 8800 *bytes* ou de 4096 *bytes*, ao passo que em redes IP utiliza MTU, em geral, de 1500 *bytes*, ou seja, uma perde de um único pacote em NDN, perde-se mais dados de aplicação se comparado às redes IP.

## 3.4 Sumário do Capítulo

Neste capítulo, apresentou-se uma avaliação crítica acerca de um conjunto de propostas para distribuição de mídias ao vivo, considerando-se os aspectos arquiteturais e de modelo de serviço de cada solução. Destacou-se as propostas concorrentes, contrapondo-se as limitações de cada uma frente ao GMTP, esclarecendo-as individualmente.

No próximo capítulo, detalha-se o funcionamento do GMTP e, no Capítulo ??, retoma-se outra discussão comparativa, porém exclusivamente sobre o desempenho do GMTP frente aos seus principais concorrentes: o Denicast e o CCN-TV.

## Capítulo 4

# Global Media Transmission Protocol (GMTP)

O *Global Media Transmission Protocol* (GMTP) é um protocolo que atua nas camadas de transporte e de rede (*cross-layer*), projetado para operar na Internet, a ser utilizado em sistemas de distribuição de fluxos dados multimídia ao vivo. Trata-se de um protocolo baseado em uma arquitetura híbrida P2P/CDN, transmitindo-se os dados de um ou mais sistemas através de uma rede de favores P2P constituída por roteadores de rede, que cooperam entre si a fim de obterem o conteúdo multimídia de interesse, ao mesmo tempo que ocorrem interações entre os servidores de uma ou mais redes CDNs, os quais atuam como super nós para os nós da rede P2P, auxiliando-os no envio e recebimento dos fluxos de dados. Uma aplicação GMTP em execução nos nós clientes, reproduz o conteúdo multimídia ao usuário final à medida que recebem pacotes de dados contendo partes da mídia, geradas pelos servidores da CDN, através de um processo em execução se comunicando através da rede com base no protocolo GMTP. Nesse ínterim, os roteador envolvidos na transmissão de uma mídia ao vivo, realiza parcerias com outros roteadores, os quais também possuem nós clientes interessados no mesmo conteúdo, motivados pelos seus respectivos usuários finais que o controlam.

As trocas de dados entre nós GMTP ocorrem por meio do envio e recebimento de partes de uma mídia (*chunks*), que são transmitidas por diferentes nós da rede, constituindo um fluxos de datagramas IP. Estes fluxos são transmitidos em modo *unicast* e compartilhados (*multi-unicast*) pelos roteadores, quando são entregues aos nós clientes em modo *multicast*,

realizando-se controle de congestionamento sem garantia de entrega, em ambos os modos de transmissão. A escolha do modo de transmissão para disseminar um fluxo de dados ocorre sem a influência da aplicação, que precisa simplesmente “sintonizar” sua conexão em um determinado canal *multicast* definido pelo roteador. Tal abstração para a camada de aplicação ocorre de modo que os processos em execução utilizam o GMTP através de uma API compatível com as especificações de socket BSD e POSIX, o que permite adaptações mais simples das atuais aplicações de rede de transmissão de mídias ao vivo.

Por conseguinte, com o uso do GMTP, permite-se o estabelecimento de conexões e a cooperação entre diferentes fornecedores de aplicações, em prol de tão logo quanto possível obter as partes de uma mídia a serem reproduzidas. Isto significa que o GMTP torna as aplicações compatíveis entre si, uma vez que o protocolo desacopla a forma como os dados são transportados da forma como estes são exibidos ao usuário final, emulando os sistemas tradicionais de TV. Sendo assim, promove-se a integração do GMTP em aplicações já existentes, quando se considera futuras adoções de tal protocolo, ao passo que se permite a utilização dos novos recursos introduzidos no protocolo, reduzindo-se a complexidade na construção de sistemas de distribuição de mídias ao vivo, especialmente aqueles baseados em arquitetura P2P/CDN.

Com base na ilustração da Figura 4.1, nas próximas seções deste capítulo, detalham-se os aspectos teóricos e computacionais empregados do GMTP, a fim de construir uma rede de sobreposição formada por roteadores, pela execução de quatro grandes passos:

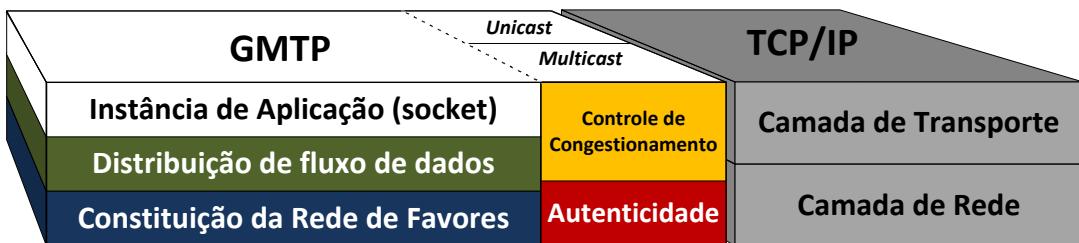


Figura 4.1: Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP.

1. *Constituição da rede de favores*: descobrir, definir, efetivar e desfazer parcerias entre os roteadores de acordo com o evento ao vivo a ser transmitido.
2. *Distribuição de fluxos de dados através de uma camada de socket*: conectar os nós

clientes interessados em receber um fluxo de dados de um evento ao vivo, bem como transmitir tal fluxo de dados através da rede de sobreposição constituída no Passo 1.

3. *Controle de congestionamento*: controlar a taxa de transmissão dos fluxos de dados distribuídos e utilizar as informações sobre a capacidade de transmissão de um canal para sugerir novas parcerias.
4. *Autenticidade do conteúdo*: verificar a autenticidade do fluxo de dados antes de repassá-los aos nossos clientes, evitando-se ataques de poluição.

Com base nesse passos, organizou-se a estrutura deste capítulo da seguinte forma:

- Na Seção 4.1, apresenta-se uma visão geral do protocolo, como cenário de atuação, arquitetura, canais de comunicação e tipos de nós e pacotes.
- Na Seção 4.2, formalizam-se as definições e restrições do protocolo.
- Na Seção 4.3, descrevem-se o processo de constituição da rede de favores e os aspectos de conexão multi-ponto através da introdução de conceitos como sockets P2P, registro de participação de um nó e a seleção de nós parceiros.
- Na Seção 4.4, discutem-se os aspectos de transmissão e recepção de fluxos de dados, relacionando os algoritmos utilizados para compartilhar os fluxos de dados e as estratégias de disponibilização e obtenção das partes de uma mídia.
- Na Seção 4.5, apresentam-se detalhes de funcionamento dos algoritmos de controle de congestionamento utilizados no GMTP, bem como estes influenciam no processo de formação de parcerias.
- Na Seção 4.6, discutem-se os aspectos relacionados à autenticidade de um fluxo de dados.
- Na Seção 4.7, apresentam-se outros aspectos relacionados ao GMTP, tais como finalização de conexão, tolerância à desconexão e eleição de nós relatores para o funcionamento do algoritmo de controle de congestionamento em modo *multicast*.
- E, por fim, na Seção 4.8, apresenta-se o sumário deste capítulo, elencando brevemente os principais pontos discutidos.

## 4.1 Visão Geral

O protocolo GMTP é composto por dois módulos chamados de *GMTP-Intra* e *GMTP-Inter*, que operam na camada de transporte e de rede, respectivamente, definindo assim sua arquitetura, ilustrada na Figura 4.2.

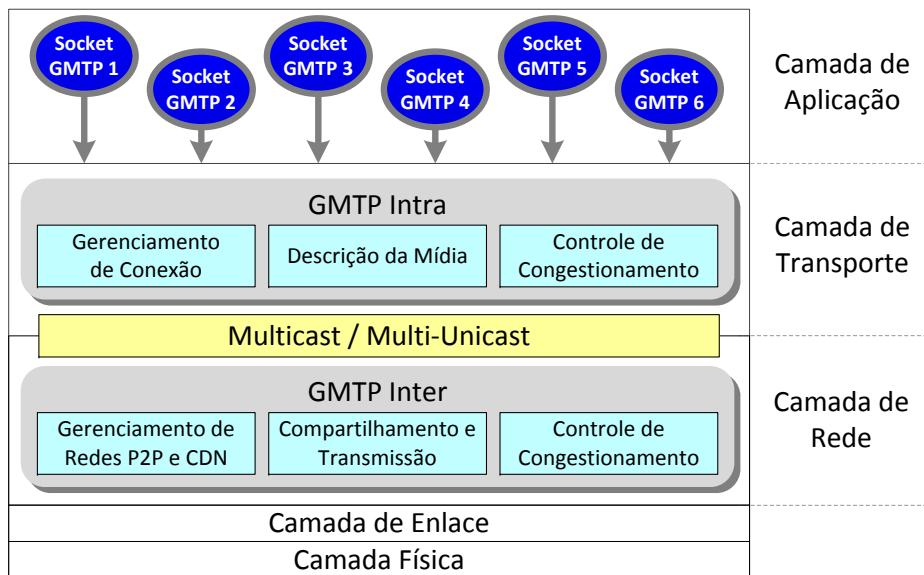


Figura 4.2: Arquitetura do Protocolo GMTP.

As responsabilidades dos módulos GMTP-Intra e GMTP-Inter são:

- **GMTP-Intra:** fornecer serviços às aplicações de rede a fim de abstrair a complexidade na execução de tarefas comuns a qualquer sistema final, tais como conexão multiponto, multiplexação/demultiplexação de segmentos IP entre as camadas de transporte/rede/aplicação e controle de congestionamento. Este módulo compreende a instância do GMTP em execução no sistema operacional do nó cliente, acessível através de uma API de socket GMTP. Um socket GMTP é a representação de uma instância do protocolo GMTP em execução, sendo responsável por gerenciar todas as atividades de comunicação da aplicação correspondente ao meio externo (outros processos GMTP). No contexto de uma conexão, o GMTP-Intra mantém diversas variáveis de estado relacionadas à execução dos algoritmos para gerenciamento de conexão (estabelecimento e desconexão), controle de congestionamento *multicast*, multiplexação e demultiplexação dos datagramas, eleição de nós relatores e determinação do formato e

preenchimento dos parâmetros que definem uma mídia, permitindo-se que a aplicação defina os valores de tais parâmetros ou obtenham acesso aos seus valores.

- **GMTP-Inter:** constituir uma rede de favores P2P composta por roteadores, os quais funcionam como pontes de acesso aos servidores de uma rede CDN. Trata-se do módulo em execução nos roteadores que cooperam entre si para constituir a rede de favores, aceitando conexões oriundas de um nó cliente, bem como instruções do nó servidor sobre formar parcerias com outros roteadores. No contexto de uma conexão, o GMTP-Inter mantém variáveis de estado relacionadas às funções de sua responsabilidade, tais como estabelecimento de conexão com nós servidores ou roteadores, seleção de nós roteadores parceiros, eleição de nós relatores, compartilhamento de fluxos multimídia e controle de congestionamento assistido pela rede. No GMTP-Inter, permite-se a configuração de parâmetros iniciais de configuração da rede de favores e da integração com servidores de uma ou mais CDN, como ilustra-se na Figura 4.3. Nesse caso, o usuário administrador de um nó repassador pode definir os seguintes parâmetros:

- configurações sobre registro de participação em uma ou mais redes CDN;
- largura de banda máxima (*download* e *upload*) que o nó repassador está autorizado a compartilhar;
- o período que o roteador funcionará como nó repassador (dias e horários);
- quantidade máxima de parcerias que podem ser realizadas;
- quantidade máxima de fluxos de dados que podem ser compartilhados;
- parâmetros avançados relacionados aos algoritmos de controle de congestionamento. E
- configurações acerca dos certificados digitais, tais como *download* automático e realização de *cache*.

Para viabilizar a disseminação de conteúdos multimídia, cada nó roteador localizado no caminho entre o servidor transmissor da mídia e o cliente interessado em obtê-la, pode repassar os pacotes de dados para seus clientes locais e replicá-los para outros roteadores

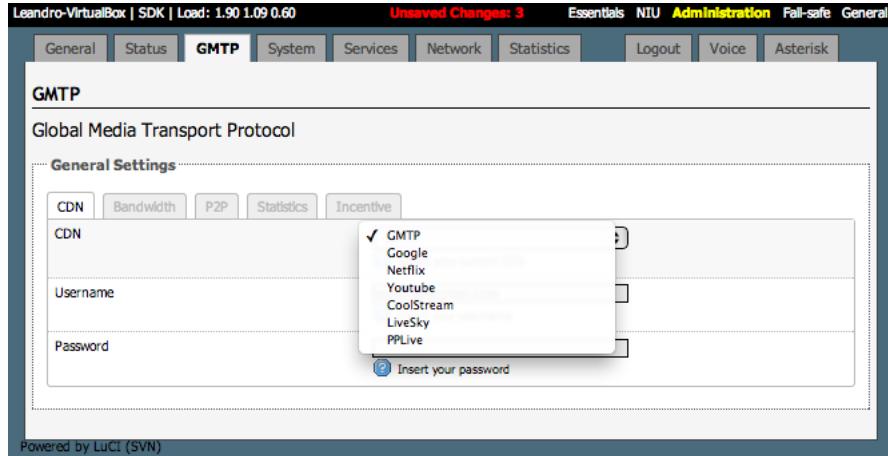


Figura 4.3: Tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP. Nessa tela, permitir que o administrador do roteador configure parâmetros do módulo GMTP-Inter.

interessados em receber o fluxo de dados correspondente, motivados pelos interesses de seus respectivos clientes. Sendo assim, permite-se que um roteador atenda à demanda dos seus clientes locais, ao passo que ajuda os outros roteadores a fazerem o mesmo, evitando múltiplas conexões para obter um mesmo fluxo de dados no nó servidor.

Na Figura 4.4, observa-se o cenário geral de atuação do protocolo GMTP, onde ilustram-se os nós *Clientes GMTP* interessados em obter o conteúdo de um determinado evento ao vivo. Neste caso, observa-se também um *Servidor GMTP*, que está conectado a uma rede CDN e atua como fonte geradora de dados; ao passo que os nós *Clientes GMTP* se conectam a um nó *Repassador GMTP* que é um roteador de rede. Com base na Figura 4.4, definiu-se a Figura 4.5, onde se observa os seguintes tipos de nós GMTP:

- **Cliente GMTP:** é capaz de reproduzir e gerar conteúdos multimídia ao vivo. Em geral, um *Cliente GMTP* é um sistema final que executa um processo a nível de sistema operacional, representando uma aplicação manipulada pelo usuário final. A maioria dos *Clientes GMTP* funciona apenas de forma passiva, recebendo o fluxo de dados de um conteúdo multimídia e entregando-o para um processo de aplicação em execução, sendo um sub-conjunto destes, contribuidores efetivos no processo de execução do algoritmo de controle de congestionamento em transmissões *multicast*.
- **Servidor GMTP:** é um sistema final que participa de uma rede CDN e obtém a mídia a ser transmitida através de três formas: i) diretamente a partir de uma unidade gera-

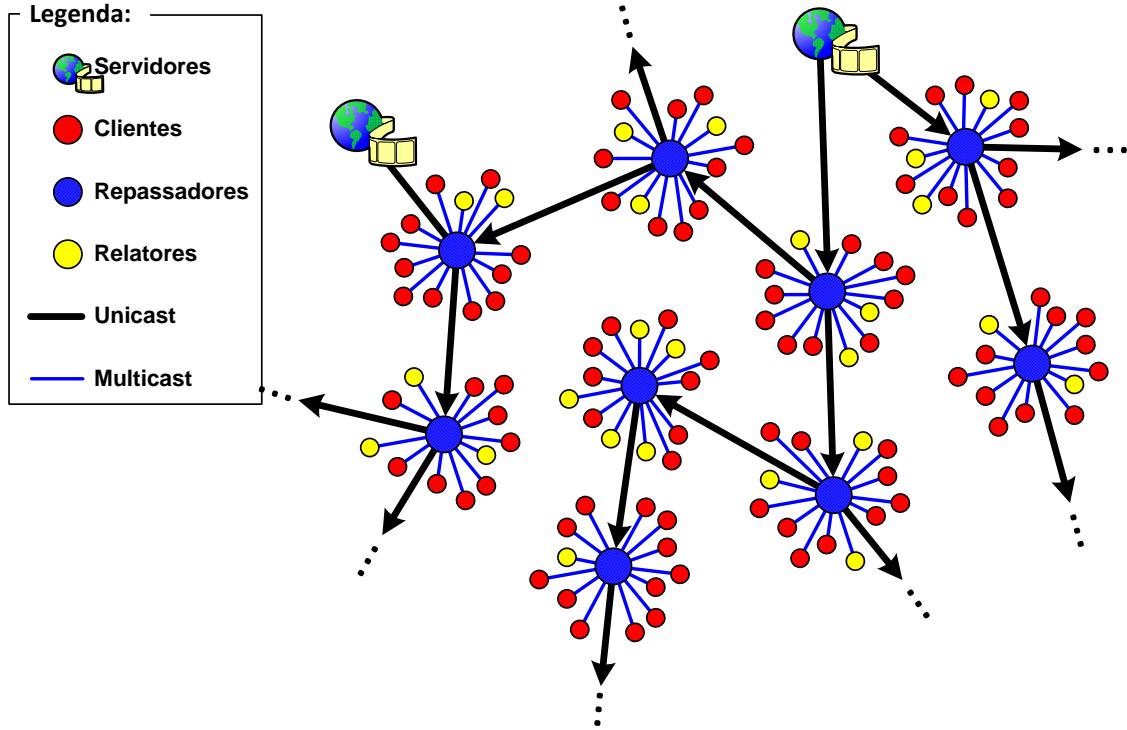


Figura 4.4: Rede de sobreposição construída dinamicamente pelo GMTP com a presença de nós repassadores e relatores.

dora de conteúdo (filmadora e/ou microfone); ii) a partir de um *Cliente GMTP*; e/ou iii) a partir de outro *Servidor GMTP* (troca de dados entre os servidores da CDN). Os *Servidores GMTP* recebem sinalizações de controle contendo requisições dos nós *Repassadores GMTP* que, ao receberem uma resposta correspondente a sua requisição, atendem à demanda de um ou mais nós *Clientes GMTP*.

- **Repassador GMTP**: roteadores que participam efetivamente da rede de favores, com a responsabilidade de repassar os fluxos de dados originados em um ou mais *Servidores GMTP* para outros nós *Repassador GMTP* até que os pacotes de dados alcancem os nós *Clientes GMTP*.
- **Relator GMTP**: é um *Cliente GMTP* com habilidades de enviar relatórios periódicos ao nó *Repassador GMTP* sobre o estado da transmissão.

Deste ponto em diante, os termos *Cliente GMTP*, *Servidor GMTP*, *Repassador GMTP* e *Relator GMTP* serão utilizados em sua forma simplificada, ou seja, *cliente*, *servidor*, *repassador* e *relator*, respectivamente. Estes termos não serão mais formatados em itálico, bem

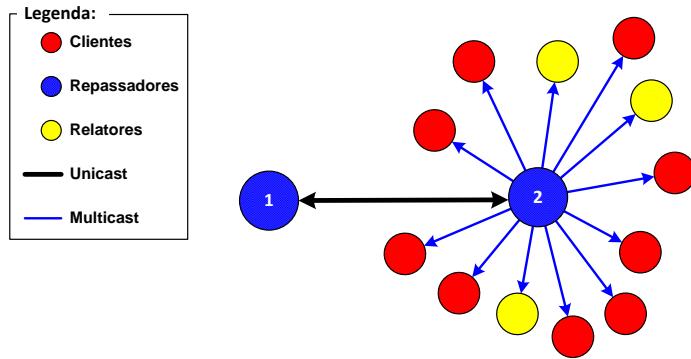


Figura 4.5: Tipos de Nós e modos de conexões do GMTP.

como os termos *socket*, *cache*, *unicast*, *multi-unicast* e *multicast*. Quando o termo *transmissão* ou *transmissão de um evento ao vivo* for mencionado, denotar-se-á a transmissão de um fluxo de datagramas IP correspondente a um evento ao vivo com o uso do protocolo GMTP. Embora alguns autores considerem os termos “repasse” e “roteamento” como conceitos distintos, neste trabalho ambos os termos são considerados sinônimos e devem ser interpretados como a capacidade que um nó GMTP tem de receber dados em uma interface de rede de entrada e encaminhar estes dados através de uma ou mais interface de rede de saída, ao mesmo tempo, permitindo-se que uma mesma interface de rede seja utilizada como entrada e saída ao mesmo tempo. Ademais, nas seções subsequentes, as palavras “deve”, “não deve”, “requerido”, “pode”, “não pode”, “recomendado” e “opcional”, incluindo suas variações morfológicas, devem ser interpretadas como descrito na RFC 2119 [249].

Quando um nó cliente deseja reproduzir um determinado evento, este envia uma requisição em direção ao nó servidor que está transmitindo o conteúdo de interesse, como atualmente acontece em qualquer conexão na Internet. A diferença é que um nó repassador pode interceptar tal requisito durante seu trajeto até o nó servidor, que então determina os melhores parceiros para atendê-la. Em geral, isto ocorre já no roteador de borda do nó cliente, que funciona como nó repassador de origem. Caso o nó repassador não encontre nenhum nó parceiro capaz de repassar a mídia de interesse que já esteja recebendo o referido fluxo, encaminha-se a requisição ao nó servidor que transmite a mídia correspondente, já que o pedido de conexão é intencionalmente endereçado ao nó servidor que transmite o fluxo de dados de interesse. Em todo caso, sempre o nó repassador de origem assumirá o controle de uma requisição do nó cliente, habilitando-se como candidato a parceiro para outros nós repassadores, quando

motivados por requisições originadas pelos seus respectivos nós clientes.

O posicionamento dos nós repassadores e suas habilidades permitem a redução do número de fluxos de dados na rede correspondente a um mesmo evento, ao tempo que maximiza a quantidade de nós clientes interessados em receber o mesmo fluxo (escalabilidade). Por este mesmo motivo, o protocolo GMTP é flexível para permitir que um nó repassador atue somente encaminhando conteúdos multimídias entre duas ou mais redes distintas, mesmo que este não tenha demandas explícitas dos seus nós clientes por tal conteúdo. Desta forma, maximiza-se o uso dos canais de transmissão ociosos, em particular das redes residenciais, principalmente quando seus usuários estão ausentes e portanto sem fazer uso dos recursos de redes disponíveis. Isto pode ocorrer sem a necessidade de manter um determinado computador ligado e conectado à rede, bastando apenas manter o roteador de rede ligado, diferentemente de todas as outras soluções existentes baseadas em arquitetura P2P ou P2P/CDN, que requer pelo menos um nó cliente conectado à rede, ou seja, ativo e operante.

As requisições de conexão podem ser originados não apenas por nós clientes para seu respectivo nó repassador, mas também estas podem ocorrer entre nós repassadores que, motivados pelos interesses dos seus nós clientes, formam parcerias entre si. Isto significa que um nó repassador pode agir como se fosse um nó servidor, respondendo às requisições originadas por seus nós clientes GMTP ou por outros nós repassadores, como se a requisição estivesse alcançado o nó servidor que originalmente transmite o conteúdo. Essa estratégia gera uma diminuição significativa do número de requisições de conexão aos nós servidores, evitando-se portanto a tragédia dos bens comuns, como discutiu-se na Seção 1.2. Além disso, na Figura 4.4, observa-se um grupo especial de nós chamados de nós *Relatores GMTP*. Estes nós são responsáveis por enviar relatórios periódicos sobre o estado da transmissão ao seu nó *Ressassador GMTP*, que os utiliza para regular a taxa de transmissão de um ou mais fluxos de dados, impedindo que a rede entre em congestionamento.

### 4.1.1 Tipos de Pacotes

Toda comunicação entre dois ou mais nós GMTP ocorre através da troca de pacotes IP, os quais carregam sinalizações de controle e/ou dados da aplicação. No mundo real (Internet), será necessário registrar na *Internet Assigned Numbers Authority – IANA*<sup>1</sup> o uso de um

---

<sup>1</sup>IANA: <http://www.iana.org/>

código para o campo *Protocolo* do cabeçalho de um datagrama IP. Com a padronização do protocolo GMTP e a publicação da sua RFC, provavelmente será utilizado o código 100, como já está definido no documento *Protocol Numbers*<sup>2</sup> da IANA.

No cabeçalho dos pacotes GMTP, existe um campo denominado *tipo do pacote* com tamanho de 5 bits, que são descritos a seguir. Este campo determina qual tipo de informação está contida em um determinado pacote GMTP e, ao processá-lo, o nó GMTP deve executar uma determinada ação.

0. *GMTP-Request*: o nó cliente envia requisição para obter um fluxo de dados multimídia com base no nome do fluxo de interesse;
1. *GMTP-RequestNotify*: o nó repassador notifica um cliente que um fluxo de dados está prestes a ser transmitido ou já está sendo transmitido em um determinado canal de repasse multicast. O campo de dados desse tipo de pacote contém a descrição da mídia a ser reproduzida;
2. *GMTP-Response*: o nó repassador confirma o estabelecimento de uma parceria com outro nó repassador, dado um determinado fluxo de dados;
3. *GMTP-Register*: o nó repassador registra participação no servidor para funcionar como distribuidor de um fluxo de dados;
4. *GMTP-Register-Reply*: o nó servidor responde sobre o pedido de registro de participação enviado por um nó repassador;
5. *GMTP-Route-Notify*: pacote que contém um caminho (rota) entre o nó repassador e um nó servidor. Em geral, o nó repassador envia esse tipo de pacote ao nó servidor;
6. *GMTP-RelayQuery*: o nó repassador pode solicitar ao servidor uma lista de possíveis nós repassadores parceiros;

---

<sup>2</sup>O código 100 foi utilizado no passado por um outro protocolo de mesma sigla, mas foi descontinuado e se tornou obsoleto. No momento da escrita desse documento, o uso de tal identificador está sendo negociado junto a IETF e a IANA <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>

7. *GMTP-RelayQuery-Reply*: o nó servidor envia uma resposta ao nó repassador com uma lista de candidatos a parceiros;
8. *GMTP-Data*: qualquer nó utiliza esse tipo de pacote para transmitir dados da aplicação;
9. *GMTP-Ack*: qualquer nó utiliza esse tipo de pacote para confirmar a recepção de um determinado pacote, seja pacotes previamente contendo dados ou não;
10. *GMTP-DataAck*: combinação dos pacotes GMTP-Data e GMTP-Ack (*PiggyBack*);
11. *GMTP-MediaDesc*: o nó servidor transmite esse pacote para descrever informações sobre a mídia sendo transmitida em uma determinado fluxo de dados (conexão);
12. *GMTP-DataPull-Request*: o nó repassador envia um pedido para obter o mapa de buffer atual de um outro repassador parceiros;
13. *GMTP-DataPull-Response*: resposta ao pedido para obtenção de um mapa de buffer;
14. *GMTP-Elect-Request*: o nó repassador envia para um cliente o pedido para tal cliente atuar como nó relator;
15. *GMTP-Elect-Response*: o nó cliente envia para o repassador uma confirmação de que pode atuar como relator;
16. *GMTP-Close*: os nós servidor, repassador ou cliente solicitam o término de uma conexão;
17. *GMTP-Reset*: determina, incondicionalmente, a finalização de uma conexão;
18. *Reservado*: a partir deste identificador ao 31, tratam-se de valores reservados para uso futuro e ignorado pelos nós que o processa.

#### 4.1.2 Resumo das Principais Funcionalidades

- Registro de participação de um nó repassador em um nó servidor. Isto permite que um nó repassador sinalize interesse em participar de uma rede CDN. Assim, pode-se pré-selecionar nós parceiros filtrados por métricas que influenciam na qualidade de serviço oferecido aos sistemas finais. Este assunto será retomado na Seção 4.3.1.

- Acesso a uma transmissão de um evento ao vivo através de um processo de conexão em três-vias (3WHS), com a requisição de conexão transmitida ao servidor e podendo ser interceptada por um nó repassador em seu trajeto ao servidor, com suporte automático de detecção e uso dos modos de transmissão suportados pelo nó repassador (unicast e/ou multicast). Este assunto será retomado na Seção 4.4.2.
- Descoberta de nós parceiros entre redes distintas e negociação de parcerias, com suporte a formação de parcerias baseadas em métricas de rede, tal como a largura de banda fim-a-fim. Além disso, o GMTP é capaz de distribuir um fluxo de dados em múltiplas taxas de transmissão de acordo com a largura de banda dos nós repassadores. Para isto, o GMTP segmenta os canais de transmissão (rota entre um nó servidor e os nós repassadores) quando existem múltiplos nós repassadores em uma determinada rota e estes estão interessados em receber o mesmo fluxo de dados. Este assunto será retomado nas Seções 4.3 e 4.5.
- Envio e recebimento de fluxos de dados compartilhados entre nós da mesma rede através do uso do modo de transmissão multicast, sendo o modo de transmissão unicast restrito apenas para transportar os fluxos de dados entre redes distintas, evitando-se a relação de uma conexão por cliente ao nó servidor. Este assunto será retomado na Seção 4.4.
- Uso de algoritmo de controle de congestionamento assistidos pela rede, em transmissões em modo unicast; e uso do *TCP Friendly Rate Control* (TFRC) adaptado às transmissões de fluxos de dados em modo multicast. Este assunto será retomado na Seção 4.5.
- Verificação de autenticidade dos dados multimídia, por meio do uso de assinaturas digitais disponibilizadas pelos nós servidores, impedindo assim ataques de poluição de conteúdo. Este assunto será retomado na Seção 4.6.
- Eleição de nós relatores com suporte a tolerância a desconexões de nós, com notificação e reeleição de novos nós. Este e outros assuntos relacionados serão retomados na Seção 4.7.

## 4.2 Definições, Relações e Restrições

Para melhor organizar as discussões sobre o funcionamento do GMTP, nesta seção, descrevem-se suas definições, relações e restrições. Para isto, faz-se uso de fundamentos de álgebra booleana, lógica proposicional, teoria de conjuntos e teoria dos grafos [250–253].

1. Seja o conjunto finito dos nós repassadores, definido por  $R = \{r_1, r_2, r_3, \dots, r_d\}$ , tal que  $d \in \mathbb{N}$ .
2. Seja o conjunto finito dos roteadores de uma rede de computadores, definido por  $B = \{b_1, b_2, b_3, \dots, b_e\}$ , tal que  $e \in \mathbb{N}$ . Existe uma relação  $R \rightarrow B$  que determina a sobreposição dos nós repassadores  $r_d \in R$  sob os roteadores em  $B$ .
3. Seja o conjunto finito dos nós servidores, definido por  $S = \{s_1, s_2, s_3, \dots, s_a\}$ , tal que  $a \in \mathbb{N}$ .
4. Seja o conjunto finito dos nós clientes, definido por  $C = \{c_1, c_2, c_3, \dots, c_f\}$ , tal que  $f \in \mathbb{N}$ .
5. Seja o conjunto *totalmente ordenado (toset)* dos pacotes de dados gerados pelos nós  $s_a \in S$  durante a transmissão de um evento ao vivo  $\mathcal{E}$ , definido por  $(\mathbb{P}, \prec) = \{p_1, p_2, p_3, \dots, p_h\}$ , tal que  $h \in \mathbb{N}$ . Note que se utiliza o símbolo  $\prec$  para representar precedência entre dois elementos.
6. Seja um grafo determinado pelo conjunto de vértices  $Z$ , que podem estar interligados entre si por um conjunto de diferentes arestas, chamadas de caminhos  $W$ , por onde se transmite o fluxo de dados  $P$ , definido por  $\eta = G(Z, W)$ , tal que:
  - (a)  $Z = S \cup R$ ;
  - (b) Sejam as relações e restrições estabelecidas entre os diferentes tipos de nós de uma transmissão de um evento  $\mathcal{E}$ , definida por  $\mathcal{T} = \{Z, P, C_i\}$ , tal que:
    - i. Seja  $P$ , o conjunto *parcialmente ordenado (poset)* dos pacotes de dados  $p_x$  transmitidos por um nó  $r_d$ , também chamado de fluxo de pacotes de dados ou apenas fluxo de dados, definido por  $(P, \prec) = \{p_1, p_2, p_3, \dots, p_x\}$ , tal que  $x \in \mathbb{N}$ . Trata-se de um *poset* porque o GMTP não garante entrega de  $p_x$ ;

- ii. Seja  $C_i$ , uma função que denota os nós  $c_f$  relacionados a um nó  $r_d$ , de modo que nenhum nó  $c_f \in C$  pode estar relacionado com dois ou mais nós  $r_d$ , definida por  $C_i : r_d \rightarrow 2^C$ ,  $\forall r_d, r_q \in R$ ,  $C_i(r_d) \cap C_i(r_q) = \{\emptyset\}$ , tal que  $q \neq d$  e  $q \in \mathbb{N}$ ;
- iii. Seja  $L$ , o conjunto finito dos nós relatores, definido por  $L = \{l_1, l_2, \dots, l_w\}$ . Como todo nó  $c_f$  pode atuar como  $l_w$ , tem-se que  $\exists L_\theta \in 2^{C_i(r_d)}$ , tal que  $l_w \in L_\theta$ . Pelo item 6(b)ii, tem-se portanto que  $L_\theta \subset L$  e  $L_\theta \cup C_i(r_d) = C_i(r_d)$ .
- (c)  $W = \bigcup_{v=1}^j W_v$ , onde  $j \in \mathbb{N}$  e corresponde à quantidade de todos os possíveis caminhos *toset*  $(W_v, \prec)$ , que denota um dos possíveis caminhos por onde um fluxo de dados  $P$  pode ser transmitido, obrigatoriamente a partir de um nó servidor  $s_a$  até um nó  $r_1$ , tal que:
- i.  $(W_v, \prec) = \{w_m \mid s_a, r_1, r_2, r_3, \dots, r_d\}$ ,  $\forall w_m, w_{m+1} \in W_v : w_m \prec w_{m+1}$  e  $|W_v| \geq 2$ ;
  - ii. Um caminho  $W_v$  é dito *caminho semi-completo*, representado por  $W_v^\circ$ , se e somente se  $W_v \leftrightarrow \exists B_\theta$  (bijetora), tal que  $B_\theta \in 2^B$  e  $B_\theta \neq \{\emptyset\}$ . Isto é, todos os roteadores  $b_e \in B$  são sobrepostos por um nó  $r_d \in W_v^\circ$ ;
  - iii. Um caminho  $W_v$  é dito *caminho completo*, representado por  $W_v^\bullet$ , se for  $W_v^\bullet$  e se  $W_v \subset T$ , tal que  $T \subset Z$  é o conjunto dos nós  $r_d$  que transmitem os pacotes de dados  $p_x \in P$  a seus nós  $c_f \in C_i(r_d)$ , definido por  $T = \{t_u \mid \varphi(t_u, P) = 1\}$ ,  $u \in \mathbb{N}$ , tal que:
    - A.  $\varphi$  uma função booleana que determina se um nó  $t_u \in T$  transmite os pacotes  $p_x \in P$  para  $c_f \in C_i(t_u)$ , definida por  $\varphi : (t_u, P) \rightarrow \{0, 1\}$ ,  $\forall (t_u, P) \in \{T \times \{P\}\}$ , onde 0 e 1 denotam, respectivamente, *falso* e *verdadeiro*.
- (d) Seja  $\sim$ , reversa de um conjunto *toset*, tal que  $\sim : (W_v, \prec) \rightarrow (W_v, \succ)$ . Isto é, para um conjunto  $(W_v, \prec) = \{w_m \mid s_a, r_1, r_2, \dots, r_d\}$ , então  $\sim(W_v)$  produzirá  $(W_v, \succ) = \{w_m \mid r_d, r_{d-1}, r_{d-2}, \dots, r_1, s_a\}$ ;
- (e) Seja  $\delta$ , uma função que define um sub-caminho de  $W_v$ , representado por  $W_v^\triangleleft$ , a partir de um nó  $t_u \in W_v$  até um nó  $t_1 \in W_v$ , tal que  $\delta : (t_u, W_v) \rightarrow (W_v^\triangleleft, \prec)$ . Ou seja, para um caminho qualquer  $(W_v, \prec) = \{t_{u+2}, t_{u+1}, t_u, t_{u-1}, t_{u-2}, \dots, t_2, t_1\}$ ,

$$\delta(t_u, W_v) = W_v^\lhd = \{t_u, t_{u-1}, t_{u-2}, \dots, t_2, t_1\}.$$

- (f) Seja  $\zeta$  uma função que calcula o custo total para transmitir um pacote  $p_x \in P$  através de um caminho  $W_v$ , definida por  $\zeta : \sum_{v=1}^{|W_v|} \gamma(w_m, w_{m+1})$ , tal que  $\gamma$  é uma função que determina o custo para transmitir o pacote  $p_x$  entre dois nós distintos  $\forall w_m, w_{m+1} \in W_v$ . No GMTP, a função  $\gamma$  calcula o custo apenas entre dois nós  $t_u, t_{u+1}$ , com base pela largura de banda disponível nos nós  $t_u$ . Porém, pode-se definir outras métricas, por exemplo, o número total de saltos no caminho  $W_v$  ou o RTT entre o nó  $s_a$  e um nó  $r_d$ ;
- (g) *Conjectura 1:*  $\forall r_d \in R$  e  $\forall c_f \in C$ ,  $r_d$  é mais estável que qualquer  $c_f$  com relação a sua disponibilidade e participação em uma rede de favores  $\eta$ . Em uma rede comutada por pacotes IP, um nó  $b_e \in B$ , ou seja, um nó  $r_d$  fica mais disponível se comparado aos seus nós  $C_i(r_d)$ . Por exemplo, nas transmissões de dados na Internet, a participação de um roteador no processo de transmissão de um fluxo de dados  $P$  é fundamental, mesmo que seja apenas para rotear os respectivos pacotes. Apesar de óbvia, tal observação é importante porque para qualquer nó  $c_f$  receber os pacotes de dados  $p_x \in P$ , primeiramente os pacotes de dados  $p_x$  passam, obrigatoriamente, pelo roteador de  $c_f$ , ou seja, o seu roteador padrão. Sendo assim, quando um nó  $r_d$  se desconecta, todos seus nós  $C_i(r_d)$  tornam-se capazes de receber  $P$ , mas a recíproca não é verdadeira – se um nó  $c_f$  se tornar indisponível, não necessariamente  $r_d$  também se torna indisponível. Com a aceitação dessa conjectura para a rede  $\eta$ , permite-se que outros nós  $c_f$  possam continuar recebendo  $P$ , mesmo ocorrendo a desconexão de um nó  $c_f$  que também esteja recebendo  $P$ . No GMTP, adota-se tal estratégia quando um nó  $r_d$  passa a manter estado sobre a transmissão de  $P$  e não mais os nós  $c_f$ , antes prática comumente adotada em soluções tradicionais de distribuição de conteúdos multimídia baseado em uma arquitetura P2P ou em qualquer protocolo disponível no estado da arte;
- (h) *Conjectura 2:* as tabelas de roteamento dos nós  $w_m \in W_v$  não mudam frequentemente e são independentes umas das outras. Em redes comutadas por pacotes IP, as rotas entre quaisquer nós  $c_{f_1}$  e  $c_{f_2} \in C$  não se alteram com uma frequência que desestabilize a comunicação entre estes. Mesmo se estas mudanças ocorrerem

em uma rota de um caminho  $W_v$ , o impacto causado é temporário e insignificante para a transmissão de um evento  $\mathcal{E}$  quando se utiliza um conjunto de algoritmos que tratem essas mudanças. Com base na aceitação dessa conjectura, pode-se antecipar a formação de parcerias pré-selecionando nós  $r_d$  em  $Z$  antes da efetiva transmissão de um fluxo de dados  $P$ . No GMTP, adota-se tal estratégia ao permitir que no processo de conexão, todos os nós  $r_d \in R$  informem sua posição na mensagem de requisição transmitida ao nó  $s_a$ . Quando o nó  $s_a$  recebe tal mensagem, este passa a conhecer o caminho até o referido nó  $r_d$ . Posteriormente, o nó  $s_a$  utiliza o conjunto de caminhos conhecidos para sugerir parcerias entre os nós  $r_d$ .

Desta forma,  $\eta$  representa formalmente a rede de sobreposição constituída pelo GMTP, definindo-se as relações, restrições estabelecidas em  $\mathcal{T}$  e as conjecturas consideradas para a execução de tal protocolo.

### 4.3 Constituição da Rede de Favores $\eta$

A constituição da rede de favores  $\eta$  ocorre por meio do registro de participação de um ou mais nós  $r_d \in R$  a um ou mais nós  $s_a \in S$ . Isto ocorre de forma direta ou indiretamente por meio de outros nós  $r_q \in R$ . Todo esforço realizado nesse processo objetiva transmitir um determinado fluxo de dados  $P$  para um ou mais nós  $c_f \in C$ , podendo ser distribuído pelos nós  $r_d$  por meio de diferentes caminhos  $W_v \in W$ .

O GMTP tenta determinar um caminho sub-ótimo  $W_\theta$  através do qual os pacotes de dados  $p_x \in P$  sejam entregues o mais rápido possível ao nó  $c_f$  interessado em obter  $P$ . Para isto, deve-se determinar  $W_\theta$ , tal que  $W_\theta = \min(\zeta(\forall W_v))$  e, sempre que possível, que  $W_\theta$  seja um caminho completo  $W_\theta^\bullet$ . Sempre buscar um caminho completo é importante porque, como todos os nós de tal caminho são roteadores sobrepostos por  $r_d$  e utilizados para transmitir  $P$ , pode-se distribuir  $P$  para mais nós  $c_f$  sem que sejam necessárias múltiplas conexões em  $s_a$ , evitando a tragédia dos bens comuns, discutida no Capítulo 1. Além disso, quanto mais nós  $r_d$  estiverem disponíveis na rede, menor será o impacto causado pelas desconexões nos sistemas finais que recebem o fluxo de dados  $P$ .

### 4.3.1 Registro de Participação de $r_d$ em $\eta$

Por analogia, o registro de participação faz com que o roteador de uma rede funcione como se fosse uma antena de recepção de uma transmissora de TV, podendo-se receber um ou mais sinais de canais de TV diferentes. Em seguida, estes sinais são repassados para os clientes conectados diretamente à antena, ou melhor, ao roteador.

O procedimento de registro de participação de um nó  $r_d$  em uma rede  $\eta$  é o primeiro passo e um dos mais importante. O registro de participação permite que um nó  $r_d$  se registre a um nó  $s_a$  para sinalizar interesse em funcionar como um nó repassador de um fluxo de dados  $P$ . O registro de participação pode ocorrer antes do nó  $s_a$  iniciar a transmissão de um fluxo de dados  $P$ , ou durante sua transmissão. Em ambos os casos, o algoritmo de registro de participação é similar, com uma diferença: se um nó  $r_d$  solicitar previamente um registro de participação a um  $s_a$  sem interesse por um fluxo de dados  $P$  qualquer, será possível mapear antecipadamente e selecionar um subconjunto de possíveis nós parceiros  $r_q$  para executar a distribuição de um fluxo de dados  $P$ . Neste caso, pode-se utilizar  $r_d$  para repassar pacotes de dados  $p_x \in P$  mesmo quando  $C_i(r_d) = \{\emptyset\}$ , ou seja, mesmo se o nó repassador não tiver nós clientes para repassar o fluxo de dados  $P$ . Assim, os nós  $r_d$  passam a funcionar como se fossem servidores de uma rede CDN, que podem ser acionados dinamicamente, quando conveniente. Na prática, o registro de participação prévio será utilizada, geralmente, por nós  $r_d$  controlados por usuários finais (redes residenciais e empresariais).

Para realizar um registro de participação, um nó  $r_d$  envia uma mensagem para um nó  $s_a$  utilizando o pacote *GMTP-Register* que, como resposta, envia um pacote do tipo *GMTP-Register-Reply*. O uso do pacote do tipo *GMTP-Register-Reply* permite a descoberta de um caminho  $W_v$ . Isto porque todos os nós  $r_d$  existentes no caminho entre  $s_a$  e  $r_d$  devem adicionar seu identificador no pacote *GMTP-Register-Reply* antes de roteá-lo para o próximo salto da rota em direção ao nó  $s_a$ . Para definir um identificador de um nó  $r_d$ , gera-se um código *hash* da soma dos endereços MAC (*Media Access Control*) de todas as interfaces de rede do roteador. Quando o pacote *GMTP-Register-Reply* alcançar o nó  $r_d$ , este envia o caminho  $W_v$  contido no pacote *GMTP-Register-Reply* de volta ao nó  $s_a$  utilizando o pacote do tipo *GMTP-Route-Notify*. A partir desse ponto, o nó  $s_a$  conhece o caminho  $W_v$  que utilizará para enviar qualquer fluxo de dados  $P$  até alcançar  $r_d$ , armazenando-o em uma estrutura de dados do tipo grafo. Esse procedimento em três vias confirma o registro de participação de  $r_d$  em

$s_a$ , ao passo que  $s_a$  poderá utilizar  $W_v$  para instruir os nós  $r_d$  a realizarem parcerias a fim de distribuir um fluxo de dados  $P$ , como se discute na Seção 4.3.3.

Dessa forma, se um nó  $r_d$  for um nó comum entre dois caminhos, será necessário apenas enviar um fluxo de dados  $P$  até  $r_d$  e este replicará o referido fluxo de dados para os nós  $r_{d+1}$ ,  $r_{d+2}$ ,  $r_{d+3}$  e assim por diante. De forma similar, se  $\exists c_f \in C_i(r_d)$  interessado em obter  $P$ , com  $\varphi(r_d, P) = 1$ , ou seja, quando um nó  $r_d$  já está recebendo  $P$ , o registro de participação já terá ocorrido e o fluxo já estará sendo recebido pelo nó  $r_d$ , vindo diretamente do nó  $s_a$  ou repassado por outros nós  $r_d$ . Como consequência, reduz-se o tempo de início de reprodução do referido fluxo de dados  $P$  para aqueles nós  $c_f$  que solicitarem o mesmo fluxo de dados  $P$  após o primeiro nó repassador pedir, bastando apenas que os próximos nós  $c_f$  “sintonizem” sua interface de comunicação (socket de rede) no canal apropriado e informado por  $r_d$ , pois a transmissão ocorre em modo multicast.

No Algoritmo 1, executado por um nó  $r_d$ , resume-se os passos para o envio do pedido de registro de participação em um nó  $s_a$ . Note que não é requerido que o nó  $r_d$  informe qual fluxo de dados  $P$  está interessado em obter. Se  $P$  for especificado, o nó  $s_a$  executará um procedimento para determinar se aceita ou não o pedido de registro de participação e logo começar a transmitir  $P$  a  $r_d$ . Em caso de aceite, inicia-se a transmissão de  $P$  a partir de  $s_a$  em direção  $r_d$  em modo unicast. Caso contrário, o nó  $s_a$ , com base nos caminhos  $W_v$  conhecidos, instruirá um outro nó  $r_q$  a transmitir o referido fluxo de dados ao nó  $r_d$  solicitante, tal como detalhou-se anteriormente. Já no Algoritmo 2, resume-se os passos após um nó  $r_d$  receber uma resposta do tipo *GMTP-Register-Reply*, transmitida pelo nó  $s_a$ , referente ao pedido de registro de participação transmitido anteriormente por  $r_d$ . Note que  $r_d$  transmite de volta a  $s_a$  o caminho  $W_v$ .

É importante salientar que toda transferência de pacotes de controle entre nós  $r_d$  ocorre com garantia de entrega, representando-se tais ações pelas funções com nomes contendo o sufixo *Rdt* (*Reliable data transfer*). Uma outra decisão importante tomada no GMTP é que um nó  $r_d$  deve periodicamente sinalizar sua participação na rede de favores  $\eta$  através de um método conhecido por *keep-alive*, comumente utilizado em outros protocolos de rede consolidados, como o TCP. Nesse aspecto, o GMTP segue a RFC 1122, *Requirements for Internet Hosts - Communication Layers* [254].

Um nó  $r_d$  pode sinalizar explicitamente sua desistência de participação diretamente ao nó

$s_a$ , quando não desejar mais participar da rede de favores  $\eta$  ou continuar recebendo um fluxo de dados  $P$ . Para isto, deve-se enviar um pacote do tipo *GMTP-Close*. Em qualquer um dos casos de desconexão, por expiração do tempo (devido ao procedimento de *keep-alive*) ou explicitamente através do envio do pacote do tipo *GMTP-Close*, o nó  $s_a$  deve desconsiderar  $r_d$  em futuras formações de parcerias, finalizando o procedimento de desconexão com o envio do pacote do tipo *GMTP-Reset*. Na Seção 4.7, discute-se em mais detalhes sobre o processo de desconexão de um nó  $r_d$ , pois a suspensão de transmissão dos pacotes de dados  $P$  não ocorre instantaneamente, dependerá se o nó  $r_d$  em fase de desconexão está ou não repassando  $P$  para algum nó parceiro  $r_q$ .

**Algoritmo 1:** registerRelay( $s_a$ : PeerServer,  $p_x = \text{GMTP-Request}$ )

---

```

/* The node  $r_d$  executes this algorithm to send a register
of participation to a given node  $s_a$ . If  $p_x$  is given,
node  $c_f$  wants to receive the flow  $P$ , so notify  $s_a$ . */
1 if  $p_x \neq \text{NULL}$  then
2    $P \leftarrow \text{getPacketFieldValue}(p_x, \text{'flow'})$ ; /* Extracts  $P$  in  $p_x$  */
3    $c_f \leftarrow \text{getPacketFieldValue}(p_x, \text{'client'})$ ; /* Extracts  $c_f$  in  $p_x$  */
4    $\text{channel} \leftarrow \text{isFlowBeingReceived}(P)$ ; /* Ver Seção 4.3.2 */
      /* Add  $c_f$  in the list of receivers waiting  $P$ . */
5    $\text{addClientWaitingFlow}(c_f, P)$ ;
6   if  $\text{channel} \neq \text{NULL}$  then
7     /* Let  $c_f$  know that  $P$  is already registered in this
        $r_d$  and is available from a multicast channel. */
8      $\text{respondToClients}(\text{GMTPRequestReply}(\text{channel}))$ ;
9     return 0;
10  else                                /* Flow  $P$  not registered yet. */
11    /* Send request to  $s_a$  and wait registration reply.
       When GMTP-Register-Reply is received, executes
       onReceiveGMTPRegisterReply (Algorithm 2). */
12     $\text{isWaitingRegisterReply}(P, \text{true})$ ;
13     $\text{sendPktRdt}(\text{GMTPRegister}(s_a, P))$ ;
14  end
15  /* Ask  $C_i(r_d)$  to wait registration reply for  $P$ . */
16   $\text{respondToClients}(\text{GMTPRequestReply}(P))$ ;
17  return 0;
18 end
19 if  $\text{not } \text{isWaitingRegisterReply}(s_a)$  then
20   return  $\text{sendPktRdt}(\text{GMTPRegister}(s_a))$ ;
21 end
22 return 0;

```

---

**Algoritmo 2:** onReceiveGMTRegisterReply( $p_x = \text{GMTP-Register-Reply}$ )

---

```

/* The node  $r_d$  executes this algorithm when receives a
packet of type GMTP-Register-Reply, as response for a
registration of participation sent to a  $s_a$  node.      */

1 setWaitingRegisterReply( $P, false$ );
2 if  $p_x = OK$  then                                /*  $s_a$  confirmed registration */
3    $W_v \leftarrow \text{getPacketFieldValue}(p_x, 'way');$     /* Gets  $W_v$  in  $p_x$  */
4    $s_a \leftarrow \text{getPacketFieldValue}(p_x, 'server');$   /* Gets  $s_a$  in  $p_x$  */
5    $P \leftarrow \text{getPacketFieldValue}(p_x, 'flow');$        /* Gets  $P$  in  $p_x$  */
6   if  $P \neq \text{NULL}$  then          /* Reply to  $C_i(r_d)$ , waiting for  $P$  */
7     if  $s_a$  enabled security layer then           /* Section 4.6.4 */
8       getAndStoreServerPublicKey( $s_a$ );
9     end
10     $channel \leftarrow \text{createMulticastChannel}(s_a, P);$ 
11    updateFlowReceptionTable( $channel$ ); /* Section 4.3.2 */
12    /* Let  $c_f \in C_i(r_d)$  know the multicast channel to
13       receive  $P$  (Section 4.4.2 for more details).      */
14    respondToClients(GMTPRequestReply( $channel$ ));
15    /* Start to relay  $P$  to clients (Section 4.4.6).      */
16    startRelay( $channel$ );
17  end
18  /* It was just a reply of a registration of
19   participation. Update flow reception table.      */
20  updateFlowReceptionTable( $s_a$ );             /* Section 4.3.2 */
21  sendWayBackToServer( $W_v$ );
22
23 else
24   /*  $s_a$  refused to accept the registration of
25    participation. This  $r_d$  must notify the clients
26    waiting for receiving  $P$ .                      */
27    $errorCode \leftarrow \text{getPacketFieldValue}(p_x, 'error');$ 
28   respondToClients(GMTPRequestReply( $errorCode, P$ ));
29
30 end

```

---

Por fim, salienta-se que o registro de participação do GMTP permite que quanto mais nós  $r_d$  se registrarem em nós  $s_a$ , mais caminhos  $W_v$  sejam conhecidos. Quanto mais caminhos forem conhecidos, mais parcerias poderão ser formadas entre os nós  $r_d$ . Quanto mais parcerias forem formadas, maior será o número de nós  $c_f$  capazes de receber um fluxo de dados  $P$  originado em  $s_a$ , disponibilizado indiretamente através dos seus respectivos nós  $r_d$ , sem nenhuma influência da camada de aplicação. No mundo real (Internet), os nós  $r_d$  podem passar a constituir dinamicamente a rede de distribuição de conteúdos de uma empresa. Por exemplo, um usuário de uma conexão residencial xDSL pode configurar seu roteador para registrar-lo em múltiplas redes de distribuição, como ilustrou-se na Figura 4.3. Nesses casos, as redes de distribuição podem fazer uso do roteador desse usuário em momentos ociosos de recepção e transmissão de dados através da Internet. Como consequência, relações comerciais podem ser construídas entre o usuário e os provedores de rede, mas essa discussão está fora do escopo deste trabalho.

### **Manutenção do registro de participação:**

O procedimento de manutenção de um registro de participação deve ser feito usando o pacote do tipo *GMTP-Ack*, em um tempo  $t = \max(300, t_{user})$ , onde  $t_{user}$  é definido em segundos e corresponde a um tempo definido pelo administrador do nó  $r_d$ , caso deseje um tempo menor que 300 s para mantém o registro de participação ativo. Quando  $s_a$  receber um pacote do tipo *GMTP-Ack* do nó  $r_d$ , este deve enviar um pacote do mesmo tipo. Caso  $r_d$  não receba *GMTP-Ack* no período de  $4 \times RTT$ , deve-se repetir tal procedimento por 3 vezes e somente após essas tentativas, o nó  $r_d$  deve considerar a conexão finalizada por tempo de expiração (*timeout*) e enviar um pacote do tipo *GMTP-Reset*. Na RFC 5482 [255], discute-se sobre outros aspectos de expiração no tempo que podem ser adaptados para o GMTP.

#### **4.3.2 Tabela de Recepção de Fluxos de Dados**

Antes de seguir com a explicação sobre o processo de estabelecimento de conexão do GMTP, é importante entender que cada nó  $r_d$  mantém uma tabela chamada *Tabela de Recepção de Fluxos de Dados*, como ilustra-se na Figura 4.6. O nó  $r_d$  utiliza tal tabela para registrar todos os fluxos de dados que estão sendo repassados para seus nós  $c_f \in C_i(r_d)$  e os respectivos

canais multicast utilizados, mantendo-se as seguintes informações:

#	Nome do Fluxo de Dados (P)	Servidores $s_a$	Repassadores $r_d$	Porta de Recepção de P	End. do Canal Multicast	Porta do Canal Multicast
- - - Vazia - - -						

Figura 4.6: Exemplo de uma tabela de recepção de fluxo mantida por um nó  $r_d$ .

- **Nome do Fluxo de Dados  $P$ :** é uma sequência de 128 bits que determina o nome de um fluxo de dados, como se descreve na Seção 4.4.1;
- **Servidores  $s_a$ :** o endereço IP do nó  $s_a$  que gera o fluxo de dados  $P$ ;
- **Repassadores  $r_q$ :** o endereço IP do nó  $r_q$ , parceiro de  $r_d$ , que está transmitindo o fluxo de dados  $P$  para  $r_d$ . Se nulo, significa que o fluxo de dados  $P$  está sendo recebido diretamente do nó  $s_a$ ;
- **Porta de Recepção de  $P$ :** o número da porta do nó remoto que está transmitindo o fluxo de dados  $P$  para  $r_d$ . Nesse caso, o nó remoto pode ser o nó  $s_a$ , em caso de conexão direta com o servidor, ou um nó  $r_q$ , parceiro de  $r_d$ ;
- **Endereço do Canal Multicast:** o endereço IP multicast utilizado pelo nó  $r_d$  para repassar o fluxo de dados  $P$  para os nós clientes  $c_f \in C_i(r_d)$ ; e
- **Porta do Canal Multicast:** o número da porta multicast utilizada pelo nó  $r_d$  para repassar o fluxo de dados  $P$  para os nós clientes  $c_f \in C_i(r_d)$ .

Conceitualmente, quando um nó  $r_d$  adiciona um registro na tabela de recepção de fluxos de dados, define-se  $\varphi(r_d, P) = 1$ , ou seja,  $r_d \in T$ . Um nó  $r_d$  consulta a tabela de recepção de fluxos de dados quando recebe um pedido de conexão (*GMTP-Register*) para obter um fluxo de dados  $P$ , tal como apresentou-se na Linha 6 do Algoritmo 1, Seção 4.3.1. Além disso, um nó  $r_d$  atualiza a tabela de recepção de fluxos de dados após receber uma confirmação do registro de participação, tal como apresentou-se na Linha 11 do Algoritmo 2, Seção 4.3.1. Mais adiante, na Seção 4.4.2, discute-se em mais detalhes as ações de consulta e atualização da tabela de recepção de fluxos de dados.

### 4.3.3 Formação de Parcerias

Dado que as parcerias ocorrem entre os nós  $r_d \in R$  e não entre os nós  $c_f \in C$ , a formação de parcerias consiste em determinar intersecções de caminhos  $W_v$ , considerando o nó *pivot*  $s_a$  e diversos nós  $r_d$  interessados em obter  $P$ , a pedido de seus nós  $c_f \in C_i(r_d)$ . Este processo pode ocorrer antes e durante a transmissão de um fluxo de dados  $P$  gerado por um no  $s_a$ , de forma transparente para a aplicação em execução em  $c_f$ , durante seu pedido de conexão transmitido em direção ao nó  $s_a$ . Como consequência, constitui-se um ou mais caminhos  $W_v \in W$ , os quais interconectam um nó  $s_a$  e os nós  $c_f \in C_i(w_m)$ , tal que  $\exists W_v \mid w_m \in W_v$ . Como regra geral para formação de parcerias, definem-se três critérios:

1. o melhor nó  $s_a$  para servir um nó  $r_d$  é aquele que está especificado em seu pedido de registro de participação, pois deve-se respeitar as regras de balanceamento de carga definida pela CDN. Em geral, o servidor DNS define tais regras com base no endereço IP do nó solicitante;
2. se  $\varphi(w_m, P) = 1$ , então  $w_m$  pode agir como se fosse um nó  $s_a$ ;
3. se o nó  $w_m \in W_v$ , tal que  $W_v$  é parte ou todo do caminho entre  $r_d$  e  $s_a$ ; e se  $w_m$  se enquadra no Item 2, então o melhor nó  $s_a$  para servir  $r_d$  será o mesmo que serve o nó  $w_m$ .

Para entender detalhes desse processo, considere a Figura 4.7. No Passo 1, ilustra-se um cenário de rede  $\eta = G(Z, W)$ , onde  $Z = \{s_1, r_{1..19}\}$ ,  $W = \{\emptyset\}$  e  $\mathcal{T} = \{\{\emptyset\}, \{\emptyset\}, \{\emptyset\}\}$ , ou seja, sem qualquer fluxo de dados  $P$  sendo transmitido, tampouco nenhuma parceria efetivada e suprimindo-se os nós  $c_f \in C_i(r_{1..19})$ . Já no Passo 2, ilustra-se a mesma rede  $\eta$ , porém com  $\mathcal{T} = \{\{s_1, r_{5..9}\}, P, C_i(r_9)\}$ , constituindo-se o caminho  $W_1 = \{s_1, \dots, r_9\}$  (linha tracejada e vermelha) e, portanto,  $W = \{W_1\}$  com  $\varphi(r_9, P) = 1$ . Nesse exemplo do Passo 2, o nó  $r_9$  recebe o fluxo de dados  $P$  em modo unicast e repassa  $P$  para todos os nós  $c_f \in C_i(r_9)$  em modo multicast. Para constituir o caminho  $W_1$ , o nó  $r_9$  deve transmitir o pedido de registro de participação ao nó  $s_1$  (como discutiu-se na Seção 4.3.1) e, a partir de sua confirmação, processada pelo nó  $s_1$  e enviada ao nó  $r_9$ , este começa a receber os pacotes  $p_x \in P$ . Com este procedimento, o nó  $s_1$  passa a conhecer o caminho  $W_1$ , que pode ser

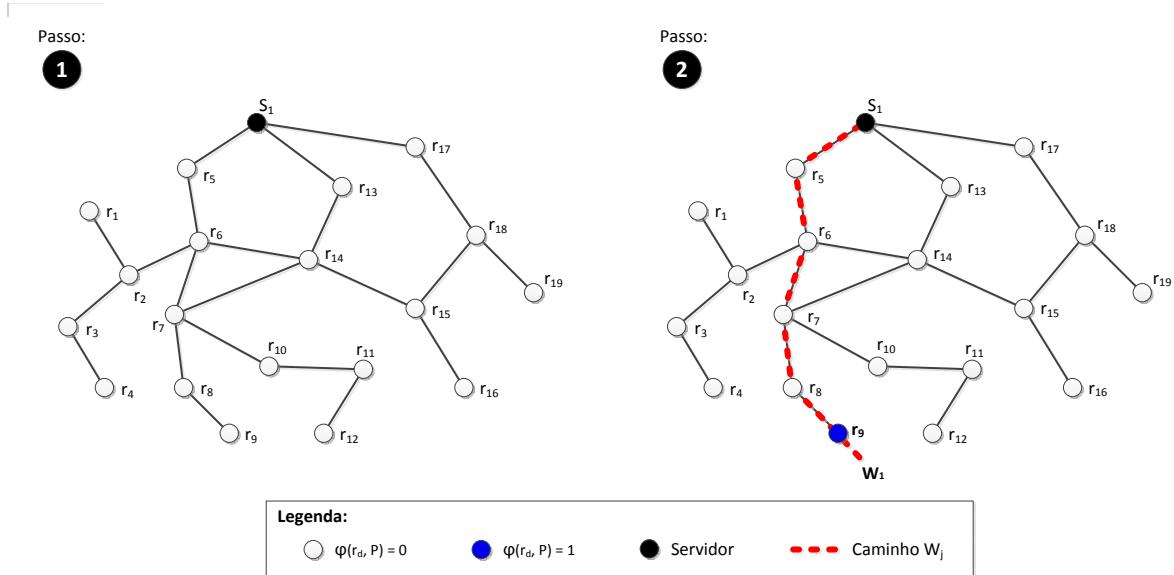
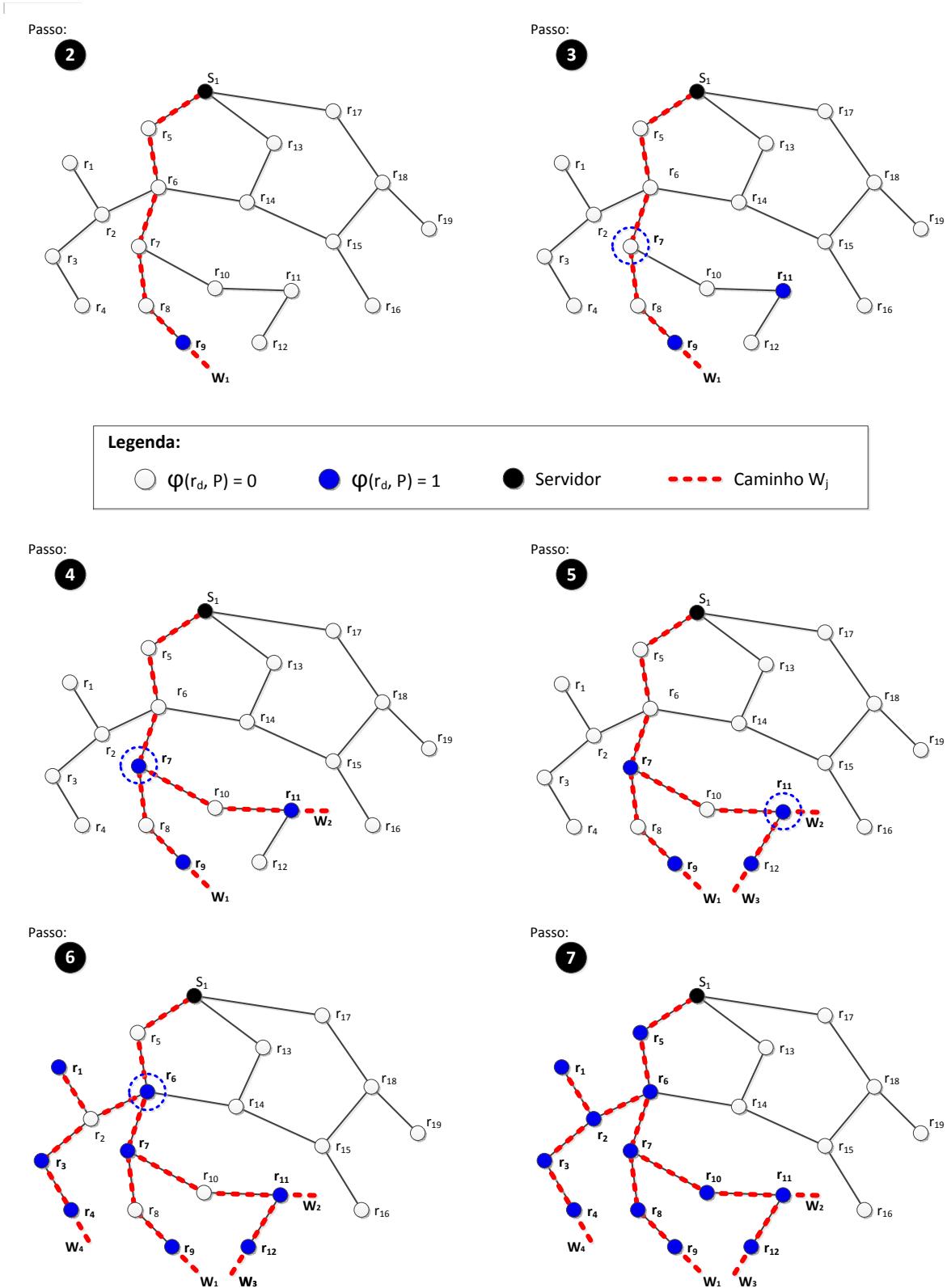


Figura 4.7: Cenário e passos para seleção de nós (exemplo 1).

utilizado para determinar futuras parcerias. Desse ponto em diante, utilizar-se-á tal exemplo como base para explicar outros aspectos do processo de formação de parceria do GMTP.

Na Figura 4.8, considera-se a formação de parceria por intersecção do fluxo de dados  $P$ , a partir do Passo 2 da Figura 4.7. Este procedimento ocorre quando um outro nó  $r_d$  envia um pedido de registro de participação em direção ao nó  $s_1$ , a fim de obter o fluxo de dados  $P$ , motivado por algum nó  $c_f \in C_i(r_d)$ . Nesse caso, se um nó  $r_d$  transmitir um pedido de registro de participação através de um sub-caminho  $W_v^\triangleleft$  tal que  $\exists W_v \in W$ , o nó  $s_a$  determina a intersecção de ambos e instrui o nó comum  $w_m$  a repassar o fluxo de dados  $P$  também para  $r_d$ , sem a necessidade de enviar um segundo fluxo de dados na mesma direção de  $W_v^\triangleleft$ . Sendo assim, a resposta de  $s_1$  não resulta em uma nova transmissão do fluxo de dados  $P$ , mas sim em uma mensagem de controle para o nó  $w_m$ , após identificá-lo como o nó comum a dois ou mais caminhos  $W_v$ . Isto implicará que o referido nó  $w_m$  replique o fluxo de dados  $P$ , mesmo quando  $|C_i(w_m)| = 0$ , mas de modo conveniente para evitar múltiplas transmissões do fluxo de dados  $P$ , originadas no nó  $s_a$ . A fim de compreender o funcionamento desse procedimento, acompanhe a explicação a seguir, com base na ilustração da Figura 4.8 e no caminho  $W_1$ .

Se qualquer um dos nós  $r_{7,8,10,11,12}$ , suponha  $r_{11}$ , enviar um registro de participação em direção à  $s_1$  para obter um fluxo de dados  $P$  (Passo 3 da Figura 4.8), o nó  $s_1$  descobrirá o

Figura 4.8: Cenário para seleção de nós por interseção de caminhos  $W_v$ .

caminho  $W_2 = \{r_5, r_6, r_7, r_{10}, r_{11}\}$  (Passo 4). Em seguida, pela intersecção ( $W_1 \cap W_2$ ), o nó  $s_1$  determinará que o nó  $r_7$  é o nó comum e portanto instruirá que  $r_7$  repasse o fluxo de dados  $P$  também para o nó solicitante  $r_{11}$ . A instrução de  $s_1$  para  $r_7$  deve determinar  $\varphi(r_7, P) = 1$ . Em termos práticos, isto obriga o nó  $r_7$  a adicionar uma nova entrada na tabela de recepção de fluxos de dados referente a  $P$ , mesmo se  $|C_i(r_7)| = 0$  para  $P$ . É óbvio que, se posteriormente  $|C_i(r_7)| > 0$  para  $P$ , será necessário apenas  $r_7$  criar um canal multicast para a transmissão local de  $P$ , evitando-se um novo registro de participação em  $s_1$ . Na Seção 4.4.2, discute-se em mais detalhes este aspecto do GMTP, explicando-se os procedimentos de pedido de conexão de um nó  $c_f$ .

Ao estender a discussão sobre o cenário ilustrado na Figura 4.8, percebe-se que se o nó  $r_{10}$  necessitar obter o mesmo fluxo de dados  $P$ , seu pedido de registro de participação será interceptado pelo nó  $r_7$  e parte do procedimento supracitado se repete. Uma situação similar ocorre se o nó  $r_{12}$  ou qualquer nó  $r_d \in W_4$  também desejar obter o fluxo de dados  $P$ , tal que  $W_4 = \{r_1, r_2, r_3, r_4\}$  (Passo 5 e 6). Para o caso do nó  $r_{12}$ , o nó  $r_{11}$  interceptará o pedido de registro de participação de  $r_{12}$ , ao passo que se for qualquer nó  $r_d \in W_4$ , o nó  $r_6$  realizará tal interceptação, pois o nó  $s_1$  determinará  $\varphi(r_6, P) = 1$ , depois do primeiro pedido de registro de participação originado por qualquer nó  $r_d \in W_4$ . A única diferença nesses últimos casos é que, como  $\varphi(r_7, P) = 1$  e  $\varphi(r_{11}, P) = 1$ , o nó  $r_7$  tem autonomia para responder ao nó  $r_{10}$  e ao nó  $r_{11}$  como se fosse o nó  $s_1$ , sem repassar tal pedido em direção ao nó  $s_1$ .

Para generalizar essa discussão sobre o processo de formação de parcerias do GMTP, caso existam outros nós  $r_q$  interessados em obter um fluxo de dados  $P$  e estão interligados direto ou indiretamente a  $r_d$ , tal que  $\varphi(r_d, P) = 1$ , o nó  $r_d$  sempre interceptará o pedido de registro de participação dos nós  $r_q$  e atuará como se fosse o nó  $s_1$ . No caso do exemplo que se discute, independente da ordem em que as requisições de registro de participação sejam enviadas por  $w_m \in (W_1 \cup W_2 \cup W_3 \cup W_4)$ , será necessário transmitir apenas um fluxo de dados  $P$  para “alimentar” os quatro caminhos referidos. Isto significa que todos os nós  $c_f \in C_i(W_1 \cup W_2 \cup W_3 \cup W_4)$  receberão um único fluxo de dados, com repasse dos pacotes  $p_x \in P$  realizado em modo multicast em cada sub-rede de cada nó  $w_m$  (Passo 7). Como a transmissão será em modo multicast, torna-se indiferente a quantidade de nós  $c_f$  desses caminhos, mas faz-se necessário um mecanismo para controle de congestionamento em modo multicast, a ser discutido na Seção 4.5.

Note que, o nó  $r_d$  que interceptar um pedido de conexão para um fluxo de dados  $P$ , deve transmitir para o nó  $s_a$  uma notificação sobre a(s) parceria(s) formada(s) por intersecção. No caso do exemplo anterior, os nós  $r_6$ ,  $r_7$  e  $r_{11}$  devem realizar tal notificação enviando um pacote do tipo *GMTP-Register*, como explicado na Seção 4.3.1. Para isso, deve-se ativar o bit *intercepted* do pacote *GMTP-Register*. Esta ação é importante devido aos aspectos gerenciais de uma transmissão, onde uma aplicação poderá contabilizar os nós  $r_d$  que estão recebendo  $P$ , mesmo que indiretamente, por meio da interceptação de registros de participação. A propósito, em ICN isso não é possível.

Na prática, não se faz necessário que o nó  $r_d$  envie a notificação de interceptação instantaneamente. Em vez disso, um nós  $r_d$  pode acumular diversos registros de participação durante um determinado intervalo de tempo e, em seguida, transmiti-los para o nó  $s_a$ . Como se trata de um aspecto a nível de implementação, tal decisão está fora do escopo dessa discussão. No caso da implementação do GMTP realizada em simulador e utilizada neste trabalho, define-se que para todo registro de participação interceptado, gera-se e transmite-se uma notificação ao nó  $s_a$ .

No Algoritmo 3, resume-se os passos descritos anteriormente na perspectiva do nó  $s_a$ , a fim de determinar a formação de parcerias por intersecção. Executa-se tal algoritmo quando o nó  $s_a$  recebe um pedido de registro de participação enviado por um nó  $r_d$  para obter um fluxo de dados  $P$ . Através dessa estratégia de formação de parceria, permite-se repasses de pacotes de dados levando-se em consideração o nome do fluxo de dados de interesse e não o nó que o produz e transmite. Em todo caso, o destino da requisição é sempre o nó servidor, garantindo-se que se nenhum nó repassador interceptar o pedido de registro de participação, com certeza tal pedido alcançará o nó servidor e o estabelecimento de conexão ocorrerá normalmente. Por isso é importante a continuidade do uso de endereçamento IP, ausente em redes ICN. Além disso, esta decisão é fundamental para manter a compatibilidade com as aplicações de rede existentes na Internet, que transmitem um pedido de conexão e recebe uma resposta que pode ser ou não oficialmente gerada pelo nó servidor, mas o importante é receber o fluxo de dados  $P$  – o GMTP faz com que a rede cuide disso.

**Algoritmo 3:** handleRegisterParticipation( $r_d$ : PeerRelay,  $p_x = GMTP-Register$ )

---

```

/*  $s_a$  executes this algorithm to finds the first node  $w_m$ 
   common to a known path  $W_v$  and the path  $W_{r_d}$ .  $W_v$  is
   already used for transporting  $P$  to node in  $\delta(w_m, W_v)$ ,
   and  $W_{r_d}$  contains all nodes between  $r_d$  (requester) and
    $s_a$ . The packet  $p_x$  carries  $W_{r_d}$  and the  $P$  flow name. */

1 done  $\leftarrow$  false;           /* It becomes true when  $w_m$  is found */
2  $P \leftarrow$  getPacketFieldValue( $p_x$ , 'flow');      /* Extracts  $P$  in  $p_x$  */
3  $W_{r_d} \leftarrow \sim(\text{getPacketFieldValue}(p_x, 'path'))$ ;
4  $W_P \leftarrow$  getKnownPathsOfFlow( $P$ );          /*  $W_P \subset W$  */
5 foreach  $W_v \in W_P$  do
6   foreach  $w_m \in W_v$  do
7     if  $w_m \in W_{r_d}$  then
8       /* The node  $w_m$  is common in  $W_v$  and in  $W_{r_d}$ . */      */
9        $done \leftarrow true$ ;
10      break;
11    end
12  end
13 if  $done$  then
14   /*  $s_a$  stores  $W_{r_d}$  as a known path and replies to  $r_d$ ,
      asking  $w_m$  to act as a relay for  $P$ .  $s_a$  actives
      flag 'relay' of the  $GMTP-RegisterReply$ . */                  */
15    $W_P[\text{length}(W_P)] \leftarrow W_{r_d}$ ;
16   return GMTPRegisterReply( $w_m$ , relay=1);
17 end
18 end

/*  $s_a$  must register  $W_{r_d}$  as a known path and reply to  $r_d$  by
   accepting its connection request, since no node  $w_m$  is
   intersecting  $W_{r_d}$ . In this case,  $s_a$  starts the
   transmission of  $p_x \in P$  to  $r_d$ . */                         */

19  $W[\text{length}(W)] \leftarrow W_{r_d}$ ;
20 return GMTPRegisterReply( $r_d$ , relay=0);

```

---

Com relação à praticidade do processo de formação de parcerias empregado no GMTP, um aspecto técnico muito importante deve ser ressaltado: apenas o nó  $r_d$  que repassar  $p_x \in P$  para seus nós  $c_f \in C_i(r_d)$  deve manter uma entrada sobre  $P$  na tabela de recepção de fluxos de dados, exceto quando sinalizado pelo nó  $s_a$ , como é o caso dos nós  $r_6$  e  $r_7$  do exemplo anterior. Além disso, como a transmissão de um fluxo de dados  $P$  entre um nó  $r_d$  e seus nós  $c_f \in C_i(r_d)$  ocorrerá sempre em modo multicast, sendo necessária apenas uma entrada na tabela de recepção de fluxos de dados sobre  $P$ . Com essa estratégia, espera-se permitir uma quantidade significativa de nós  $c_f$  capazes de reproduzir um fluxo de dados  $P$ , sem sobre-carregar a rede com demasiadas transmissões do mesmo fluxo de dados  $P$ , além de reduzir o tempo de inicialização para reproduzir o fluxo de dados  $P$  e o índice de continuidade para um fluxo de dados  $P$ . Ademais, apresentou-se procedimentos que não são adotados em nenhum protocolo de rede pesquisa no estado da arte. Trata-se da primeira solução em que o nó servidor dar suporte aos roteadores no processo de formação de parcerias, delegando-se para estes a responsabilidade de distribuir um determinado fluxo de dados  $P$ , tudo de forma transparente para as aplicações. Como resultado, pode-se afirmar que os roteadores passam a funcionar como se fossem servidores de uma CDN, só que participando dinamicamente sempre que conveniente.

Por fim, um outro aspecto no processo de formação de parcerias do GMTP é o uso de informações sobre a capacidade de transmissão de um caminho  $W_v$  para determinar as parcerias entre os nós  $r_d$ , bem como a qualidade do fluxo de dados a ser transmitido pelo nó servidor. Diferentemente se comparado a soluções como as empregadas com o uso de HTTP (por exemplo, DASH), em vez de um servidor GMTP gerar múltiplos fluxos de dados, codificados em diferentes taxas de bits, gera-se o fluxo de dados com a taxa de bits correspondente a capacidade máxima de transmissão do caminho  $W_v$  em um determinado instante. Isto ocorre porque o GMTP expõe, ao nó  $s_a$ , a informação de controle de congestionamento percebida pelos nós  $w_m \in W_v$ . Sendo assim, o nó  $s_a$  codifica a mídia em uma qualidade possível de ser transportada através de  $W_v$  utilizando um codificador do tipo VBR (*Variable Bit Rate*), como o H.264 [256]. Na Seção 4.5.1, discute-se essa função em mais detalhes.

## 4.4 Transmissão de $p_x \in P$ através de $\eta$

No GMTP, transmite-se os pacotes de dados  $p_x \in P$  utilizando uma estratégia híbrida *push/pull*. Utiliza-se o método *push* por padrão, onde os nós  $s_a$  iniciam a transmissão de  $p_x \in P$  para os demais nós  $w_m \in W_v$ . Já o método *pull* é utilizado somente quando um nó  $c_f$  precisa obter parte de uma mídia que está na iminência de ser reproduzida e ainda não foi repassada por um nó  $r_d$  via *push*, de acordo com o seu mapa de *buffer*.

Nessa seção, apresentam-se detalhes sobre como se realiza a disseminação de pacotes de dados  $p_x \in P$  e como os nós  $c_f$  recebem tal conteúdo para reprodução, discutindo-se aspectos sobre indexação, requisição, recepção e compartilhamento de um fluxo de dados  $P$ .

### 4.4.1 Indexação de Conteúdo

No GMTP, um fluxo de dados  $P$  tem um nome único que o identifica em qualquer nó. Na prática, cada fluxo de dados  $P$  corresponde a uma mídia gerada a partir de um evento ao vivo  $\varepsilon$ , por exemplo, a transmissão de um jogo de futebol, corrida de fórmula 1, etc.

Define-se um nome de um fluxo de dados  $P$  por um código de *hash* no formato UUID (*Universally Unique Identifier*) de 128 bits [257]. Na sua forma canônica, representa-se  $P$  por uma sequência de 32 dígitos hexadecimal, exibidos em cinco grupos separados por hífen, na forma de {8}-{4}-{4}-{4}-{12}. Por exemplo,  $P = 641f931f-d3ac-50e3-b625-537574541f1f$ . O nome de um fluxo de dados  $P$  sempre será informado no campo *nome do fluxo de dados (data flow name)*, disponível no cabeçalho de transporte dos pacotes *GMTP-Register*, *GMTP-Request*, *GMTP-Data* e *GMTP-Ack*.

Na prática, para gerar o nome para um fluxo de dados  $P$ , utiliza-se uma função de *hash* do tipo SHA1. Sendo assim, para determinar o nome de um fluxo de dados  $P$ , disponibilizado por um servidor  $s_a$ , utiliza-se  $\text{SHA1}(\text{IP}_{s_a} + : + \text{PORTA}_{s_a})$ . Por exemplo, suponha que um servidor esteja disponibilizando um fluxo de dados  $P$  através do endereço 200.17.113.98, na porta 21200. O nome do fluxo de dados  $P$  será definido por  $\text{SHA1}("200.17.113.98:21200") = f8ea01fd-4d71-5d95-89ec-35646e11d7fe$ . Opcionalmente, o nó  $s_a$  pode divulgar o nome do fluxo de dados através do serviço DNS. Já com relação ao título do conteúdo e sua descrição, tais informações podem ser divulgadas por meio de um serviço web, ou por meio de uma busca de diretório via um *Web Services*. Independente da forma que o nó  $s_a$  disponibilize os

nomes dos fluxos de dados  $P$ , de posse de um identificador de um fluxo de dados  $P$ , um nó GMTP poderá solicitar os pacotes de dados  $p_x \in P$ . Além disso, os nós  $r_d$  mantêm a tabela de recepção de fluxo de dados que estão repassando para seus clientes  $c_f \in C_i(r_d)$  e, sendo assim, podem compartilhá-la para outros nós repassadores. Atualmente, não se explora o compartilhamento da tabela de repasse, mas pode ser feito para formar parcerias entre dois nós  $r_d$  sem precisar consultar o nó  $s_a$ , por exemplo.

Com esse esquema de nomes baseado no endereço IP e porta, o GMTP não requer alteração na camada de aplicação para informar o fluxo de dados de interesse – a aplicação continua informando endereço IP e número da porta no momento do estabelecimento de uma conexão, mantendo-se a compatibilidade com as aplicações existentes e, portanto, futuras adoção do GMTP na Internet.

No caso do uso do DNS, o nó  $s_a$  divulga os identificadores de todos os eventos sendo transmitido por meio de um mecanismo de atualização dinâmica de registro de DNS, como especificado na RFC 2136 [258]. Para o GMTP, criou-se um novo tipo de registro de DNS chamado de SID (*Streaming IDentifier*).

No Quadro 4, ilustra-se um exemplo de uma requisição DNS, utilizando a ferramenta *dig*, um comando de terminal para Linux. Nesse exemplo, apresenta-se a lista dos nomes dos fluxos de dados transmitidos pelo domínio administrativo *globo.com*. Por ser uma consulta simples de DNS, qualquer sistema final conectado à Internet pode realizar tal procedimento, enaltecendo-se a facilitar de adaptar aplicações multimídia existentes para utilizar o GMTP. Ao indexar o conteúdo através de um serviço de DNS, permite-se desacoplar a forma de indexar um determinado conteúdo e a forma de obtê-lo, que passa a ser de responsabilidade da infra-estrutura de rede e não de uma ou mais aplicações isoladamente, como nas soluções apresentadas no Capítulo 3. Isto pode permitir o aumento das aplicações multimídia sem se preocupar como localizar um determinado conteúdo, extrapolando-se as barreiras administrativas de cada sistema de geração de conteúdos multimídia, bastando para isso apenas todas as aplicações utilizarem o protocolo GMTP. Consequentemente, um fluxo de dados  $P$ , gerado por uma aplicação qualquer APL1, em execução em um nó  $s_a$ , poderá ser reproduzido por uma aplicação APL2, em execução em um nó  $c_f$ , independentemente de seus fornecedores (implementadores). Para que essa visão seja empregada, definiu-se uma função para descrever a mídia transmitida em um fluxo de dados  $P$ .

---

**Quadro 4:** Exemplo de requisição e resposta da lista de nomes dos fluxos de dados  $P$  de um distribuidor de conteúdos multimídia.

---

```

1 dig -t SID globo.com;                                /* comando de requisição */
2 QUESTION SECTION:
3   globo.com.  IN  SID
4 ANSWER SECTION:
5   globo.com.  IN  SID  "111f931f-d3ac-10e3-b62f-f17f74541f1f"
6   globo.com.  IN  SID  "72c44591-7d82-427c-825f-722f015787c1"
7   globo.com.  IN  SID  "0bb0b9f5-f57d-4da5-8a6c-13acf1965188"
8 SUMMARY:
9   Query time: 4 msec
10  SERVER: 192.168.1.252:53(192.168.1.252)
11  WHEN: Tue Jul 16 15:44:25 2013

```

---

### Descrição de um fluxo de dados $P$ :

O GMTP permite nativamente a descrição da mídia a ser transmitida e com isso promover a compatibilidade entre diferentes aplicações. Para isto, incorporou-se ao GMTP o protocolo o SDP (*Session Description Protocol*), definido na RFC 2327 [98], permitindo-se que as aplicações consigam obter mais detalhes sobre a mídia transmitida, flexibilizando-se o acesso a um determinado conteúdo, pois as aplicações decodificará o conteúdo e o reproduzirá ao usuário através da aplicação final, independente de qual sistema o gerou.

Com esta decisão, torna-se mais fácil implementar novas aplicações multimídia, ao passo que fica mais fácil adaptar aplicações existentes para fazer uso do GMTP, uma vez que, em sua grande maioria, já se utiliza o SDP. Do ponto de vista de engenharia de software, isto evitará a repetição de esforço com implementações já consolidadas e que, com o passar dos anos, provou-se funcionar a contento, como foi o caso do SDP. Consequentemente, caso seja necessário a atualização do referido padrão, tal atualização será realizada internamente no GMTP e todas as aplicações automaticamente já poderão usufruir dos novos recursos disponibilizados. Na prática, isto significa uma atualização a nível de sistema operacional. A título comparativo, considerando os protocolos baseado em HTTP e os sistemas de dis-

tribuição multimídia, descritos no Capítulo 3, cada um possui uma proposta diferente de descrição da mídia, predominando o uso de XML, porém não se mantendo o mesmo formato. A consequência disso é que a aplicação cliente, para suportar diversos sistemas, terá que implementar cada uma dessas propostas e usar XML gasta-se mais espaço se comparado ao SDP.

Para uma aplicação em execução no nó  $s_a$ , faz-se necessário apenas determinar as informações da mídia e as fornece ao GMTP através de passagem de parâmetro via socket. Em seguida, o GMTP fica pronto para enviar a descrição da mídia como resposta ao pedido de conexão, dentro do campo de dados do pacote do tipo *GMTP-Register-Reply* ou *GMTP-MediaDesc*. Como um nó  $r_d$  pode interceptar um pedido de conexão,  $r_d$  também pode transmitir a descrição da mídia aos seus nós parceiros  $r_q$ . No Quadro 5, apresenta-se um exemplo de uma mensagem SDP e, a seguir, descreve-se cada um dos possíveis atributos de uma mensagem SDP.

- $v$ , a versão do SDP;
- $o$ , a lista de nós  $s_a$  que a distribui;
- $s$ , o nome da mídia, como discutido na Seção 4.4.1;
- $i$ , o título da mídia;
- $u$ , a URI que descreve detalhes sobre a mídia;
- $c$ , as informações de conexão, como a versão do protocolo de rede e o endereço do nó  $r_d$ ;
- $f$ , o certificado digital emitido pelo nó  $s_a$  para verificação de autenticidade dos pacotes  $p_x \in P$  (opcional). Este assunto será retomado na Seção 4.6;
- $m$ , o tipo da mídia, a porta de conexão e protocolo de transporte; e
- $a$ , atributos adicionais sobre a mídia como, por exemplo, qualidade, idioma, taxa de bits mínima e máxima necessária para transmitir a mídia, em bytes.

**Quadro 5:** Exemplo de uma mensagem SDP no pacote *GMTP-MediaDesc.*


---

```

1   v=0
2   o=- IN IP4 177.135.177.241, IP4 186.192.82.163, IP6 2001:0db8:85a3::7344
3   s=72c44591-7d82-427c-825f-722f015787c1;      /* ver Seção 4.4.1 */
4   i=An Introduction about Global Media Transmission Protocol (GMTP).
5   u=http://www.ic.ufal.br/projects/gmtp/introduction.ps
6   c=IN IP4 200.17.113.100
7   f=x509:http://vid12.akamai.com/certs/cert.crt      /* ver Seção 4.6 */
8   m=audio 49170 GMTP/RTP/AVP 16000-20000
9   m=video 51372 GMTP/RTP/AVP 163840-655360
10  a=type:multicast
11  a=sendrecv
12  a=quality:10
13  a=lang:en                                         /* ver RFC1766 [259] */
14  a=framerate:23.0

```

---

No exemplo apresentado no Quadro 5, utiliza-se a primeira versão do protocolo SDP e descreve-se a transmissão de dois fluxos de dados  $P$  (Linhas 10 e 11), sendo um de áudio e outro de vídeo. A distribuição dos fluxos de dados  $P$  ocorre com a geração dos pacotes de dados  $p_x \in P$  em três nós  $s_a$  (Linha 2), dos quais dois são acessíveis através de endereços IPv4 e um através de um endereço IPv6. Os fluxos de áudio e vídeo são repassados por um nó  $r_d$ , acessível por um endereço IPv4 (Linha 6), através das portas 49170 e 51372, respectivamente (Linhas 9 e 10). As informações de endereço IP e porta do nó  $r_d$  são utilizadas para que os nós  $c_f \in C_i(r_d)$  possam sintonizar seus sockets de conexão e iniciar a reprodução da mídia, através do modo de transmissão multicast (Linha 10). Em seguida, na Linha 8, observa-se uma URL do certificado digital a ser utilizado pelo nó  $r_d$  para verificar a autenticidade do conteúdo de pacote de dados  $p_x \in P$  – na Seção 4.6, discute-se este assunto em mais detalhes. Por fim, entre as Linhas 11 e 17 especificam-se outros parâmetros para descrever a mídia, tais como o nível de qualidade da mídia, que varia entre 1 e 10, as taxas de bits para cada fluxo de dados, sendo para o áudio variando entre 16000 *Bytes* e 20000 *Bytes* e, para o vídeo, variando entre 156250 *Bytes* e 625000 *Bytes*. É importante salientar que os nós

$r_d$  utilizam as informações de taxa de bits para determinar o tamanho do buffer necessário para permitir a transmissão da mídia, o que ocorre ao adicionar uma nova entrada na tabela de recepção de fluxos de dados.

#### 4.4.2 Estabelecimento de Conexão entre $c_f$ e $s_a$ para Obter $P$

Divide-se o processo de estabelecimento de conexão em três fases. A Fase 1 acontece quando, por exemplo, um nó qualquer  $c_1 \in C_i(r_d)$  deseja obter  $P$  transmitido por um nó  $s_1$  e não existe nenhum outro nó  $c_f \in C_i(r_d)$  em sua rede local recebendo  $P$ . Já a Fase 2 acontece quando um outro nó  $c_2 \in C_i(r_d)$  precisa obter o mesmo fluxo de dados  $P$ , solicitado previamente pelo nó  $c_1$ . E, por fim, a Fase 3 acontece quando o nó  $r_d$  começa a buscar novos nós parceiros  $r_q$  a fim de obter  $P$ . Na Figura 4.9, ilustram-se um nó  $s_a$ , que gera um fluxo de dados  $P$  e 12 nós  $r_d$ , que constituem uma rede de diferentes domínios administrativos, sendo o nó  $r_1$  o repassador de um desses domínios, composto por 6 nós  $c_f \in C_i(r_1)$  (Rede Local).

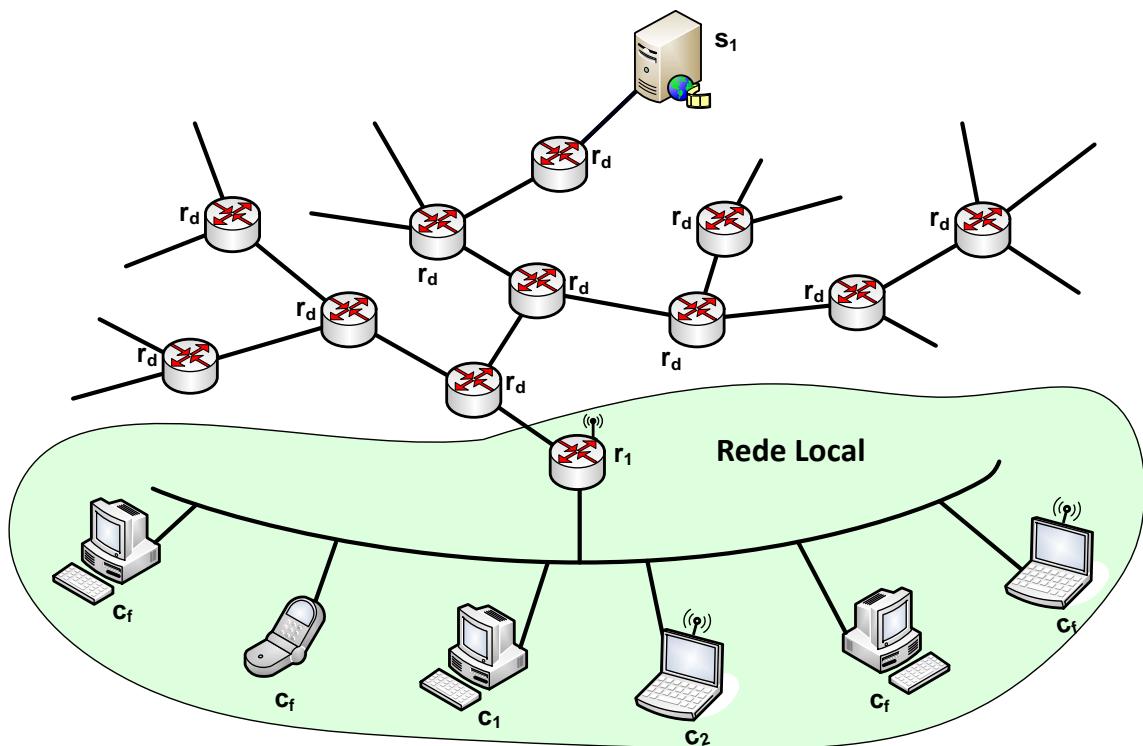


Figura 4.9: Exemplo de rede para o estabelecimento de conexão do GMTP.

A regra geral é que um nó  $r_d$  deve consultar a tabela de recepção de fluxo de dados todas as vezes que receber um pacote do tipo *GMTP-Request* ou do tipo *GMTP-Register*,

transmitido por um nó  $c_f \in C_i(r_d)$ . Com base no estado da referida tabela, que define a fase de conexão para um determinado fluxo de dados  $P$  solicitado, o nó  $r_d$  realiza uma determinada ação de registro de participação e repasse.

#### 4.4.3 Fase 1: Primeira Requisição a um Fluxo de Dados $P$

A Fase 1 ocorre quando nenhum nó  $c_f \in C_i(r_d)$  está recebendo um fluxo de dados  $P$ . Com base na Figura 4.10, onde ilustra-se um exemplo de conexão na Fase 1, considere:

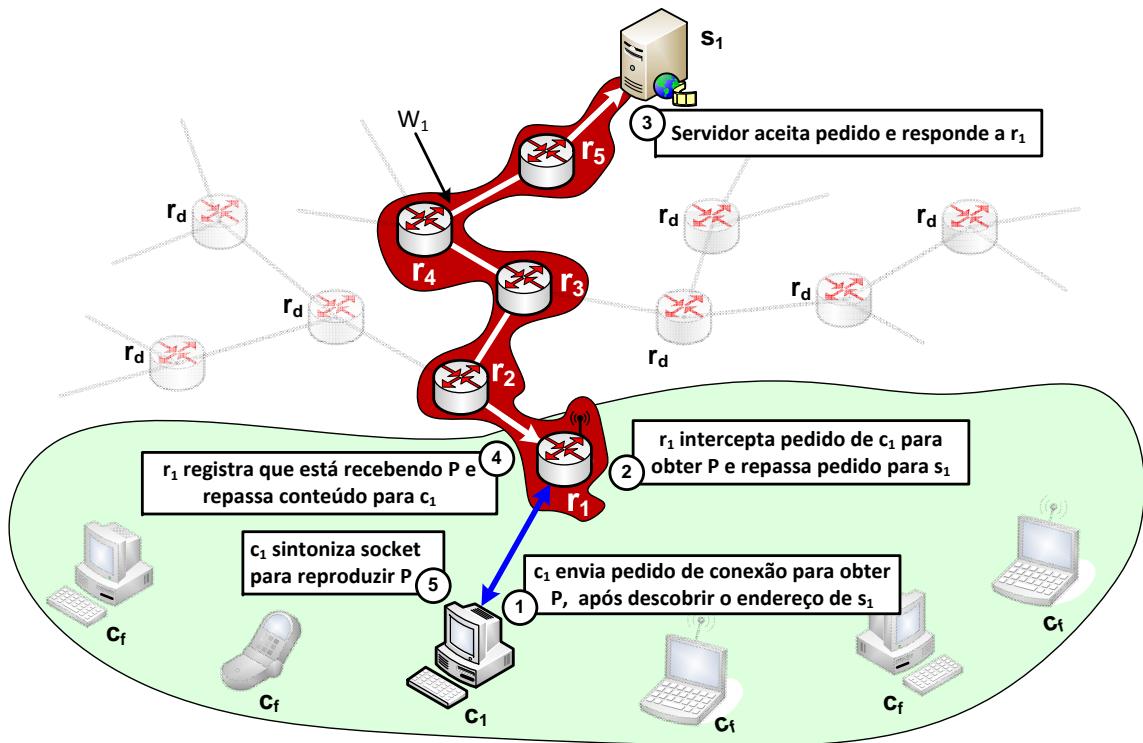


Figura 4.10: Passos do processo de estabelecimento de conexão do GMTP (Fase 1).

- $P$ , um fluxo de dados;
- $s_1$ , o nó servidor que gera os pacotes de dados  $p_x \in P$ ;
- $r_1$ , o nó repassador para os clientes  $c_f \in C_i(r_1)$ ; e
- $c_1$ , um nó cliente que deseja obter um fluxo de dados  $P$ , tal que  $c_1 \in C_i(r_1)$ .

Para obter o fluxo de dados  $P$ , o nó  $c_1$  inicia o canal de controle GMTP (detalhado na Seção 4.7.1) e transmite um pacote do tipo *GMTP-Request* (Figura 4.10, Passo 1). Para

construir o pacote do tipo *GMTP-Request*, qualquer nó  $c_f$  deve especificar o valor para o endereço IP de destino como sendo o endereço do nó  $s_a$  que transmite  $P$ , com o valor para o campo do cabeçalho de rede  $TTL=1$ . Além dos valores para o IP de destino e para o  $TTL$ , o nó  $c_f$  também deve informar o nome do fluxo de dados  $P$  que o usuário deseja reproduzir, presente no cabeçalho de transporte do pacote do tipo *GMTP-Request*. O valor de  $TTL=1$  é intencional, pois faz com que o nó  $r_d$  intercepte o referido pacote de requisição, evitando-se extrapolar o domínio administrativo de sua rede local. Este nível de detalhe é essencial para garantir que o roteador gerará uma interrupção através do processo que controla a fila de roteamento quando o  $TTL=0$ , obrigando o roteador analisar o pacote e nesse momento perceberá que é um pacote do tipo *GMTP-Request*. Isso evita que o roteador tenha que verificar cada pacote a ser roteado se corresponde ao tipo *GMTP-Request*.

Quando o pacote *GMTP-Request* alcançar o nó  $r_1$  (Passo 2 da Figura 4.10), este consulta a tabela de recepção de fluxos de dados e constata que não há qualquer registro para o fluxo de dados  $P$ . Nesse instante, o nó  $r_d$  inicia um processo de registro de participação para obter o fluxo de dados  $P$ . Isto significa que a execução do procedimento *registerRelay*( $s_a, p_x$ ) (Seção 4.3.1), onde  $p_x$  é o pacote do tipo *GMTP-Request*, fará o nó  $r_1$  transmitir um pacote do tipo *GMTP-Register* em direção ao nó  $s_1$ . À medida que os nós  $r_d$  repassam o pacote *GMTP-Register* até alcançar o nó  $s_1$ , constitui-se o caminho  $W_1 = \{r_1, r_2, r_3, r_4, r_5, s_1\}$  (Passo 3 da Figura 4.10 e destacado na cor vermelha), conforme discutiu-se na Seção 4.3.3.

Em seguida, ao receber o pacote do tipo *GMTP-Register-Reply*, como resposta ao registro de participação, o nó  $r_1$  cria um canal multicast e envia um pacote do tipo *GMTP-RequestNotify* para um ou mais clientes  $c_f \in C_i(r_1)$  (Passo 4 da Figura 4.10). Esta notificação permitirá os nós  $c_f$ , aguardando para obter  $P$ , “sintonizarem” seus respectivos sockets no canal multicast correspondente. No caso do exemplo supracitado, o nó  $c_1$ , após sintonizar o socket no canal multicast informado pelo nó  $r_1$ , começa a receber os pacotes de dados  $p_x$  do tipo *GMTP-Data* ou *GTMP-DataAck* (Passo 5 da Figura 4.10).

No Algoritmo 6, resume-se os passos descritos anteriormente para iniciar a transmissão dos pacotes de dados  $p_x \in P$  aos nós  $c_f \in C_i(r_d)$ , após  $r_d$  receber o pacote do tipo *GMTP-RequestReply*. Note que, o nó  $r_d$  invoca tal procedimento nas Linhas 7 e 14 do Algoritmo 1 e nas Linhas 12 e 19 do Algoritmo 2 (Seção 4.3.1). Como resultado da Fase 1, gera-se uma nova entrada na tabela de recepção de fluxos de dados do nó  $r_d$ , tal como ilustra-se na

Figura 4.11. Com base no exemplo citado, a tabela de recepção antes vazia, agora contém uma entrada que informa a ocorrência de recepção do fluxo de dados  $P = 72c44591-7d82-427c-825f-722f015787c1$ , originado no nó  $s_a$ , cujo endereço é 177.135.177.241, com porta de recepção 49170. Além disso, define-se o canal multicast no endereço 239.192.68.79 e porta 1900, através do qual os nós  $c_f \in C_i(r_d)$  podem receber os pacotes de dados  $p_x \in P$ .

#	Nome do Fluxo de Dados (P)	Servidores $s_a$	Repassadores $r_d$	Porta de Recepção de P	End. do Canal Multicast	Porta do Canal Multicast
1	72c44591-7d82-427c-825f-722f015787c1	177.135.177.241	nulo	49170	239.192.68.79	1900

Figura 4.11: Tabela de recepção de fluxos de dados após a Fase 1.

**Algoritmo 6:** respondToClients( $p_x$ : GMTP-RequestNotify)

---

```

/* A  $r_d$  node executes this Algorithm to respond to clients
   waiting for receiving a flow  $P$ . This algorithm is
   invoked in Lines 7 and 14 of Algorithm 1 and in
   Lines 12 and 19 of the Algorithm 2. */
```

**1**  $destAddress \leftarrow \text{getCtrlChannel}();$  /\* 238.255.255.250:1900 \*/

**2**  $\text{setPacketFieldValue}(p_x, \text{'destinationAddress'}, destAddress);$

**3**  $P \leftarrow \text{getPacketFieldValue}(p_x, \text{'flow'});$  /\* Extracts  $P$  in  $p_x$  \*/

**4**  $\text{errorCode} \leftarrow \text{getPacketFieldValue}(p_x, \text{'errorCode'});$

**5** **if**  $\text{errorCode} \neq \text{NULL}$  **then**

**6**     $\text{removeClientsWaitingForFlow}(P);$  /\* See Algorithm 1 \*/

**7**     $\text{sendPkt}(p_x);$

**8**    **return** 0;

**9** **end**

**10**  $\text{channel} \leftarrow \text{getPacketFieldValue}(p_x, \text{'channel'});$

**11** **if**  $\text{channel} \neq \text{NULL}$  **then**

/\* Node  $r_d$  is already receiving  $P$  and clients  $C_i(r_d)$ 
 must know the media description. \*/

**12**     $\text{mediaDescription} \leftarrow \text{getMediaDescription}(P);$

**13**     $\text{setPacketFieldValue}(p_x, \text{'data'}, mediaDescription);$

/\* In Algorithm 1, Line 5,  $c_f$  nodes are added in a list
 of clients waiting for flow  $P$ . Now,  $r_d$  notifies
 them, wait confirmation (ACKs) from them and start
 relaying  $p_x \in P$  to them through given channel. \*/

**14**     $\text{sendPkt}(p_x);$

**15**     $C_i(r_d) \leftarrow \text{getClientsWaitingForFlow}(P);$

**16**     $\text{waitAck}(C_i(r_d), P);$

**17** **else** /\* Let  $C_i(r_d)$  know  $r_d$  is waiting for registration. \*/

**18**     $\text{setPacketFieldValue}(p_x, \text{'waitingRegistration'}, true);$

**19**     $\text{sendPkt}(p_x);$

**20** **end**

**21** **return** 0;

---

#### 4.4.4 Fase 2: Próximas Requisições para Obter $P$

A Fase 2 de conexão ocorre quando futuras requisições para obter o fluxo de dados  $P$  são originadas por qualquer nó  $c_f \in C_i(r_1)$ . Considerando o exemplo anterior, citado na Fase 1, se um nó  $c_2 \in C_i(r_1)$  também solicitar  $P$ , o nó  $r_1$  simplesmente informará o canal multicast correspondente ao fluxo de dados  $P$ , como ilustra-se na Figura 4.12 (Passo 1 da Figura 4.12). Para isto, o nó  $r_1$  intercepta a requisição do nó  $c_2$ , consulta a tabela de recepção de fluxos de dados e dessa vez constata a recepção do fluxo de dados  $P$ , criando o pacote do tipo *GMTP-Request-Reply* (Passo 2). Este procedimento ocorre no registro de participação, especificamente no trecho de código definidos entre as Linhas 2-8 do Algoritmo 1. Em seguida, transmite-se o pacote do tipo *GMTP-Request-Reply* ao nó  $c_2$ , como descreve-se no trecho de código entre as Linhas 10-16 do Algoritmo 6, que então “sintoniza” seu socket para o canal multicast informado por  $r_1$  (Passo 3). Tal procedimento se repete para cada novo nó  $c_f \in C_i(r_1)$  interessado em obter  $P$ .

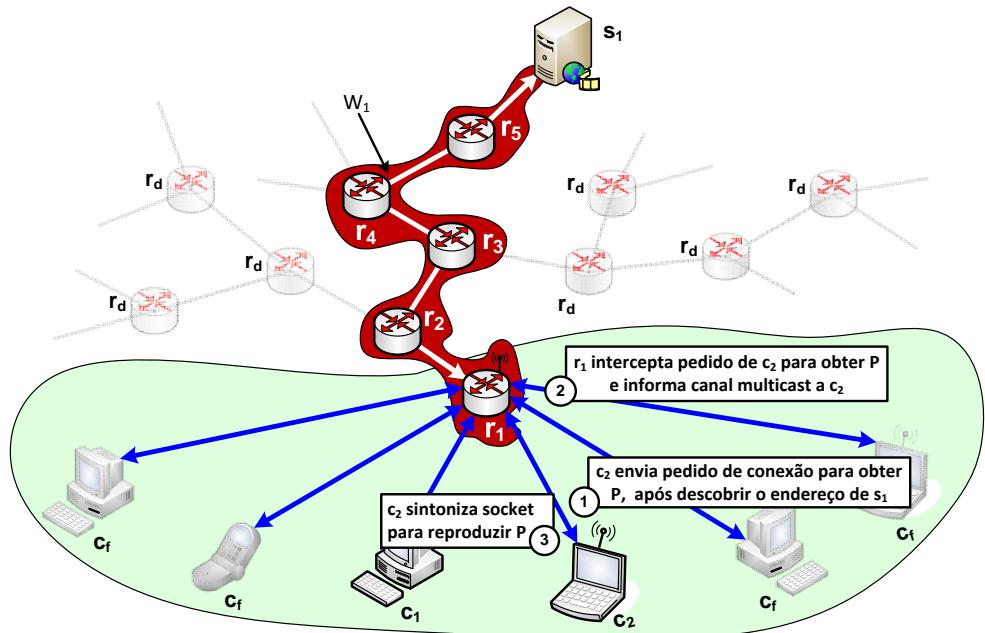


Figura 4.12: Passos do processo de estabelecimento de conexão do GMTP (Fase 2).

#### 4.4.5 Fase 3: Busca por Mais Parceiros $r_q$ para Obter $P$

Na Fase 3, o nó  $r_d$  inicia um processo de aumentar suas parcerias a fim de obter mais rapidamente os pacotes  $p_x \in P$  através de caminhos  $W_v$  alternativos. Nesse contexto, seja um nó  $r_3$  que esteja recebendo  $P$  originado em um nó  $s_a$ . Para conseguir mais nós parceiros  $r_q$ , o nó  $r_3$  envia uma requisição do tipo *GMTP-RelayQuery* para  $s_a$  e obtém um subconjunto de nós  $r_q$  candidatos a parceiro de  $r_3$ , como ilustra-se na Figura 4.13. O nó  $s_a$  constrói a lista de nós parceiros e envia ao nó  $r_d$ , funcionando como um indexador (*tracker*) de nós parceiros  $r_q$ , pré-selecionando parceiros para  $r_d$ . No caso do exemplo supracitado, essa pré-seleção ajuda o nó  $r_3$  a escolher os melhores parceiros disponíveis, de acordo com os seguintes critérios de prioridade:

1. Maior capacidade de transmissão do caminho  $W_v$ . Define-se este critério com base na menor taxa de transmissão disponível entre todos os nós  $w_m \in W_v$  em um determinado instante  $t$ . Na Seção 4.5, discute-se os algoritmos de controle de congestionamento do GMTP e o procedimento para determinar a taxa de transmissão de um caminho  $W_v$ ;
2. Se for um caminho for  $W_v^\bullet$ , determinado através da verificação da condição  $|W_v| = ttl(r_d, W_v)$ , onde  $ttl$  é uma função que determina o número de saltos entre o nó  $r_d$  até o nó  $s_a$ . Este critério é importante porque quanto mais nós GMTP estiverem no caminho, maior será a possibilidade de interceptação para obter um fluxo de dados  $P$ ;
3. Escolha aleatória de  $W_v$  entre os caminhos  $W$  conhecidos. Note que, por exemplo, no CoolStreaming, a escolha aleatória é feita a nível de nó cliente, ao passo que no GMTP a escolha é com base na capacidade de transmissão dos caminhos  $W_v \in W$ .

Sendo assim, define-se a Fase 3 do processo de estabelecimento de conexão do GMTP em três passos:

1. um nó  $r_d$  envia periodicamente requisições do tipo *GMTP-RelayQuery* para o nó  $s_a$  a fim de descobrir melhores parceiros e aumentar o número de parcerias. Por se tratar de fluxos de dados ao vivo, não necessariamente quanto mais parceiros um nó  $r_d$  tem, melhor será a qualidade do fluxo de dados  $P$ . Por isso, um nó  $r_d$  sempre mantém uma lista de candidatos a parceiros  $r_q$  fornecida pelo nó  $s_a$ , porém não necessariamente

estabelece parcerias com todos. Em vez disso, executa-se repetidamente as seguintes ações:

- Um nó  $r_d$  inicia uma nova parceria se a quantidade atual de parcerias reduzir por desconexão de um nó parceiro  $r_q$  ou se o buffer de recepção estiver com menos de  $\frac{1}{3}$  de sua capacidade. O objetivo é evitar o esvazamento do buffer para o fluxo de dados  $P$ , mantendo continuamente o repasse de pacotes de dados  $p_x \in P$  aos nós  $c_f \in C_i(r_d)$ .
  - A quantidade de parcerias em um determinado instante  $t$  é inversamente proporcional a quantidade de pacotes de dados  $p_x \in P$  que chegam repetidos ao nó  $r_d$ . Nesse caso, se um mesmo pacote  $p_x$  chegar repetidamente na mesma quantidade de parcerias estabelecidas, o nó  $r_d$  desconecta-se daquele nó parceiro  $r_q$  cujo pacote  $p_x$  chegou por último.
2. Após obter a lista de candidatos a parceiros (Passo 1), o nó  $r_3$  forma uma parceria com um dos candidatos da lista de possíveis parceiros ao enviar requisições do tipo *GMTP-Request* em direção a outro nó  $r_q$  que já esteja recebendo um fluxo de dados  $P$ . Como ilustra-se na Figura 4.14, este procedimento ocorre da seguinte forma: após o passo anterior, o nó  $r_3$  envia uma requisição do tipo *GMTP-Request* para o nó  $r_2$  contendo uma chave de autorização conhecida por ambos e informada pelo nó  $s_a$ . Caso a chave de autorização esteja correta, o nó  $r_2$  deve enviar um resposta do tipo *GMTP-Response* ao nó  $r_3$  e então começar a repassar os pacotes  $p_x \in P$ . O uso da chave de autorização é importante para evitar que um nó  $r_d$  se conecte a outro nó  $r_q$  sem que o nó  $s_a$  seja notificado sobre isto. As chaves de autorização são geradas pelo nó  $s_a$  e transmitidas como resposta no pacote do tipo *GMTP-Register-Reply*. Cada entrada disponível no pacote do tipo *GMTP-RelayQuery-Reply* contém endereço IP do nó repassador candidato a parceiro e sua respectiva chave de autorização;

A periodicidade de requisições do tipo *GMTP-RelayQuery* e a quantidade máxima de parcerias efetivas são parâmetros controláveis pelo administrador do nó  $r_d$ . Na implementação do GMTP utilizada neste trabalho, definiu-se o tempo de 5 minutos para a periodicidade de requisições do tipo *GMTP-RelayQuery* e 5 para a quantidade de parcerias efetivas. Os va-

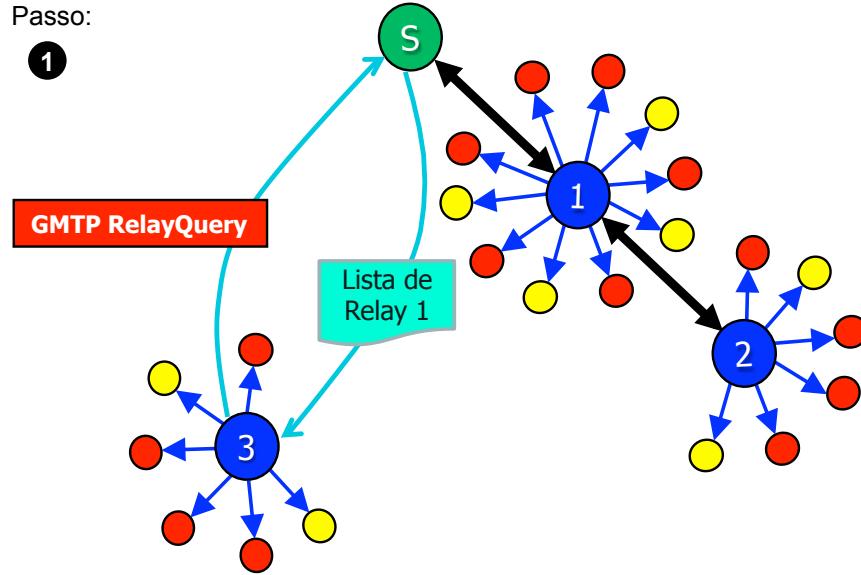


Figura 4.13: Fase 3 de conexão do GMTP (Passo 1).

lores destes parâmetros são considerados padrões em aplicações tradicionais de distribuição de conteúdos multimídia ao vivo baseados em redes P2P.

Com a execução da Fase 3 do processo de conexão do GMTP, pode-se expandir a quantidade de parcerias e essas informações são registradas na tabela de recepção de fluxos de dados, tal como ilustra-se Figura 4.15. Nesse exemplo, observa-se que um nó  $r_d$  está recebendo e repassando aos seus nós  $c_f \in C_i(r_d)$  quatro fluxos de dados diferentes, originados em quatro nós  $s_a$ , porém recebendo fluxos de dados de diferentes nós parceiros  $r_q$ . Por exemplo, dentre os fluxos de dados que o nó  $r_d$  está recebendo, um deles é o  $P = 72c44591-7d82-427c-825f-722f015787c1$ , cujos pacotes de dados  $p_x \in P$  são transmitidos por três nós  $r_q$  identificados pelos endereços IP e porta  $182.111.88.21:49170$ ,  $90.39.135.46:62242$  e  $83.67.132.41:53434$ . Para esse fluxo de dados  $P$ , o nó  $r_d$  repassa os pacotes de dados  $p_x$  para os nós  $c_f \in C_i(r_d)$  através do canal multicast  $239.192.68.79:1900$ .

Desta forma, o processo de conexão do GMTP é fundamental para a efetiva distribuição de mídias ao vivo, pois permite que as aplicações compartilhem fluxos de dados entre si, mesmo que estas não tenham sido desenvolvidas pela mesma equipe. Esta unificação ajuda no processo de distribuição do fluxo de dados  $P$ , pois, na prática, até mesmo uma aplicação *standalone* e um objeto de vídeo imbutido em uma página Web podem obter o mesmo fluxo de dados sem que estas conheçam uma a outra. Consequentemente, reduz-se para 1 o número

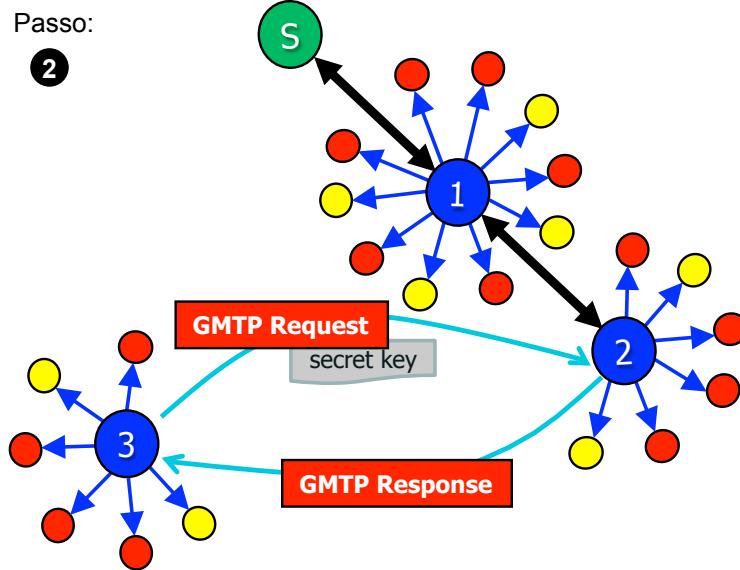


Figura 4.14: Fase 3 de conexão do GMTP (Passo 2).

#	Nome do Fluxo de Dados (P)	Servidores $s_a$	Repassadores $r_d$	Porta de Recepção de P	End. do Canal Multicast	Porta do Canal Multicast
1	72c44591-7d82-427c-825f-722f015787c1	177.135.177.241	nulo	49170	239.192.68.79	1900
2	72c44591-7d82-427c-825f-722f015787c1		90.39.135.46	62242	239.192.68.79	1900
3	72c44591-7d82-427c-825f-722f015787c1		83.67.132.41	53434	239.192.68.79	1900
4	e6ab15af-09ef-4985-a6ef-1777e41ffeb0	67.203.202.33	196.163.34.64	14928	239.192.226.179	6860
5	e6ab15af-09ef-4985-a6ef-1777e41ffeb0		204.36.89.52	58182	239.192.226.179	6860
6	fe222be9-8844-4ee9-bba1-0a90b2bea437	183.235.181.135	212.80.75.162	39345	239.192.57.10	1167
7	fe222be9-8844-4ee9-bba1-0a90b2bea437		174.195.228.32	32646	239.192.57.10	1167
8	721e1575-2a89-46f0-a8c7-340c81fc5de5	158.37.63.151	158.37.63.151	25848	239.192.161.45	7001
...	...	...	...	...	...	...

Figura 4.15: Tabela de recepção de fluxos de dados após a Fase 3.

de transmissões para um mesmo fluxo de dados  $P$  originado em  $s_a$  e destinados a uma mesma rede ou para um subconjunto de redes adjacentes. Além dessa diferença, a forma de conexão do GMTP supre uma antiga deficiência das soluções tradicionais de transmissão multicast, pois as aplicações tinham que se adaptar às configurações estáticas dos canais multicast definidos pelo administrador de rede e, até os próprios administradores de rede tinham que fazer tal configuração de forma manual, obrigatoriamente em todos os nós roteadores de um determinado caminho. Isto é impraticável devido à independência dos diferentes domínios administrativos.

Até o presente momento, não se conhece nenhuma solução que permita configuração dinâmica de canais multicast aliado a formação de uma rede de sobreposição constituída entre roteadores, com estratégia de formação de parcerias não mais considerando os sistemas finais, mas a intersecção de caminhos levando em consideração um nó *pivot* servidor, compartilhando-se os fluxos de dados entre aplicações distintas, promovendo um uso mais eficaz dos recursos computacionais e de rede, como se demonstra mais adiante no Capítulo ?? (Resultados).

#### 4.4.6 Envio e Recebimento de $p_x \in P$ em $\eta$

Após o estabelecimento de conexão, os nós  $r_d$  trocam dados entre si em modo unicast a fim de distribuir os pacotes de dados  $p_x \in P$  do tipo *GMTP-Data* e *GMTP-DataAck*. De forma similar, os nós  $r_d$  utilizam os mesmos tipos de pacotes para enviar  $p_x \in P$  para os nós  $c_f$ , porém em modo multicast. Quando o GMTP estiver em funcionamento em um nó  $s_a$  ou em um  $r_d$ , o estado é o de *transmitindo dados*, ao passo que quando executado em um cliente o estado é o de *recepção de dados*. Para o transporte dos pacotes de dados  $p_x$ , um nó  $s_a$  deve transmitir pacotes do tipo *GMTP-Data* ou o *GMTP-DataAck* em direção aos nós  $r_d$  de acordo com os registros de participação. Nesta seção, detalha-se como se executa os procedimentos de transmissão e recepção desses pacotes de dados.

##### **Buffer de Envio e Recepção:**

A transmissão de um evento  $\mathcal{E}$  consiste no processo de disseminação dos pacotes  $p_x \in P$  através dos nós interessados em obtê-lo. Para isto, cada nó GMTP controla um buffer de envio e recepção definido por uma estrutura de dados do tipo array circular (*ring buffer*), onde cada posição é utilizada para armazenar um pacote  $p_x$ , como ilustra-se na Figura 4.16. Ao receber  $p_x$ , um nó GMTP armazena-o no buffer e posteriormente o entrega para a aplicação, que o reproduz para o usuário final. Para o envio ou repasse de um pacote, o nó GMTP consome os pacotes  $p_x$  do buffer e transmite para o(s) nó(s) interessado(s), seja em modo unicast e/ou em modo multicast. Isto porque é possível que um nó  $r_1$  repasse  $p_x$  para um outro nó  $r_2$  (unicast) ao mesmo tempo que  $r_1$  pode repassar  $P$  para seus nós  $c_f$  (multicast).

O buffer de envio e recepção do GMTP tem seu tamanho definido no processo de estabe-

lemento de conexão, de acordo com o tipo da mídia sendo transmitido, mas o nó  $r_d$  pode determinar um limite a fim de evitar ataques de negação de serviço. Isto pode ocorrer porque o GMTP permite uma aplicação definir o tamanho do buffer que cada nó  $w_m$  deverá alocar para repassar os pacote de dados  $p_x$  de um fluxo de dados  $P$ . Então, uma aplicação maliciosa poderia alocar um espaço de buffer maior do que a que o roteador suporta, ou no mínimo monopolizar tal buffer. Após definir o tamanho inicial do buffer circular para um fluxo de dados  $P$ , este pode variar de acordo com a capacidade de transmissão do canal em um determinado instante. Essa decisão é importante porque permite um nó  $s_a$  alocar previamente o recurso necessário para o transporte de um determinado fluxo de dados  $P$ . O tamanho do buffer é especificado pelo nó  $s_a$  e propagado para os demais nós em um caminho  $W$ , no cabeçalho do pacote do tipo *GMTP-Register-Reply* ou *GMTP-MediaDesc*.

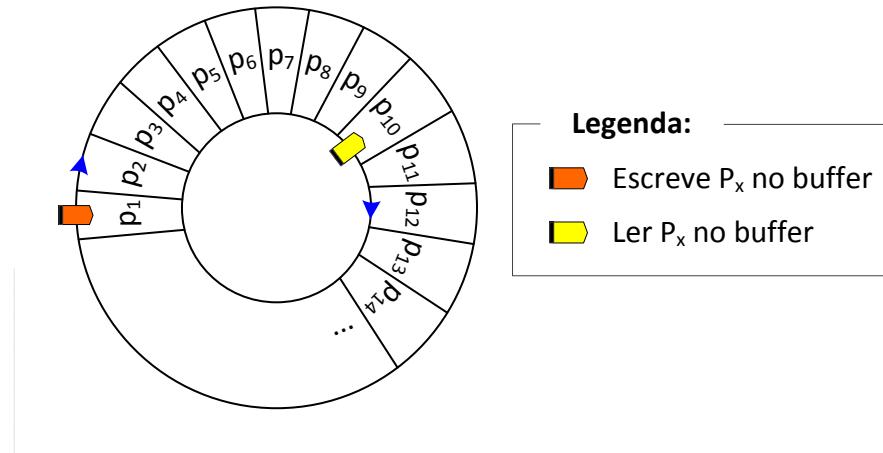


Figura 4.16: Exemplo da estrutura do buffer de envio e recepção de um nó GMTP com dois ponteiros, um para escrever e outro para ler pacotes  $p_x$ .

### Mapa de Buffer:

O mapa de buffer do GMTP descreve o estado atual do buffer de envio e recepção de um nó GMTP. Como ilustrado na Figura 4.17, trata-se de uma estrutura de dados que determina se um pacote  $p_x$  está ou não presente no buffer de um respectivo nó GMTP. O conteúdo de cada posição é o número de sequência do pacote, que determina a ordem que um pacote foi gerado e transmitido pelo nó  $s_a$ , mas o nó  $r_d$  não os ordena, pois tal responsabilidade é delegada aos nós  $c_f$  no momento de sua reprodução ao usuário final.

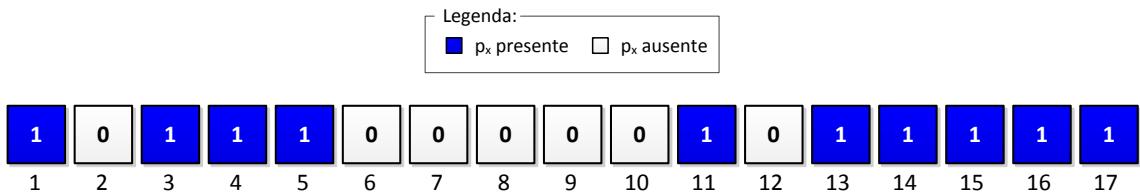


Figura 4.17: Exemplo do mapa de buffer de um nó GMTP com tamanho de 17  $p_x$ .

Um nó GMTP utiliza o mapa de buffer para sinalizar seu atual estado com relação a um determinado fluxo de dados  $P$ . Um nó GMTP pode enviar o mapa de buffer completo, como ilustrado na Figura 4.17, ou o mapa de buffer apenas dos  $p_x$  presentes ou ausentes. Na prática, quando deseja indicar a sua atual disponibilidade, um nó  $r_d$  envia para um nó parceiro  $r_q$  o mapa de buffer dos  $p_x$  presentes e, quando desejar obtê-los, envia o mapa de buffer dos  $p_x$  ausentes. Para diferenciar o tipo de requisição, utiliza-se uma sinalização binária (*flag*) chamada *request-type*, onde 0 significa que o mapa de buffer contém pacotes disponíveis e 1, pacotes ausentes. Note que, quando um nó  $r_d$  transmite um mapa de buffer para um outro nó qualquer  $r_q$ , caracteriza-se automaticamente o uso do método *pull*, em vez do método *push*, que é o modo padrão do GMTP. Salienta-se ainda que se deve evita o método *pull* devido à transitoriedade dos pacotes de dados  $p_x$  (transmissão de dados ao vivo) e um nó  $r_d$  deve apenas realiza tal procedimento após completar a Fase 3 do processo de estabelecimento de conexão. Isto porque um nó  $r_d$  pode nunca receber a resposta para uma requisição do tipo *pull*. Essa sinalização ocorre através do uso do pacote do tipo *GMTP-DataPull-Request*, que é preenchido com o mapa de buffer dos pacotes ausentes e transmitido aos respectivos nós parceiros. Ao receber esse tipo de requisição, um nó parceiro avalia seu conteúdo e responde com o pacote do tipo *GMTP-DataPull-Response*, o qual contém o mapa de buffer dos pacotes disponíveis, seguido dos pacotes  $p_x$  do tipo *GMTP-Data*. Note que os pacotes do tipo *GMTP-DataPullRequest* e *GMTP-DataPull-Response* são transmitidos com garantia de entrega.

Na prática, o mapa de buffer utilizado para sinalizar a presença ou ausência de  $p_x$  é representado por faixas de acordo com o índice do buffer. Por exemplo, para representar o mapa de buffer dos pacotes ausentes ilustrados na Figura 4.17, o nó GMTP preenche o pacote do tipo *GMTP-DataPull-Request* com a sequencia 2;6-10;12. Ao receber esta sequência, o

nó parceiro  $r_q$  responde com o pacote do tipo *GMTP-DataPull-Response*, que contém o mapa de buffer de quais pacotes serão enviados e começa a transmití-los.

#### Descarte de pacotes:

O descarte de pacotes  $p_x$  ocorre sempre no nó  $r_d$  e em duas situações:

1. **Por transbordo do buffer:** o transbordo do buffer pode ocorrer devido ao mecanismo de controle de congestionamento empregado no GMTP, que pode reduzir a taxa de transmissão enquanto novos pacotes precisam ser alocados no buffer. Sendo assim, deve-se descartar os primeiros pacotes  $p_x$  recebidos se o buffer alcançou seu limite, mesmo que ainda não tenham sido repassados. Uma otimização não explorada neste trabalho, mas que é possível de ser realizada, é o descarte seletivo de pacotes, primeiro os que tenham menos impacto na qualidade da mídia, por exemplo, pacotes de dados contendo quadros B (codificação MPEG4, tipo 2). O descarte seletivo de pacotes não impede que o vídeo seja reproduzido, porém com perda de qualidade, mas ao menos se permite a transmissão dos pacotes de dados  $p_x$  de acordo com a largura de banda disponível;
2. **Por duplicação:** ocorre quando o pacote  $p_x$  já foi recebido anteriormente. Tal verificação é feita de acordo com o número de sequência presente em cada pacote  $p_x$ . Note que essa contagem é importante e pode determinar que um nó  $r_d$  desconecte de um nó parceiro  $r_q$ , tal como explicou-se na Seção 4.4.5.

## 4.5 Controle de Congestionamento em $\eta$

No GMTP, executa-se um algoritmo para controle de congestionamento híbrido, cujo comportamento dependerá do modo de transmissão sendo utilizado para transportar os pacotes de dados  $p_x \in P$  (unicast ou em multicast). Como ilustra-se na Figura 4.18, trata-se de dois algoritmos para controle de congestionamento, um que atua em transmissões unicast, chamado de *GMTP Unicast Congestion Control* (GMTP-UCC) e outro que atua em transmissões multicast, chamado de *GMTP Multicast Congestion Control* (GMTP-MCC). Nas próximas seções, discute-se o funcionamento de cada um desses algoritmos.

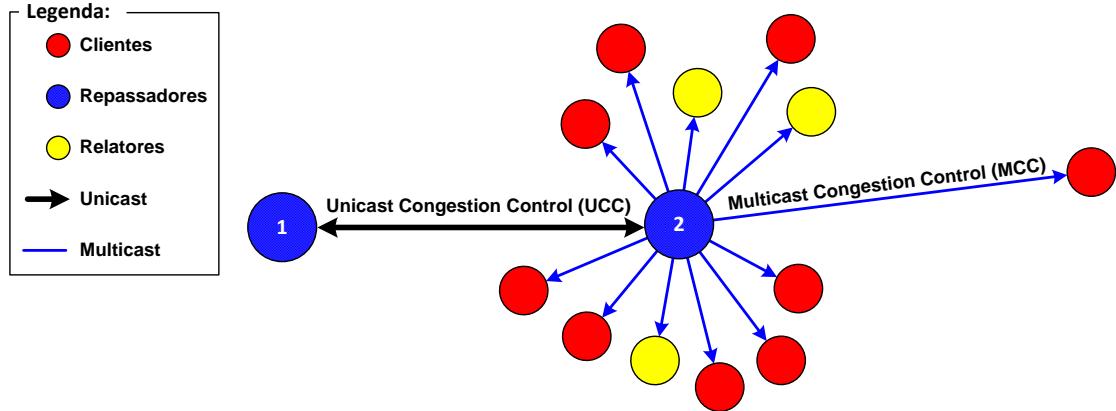


Figura 4.18: Organização do algoritmo de controle de congestionamento no GMTP.

Em modo de transmissão unicast, utilizado na comunicação entre os nós  $r_d$ , define-se a taxa de transmissão de um nó GMTP através de uma versão modificada do protocolo RCP [57]. Já em modo de transmissão multicast, executa-se uma versão modificada do TFRC [160], utilizando-se relatórios transmitidos por nós relatores  $l_w \in C_i(r_d)$ , eleitos em cada rede e controlados por um nó  $r_d$ .

#### 4.5.1 Controle de Congestionamento Unicast

O GMTP-UCC funciona de forma similar ao protocolo RCP, porém com alguns diferenciais a serem discutidos a seguir. O RCP é um protocolo para controle de congestionamento assistido pela rede que tenta emular um Comutador Compartilhado (*Processor Sharing* – PS), por exemplo, um roteador [55]. Nesse contexto, entende-se que se um roteador pudesse obter a informação exata sobre o número de fluxos de entrada em um instante  $\hat{t}$ , a taxa de transmissão ideal para cada fluxo de dados seria  $R_{ps}(\hat{t}) = \frac{C}{N(\hat{t})}$ , onde  $C$  corresponde à capacidade do *enlace* e  $N(\hat{t})$  o número de fluxos no instante  $\hat{t}$ .

Partindo desse ponto, Nandita et. al [57] argumenta que para um roteador funcionar de forma equânime, este deve oferecer a mesma taxa de transmissão para todos os fluxos transmitidos através dele, mantendo-se o número de pacotes na fila de roteamento perto de zero, a fim de evitar que apenas os fluxos que tem pacotes na fila de repasse compartilhem a largura de banda disponível. Com base nisso, Nandita et. al [57] determinou a Equação 4.1, onde  $R(\hat{t})$  é a taxa de transmissão que deve ser oferecida para cada fluxo de dados que passa pelo roteador. Pela Equação 4.1, estima-se a largura de banda disponível em um determinado

canal, representada pela porção  $\alpha(C - y(\hat{t})) - \beta \frac{q(\hat{t})}{h_0}$  (mudança agregada) e a divide por  $N(\hat{t})$ . Porém, como é impossível determinar o valor exato de  $N(\hat{t})$ , estima-se  $\hat{N}(\hat{t}) = \frac{C}{R(\hat{t}-H)}$  e para atualizar  $R(\hat{t})$  com mais frequência do que no tempo de um RTT, escala-se a mudança agregada por  $\frac{H}{h_0}$ , resultando na Equação 4.2, onde:

$$R(\hat{t}) = R(\hat{t} - h_0) + \frac{\alpha(C - y(\hat{t})) - \beta \frac{q(\hat{t})}{h_0}}{\hat{N}(\hat{t})} \quad (4.1)$$

$$R(\hat{t}) = R(\hat{t} - H) \left[ 1 + \frac{\frac{H}{h_0} \left( \alpha(C - y(\hat{t})) - \beta \frac{q(\hat{t})}{h_0} \right)}{C} \right] \quad (4.2)$$

- $h_0$ , é a média móvel dos valores de  $RTT_s$ , calculada através da Equação 4.3, onde  $\theta$  é o ganho e corresponde a 0.02. Note que quanto maior o valor de  $\theta$ , mais rápida será a convergência de  $h_0$  ao valor de  $RTT_s$ .  $RTT_s$  é o tempo de ida e volta calculado entre o nó transmissor e o receptor.

$$h_0 = \theta \times RTT_s + (1 - \theta) \times h_0 \quad (4.3)$$

- $H = \min(RTT_{user}, h_0)$ , sendo  $RTT_{user}$  um tempo definido pelo usuário (administrador do roteador), caso seja necessário atualizar  $R(\hat{t})$  mais rápido do que o tempo de  $h_0$ ;
- $R(\hat{t} - H)$ , é a última taxa de transmissão medida;
- $y(\hat{t})$ , é a taxa de tráfego de entrada medida no intervalo entre a última atualização da taxa de transmissão e  $h_0$ ;
- $q(\hat{t})$ , é o tamanho instantâneo da fila de repasse, em bytes. No GMTP, esse valor é obtido pela soma de todos os pacotes  $p_x$  presentes no buffers circular, para todos os fluxos de dados  $P$  registrado na tabela de repasse. Nesse caso, um nó  $r_d$  mantém um buffer geral e um buffer para cada fluxo de dados  $P$ , que esteja repassando aos seus nós  $c_f \in C_i(r_d)$ . Utiliza-se o buffer geral para pacotes de dados que não precisam de tratamentos especiais, como pacotes de protocolos tradicionais, como TCP;
- $\alpha$  e  $\beta$ , são parâmetros pré-definidos que determinam a estabilidade e o desempenho;

- $C$ , é a capacidade do link.

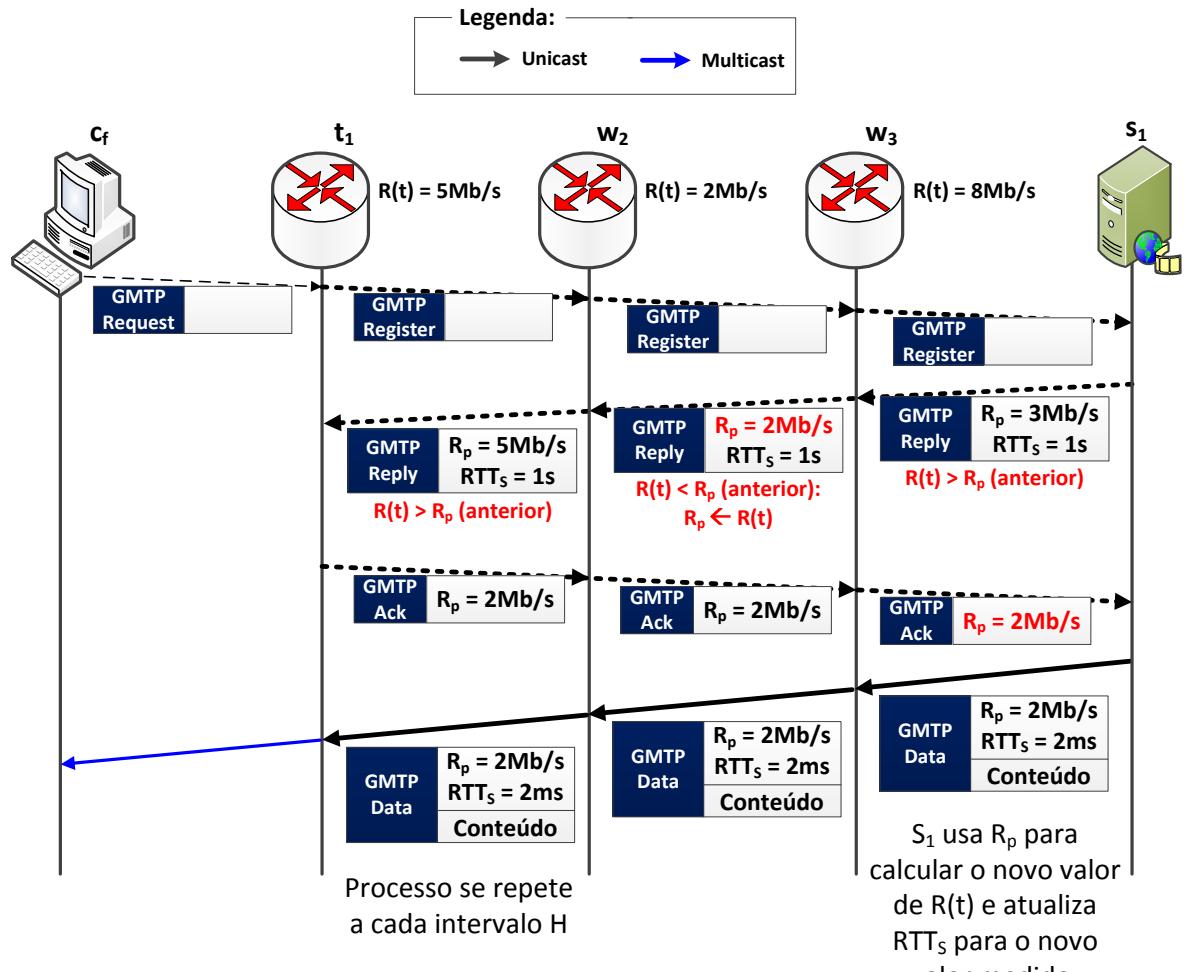


Figura 4.19: Cada  $r_d$  mantém uma única taxa de transmissão  $R(\hat{t})$  que é atribuída no cabeçalho de todos os pacotes transmitidos do nó  $s_a$  aos nós  $w_m \in W_v$ . À medida que o pacote passa em cada  $w_m$ , se a taxa atual  $R(\hat{t})$  no roteador for menor do que  $R_p$ , informada no pacote sendo processado,  $R_p \leftarrow R(\hat{t})$ . Quando o pacote alcançar o último nó  $w_m$ , este envia para  $s_a$  o valor de  $R_p$ , que é a máxima taxa de transmissão suportada no caminho  $W_v$ . Ao receber o valor de  $R_p$ ,  $s_a$  atualiza  $R(\hat{t})$  e o valor de  $h_0$ , utilizando  $R(\hat{t})$  para transmitir os próximos pacotes de dados. Este procedimento se repete a cada intervalo de tempo  $H$ .

Antes de detalhar o funcionamento do GMTP-UCC, a ideia básica é a seguinte: para quaisquer dois nós  $t_1, t_2 \in W_v$ , a taxa de transmissão a ser utilizada por  $t_1$  e  $t_2$  será definida pela menor taxa de transmissão oferecida pelos nós  $w_m \in W_v$  posicionados entre  $t_1$  e  $t_2$ . Isto

significa que o GMTP-UCC segmenta um caminho  $W_v$  em vários sub-caminhos  $W_v^\triangleleft$ . Com isto, se existir largura de banda disponível entre  $t_1$  e  $t_2$ , ou seja,  $C - y(\hat{t}) > 0$ , então o GMTP-UCC compartilhará igualmente o canal entre todos os fluxos, inclusive para o fluxo partindo de  $t_1$  em direção a  $t_2$ . Caso contrário, ou seja, se  $C - y(\hat{t}) < 0$ , considera-se o canal saturado e o GMTP-UCC reduzirá a taxa de transmissão igualmente para todos os fluxos, inclusive para o fluxo partindo de  $t_1$  para  $t_2$ . Por este motivo, o tempo  $H$  é definido entre dois nós  $t_u$  e  $t_{u+1}$  contidos em um caminho  $W_v$ . A consequência dessa estratégia de segmentar um caminho é muito importante e por esse motivo o GMTP-UCC é relativamente diferente se comparado ao RCP, como se observa na discussão a seguir.

O algoritmo adotado no GMTP-UCC, adaptado do RCP, funciona da seguinte forma (acompanhe os passos de acordo com a Figura 4.19):

1° Seja um caminho  $W_v$ , todo nó  $w_m \in W_v$  mantém uma única taxa de transmissão local  $R(\hat{t})$ , que é atribuída em qualquer pacote gerado em  $s_a$  de um fluxo de dados  $P$ , transmitido em direção a qualquer nó  $w_m$ .

2° Todos os pacotes gerados pelo nó  $s_a$  (*GMTP-Register-Reply*, *GMTP-RelayQuery-Reply*, *GMTP-Data*, *GMTP-MediaDesc*, *GMTP-DataPull-Request* e *GMTP-DataAck*) carregam duas informações de controle no campo de cabeçalho:

- *taxa de transmissão proposta* ( $R_p$ ): corresponde à taxa de transmissão inicialmente desejada pelo nó  $s_a$  para transmitir os pacotes de dados  $p_x \in P$  ou a taxa de transmissão do nó  $w_m \in W_v$  com menor capacidade de transmissão em um instante  $t$ ;
- *RTT na fonte* ( $RTT_s$ ): corresponde ao RTT estimado entre quaisquer nós  $t_u$  e  $t_{u+1} \in W_v$ , ou seja, o RTT entre dois nós consecutivos que processam pacotes de dados *GMTP-Data* de um fluxo de dados  $P$ , a fim de repassar aos seus nós  $c_f \in (C_i(t_u) \cup C_i(t_{u+1}))$ . Essa é uma das mudanças do GMTP-UCC se comparado ao RCP. No RCP, utiliza-se apenas os sistemas finais localizados nas extremidades (entre o servidor e o receptor) para determinar esse valor.

3° No início da transmissão de um fluxo de dados  $P$  por parte de  $s_a$ , o nó  $w_m$ , motivado por um ou mais nós  $c_f \in C_i(w_m)$ , transmite um pedido de registro de participação ao

nó  $s_a$ . Ao receber o pacote *GMTP-Register*, o nó  $s_a$  envia um pacote *GMTP-Register-Reply* com o campo  $R_p$  contendo a taxa de transmissão necessária para transmitir o fluxo de dados  $P$ , com o campo  $RTT_s$  igual a 1 s [260]. O valor inicial de  $RTT_s = 1$  s é bastante conservador, mas se decidiu utilizá-lo por ser o valor inicial adotado no protocolo TCP. Além disso, inicia-se um temporizador para medir o próximo RTT instantâneo, que de fato será o primeiro valor correto e alimentará a média móvel  $h_0$ .

4º Todo nó  $w_m \in W_v$ , ao receber qualquer pacote gerado no nó  $s_a$ , verifica se sua capacidade atual de transmissão  $R(\hat{t})$  é menor do que  $R_p$  presente no referido pacote. Em caso afirmativo, atualiza-se  $R_p \leftarrow R(\hat{t})$ , caso contrário, não se realiza nenhuma modificação nesse campo e repassa o pacote a diante. Nesse ínterim, se  $\varphi(w_m, P) = 1$ , ou seja,  $w_m = t_u \in T$ ,  $t_u$  também executa as seguintes ações:

- (a) se o pacote for do tipo *GMTP-Data*, repassa-se o pacote para seus nós  $c_f \in C_i(t_u)$  através do canal multicast, a uma taxa de transmissão definida pelo algoritmo GMTP-MCC, como se discute na próxima seção; e também em direção ao próximo nó  $t_{u+1} \in W_v$ , se houver. A transmissão dos pacotes *GMTP-Data* partindo de  $t_u$  em direção ao nó  $t_{u+1}$  (ou seja, *down-streaming*), ocorre a uma taxa de  $R(\hat{t})$  atualmente definida em  $t_u$ ;
- (b) cria-se um pacote *GMTP-Ack* informando o valor de  $R_p$  e o envia de volta em direção a  $t_{u-1}$ , se  $t_u \neq s_a$ . Ao receber esse pacote de *GMTP-Ack*,  $t_{u-1}$  utilizará  $R_p$  para calcular o novo valor para  $R(\hat{t})$ , pois se trata da menor taxa de transmissão oferecida ao longo do sub-caminho  $W'_v \subset W_v$ , tal que  $W'_v = \{t_u, \dots, w_m, w_{m+1}, w_{m+2}, \dots, t_{u-1}\}$ .

*Observação:* Pela definição de  $W_v$ ,  $t_u$  pode ser o nó  $s_a$ , então esse mecanismo segmenta o caminho  $W_v$  a cada nó  $w_m \in W_v$  e  $w_m \in T$ , incluindo o nó servidor. O objetivo disso é criar sub-fluxos de transmissão dentro de um caminho  $W_v$  de acordo com a capacidade de transmissão e recepção a cada dois nós  $t_u$  e  $t_{u+1}$ . Trata-se de uma peculiaridade GMTP-UCC, pois se evita que um nó  $t_u$  com uma maior capacidade de recepção seja penalizado caso a capacidade de recepção  $R_p$  do próximo nó  $t_{u+1}$  seja menor do que a sua própria capacidade de transmissão.

Este assunto será detalhado na próxima seção.

- 5º A cada instante  $H$ , cada nó  $w_m$  atualiza  $h_0$  de acordo a Equação 4.3), utilizando como parâmetro o valor do campo  $RTT_s$  do último pacote recebido. Em seguida, recalcula-se a taxa de transmissão local  $R(\hat{t})$  usando a Equação 4.2.

**Segmentação de um caminho  $W_v$ :**

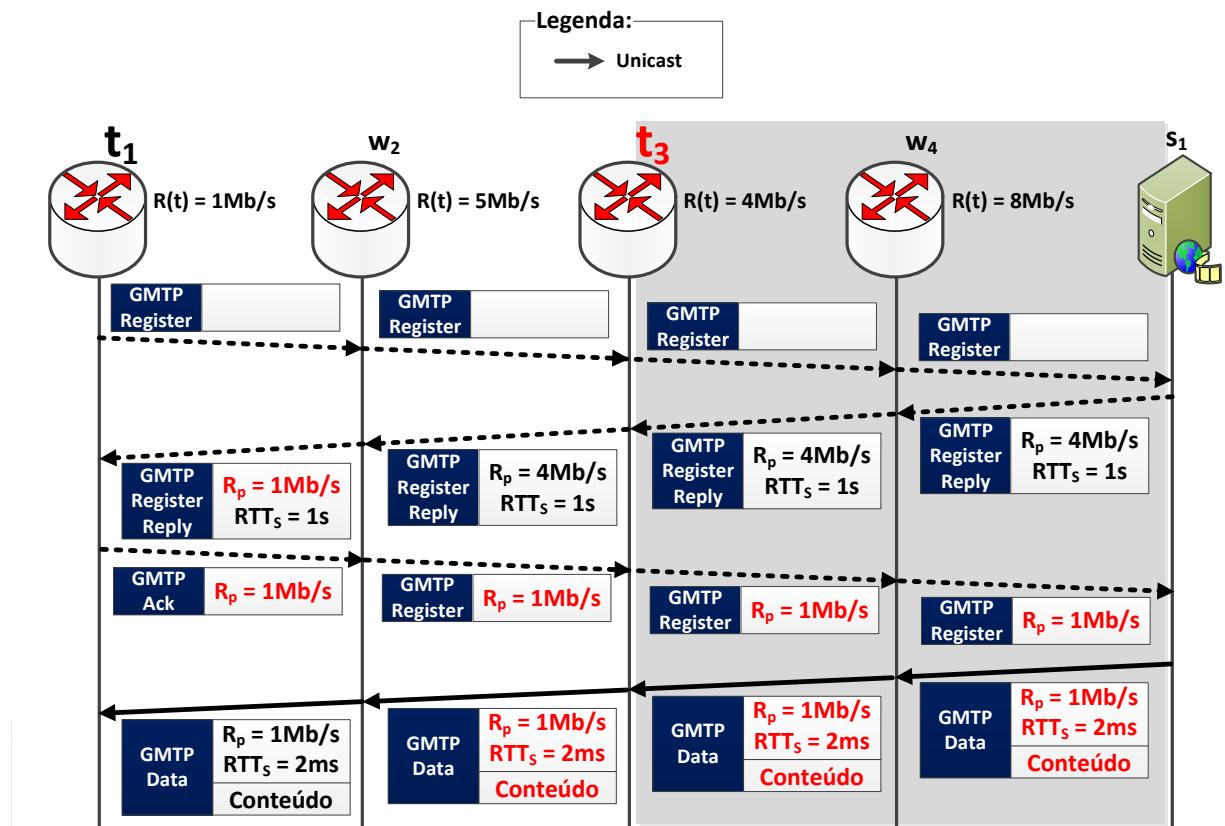


Figura 4.20: O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão, porém isto pode limitar alguns nós a receberem os pacotes de dados em uma taxa maior. Nesse caso, o nó  $t_3$  tinha capacidade para receber o conteúdo a uma taxa de transmissão de  $4\text{ Mb/s}$ , porém a taxa de máxima relatada por  $t_1$  é de  $1\text{ Mb/s}$ , fazendo com que todo o caminho  $W_v$  passasse a receber o fluxo de dados  $P$  a  $4\text{ Mb/s}$ .

O RCP considera todo o caminho entre o nó transmissor e o nó receptor para determinar o novo valor da taxa de transmissão do nó transmissor, especificado por  $R(\hat{t})$  e atualizado a cada instante  $H$  utilizando o novo valor medido de  $R_p$  e  $h_0$ . Porém, essa estratégia pode

limitar alguns nós  $c_f$  a receberem os pacotes de dados  $p_x \in P$  em uma taxa maior, quando disponível. Por exemplo, na Figura 4.20, ilustra-se um cenário de transmissão usando apenas o RCP, abstraindo-se os nós  $c_f \in C_i(w_m)$ . Nesse cenário, observa-se um caminho  $W_v = \{t_1, w_2, t_3, w_4\}$ . Isto significa que existem nós  $c_f \in (C_i(t_1) \cup C_i(t_3))$  interessados em receber os pacotes de dados  $p_x$ . Ao utilizar apenas o RCP, o nó  $s_a$  transmitirá pacotes de dados  $p_x$  a uma taxa de transmissão de  $1 \text{ Mb/s}$  tanto para os nós  $c_f \in C_i(t_1)$  quanto para os nós  $c_f \in C_i(t_3)$ . Porém, isso faz sentido apenas para os nós  $c_f \in C_i(t_1)$  e não para os nós  $c_f \in C_i(t_3)$ , visto que em  $t_3$  o valor de  $R(\hat{t})$  é igual a  $4 \text{ Mb/s}$  e o nó  $w_4$  não limita o uso dessa taxa de transmissão para os nós  $c_f \in C_i(t_3)$ , uma vez que em  $w_4$  o valor de  $R(\hat{t})$  corresponde a  $8 \text{ Mb/s}$ . No caso do GMTP-UCC, esta limitação foi superada ao determinar que se  $\varphi(w_m, P) = 1$ , ou seja,  $w_m = t_u \in T$ , então a taxa de transmissão informada em  $R_p$  será utilizada por  $w_m$ , porém não será considerada para determinar a taxa de transmissão do próximo nó  $t_{u+1}$ . Por isso, o GMTP-UCC segmenta o caminho  $W_v$  de acordo com a menor taxa de transmissão entre dois nós  $t_u$  e  $t_{u+1}$ , como já foi discutido anteriormente.

Sendo assim, considerando o mesmo exemplo ilustrado na Figura 4.20, mas adotando essa estratégia de segmentar o caminho  $W_v$ , tal cenário corresponde ao ilustrado na Figura 4.21. Note que, no sub-caminho  $W_1^\triangleleft = \{t_3, w_2, t_1\}$ , a taxa de transmissão de  $t_3$  em direção a  $t_1$  será de  $1 \text{ Mb/s}$ , ao passo que no sub-caminho  $W_2^\triangleleft = \{s_1, w_4, t_3\}$  será de  $4 \text{ Mb/s}$ . Sendo assim, os nós  $c_f \in C_i(t_1)$  receberão o fluxo de dados  $P$  em uma taxa de  $1 \text{ Mb/s}$ , ao passo que os nós  $c_f \in C_i(t_3)$  receberão os pacotes de dados  $p_x$  a uma taxa de  $4 \text{ Mb/s}$ .

### **Ordenação dos melhores caminhos com base em $R(\hat{t})$ :**

Na Seção 4.4.5, discutiu-se que na Fase 3 de conexão do GMTP, um nó  $s_a$  pode sugerir possíveis nós  $r_q$  como candidatos a parceiros de um nó solicitante  $r_d$ . O primeiro critério para sugerir nós parceiros  $r_q$  é priorizar aqueles que fazem parte de um caminho  $W_v$  com maiores capacidade de transmissão. No GMTP isto é possível porque os nós  $s_a$  conhecem a capacidade de transmissão de todo o caminho, inclusive os pontos onde se inicia um sub-caminho e inicia outro, obtidos pelo Passo 4b do algoritmo GMTP-UCC. Sendo assim, dependendo da posição de um nó solicitante  $r_d$  em um caminho  $W_v$ , o nó  $s_a$  determina os parceiros  $r_q$  e os sugere ao nó  $r_d$  através da intersecção de caminhos conhecidos, utilizando como critério

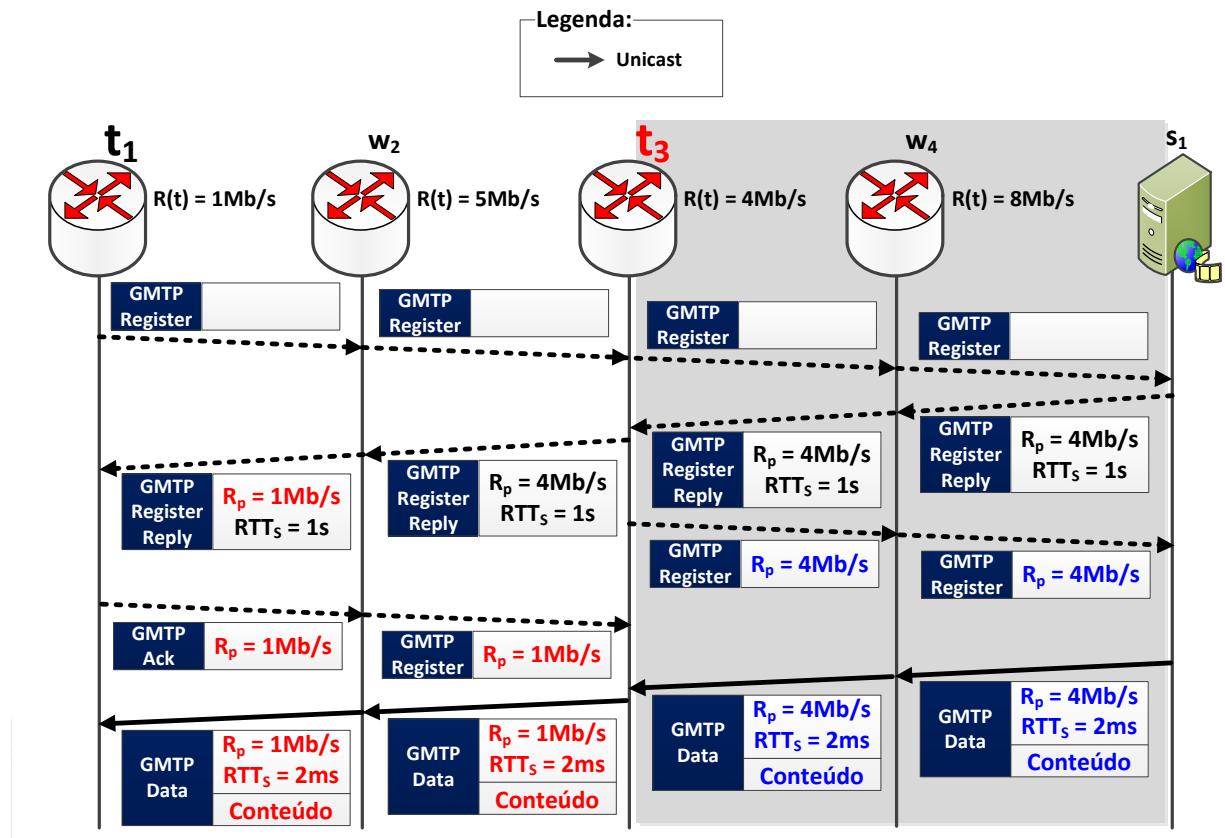


Figura 4.21: O GMTP-UCC segmenta o caminho e dessa forma não limita a taxa de transmissão de um fluxo de dados para certos nós capazes de receber em uma taxa de transmissão maior.

de ordenação as capacidades de transmissão dos caminhos conhecidos  $W$ .

#### Escolha do algoritmo RCP em detrimento ao TCP e ao XCP:

A motivação para o RCP é identificar um algoritmo para controle de congestionamento simples e prático para emular a capacidade de transmissão de um PS ( $R_{ps}(\hat{t})$ ), independente da característica do tráfego e das condições da rede. A abordagem adotada no RCP é diferente se comparada ao TCP e ao XCP. No RCP, em vez de monitorar a mudança de uma janela deslizante a cada tempo de RTT, busca-se determinar se existe uma taxa de transmissão a qual o roteador pode oferecer para todos os fluxos de modo a emular um PS, sem manter estado e nem filas por fluxo de dados, tampouco computação por cada pacote no roteador. Tanto o RCP quanto o XCP são os protocolos mais conhecidos do estado da arte que ten-

tam emular um PS e, por este motivo, suas equações de controle de congestionamento são similares. Porém, o modo que o RCP e o XCP tentam convergir suas respectivas taxas de transmissão  $R_{rcp}(\hat{t})$  e  $R_{xcp}(\hat{t})$  é bastante diferente, alocando-se suas taxas para cada fluxo de dados a fim de emular  $R_{ps}(\hat{t})$ . Dessa forma, foi fundamental decidir qual dos dois protocolos seria mais adequado ao GMTP-UCC e, para tomar tal decisão, estudou-se as diferenças entre tais protocolos, com base no que se apresenta a seguir e detalhado em [55, 57].

Especificamente, a principal diferença entre o RCP e o XCP está no tipo de informação enviada para um nó transmissor de um fluxo de dados para atualizar o valor de  $R_{rcp}(t)$  ou de  $R_{xcp}(t)$ . O XCP continuamente tenta convergir a taxa de transmissão para um ponto de equilíbrio onde todos os transmissores transmitirão pacotes de dados a uma taxa de transmissão  $R_{xcp}(t)$ , ao passo que o RCP calcula uma única taxa de transmissão que deve ser utilizada por todos os nós transmissores em um certo instante  $t$ . Apesar dessa diferença suscinta, deve-se entender minuciosamente o que isto significa.

No caso do XCP, o protocolo aumenta ou diminui a janela de congestionamento de um fluxo de dados de acordo com o tamanho atual da sua janela de congestionamento. Isto significa que o XCP reduz gradativamente os tamanhos da janela de congestionamento dos fluxos com  $R_{xcp}(t)$  maior do que o  $R_{ps}(t)$  estimado, aumentando-se gradativamente o tamanho das janelas de congestionamento dos fluxos com  $R_{xcp}(t)$  menor do que  $R_{ps}(t)$  estimado. Porém, como se sabe, o tamanho da janela de congestionamento é sempre menor para os fluxos iniciados mais recente. Assim, em qualquer momento, os fluxos XCP podem ter diferentes tamanhos de janela de congestionamento e de RTTs, portanto diferentes taxas de transmissão  $R_{xcp}(t)$ , resultando em valores para  $R_{xcp}(t)$  não equânimis para todos os fluxos de dados.

Para se ter uma visão mais adequada, nos gráficos da Figura 4.22, compara-se o TCP e o XCP com um PS ideal com base em uma rede simulada, com taxa de entrada de pacotes de dados definida em *Poisson* e tamanhos dos fluxos em distribuição *Pareto*, com média de 30 pacotes (1000 bytes/pacote), *shape* igual a 1.4, capacidade do link igual a 2.4 Gbps e RTT igual a 100 ms, com carga ofertada igual a 0.9. No gráfico da esquerda, ilustra-se o tempo médio de duração de um fluxo (quanto tempo o respectivo protocolo gasta para completar o fluxo) em função do tamanho do fluxo. No gráfico da direita, ilustra-se o número de fluxos ativos em função do tempo. Os valores de PS foram calculados a partir de expressões analíticas [261] e mostram que os fluxos poderiam ser finalizados uma ordem de magnitude

mais rápida do que o TCP.

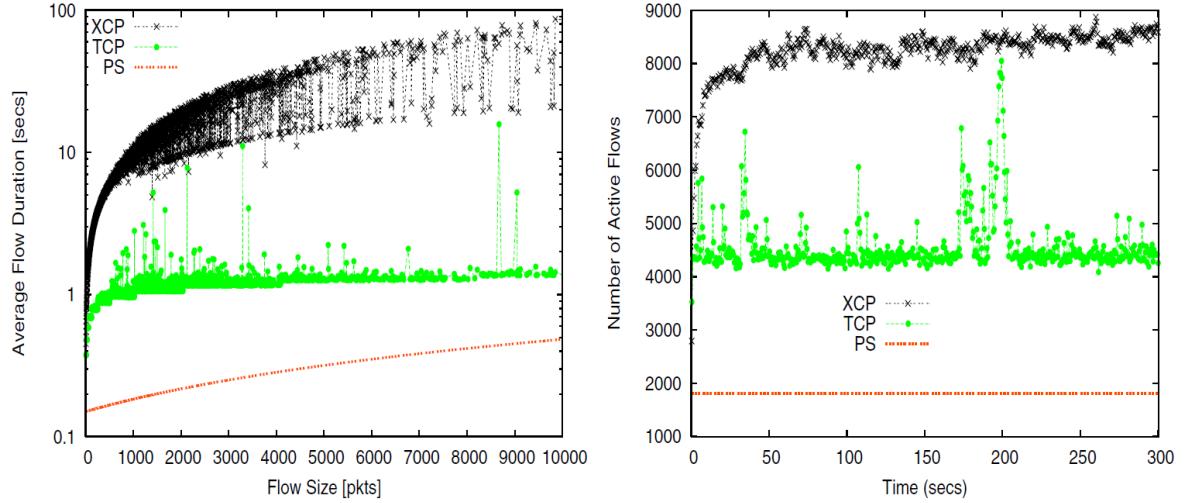


Figura 4.22: No gráfico da esquerda, ilustra-se o tempo médio de duração (quanto tempo leva para finalizar o fluxo) de um fluxo versus o tamanho do fluxo utilizando o TCP e o XCP. No gráfico da direita, ilustra-se o número de fluxos ativos versus o tempo. Ambos os gráficos são resultados de simulações com chegada de fluxo em *Poisson* e tamanhos do fluxo em distribuição *Pareto* com média de 30 pacotes (1000 bytes/pacote), *shape* igual a 1.4, capacidade do *enlace* igual a 2.4 Gbps e RTT igual a 100 ms, com carga ofertada igual a 0.9. Os valores de PS foram calculados a partir de expressões analíticas. Extraído de [55].

Com base nos gráficos da Figura 4.22, observa-se que os fluxos TCP demoram para finalizar porque consome-se múltiplos RTTs na fase de partida lenta para encontrar uma taxa de transmissão equânime, além do mais, muitas vezes, o fluxo acaba antes que tal taxa seja encontrada. Em seguida, quando o fluxo TCP entra no modo de prevenção de congestionamento, o TCP adapta-se lentamente devido ao método de aumento aditivo, o que aumenta o tempo de finalização do fluxo. Além disso, o TCP deliberadamente preenche o buffer dos roteadores saturados de modo a ajustar a taxa de transmissão com base nos descartes de pacotes, mas buffers adicionais resulta em aumento no tempo (atraso) para entregar um pacote de dados, impactando no tempo total de duração de um fluxo. Já o XCP funciona melhor em redes com altos produtos largura de banda-atraso. Os roteadores disponibilizam para as fontes transmissoras relatórios sobre as mudanças da janela de congestionamento, enviados em múltiplos RTTs, que funcionam a contento quando todos os fluxos são de longa duração. Por isso, em um ambiente dinâmico, o XCP pode aumentar a duração de cada fluxo em relação

ao PS ideal, resultando em mais fluxos de dados em trânsito na rede em qualquer instante, principalmente os fluxos de curta duração.

Já no RCP, todos os fluxos (novos e antigos) recebem a mesma taxa de transmissão  $R_{rcp}(t)$  baseada no estado atual do nó  $r_d$  com menor largura de banda disponível em um certo instante  $t$ . Isto permite que um fluxo de dados de curta duração termine o mais rápido possível, ao passo que os fluxos de dados mais longos não influenciam diretamente no compartilhamento equânime do PS, sem permitir que parte da largura de banda disponível fique ociosa por muito tempo. Este procedimento ocorre em um intervalo de tempo definido por  $H$  (vide Equação 4.2).

Já ao observar o gráfico da Figura 4.23, percebe-se que a estratégia do RCP de compartilhar uma única taxa de transmissão para qualquer fluxo com base no estado atual do roteador saturado, produz um resultado satisfatório no que diz respeito a melhor utilizar o canal de transmissão (seja quando em altos níveis de utilização quanto de ociosidade). Com base no gráfico, percebe-se que, em comparação ao XCP e a outras soluções tradicionais como o TCP, o RCP emula melhor um PS e por isso acompanha o tempo médio de finalização de um fluxo de dados à medida que se aumenta o tamanho do fluxo. Note que, para o cenário descrito, o XCP teve um desempenho pior se comparado, inclusive, ao TCP.

O XCP é computacionalmente mais complexo do que o RCP, uma vez que o XCP define diferentes valores de *feedback* para cada fluxo, envolvendo operações matemáticas (multiplicação e soma) para cada pacote, o que torna o XCP mais lento que o RCP. Pela estratégia de mudança no tamanho da janela de congestionamento, o XCP pode levar múltiplos RTTs para a maioria dos fluxos alcançarem a taxa de transmissão equânime entre eles, mas que mudam com o passar do tempo à medida que novos fluxos são injetados na rede e outros são finalizados, devido à natureza dinâmica das redes. No caso do RCP, essa complexidade é menor e há uma redução significativa na convergência entre a taxa de transmissão praticada  $R_{rcp}(t)$  e a taxa estimada do PS ( $R_{ps}(t)$ ). Isto porque se mantém uma única taxa de transmissão para todos os fluxos, não envolvendo qualquer computação adicional por pacote  $p_x$  que passa por  $r_d$ . Além disso, para determinar  $R_{rcp}(t)$ , utiliza-se apenas o tamanho da fila e a taxa agregada de entrada, sem necessitar manter estado por fluxo de dados e operações matemáticas por pacote de dados.

Desta forma, os aspectos que determinam o funcionamento do RCP são fundamentais

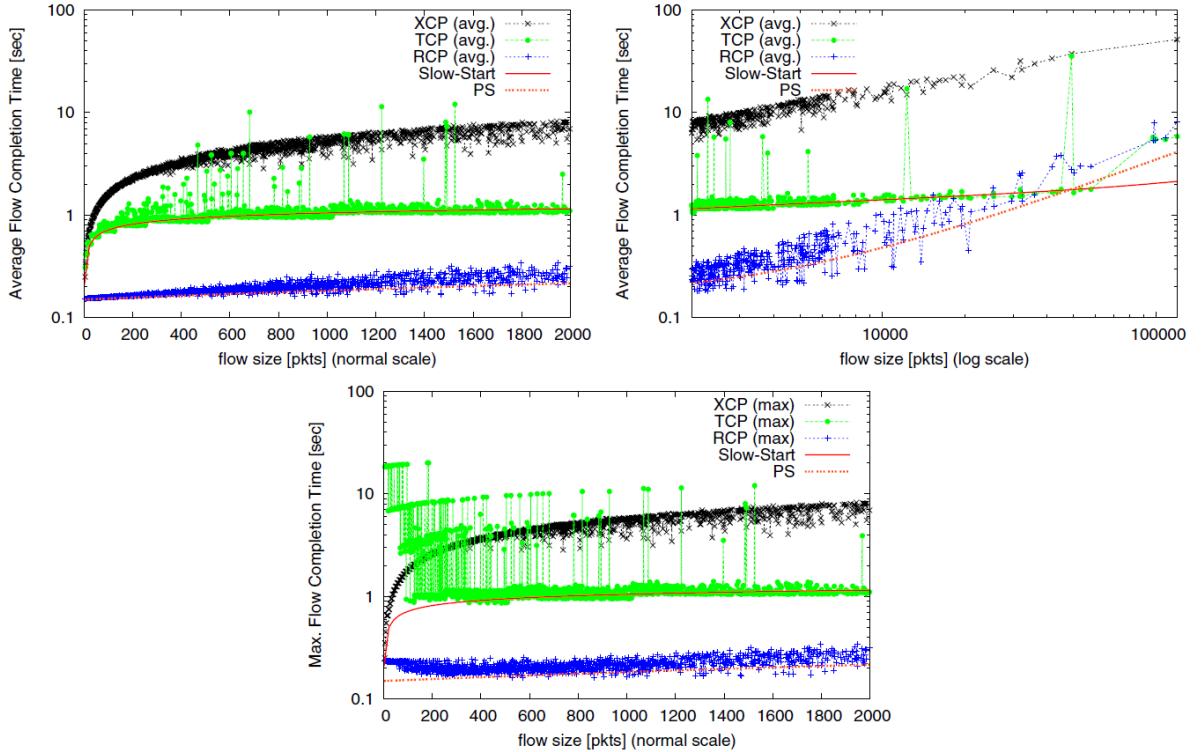


Figura 4.23: Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP como resultados de simulações com taxa de entrada de dados em *Poisson* e tamanhos do fluxo em distribuição Pareto com média de 25 pacotes (1000 bytes/pacote), *shape* igual a 1.2, capacidade do link igual a 2.4 Gbps e RTT igual a 100 ms, com carga ofertada igual a 0.9. Os valores de PS foram calculados a partir de expressões analíticas. Extraído de [55].

quando se trata de transmissão de conteúdos multimídia ao vivo, aliado às outras estratégias adotadas no GMTP. O RCP define uma taxa de transmissão equânime para todos os fluxos, sua reação é rápida às mudanças circunstanciais na rede, tanto para uma super-utilização de um canal quanto para a sua sub-utilização. Como o RCP escala naturalmente com relação à capacidade de transmissão do canal e ao RTT, o seu desempenho é invariante com relação ao tamanho de um fluxo, portanto não importa qual tipo de fluxo as aplicações geram (se de curta ou de longa duração; independente de qualquer protocolo de transporte). Com isto, pode-se permitir que fluxos de dados GMTP/RCP e TCP/RCP coexistam na Internet de forma equânime, aliado às funções do GMTP de distribuição de conteúdo assistida pela rede, evitando-se sobrecarga nos nós  $s_a$ .

### 4.5.2 Controle de Congestionamento Multicast

Da mesma forma que no GMTP-UCC, o objetivo principal do GMTP-MCC é determinar uma taxa de transmissão equânime entre os fluxos de dados transmitidos pelo GMTP e por outros protocolos, como o TCP, porém em modo de transmissão multicast. No caso GMTP-MCC, trata-se de um algoritmo responsável pelo controle de congestionamento em uma rede local constituída por  $\eta_{local} = r_d \cup C_i(r_d)$ . Na prática, os nós da rede  $\eta_{local}$  formam um grupo multicast para a transmissão e recepção de um ou mais fluxos de dados  $P$ , onde o nó  $r_d$  sempre será o transmissor e os nós  $c_f \in C_i(r_d)$  os receptores. A estratégia é que o valor a ser utilizado pelo GMTP-MCC para a taxa de transmissão de fluxo de dados  $P$  seja tão próximo ao valor da taxa de transmissão que o TCP usaria se este fosse utilizado para transmitir  $P$ , tornando-se o GMTP-MCC um algoritmo *TCP-Friendly*. Um fluxo de dados é considerado *TCP-Friendly* quando este não degrada a taxa de transmissão de um fluxo de dados TCP mais do que outro fluxo TCP degradaria se começasse a ser transmitido na rede.

O GMTP-MCC foi inspirado em um protocolo publicado pela IETF chamado *TCP-Friendly Rate Control protocol (TFRC)* (RFC 3448 [262]). O TFRC é um mecanismo para controle de congestionamento de fluxos unicast que tenta prevê a taxa de transmissão de um fluxo TCP e utilizá-la em protocolos diferentes do TCP [160]. Trata-se de uma abordagem diferente da utilizada em algoritmos baseados em janela deslizante e que utilizam pacotes de confirmação para determinar a taxa de transmissão de uma conexão, como acontece no TCP/NewReno. No TFRC, o receptor envia para o transmissor relatórios sobre as perdas observadas e, com base nesse relatório, o transmissor calcula a nova taxa de transmissão. O TFRC é categorizado com um protocolo de controle de congestionamento baseado em uma equação matemática (*Equation Based Congestion Control*) e algoritmos desse tipo são adotados em diversos protocolos, como no CCIDs 3 e 4 do DCCP [263, 264]. Em resumo, o algoritmo TFRC funciona da seguinte forma:

- 1º o receptor mede a taxa de perda de pacotes e a envia para o nó transmissor;
- 2º o nó transmissor usa esse relatório para medir o RTT até o receptor;
- 3º o nó transmissor utiliza a Equação 4.4 para determinar qual será a sua próxima taxa de transmissão em função do relatório de perdas e o RTT obtidos;

- 4º o nó transmissor ajusta sua taxa de transmissão para o valor calculado no passo anterior.

$$R(s, p) = \frac{s}{RTT \times \left( \sqrt{\frac{2 \times p}{3}} + \left( 12 \times \sqrt{\frac{3 \times p}{8}} \right) \times p \times (1 + 32 \times p^2) \right)} \quad (4.4)$$

Na Equação 4.4 [265],  $R(s, p)$  é a taxa de transmissão medida em bytes/segundo definida em função de  $s$ , que é o tamanho do pacote medido em bytes e  $p$ , que corresponde a taxa de perda de pacotes observado pelo nó receptor;  $RTT$  é o tempo de ida-volta entre o nó transmissor e o receptor, medido em segundos.

Apesar de ser uma estratégia interessante e funcionar em conexões unicast, em transmissões multicast o algoritmo descrito anteriormente não é eficiente. O algoritmo é limitado devido a um problema conhecido por *explosão de retorno (feedback implosion)*. Esse problema ocorre quando há muitos receptores enviando relatórios de perdas para o mesmo transmissor, o que resulta em uma inundação de relatórios, os quais o transmissor é incapaz de processar em tempo hábil.

Nesse contexto, para evitar o problema da *explosão de retorno*, determinou-se que apenas alguns nós  $c_f$  são obrigados a enviar tais relatórios ao nó  $r_d$ . Estes nós são chamados de nós relatores e representados por  $l_w$ . No GMTP-MCC, a versão original do TFRC foi alterada e funciona da seguinte forma:

- 1º O nó  $r_d$  executa um algoritmo de eleição de nós relatores  $l_w \in C_i(r_d)$ . Na Seção 4.7.3, descreve-se o procedimento para eleger os nós  $l_w$ .
- 2º Os nós  $l_w$  calculam a taxa de transmissão utilizando a Equação 4.4, em vez do transmissor realizar este cálculo, como na versão original do TFRC;
- 3º Os nós  $l_w$  determinam a taxa de eventos de perda, e não todos os receptores do grupo multicast. Para calcular o evento de perda  $p$ , utiliza-se o mesmo procedimento feito pelo TFRC, onde um intervalo de perda é determinado por consecutivas perdas de pacotes, desde do primeiro pacote perdido até o último pacote perdido, seguido de um pacote recebido com sucesso [262, 265];

- 4° O RTT é calculado entre o nó  $l_w$  e o nó  $r_d$ , com o temporizador controlado pelos nós  $l_w$  e não pelo nó  $r_d$ . Isto evita que o nó  $r_d$  tenha que manter estado de temporizador para cada fluxo de dados  $P$  transmitido para os nós  $c_f \in C_i(r_d)$ . Para determinar o valor do parâmetro RTT e calcular a taxa de transmissão através da Equação 4.4, o GMTP-MCC utiliza a Equação 4.3, que também é utilizada no GMTP-UCC, porém com  $\theta = 0.25$ , valor igual ao utilizado no TCP e no DCCP;
- 5° A taxa de transmissão a ser utilizada pelo nó  $r_d$  é a média aritmética de todas as taxas enviadas pelos nós  $l_w$ ;
- 6° Repete-se todos os passos a partir do passo 2 a cada intervalo igual ao RTT ou quando um intervalo de perda  $p$  é determinado.

Teoricamente, o GMTP-MCC seria um protocolo *TCP-Friendly* se  $R(s, p)$  fosse o valor máximo entre as taxas de transmissão relatadas pelos nós  $l_w$ . Porém, optou-se por utilizar a média aritmética dos valores relatados pelos nós  $l_w$  porque, na prática, diversos fatores podem alterar o estado da rede no instante da transmissão usando o valor máximo da taxa de transmissão reportada pelos nós  $l_w$ . Com esta decisão, define-se uma margem de segurança evitando-se que o GMTP-MCC alcance o limite superior para o valor da taxa de transmissão de um fluxo transmitido com TCP. Além disso, a média aritmética suaviza os valores subsequentes para a taxa de transmissão a ser utilizada pelo nó  $r_d$ .

Um aspecto importante na medição do RTT está relacionado com o início de uma conexão GMTP, pois não se sabe o valor para inicial para RTT até o final do processo de estabelecimento de uma conexão. Nesse caso, deve-se utilizar um valor consideravelmente alto para evitar taxas de transmissões maiores do que a rede tem capacidade de suportar. No GMTP, utiliza-se o valor inicial de RTT igual a  $64\text{ ms}$ , que é um valor relativamente alto se considerar apenas redes locais em condições normais, que é o caso aqui. Quando um nó  $c_f$  envia um pedido de conexão utilizando o pacote do tipo *GMTP-Request*, o mesmo deve realizar a sua primeira medição do valor de RTT, iniciando-se o marcador de tempo para o cálculo do RTT quando enviar o primeiro *GMTP-Request* e parando-o quando receber o pacote do tipo *GMTP-Response*. Em seguida, deve-se acionar o mecanismo de cálculo da taxa de transmissão através da Equação 4.4, caso o respectivo nó  $c_f$  seja eleito um nó relator.

## 4.6 Autenticidade de $P$

Em uma solução baseada em um modelo de serviço P2P, é possível que nós mal-intencionados  $r_d$  poluam o sistema com conteúdos que não foram gerados pelo nó servidor. Para evitar esse tipo de ataque, executa-se um procedimento para verificar a autenticidade de um fluxo de dados  $P$ . Para isto, os próprios nós  $w_m \in W_v$  verificam se o conteúdo de um pacotes de dados  $p_x \in P$  foi alterado por algum nó  $w_m$  anterior durante o procedimento de repasse. Apenas após comprovar a autenticidade de um pacote  $p_x$ , o nó  $w_m$  repassa tal pacote de dados  $p_x$  para o próximo nó  $w_{w+1}$ , transmitindo-os também para seus nós  $c_f \in C_i(w_m)$ , se houver demanda. Este procedimento evita que todos os nós  $c_f$  que receberem o fluxo de dados  $P$  tenham que verificar a autenticidade dos pacotes  $p_x$ , evitando-se que a rede repasse conteúdo multimídia errados, consequentemente não consumindo recursos computacionais desnecessários.

Na prática, o ideal seria que todos nós  $w_m$  verificassem a autenticidade de cada pacote  $p_x$ , porém, tal ação pode onerar os recursos computacionais de cada nó  $w_m$  e aumentar o tempo de entrega de  $p_x$  aos nós  $c_f \in C_i(w_m)$ . Isto porque os nós  $w_m$  também processam cada pacote de dados  $p_x$  para decidir sobre seu repasse e para executar os algoritmos de controle de congestionamento, como discutiu-se nas Seções 4.3, 4.4 e 4.5.

Para reduzir a sobrecarga de verificação de autenticidade de um fluxo de dados  $P$  em cada nós  $w_m$ , definiu-se duas regras, uma para decidir quais nós devem realizar a verificação de autenticidade (Regra 1) e a outra para determinar a quantidade de pacotes que se deve realizar tal procedimento (Regra 2). Tais regras são definidas a seguir.

1. apenas os nós  $w_m$ , tal que  $\varphi(w_m, P) = 1$  devem realizar o procedimento de verificação de autenticidade do fluxo de dados  $P$ ; e
2. os nós  $w_m$ , definidos pela Regra 1, não devem verificar todos os pacotes  $p_x \in P$ , mas apenas uma quantidade  $pc(t)$  de pacotes de dados  $p_x \in P$ , em um instante  $t$ . Nesse caso, define-se  $pc(t)$ , apresentada na Equação 4.5, em função de:
  - $bs(t, P)$ , o número de pacotes  $p_x \in P$  presentes no buffer de repasse de  $w_m$  em um instante  $t$ ;

- $\frac{1}{|W_v^\triangleleft|-1}$ , a probabilidade de um nó  $r_d \in W_v^\triangleleft$  ter alterado o conteúdo de um ou mais  $p_x$  presente(s) no buffer de repasse de  $w_m$ , onde  $W_v^\triangleleft = \sim(\delta(w_{m+1}, W_v))$  e  $W_v$  é o caminho através do qual se transmite os pacotes de dados  $p_x \in P$ ;

$$pc(t) = \left\lfloor bs(t, P) \times \left(1 - \frac{1}{|W_v^\triangleleft|-1}\right)\right\rfloor \quad (4.5)$$

Sendo assim, quanto mais distante um nó  $w_m$  estiver do nó  $s_a$ , mais pacotes  $p_x \in P$  devem ser verificados. Antes de entender o procedimento para verificar a autenticidade de um pacote  $p_x \in P$ , deve-se entender como o nó  $s_a$  deve gerar os referidos pacotes de dados para que seja possível verificar sua autenticidade. Este procedimento é explicado a seguir.

#### 4.6.1 Transmissão e Assinatura de Autenticidade de $p_x \in P$

Quando o nó  $s_a$  gerar cada pacote de dados  $p_x \in P$ , este deve gerar uma assinatura digital dos dados da aplicação a serem transportados. Em seguida, o nó  $s_a$  deve incluir a assinatura digital gerada no cabeçalho do pacote de dados  $p_x$ , no campo assinatura (*signature*). Para assinar digitalmente o conteúdo da aplicação, utiliza-se o método de criptografia assimétrica RSA, onde  $K_{s_a}^-$  e  $K_{s_a}^+$  representam a chave privada e a chave pública de  $s_a$ , respectivamente. No Trecho de Código 7, apresenta-se o procedimento de assinatura de um pacote  $p_x \in P$  adotado no GMTP, utilizando-se a mesma técnica apresentada na Seção 2.4.

---

**Algoritmo 7:** digitalSignPacket( $p_x$ : GMTP-Data)

---

```

/*  $s_a$  executes this algorithm to digital sign the packet
content using its private key  $K_{s_a}^-$  and a pre-defined
hash function, such as the well-know md5 or sha1
function.  $s_a$  get the value of data field, which is the
content that application wants to transport and
generates a signature by encrypt the hash of the data
using the  $s_a$  private key. After, put the generated
signature in the signature field of the packet  $p_x$ . The
signature field will be used later by a note  $r_d$  to
verify the packet  $p_x$  authenticity executing the
Algorithm 8. */
1 data  $\leftarrow$  getPacketFieldValue ( $p_x$ , 'data') ;
2 hashValue  $\leftarrow$  hash (data) ;
3 signature  $\leftarrow$  encrypt ( $K_{s_a}^-$ , hashValue) ;
4 setPacketFieldValue ( $p_x$ , 'signature', signature) ;
5 return  $p_x$ ;
```

---

#### 4.6.2 Verificação de Autenticidade de $p_x \in P$

Após definir as regras para verificação de autenticidade do fluxo de dados  $P$  e a quantidade de pacotes  $pc(t)$  que um nó  $w_m$  deve verificar, nesta seção discute-se como ocorre o procedimento de verificação de autenticidade de um ou mais pacotes de dados  $p_x \in P$ .

Dada a quantidade  $pc(t)$  de pacotes que  $w_m$  deve verificar suas respectivas autenticidades, o nó  $w_m$  escolhe aleatoriamente (distribuição uniforme) os pacotes  $p_x$  disponíveis no buffer de recepção, gerando um conjunto  $P' \subset P$ . Uma vez definido  $P'$ ,  $w_m$  executa o procedimento de verificação de autenticidade que funciona a seguinte forma. Para cada pacote  $p_x \in P'$ , extrai-se a assinatura do pacote  $p_x$ , gerada pelo nó  $s_a$ , como explicado na Seção 4.6.1. Em seguida, extrai-se o campo de dados para que se possa verificar sua autenticidade. Para isto, gera-se o valor de *hash* do campo de dados e compara-se com o valor de *hash* gerado pelo nó  $s_a$  no momento da transmissão do pacote  $p_x$ . Note que o valor de

*hash* gerado pelo nó  $s_a$  é obtido através de processo de decriptar a assinatura do pacote de dados  $p_x$  utilizando a chave pública do nó  $s_a$ . Assim, se o valor de *hash* gerado com base no conteúdo transportado no pacote  $p_x$  for igual ao valor de *hash* disponível na assinatura do pacote, conclui-se que o pacote  $p_x$  não foi alterado por nenhum nó  $w_m \in W_v^\triangleleft = \sim(\delta(w_{m+1}, W_v))$ . Se o pacote de dados  $p_x$  não foi alterado, marca-o como aprovado para ser repassado, caso contrário, marca-o como desaprovado e deve ser descartado. No Trecho de Código 8, apresenta-se o procedimento de verificação de autenticidade de um pacote  $p_x \in P$ .

---

**Algoritmo 8:** verifyPacketAuthenticity( $P'$ : array of GMTP-Data)
 

---

```

/*  $w_m$  executes this Algorithm to check if the content of
   a subset of packets  $P' \subset P$  was modified. It marks
   each  $p_x \in P'$  to be relayed or discarded.  $w_m$  uses the
    $s_a$  public key to decrypt the  $p_x$  signature and compares
   it to the hash value of the  $p_x$  content. It marks  $p_x$  to
   be relayed if  $p_x$  content was not modified, otherwise it
   marks  $p_x$  to be discarded, because  $p_x$  was modified by a
   node in  $W_v^\triangleleft = \sim(\delta(w_{m+1}, W_v))$ . */
```

- 1 verifiedPackets  $\leftarrow$  array of boolean;
- 2 **foreach**  $p_x \in P$  **do**
- 3      $signature \leftarrow getPacketFieldValue(p_x, 'signature');$
- 4      $data \leftarrow getPacketFieldValue(p_x, 'data');$
- 5      $verifiedPackets[x] \leftarrow (\text{hash}(data) = \text{decrypt}(K_{s_a}^+, signature));$
- 6 **end**
- 7 **return**  $verifiedPackets$ ;

---

#### 4.6.3 Habilitar / Desabilitar a Validação de Pacotes $p_x \in P$

A função de verificação de autenticidade de um fluxo de dados  $P$  do GMTP é opcional e desabilitada por padrão. Isto porque um sistema de transmissão, em execução na camada de aplicação, pode ou não desejar tal função. Por isso, considera-se que apenas o nó  $s_a$  tem o controle de habilitar tal funcionalidade, e este procedimento requer sinalizar os nós  $w_m$  para

que estes executem o procedimento de verificação de autenticidade descrito na Seção 4.6.2. Para isto, o nó  $s_a$  ativa a opção assinado (*signed*), disponível no pacote de dados *GMTP-Register-Reply*, sinalizando que todos os pacotes de dados  $p_x \in P$  conterá a assinatura da porção de dados sendo transportados naquele pacote de dados, podendo ser verificado pelos nós  $w_m \in W_v$ , desde que  $\varphi(w_m, P) = 1$ .

Note que quando um nó  $c_f \in C_i(r_d)$  solicitar um fluxo de dados  $P$ , em resposta a tal pedido, o nó  $r_d$  retornará um pacote do tipo *GMTP-Request-Notify*. No cabeçalho desse pacote, o nó  $r_d$  deve também ativar a opção assinado (*signed*) para que o nó  $c_f$  seja notificado e entenda que seu nó  $r_d$  realizará a verificação de autenticidade do fluxo de dados  $P$  da forma descrita anteriormente na Seção 4.6.2. Este procedimento permitirá que a aplicação em execução no nó  $c_f$  possa informar ao usuário final que tal funcionalidade está habilitada, por exemplo.

Além disso, como parâmetros de configuração, o usuário administrador do nó  $r_d$  pode habilitar ou desabilitar a opção de verificação de autenticidade dos fluxos de dados  $P$ , mesmo que o nó  $s_a$  possibilite tal verificação, como descrito anteriormente. Por fornecer essa função de verificação da porção de dados transportado em um pacote, no GMTP não se realiza checagem de erro por soma de verificação (*checksum*), tal como em protocolos como TCP, UDP, DCCP e SCTP.

#### 4.6.4 Obtenção da Chave Pública $K_{s_a}^+$ de $s_a$

Um nó  $r_d$  obtém a chave pública  $K_{s_a}^+$  de  $s_a$  através do certificado digital disponível na URI especificada no parâmetro  $f$  da descrição da mídia, como ilustrou-se no Trecho de Código 5, Linha 7, da Seção 4.4.1. Isto ocorre após o nó  $r_d$  receber o pacote *GMTP-Register-Reply*, que confirma o registro de participação ou a conexão para obter um fluxo de dados  $P$ , como apresentou-se no Trecho de Código 1, Linha 7, Seção 4.3.1.

Após obter o referido certificado digital do nó  $s_a$ , o nó  $r_d$  pode realizar *cache* do certificado, que pode ser utilizado quando os próximos nós  $c_f$  realizarem outras requisições ao nó  $s_a$ , evitando ter que obtê-lo a todo instante. De forma alternativa, o usuário administrador do nó  $r_d$  pode obter o arquivo de certificação digital do nó  $r_d$  e informá-lo, por meio de *upload* nas configurações do nó  $r_d$ . Deve ser opcional também para o usuário administrador do nó  $r_d$  escolher se tal nó deve ou não realizar *cache* dos certificados digitais dos nós  $s_a$ .

## 4.7 Outras Considerações

Nesta seção, apresentam-se brevemente outras funcionalidades do GMTP, tais como os canais de comunicação, o procedimento de desconexão e falha de um nó repassador, adaptação de fluxo, eleição de nós relatores.

### 4.7.1 Canais de Comunicação

No GMTP, utilizam-se três canais de comunicação para executar suas funcionalidades, o canal de controle, o de transmissão unicast e o de transmissão multicast. A seguir, definem-se tais conceitos.

#### **Canal de Controle:**

Quando um nó repassador iniciar uma instância do protocolo GMTP, este deve criar um socket multicast no endereço IP 238.255.255.250 e na porta 1900, em toda interface de rede local, ou seja, nas interfaces por onde se permite acesso aos nós clientes. Através desse socket, um nó GMTP é capaz de enviar e receber pacotes de controle utilizados para negociar as funções de transmissão de um determinado fluxo de dados de mídia ao vivo. Por exemplo, utiliza-se este canal para permitir que um nó cliente envie pedidos de conexão e descobrir quais fluxos de dados já estão sendo recebidos e qual canal multicast cada um deles está disponível.

A decisão do uso do endereço IP multicast 238.255.255.250 foi baseada na RFCs 2365 [266], que define o escopo administrativo do uso dos endereços multicast entre 239.0.0.0 e 239.255.255.255. O endereço 238.255.255.250 é definido no escopo de uso global e sua alocação deve ser confirmada pela IANA antes do uso massivo do GMTP na Internet.

#### **Canal de transmissão unicast:**

O canal de controle e recepção unicast é criado por todos os nós repassadores ao iniciar uma instância do protocolo GMTP. Na prática, trata-se de um socket que os nós repassadores formam as devidas parcerias para transmitir os fluxos de dados uns para os outros e, posteriormente, serem disseminados em modo multicast pelos respectivos nós repassadores aos

seus clientes.

Do ponto de vista de roteamento, todo nó repassador deve avaliar os datagramas GMTP e realizar as ações apropriadas, definidas nas próximas seções deste capítulo. Por exemplo, no processo de estabelecimento de conexão, a ser detalhado na Seção 4.4.2, ao processar um pacote GMTP transmitido por um nó cliente, o nó repassador deve verificar se o pacote é do tipo *GMTP-Request* e, em caso positivo, deve-se retornar um pacote do tipo *GMTP-Response* ao nó cliente, se o fluxo de dados de interesse do nó cliente especificado no pacote *GMTP-Request* já estiver sendo recebido por tal nó repassador.

### Canal de repasse multicast

O canal de repasse multicast é utilizado por um nó repassador para encaminhar datagramas vindos de um servidor ou de outro repassador para a rede local. Na prática, esse canal de repasse é um socket multicast criado pelo nó repassador para transmitir os datagramas para todos os seus clientes com interesse em reproduzir um fluxo de dados de um evento ao vivo.

O *socket de repasse multicast* deve ser criado quando um nó repassador começa a receber um determinado fluxo de dados correspondente a um evento de interesse de pelo menos um dos seus clientes. Na prática, quando isto acontece, o repassador deve criar um socket multicast em um endereço IP e número de porta escolhida aleatoriamente na faixa de endereços IP de escopo local 239.192.0.0/14, definida na RFC 2365 [266]. Como se trata de uma faixa de endereçamento IP multicast de domínio local, não se faz necessário registrar o uso desses endereços. Isto significa que para todo fluxo de dado de um evento ao vivo, deve-se alocar um endereço IP e uma porta. No caso do esquema de endereçamento IPv4, será possível definir a transmissão de exatos 17.179.607.040 (dezessete bilhões, cento e setenta e nove milhões, seiscentos e sete mil e quarenta) diferentes fluxos de dados em uma rede local, o que é mais do que suficiente e escalável por vários séculos.

#### 4.7.2 Procedimentos para Desconexão de nós $c_f$ , $l_w$ e $r_d$

O processo de finalização de uma conexão GMTP ocorre com algumas diferenças se comparado com outros protocolos orientados à conexão. Para sinalizar uma desconexão, um nó  $c_f$  transmite um pacote do tipo *GMTP-Close* pelo canal de controle, contendo o nome do fluxo

que deseja se desconectar. Ao receber este tipo de pacote, o nó  $r_d$  transmite ao nó  $c_f$  um pacote do tipo *GMTP-Reset*, sinalizando que está ciente do fechamento da conexão. Nesse ínterim, os nós desalocam recursos relacionados à respectiva conexão. Este procedimento é suficiente para o pedido de finalização de uma conexão de um cliente GMTP, porém para finalizar uma conexão de um nó  $l_w$  e  $r_d$  outros procedimentos são necessários.

#### **Desconexão de um nó $l_w$ :**

Como apresentado na Seção 4.5.2, um nó  $l_w$  é responsável por relatar ao nó  $r_d$  as condições de recepção de pacotes  $p_x \in P$  em uma transmissão multicast e assim determinar a taxa de transmissão que deve ser utilizada para repassar o referido fluxo de dados. Sem os nós  $l_w$ , tal procedimento não seria possível. Sendo assim, deve-se realizar um procedimento para eleger um novo nó  $l_w$  quando um nó com tal responsabilidade solicite desconexão. Os candidatos a se tornar novo  $l_w$  são os nós  $c_f$  já recebendo o fluxo de dados  $P$ , sendo que o nó  $l_w$  em procedimento de desconexão deve esperar que o procedimento de nova eleição seja concluído. Nesse ínterim, o nó  $l_w$  em processo de desconexão deve continuar enviando pacotes do tipo *GMTP-Ack* para o nó  $r_d$ .

#### **Desconexão de um nó $r_d$ :**

Um nó  $r_d$  realiza o procedimento de desconexão não por intervenção da aplicação, mas sim quando  $C_i(r_d) = 0$  para um determinado fluxo de dados  $P$ , ou quando o nó  $s_a$  explicitamente sinaliza a desconexão. Neste caso, pode ocorrer uma situação crítica para todos os nós parceiros  $r_q$  de  $r_d$ , pois teoricamente estes não poderão mais receber os pacotes de dados  $p_x \in P$ . Para evitar um período de instabilidade na recepção de  $P$  por parte dos nós parceiros de  $r_d$ , define-se um parâmetro chamado de período de carência para novas parcerias (*grace period for new partnerships*). Trata-se de um parâmetro que determina o tempo que um nó  $r_d$ , em processo de desconexão, continuará repassando o fluxo de dados  $P$  para seus parceiros  $r_q$ .

O valor para o *período de carência para novas parcerias* é transmitido para os nós parceiros  $r_q$  de  $r_d$ , que por sua vez deve iniciar o procedimento de realizar outras parcerias a fim de continuar recebendo o fluxo de dados  $P$  (Fase 3 do procedimento de conexão do GMTP). Opcionalmente, um nó  $r_d$  pode aceitar receber de seus nós parceiros  $r_q$ , o valor para o pe-

ríodo de carência, desde que não ultrapasse um limite máximo definido pelo administrador de  $r_d$ .

### 4.7.3 Eleição de nós $l_w$

Para um fluxo de dados  $P$ , o primeiro nó  $l_w$  será o nó  $c_f$  que iniciar a primeira conexão para obter o referido fluxo. Os seguintes nós  $l_w$  serão os próximos nós  $c_f$  que se conectar para receber o fluxo de dados  $P$ , até atingir um parâmetro que determinará a quantidade máxima de nós  $l_w$  por fluxo de dados  $P$ . Tal parâmetro pode ser determinado pelo administrador do nó  $r_d$ . Por padrão, utiliza-se  $\frac{1}{6}$  dos nós  $c_f \in C_i(r_d)$  como sendo nós relatores para a transmissão de um fluxo de dados  $P$ .

Sendo assim, à medida que um nó  $r_d$  recebe pacotes do tipo *GMTP-Request*, no pacote de resposta *GMTP-Response*, o nó  $r_d$  ativa um indicador sinalizando que o referido nó  $c_f$  em processo de conexão deverá se comportar como um nó  $l_w$ , passando a enviar relatórios da taxa de transmissão calculada, como discutiu-se na Seção 4.5.2.

Uma outra situação que se faz necessária a eleição de nós  $l_w$  é no procedimento de desconexão, como explicado na Seção 4.7.2. Para esse caso, quando o nó  $r_d$  receber o pacote do tipo *GMTP-Close*, este deve verificar se o referido nó  $c_f$  é um nó  $l_w$ . Em caso afirmativo, o nó  $r_d$  deve transmitir para um dos nós  $c_f$  que também recebe o referido fluxo de dados  $P$  (se houver), um pacote do tipo *GMTP-Elect-Request* e aguardar por um *GMTP-Elect-Response*. Este procedimento deve ocorrer com garantia de entrega.

## 4.8 Sumário do Capítulo

Neste capítulo, apresentou-se o *Global Media Transmission Protocol* (GMTP), um protocolo baseado em uma arquitetura híbrida P2P/CDN para distribuição de mídias ao vivo através da Internet. Para viabilizar a distribuição de um fluxo de dados referente a um evento ao vivo, uma aplicação, em execução em um servidor, obtém o conteúdo e requisita sua transmissão ao protocolo GMTP, por meio de uma API socket. Para disseminar o conteúdo da aplicação, o GMTP utiliza servidores dispostos em uma rede CDN e constitui dinamicamente uma rede P2P, formada pelos roteadores localizados entre os clientes interessados em obter o fluxo de dados multimídia e os servidores da CDN. Para isto, os roteadores expressam interesse em

participar da rede ao realizarem registros de participação (previamente ou sob-demanda) em um ou mais servidores, ao passo que os servidores começam a conhecer todas as possíveis rotas para alcançar os clientes.

Quando um cliente requisita um fluxo de dados a um servidor, seu roteador de borda, participante da rede P2P, assume a responsabilidade de obter o fluxo de dados de interesse. Nesse ínterim, o roteador do cliente transmite um pedido ao servidor e receber como resposta os pacotes de dados referentes à mídia de interesse (em modo unicast/push). À medida que transmissões diretas (servidor → cliente) ocorrem, os roteadores presentes nas rotas entre o servidor e os clientes vão sendo “alimentados” pelos pacotes de dados referente à mídia em questão, ao passo que os roteadores intermediários interceptam os mesmos pacotes de dados quando seus clientes locais também tem interesse pela mesma mídia. Dessa forma, os roteadores formam parcerias entre si, diretamente por interceptação de pacotes de dados ou com base em instruções obtidas através dos servidores, que executam um algoritmo para determinar a intersecção de rotas usadas por outros roteadores para obter o mesmo fluxo a partir do servidor. Isto ocorre quando o servidor percebe detecta pontos comuns nas rotas e então, em vez de aceitar um novo pedido de conexão, recusa-o e na mesma resposta sugere uma lista de roteadores candidatos a parceiros do roteador solicitante, de modo a evitar múltiplas conexões para obter um mesmo fluxo de dados em direção ao servidor. Um roteador pode também solicitar explicitamente uma lista de candidatos a parceiros a fim de aumentar a quantidade de parcerias ou pelo menos conhecer outros roteadores e contatá-los em caso de desconexão com os seus parceiros atuais.

A distribuição do conteúdo em uma rede local sempre ocorre em modo multicast. Para isto, o roteador cria dinamicamente canais multicast e os divulga na rede à medida que recebem pedidos de conexão para obter fluxos de dados que já estão sendo recebidos, Sendo assim, não se transmite mais de um pedido de conexão ao servidor para uma mesma mídia ao vivo. Além disso, nesse capítulo, detalhou-se todo esse processo de conexão, incluindo aspectos sobre controle de *cache* nos roteadores e outras possíveis ações de cada tipo de nó GMTP.

Em seguida, discutiu-se sobre a estratégia para realizar controle de congestionamento durante o processo de disseminação de uma mídia ao vivo. Nesse contexto, propõe-se dois algoritmos. O GMTP-UCC tem como objetivo controlar o congestionamento no núcleo da

rede P2P, além de expor as informações de seu estado aos servidores. Os servidores podem utilizar tal informação para adaptar o conteúdo multimídia de acordo com a capacidade de transmissão da rede, que sempre tende a ser próxima da capacidade de um PS. Já o GMTP-MCC tem como objetivo controlar o congestionamento na rede local, em transmissões dos fluxos de dados multicast. O roteador elege um sub-conjunto de clientes (relatores) responsáveis por enviar relatórios de suas respectivas capacidade de recepção de dados, ao passo que o roteador define a próxima taxa de transmissão com base nesses relatórios. Por utilizar uma estratégia adaptada do TFRC, o GMTP-MCC emula a taxa de transmissão que seria utilizada pelo TCP caso este fosse utilizado para transmitir o fluxo de dados multimídia, tornando o GMTP um protocolo TCP-Friendly.

Por fim, discutiu-se aspectos sobre segurança e os métodos empregados no GMTP para evitar ataques de poluição. O mecanismo de segurança do GMTP permite que os servidores assinem digitalmente os dados transmitidos, ao passo que permite que os roteadores validem se tal conteúdo não foi alterado ao longo do caminho. Além disso, discutiu-se outros aspectos relacionados ao GMTP, como as ações realizadas na desconexão dos nós repassadores, relatores e clientes, bem como a eleição de nós relatores.

As definições das funções do GMTP foram propostas com base em investigações dos trabalhos disponíveis no estado da arte e guiadas por questionamentos sobre quais funções se mostraram eficazes ao longo de 15 anos de pesquisa quando aplicadas em sistemas P2P e P2P/CDN, bem como aquelas que poderiam melhorias (ou ainda, aquelas de faziam sentido se manter na camada de aplicação). Com esta visão, decidiu-se reduz as responsabilidades dos nós clientes e aumentar a responsabilidade dos roteadores de rede no processo de disseminação dos pacotes de dados multimídia, fazendo uso de informações mais precisas sobre a capacidade de transmissão do núcleo da rede e abstraindo-se a complexidade desse processo da camada de aplicação. Consequentemente, padroniza-se a forma como as aplicações distribuem e recebem conteúdos multimídia, permitindo-se que aplicações desenvolvidas por diferentes fornecedores possam cooperar entre si para obter um mesmo conteúdo multimídia de interesse.

A decisão de utilizar roteadores para formar uma rede colaborativa a fim de disseminar o conteúdo multimídia pode melhorar o desempenho das transmissões de conteúdos multimídia ao vivo, pois reduz-se o impacto negativo causado por fatores que desestabilizam

a rede e a qualidade de serviço, tais como a capacidade de processamento e memória dos dispositivos, mobilidade e dinâmica de entrada/saída dos nós clientes (*churn*). Esses fatores são os mais críticos em se utilizar uma rede P2P para distribuição de conteúdos multimídia, principalmente com a popularização dos dispositivos móveis. Por exemplo, usar dispositivos móveis como nós contribuidores de recurso não é uma estratégia apropriada, principalmente em redes híbridas IP/celular (por exemplo, 3G). Para esse último exemplo, a partir de agora, bastará posicionar um roteador GMTP entre a rede IP e a rede celular para atender à demanda de todos os nós móveis em uma rede celular referente a um fluxo de dados multimídia.

# Bibliografia

- [1] Cisco Systems. The zettabyte era—trends and analysis. Technical report, Cisco Systems, 5 2012. [http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI\\_Hyperconnectivity\\_WP.pdf](http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf).
- [2] Yonghong Tian, J. Srivastava, Tiejun Huang, and N. Contractor. Social multimedia computing. *Computer*, 43(8):27–36, Aug.
- [3] N. Leavitt. Network-Usage Changes Push Internet Traffic to the Edge. *Computer*, 43(10):13 –15, 10 2010.
- [4] San Murugesan. Understanding web 2.0. *IT Professional*, 9(4):34–41, 2009.
- [5] Kwei-Jay Lin. Building web 2.0. *Computer*, 40(5):101–102, 5 2007.
- [6] Thiago Silva, Jussara M. Almeida, and Dorgival Guedes. Live streaming of user generated videos: Workload characterization and content delivery architectures. *Comput. Netw.*, 55(18):4055–4068, December 2011.
- [7] Redação da TI Inside Online. Tráfego de internet deve crescer 29 % até 2016, 5 2012. <http://www.tiinside.com.br/30/05/2012/trafego-de-internet-deve-crescer-29-ate-2016/ti/280945/news.aspx>. Último acesso: 28 de Fevereiro de 2014.
- [8] Jeremy Scott. Live streaming video is growing by leaps and bounds, 3 2012. <http://www.reelseo.com/live-streaming-video-growing-leaps-bounds/>. Último acesso: 28 de Fevereiro de 2014.

- [9] Eric Franchi. Live streaming continues momentum with march madness, 3 2009. <http://www.mediapost.com/publications/article/101750/#axzz200qSKoNN>. Último acesso: 28 de Fevereiro de 2014.
- [10] Marshall Kirkpatrick. The numbers are in, live video online is blowing up, 3 2008. [http://readwrite.com/2008/06/05/live\\_video\\_big](http://readwrite.com/2008/06/05/live_video_big). Último acesso: 28 de Fevereiro de 2014.
- [11] Liz Gannes. The obama inauguration live stream stats, 1 2009. <http://gigaom.com/2009/01/20/the-obama-inauguration-live-stream-stats/>. Último acesso: 28 de Fevereiro de 2014.
- [12] Y.J. Won, J.W.-K. Hong, Mi-Jung Choi, Chan kyu Hwang, and Jae-Hyoung Yoo. Measurement of download and play and streaming iptv traffic. *Communications Magazine, IEEE*, 46(10):154–161, October.
- [13] Yanping Zhao, D.L. Eager, and M.K. Vernon. Network bandwidth requirements for scalable on-demand streaming. *Networking, IEEE/ACM Transactions on*, 15(4):878–891, Aug.
- [14] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and Shudong Jin. A hierarchical characterization of a live streaming media workload. *Networking, IEEE/ACM Transactions on*, 14(1):133–146, Feb.
- [15] A. Aurelius, C. Lagerstedt, and M. Kihl. Streaming media over the internet: Flow based analysis in live access networks. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2011 IEEE International Symposium on*, pages 1–6, June.
- [16] B. Morris and M. Trivedi. Real-time video based highway traffic measurement and performance monitoring. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 59–64, 30 2007-Oct. 3.
- [17] Chuan Wu, Baochun Li, and Shuqiao Zhao. On dynamic server provisioning in multichannel p2p live streaming. *Networking, IEEE/ACM Transactions on*, 19(5):1317–1330, Oct.

- [18] Jameson Berkow. Netflix Proclaimed “King” of North American Internet Use, 5 2011. [http://business.financialpost.com/2011/05/17/netflix-named-king-of-north-american-internet/?\\_\\_lsa=60e0-e253](http://business.financialpost.com/2011/05/17/netflix-named-king-of-north-american-internet/?__lsa=60e0-e253). Último acesso: 28 de Fevereiro de 2014.
- [19] Dave Caputo and Tom Donnelly. Global Internet Phenomena Spotlight - Netflix Rising. Technical report, Sandvine Incorporated ULC, 5 2011. [http://www.sandvine.com/downloads/documents/05-17-2011\\_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Spotlight%20-%20Netflix%20Rising.pdf](http://www.sandvine.com/downloads/documents/05-17-2011_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Spotlight%20-%20Netflix%20Rising.pdf).
- [20] N. Staelens, S. Moens, W. Van den Broeck, I. Marien, B. Vermeulen, P. Lambert, R. Van De Walle, and P. Demeester. Assessing quality of experience of iptv and video on demand services in real-life environments. *Broadcasting, IEEE Transactions on*, 56(4):458–466, Dec.
- [21] P. Brooks and B. Hestnes. User measures of quality of experience: why being objective and quantitative is important. *Network, IEEE*, 24(2):8–13, March-April.
- [22] Jin Li. On peer-to-peer (p2p) content delivery. *Peer-to-Peer Networking and Applications*, 1(1):45–63, 2008.
- [23] Bo Li and Hao Yin. Peer-to-peer live video streaming on the internet: issues, existing approaches, and challenges [Peer-to-Peer Multimedia Streaming]. *Communications Magazine, IEEE*, 45(6):94–99, 2007.
- [24] Mu Mu, J. Ishmael, W. Knowles, Mark Rouncefield, N. Race, M. Stuart, and G. Wright. P2p-based iptv services: Design, deployment, and qoe measurement. *Multimedia, IEEE Transactions on*, 14(6):1515–1527, Dec.
- [25] Darshan Purandare and R. Guha. An Alliance Based Peering Scheme for P2P Live Media Streaming. *Multimedia, IEEE Transactions on*, 9(8):1633–1644, 2007.
- [26] A. Vakali and G. Pallis. Content delivery networks: status and trends. *Internet Computing, IEEE*, 7(6):68–74, nov.-dec. 2003.

- [27] L. Kontothanassis, Ramesh Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky. A transport layer for live streaming in a content delivery network. *Proceedings of the IEEE*, 92(9):1408–1419, Sept.
- [28] C.D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, J.E. Van der Merwe, and C.J. Sreenan. Enhanced streaming services in a content distribution network. *Internet Computing, IEEE*, 5(4):66–75, Jul/Aug.
- [29] ZhiHui Lu, XiaoHong Gao, SiJia Huang, and Yi Huang. Scalable and Reliable Live Streaming Service through Coordinating CDN and P2P. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 581–588, Dec.
- [30] 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.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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.
- [36] 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.
- [37] 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.
- [38] 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.
- [39] Yee-Ting Li, D. Leith, and R.N. Shorten. Experimental evaluation of tcp protocols for high-speed networks. *Networking, IEEE/ACM Transactions on*, 15(5):1109–1122, 10 2007.
- [40] Douglas E. Comer. *Interligação em Redes com TCP/IP: Princípios, Protocolos e Arquitetura*. ELSEVIER, 2 edition, 9 2004.
- [41] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM.
- [42] Kunwoo Park, Dukhyun Chang, Junghoon Kim, Wonjun Yoon, and T. Kwon. An analysis of user dynamics in p2p live streaming services. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6, May.
- [43] D. Ciullo, M.A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia. Network awareness of p2p live streaming applications: A measurement study. *Multimedia, IEEE Transactions on*, 12(1):54–63, Jan 2010.
- [44] B.A. Miller, T. Nixon, C. Tai, and M.D. Wood. Home networking with universal plug and play. *Communications Magazine, IEEE*, 39(12):104–109, Dec.

- [45] Chi-Chung Cheung, Man-Ching Yuen, and A.C.H. Yip. Dynamic dns for load balancing. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 962–965, May.
- [46] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Kat-saros, and G. Polyzos. A survey of information-centric networking research. *Communications Surveys Tutorials, IEEE*, PP(99):1–26, 10 2013.
- [47] B. Ahlgren, C. Dannowitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 7 2012.
- [48] J. Pan, S. Paul, and R. Jain. A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26–36, July 2011.
- [49] Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking: Seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks, HotNets-X*, pages 1:1–1:6, New York, NY, USA, 2011. ACM.
- [50] C. Severance. Van Jacobson: Content-Centric Networking. *Computer*, 46(1):11–13, 2013.
- [51] Cheng Wanxiang, Shi Peixin, and Lei Zhenming. Network-assisted congestion control. In *Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences on*, volume 2, pages 28–32.
- [52] Dina Katabi. *Decoupling congestion control and bandwidth allocation policy with application to high bandwidth-delay product networks*. PhD thesis, Cambridge, MA, USA, 2003.
- [53] N. Dukkipati, G. Gibb, N. McKeown, and Jiang Zhu. Building a rcp (rate control protocol) test network. In *High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on*, pages 91–98, Aug.

- [54] H. Balakrishnan, N. Dukkipati, N. McKeown, and C.J. Tomlin. Stability analysis of explicit congestion control protocols. *Communications Letters, IEEE*, 11(10):823–825, October.
- [55] Nandita Dukkipati, Masayoshi Kobayashi, Rui Zhang-Shen, and Nick McKeown. Processor Sharing Flows in the Internet. In *Proceedings of the 13th international conference on Quality of Service*, IWQoS’05, pages 271–285, Berlin, Heidelberg, 2005. Springer-Verlag.
- [56] Nandita Dukkipati and Nick McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.*, 36(1):59–62, January 2006.
- [57] Nandita Dukkipati. *Rate Control Protocol (RCP): congestion control to make flows complete quickly*. PhD thesis, Stanford, CA, USA, 2008. AAI3292347.
- [58] Ashvin Lakshmikantha, R. Srikant, Nandita Dukkipati, Nick McKeown, and Carolyn L. Beck. Buffer sizing results for rcp congestion control under connection arrivals and departures. *Computer Communication Review*, 39(1):5–15, 2009.
- [59] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):89–102, 8 2002.
- [60] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM ’02, pages 89–102, New York, NY, USA, 2002. ACM.
- [61] Yong Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit is Enough. *Networking, IEEE/ACM Transactions on*, 16(6):1281–1294, Dec.
- [62] Huixiang Zhang, Guanzhong Dai, Lei Yao, and Hairui Zhou. Fast convergence of variable-structure congestion control protocol with explicit precise feedback. In Franco P. Preparata, Xiaodong Wu, and Jianping Yin, editors, *Frontiers in Algorithmics*,

- volume 5059 of *Lecture Notes in Computer Science*, pages 264–275. Springer Berlin Heidelberg, 2008.
- [63] M. Vinodhini and P. Arockia Jansi Rani. Article: Usage of variable structure congestion control protocol (vcp) in buffer overflow attack blocker. *International Journal of Computer Applications*, 39(8):46–52, February 2012. Published by Foundation of Computer Science, New York, USA.
- [64] Huixiang Zhang, Guanzhong Dai, Lei Yao, and Hairui Zhou. Fast convergence of variable-structure congestion control protocol with explicit precise feedback. In *Proceedings of the 2nd annual international workshop on Frontiers in Algorithmics*, FAW '08, pages 264–275, Berlin, Heidelberg, 2008. Springer-Verlag.
- [65] Xing Guowen and Xue Shengjun. Study on variable-structure congestion control protocol. In *New Trends in Information and Service Science, 2009. NISS '09. International Conference on*, pages 766–769, 30 2009-July 2.
- [66] B. Briscoe, R. Woundy, and A. Cooper. Congestion exposure (conex) concepts and use cases, 12 2012. <http://www.ietf.org/rfc/rfc6789.txt>. Último acesso: 28 de Fevereiro de 2014.
- [67] Marcelo Bagnulo and Nandita Dukkipati. Congestion exposure group, 6 2010. <https://datatracker.ietf.org/doc/charter-ietf-conex/>. Último acesso: 28 de Fevereiro de 2014.
- [68] Philip Eardley, Michalis Kanakakis, Alexandros Kostopoulos, Tapi Levä, Ken Richardson, and Henna Warma. The Future Internet. chapter Deployment and adoption of future internet protocols, pages 133–144. Springer-Verlag, Berlin, Heidelberg, 2011.
- [69] I. Grosclaude and B. Mathieu. Network multicast opened by network operators to be used by p2p applications. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, pages 44–51, Oct 2013.
- [70] K.C. Almeroth. The evolution of Multicast: From the MBone to Interdomain Multicast to Internet2 Deployment. *Network, IEEE*, 14(1):10–20, Jan 2000.

- [71] F. Bronzino, R. Gaeta, M. Grangetto, and G. Pau. An adaptive hybrid cdn/p2p solution for content delivery networks. In *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pages 1–6, Nov.
- [72] Zhonghua Wei and Jianping Pan. Modeling BitTorrent-Based P2P video streaming systems in the presence of NAT devices. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, June 2011.
- [73] R Rodrigues and P Druschel. Peer-to-peer streaming systems. *Communications of the ACM*, 53(10):3–39, 2010.
- [74] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *Proceedings of the 17th ACM international conference on Multimedia, MM ’09*, pages 25–34, New York, NY, USA, 2009. ACM.
- [75] Sachin Agarwal, Jatinder Pal Singh, Aditya Mavlankar, Pierpaolo Bacchiet, and Bernd Girod. Performance of P2P live video streaming systems on a controlled testbed. In *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities, TridentCom ’08*, pages 6:1–6:10, Innsbruck, Austria, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ACM ID: 1390584.
- [76] Laurent MassouliÃ© and Andrew Twigg. Rate-optimal schemes for Peer-to-Peer live streaming. *Perform. Eval.*, 65(11-12):804–822, November 2008. ACM ID: 1453585.
- [77] Thomas Locher, Remo Meier, Stefan Schmid, and Roger Wattenhofer. Push-to-Pull Peer-to-Peer live streaming. In Andrzej Pelc, editor, *Distributed Computing*, volume 4731, pages 388–402. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [78] S. Venot and Lu Yan. Peer-to-Peer media streaming application survey. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2007. UBICOMM ’07*, pages 139–148. IEEE, November 2007.

- [79] Chao Liang, Yang Guo, and Yong Liu. Hierarchically clustered P2P streaming system. In *IEEE GLOBECOM 2007-2007 IEEE Global Telecommunications Conference*, pages 236–241, Washington, DC, USA, November 2007.
- [80] 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.
- [81] Qi Huang, Hai Jin, and Xiaofei Liao. P2P live streaming with Tree-Mesh based hybrid overlay. In *International Conference on Parallel Processing Workshops, 2007. ICPPW 2007*, pages 55–55. IEEE, September 2007.
- [82] 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.
- [83] Meng Zhang, Li Zhao, Yun Tang, Jian-Guang Luo, and Shi-Qiang Yang. Large-scale live media streaming over peer-to-peer networks through global internet. In *Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming, P2PMMS'05*, pages 21–28, Hilton, Singapore, 2005. ACM. ACM ID: 1099388.
- [84] Hideki Otsuki and Takashi Egawa. A retransmission control algorithm for Low-Latency UDP stream on StreamCode-Base active networks. In Naoki Wakamiya, Marcin Solarski, and James Sterbenz, editors, *Active Networks*, volume 2982, pages 92–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [85] 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.

- [86] Yaning Liu, Hongbo Wang, Yu Lin, Shiduan Cheng, and G. Simon. Friendly P2P: Application-Level congestion control for Peer-to-Peer applications. In *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*, pages 1–5. IEEE, December 2008.
- [87] T. K Yan and H. P Dommel. Multimedia-Aware congestion control for video streaming over the internet. In *Second International Conference on Digital Telecommunications, 2007. ICDT '07*, pages 6–6. IEEE, July 2007.
- [88] Emir Mulabegovic, Dan Schonfeld, and Rashid Ansari. Lightweight streaming protocol (LSP). In *Proceedings of the tenth ACM international conference on Multimedia, MULTIMEDIA '02*, pages 227–230, Juan-les-Pins, France, 2002. ACM. ACM ID: 641051.
- [89] Fabio Pianese. A survey of p2p data-driven live streaming systems. *Streaming Media Architectures, Techniques, and Applications: Recent Advances*, 1:295–310, 10 2011.
- [90] Chuan Wu, Baochun Li, and Shuqiao Zhao. Diagnosing network-wide p2p live streaming inefficiencies. In *INFOCOM 2009, IEEE*, pages 2731–2735, April.
- [91] Xiangyang Zhang and Hossam Hassanein. Survey a survey of peer-to-peer live video streaming schemes - an algorithmic perspective. *Comput. Netw.*, 56(15):3548–3579, October 2012.
- [92] A. Mansy and M. Ammar. Analysis of adaptive streaming for hybrid cdn/p2p live video systems. In *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, pages 276–285, Oct.
- [93] Yishuai Chen, Baoxian Zhang, and Changjia Chen. Modeling and performance analysis of p2p live streaming systems under flash crowds. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, June.
- [94] Bo Li, Susu Xie, G.Y. Keung, Jiangchuan Liu, I. Stoica, Hui Zhang, and Xinyan Zhang. An Empirical Study of the Coolstreaming+ System. *Selected Areas in Communications, IEEE Journal on*, 25(9):1627–1639, 2007.

- [95] Jeff Seibert, David Zage, Sonia Fahmy, and Cristina Nita-Rotaru. Experimental comparison of peer-to-peer streaming overlays: An application perspective. In *33rd IEEE Conference on Local Computer Networks (LCN)*, pages 20–27, Washington, DC, USA, October 2008. IEEE Computer Society.
- [96] Yang Guo, Chao Liang, and Yong Liu. A survey on peer-to-peer video streaming systems. *PeertoPeer Networking and Applications*, 1(1):18–28, 2008.
- [97] E. Setton, P. Baccichet, and B. Girod. Peer-to-peer live multicast: A video perspective. *Proceedings of the IEEE*, 96(1):25–38, Jan.
- [98] M. Handley and V. Jacobson. Sdp: Session description protocol, 4 1998. <http://www.ietf.org/rfc/rfc2327.txt>. Último acesso: 28 de Fevereiro de 2014.
- [99] 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: 28 de Fevereiro de 2014.
- [100] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp), 4 1998. <http://www.ietf.org/rfc/rfc2326.txt>. Último acesso: 28 de Fevereiro de 2014.
- [101] 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.
- [102] 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.
- [103] 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.
- [104] 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.

- [105] 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.
- [106] 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.
- [107] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Addison Wesley, trad. 3 ed. edition, 2006.
- [108] Mouna Allani, Benoit Garbinato, and Fernando Pedone. *Middleware for Network Eccentric and Mobile Applications*, volume 3, chapter Application Layer Multicast, pages 191–218. Springer Berlin Heidelberg, 9 2009.
- [109] Yao Liu, L. Guo, Fei Li, and Songqing Chen. A case study of traffic locality in internet p2p live streaming systems. In *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, pages 423–432, June.
- [110] Chuan Wu, Baochun Li, and Shuqiao Zhao. Exploring large-scale peer-to-peer live streaming topologies. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(3):19:1–19:23, 9 2008.
- [111] N. Magharei, R. Rejaie, and Yang Guo. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1424–1432, 5 2007.
- [112] 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.
- [113] Rosario G. Garropo, Stefano Giordano, Stella Spagna, Saverio Niccolini, and Jan Seedorf. Topology control strategies on p2p live video streaming service with peer churning. *Comput. Commun.*, 35(6):759–770, 3 2012.

- [114] M. Shibuya, Y. Hei, and T. Ogishi. Isp-friendly peer selection mechanism with alto-like server. In *Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific*, pages 1–8, Sept.
- [115] Liang Dai, Yanchuan Cao, Yi Cui, and Yuan Xue. On scalability of proximity-aware peer-to-peer streaming. *Comput. Commun.*, 32(1):144–153, January 2009.
- [116] S. Tang, H. Wang, and P. Van Mieghem. The effect of peer selection with hopcount or delay constraint on peer-to-peer networking. In *Proceedings of the 7th international IFIP-TC6 networking conference on AdHoc and sensor networks, wireless networks, next generation internet*, NETWORKING’08, pages 358–365, Berlin, Heidelberg, 2008. Springer-Verlag.
- [117] Nianwang Liu, Zheng Wen, K.L. Yeung, and Zhibin Lei. Request-peer selection for load-balancing in p2p live streaming systems. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 3227–3232, April.
- [118] Yan Chen, Beibei Wang, W.S. Lin, Yongle Wu, and K.J.R. Liu. Cooperative peer-to-peer streaming: An evolutionary game-theoretic approach. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(10):1346–1357, Oct.
- [119] Constantinos Vassilakis and Ioannis Stavrakakis. Minimizing node churn in peer-to-peer streaming. *Comput. Commun.*, 33(14):1598–1614, September 2010.
- [120] Y. Sakata, K. Takayama, R. Endo, and H. Shigeno. A chunk scheduling based on chunk diffusion ratio on p2p live streaming. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, pages 74–81, Sept.
- [121] Byungryeol Sim, Yeonhee Lee, and Youngseok Lee. A simulation study of application-layer traffic optimization protocol for p2p applications. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 279–282, Oct 2011.
- [122] J. Seedorf, S. Kiesel, and M. Stiemerling. Traffic localization for p2p-applications: The alto approach. In *Peer-to-Peer Computing, 2009. P2P ’09. IEEE Ninth International Conference on*, pages 171–177, Sept 2009.

- [123] Harsha V. Madhyastha, Ethan Katz-Bassett, Thomas Anderson, Arvind Krishna-murthy, and Arun Venkataramani. iplane nano: path prediction for peer-to-peer applications. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 137–152, Berkeley, CA, USA, 2009. USENIX Association.
- [124] Sachin Agarwal, Jatinder Pal Singh, and Shruti Dube. Analysis and implementation of gossip-based p2p streaming with distributed incentive mechanisms for peer cooperation. *Adv. MultiMedia*, 2007(1):8:1–8:12, 4 2007.
- [125] Jaeok Park and M. Van der Schaar. A game theoretic analysis of incentives in content production and sharing over peer-to-peer networks. *Selected Topics in Signal Processing, IEEE Journal of*, 4(4):704–717, Aug.
- [126] Sachin Agarwal and Shruti Dube. Gossip based streaming with incentives for peer collaboration. In *Proceedings of the Eighth IEEE International Symposium on Multimedia*, ISM '06, pages 629–636, Washington, DC, USA, 2006. IEEE Computer Society.
- [127] A. Habib and J. Chuang. Incentive mechanism for peer-to-peer media streaming. In *Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on*, pages 171–180, June.
- [128] Tzu-Ming Wang, Wei-Tsong Lee, Tin-Yu Wu, Hsin-Wen Wei, and Yu-San Lin. New p2p sharing incentive mechanism based on social network and game theory. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, pages 915–919, March.
- [129] YangBin Tang, Huaimin Wang, and Wen Dou. Trust based incentive in p2p network. In *E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on*, pages 302–305, Sept.
- [130] Kyuyong Shin, D.S. Reeves, and Injong Rhee. Treat-before-trick : Free-riding prevention for bittorrent-like peer-to-peer networks. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12, May.

- [131] Yuling Li, Yuhua Liu, Kaihua Xu, and Wei Chen. Analysis and balanced mechanism on free-rider in p2p network. In *Computer Modeling and Simulation, 2010. ICCMS '10. Second International Conference on*, volume 4, pages 462–466, Jan.
- [132] S. Sanghavi and B. Hajek. A new mechanism for the free-rider problem. *Automatic Control, IEEE Transactions on*, 53(5):1176–1183, June.
- [133] D. Jurca, J. Chakareski, J.-P. Wagner, and P. Frossard. Enabling adaptive video streaming in p2p systems [peer-to-peer multimedia streaming]. *Communications Magazine, IEEE*, 45(6):108–114, June 2007.
- [134] Truong Cong Thang, Hung T. Le, Hoc X. Nguyen, Anh T. Pham, Jung Won Kang, and Yong Man Ro. Adaptive video streaming over http with dynamic resource estimation. *Communications and Networks, Journal of*, 15(6):635–644, Dec 2013.
- [135] R. Pantos and W. May. HTTP Live Streaming, 10 2013. <http://tools.ietf.org/html/draft-pantos-http-live-streaming-12>. Último acesso: 28 de Fevereiro de 2014.
- [136] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *MultiMedia, IEEE*, 18(4):62–67, April 2011.
- [137] Alex Borges Vieira and Sergio Vale Aguiar Campos. *Transmissão de mídia contínua ao vivo em P2P: modelagem, caracterização e implementação de mecanismo de resiliência a ataques*. PhD thesis, Universidade Federal de Minas Gerais - UFMG, 3 2010. <http://dspace.lcc.ufmg.br/dspace/handle/1843/SLSS-85BNKG>.
- [138] Hao Yin, Chuang Lin, Qian Zhang, Zhijia Chen, and Dapeng Wu. Truststream: A secure and scalable architecture for large-scale internet media streaming. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(12):1692–1702, 12 2008.
- [139] M. Brinkmeier, G. Schafer, and T. Strufe. Optimally dos resistant p2p topologies for live multimedia streaming. *Parallel and Distributed Systems, IEEE Transactions on*, 20(6):831–844, June.
- [140] M. Abdalla, Y. Shavitt, and A. Wool. Key management for restricted multicast using broadcast encryption. *Networking, IEEE/ACM Transactions on*, 8(4):443–454, Aug.

- [141] C. K. Yeo, B. S. Lee, and M. H. Er. A framework for multicast video streaming over ip networks. *J. Netw. Comput. Appl.*, 26(3):273–289, August 2003.
- [142] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable application layer multicast. *SIGCOMM Comput. Commun. Rev.*, 32(4):205–217, August 2002.
- [143] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. Omni: An efficient overlay multicast infrastructure for real-time applications. *Comput. Netw.*, 50(6):826–841, April 2006.
- [144] Yi Cui, Baochun Li, and K. Nahrstedt. ostream: asynchronous streaming multicast in application-layer overlay networks. *Selected Areas in Communications, IEEE Journal on*, 22(1):91–106, Jan 2004.
- [145] John Jannotti, David K Gifford, Kirk L Johnson, M. Frans Kaashoek, and Jr. O’Toole. 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, San Diego, California, 2000. USENIX Association. ACM ID: 1251243.
- [146] Minseok Kwon and Sonia Fahmy. Path-aware overlay multicast. *Comput. Netw.*, 47(1):23–45, January 2005.
- [147] Feng Liu, J. Huang, Xicheng Lu, and Yuxing Peng. An efficient distributed algorithm for constructing delay- and degree-bounded application-level multicast tree. In *Parallel Architectures,Algorithms and Networks, 2005. ISPAN 2005. Proceedings. 8th International Symposium on*, pages 6 pp.–, Dec 2005.
- [148] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion Control Without Reliability. In *SIGCOMM ’06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–38, New York, NY, USA, 2006. ACM Press.
- [149] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol

- (DCCP), 3 2006. <http://www.ietf.org/rfc/rfc4340.txt>. Último acesso: 28 de Fevereiro de 2014.
- [150] L.M. de Sales, R. de Amorim Silva, H.O. Almeida, and A. Perkusich. Multi(uni)cast dccp for live content distribution with p2p support. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 3253–3258, April.
- [151] Leandro Melo de Sales, Hyggo Oliveira, and Angelo Perkusich. Multimedia content distribution of real time controlled and non-reliable datagrams between peers. In *Proceedings of IEEE Globecom 2011. 2nd IEEE Workshop on Multimedia Communications & Services*, volume 1, pages 131–146, 5 2011.
- [152] Leandro Melo de Sales. Avaliação Experimental do Protocolo DCCP para Transmissão de Conteúdos Multimídia em Redes Sem Fio 802.11g e na Internet. Master's thesis, Universidade Federal de Campina Grande, 4 2008.
- [153] Garrett Hardin. The Tragedy of the Commons. *Science*, 162(3859):1243 – 1248, 3 1968.
- [154] Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 17(1):1–14, June 1989.
- [155] Leandro Melo de Sales, Hyggo Oliveira, Angelo Perkusich, and Rafael A. Silva. Distribuição de conteúdo multimídia em tempo real com transporte de fluxos controlados e não confiáveis entre pares. In *Proceedings of Simpósio Brasileiro de Redes de Computadores 2011. VII Workshop de Redes Dinâmicas e Sistemas P2P*, volume 1, pages 131–146, 5 2011. [http://sbrc2011.facom.ufms.br/files/workshops/wp2p/ST04\\_1.pdf](http://sbrc2011.facom.ufms.br/files/workshops/wp2p/ST04_1.pdf).
- [156] J. R Iyengar, P. D Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent End-to-End paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, October 2006.
- [157] C. -M Huang and M. -S Lin. Multimedia streaming using partially reliable concurrent

- multipath transfer for multihomed networks. *IET Communications*, 5(5):587–597, March 2011.
- [158] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound tcp approach for high-speed and long distance networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1 –12, april 2006.
- [159] M. Goutelle, Y. Gu, and E. He. A Survey of Transport Protocols other than Standard TCP, 4 2004. <http://www.gridforum.org/documents/GFD.55.pdf>. Último acesso: 28 de Fevereiro de 2014.
- [160] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-Based Congestion Control for Unicast Applications. *ACM SIGCOMM Computer Communication Review*, 30(4):43–56, October 2000.
- [161] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42:64–74, July 2008.
- [162] Sally Floyd. Highspeed tcp for large congestion windows, 12 2003. <http://www.ietf.org/rfc/rfc3649.txt>. Último acesso: 28 de Fevereiro de 2014.
- [163] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch. Performance analysis of modern tcp variants: A comparison of cubic, compound and new reno. In *Communications (QBSC), 2010 25th Biennial Symposium on*, pages 80 –83, may 2010.
- [164] Cheng Peng Fu and Soung C. Liew. TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks. volume 21, pages 216–228, 2 2003.
- [165] L.S. Brakmo and L.L. Peterson. Tcp vegas: end to end congestion avoidance on a global internet. *Selected Areas in Communications, IEEE Journal on*, 13(8):1465–1480, Oct.
- [166] G. Carofiglio, L. Muscariello, D. Rossi, and C. Testa. A hands-on assessment of transport protocols with lower than best effort priority. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 8–15, Oct.

- [167] 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.
- [168] Jeng-Yuh Chang and Xiao Su. An evaluation of transport protocols in peer-to-peer media streaming. In *Networking, Architecture, and Storage, 2008. NAS '08. International Conference on*, pages 241 –247, june 2008.
- [169] E. Brosh, S. A Baset, V. Misra, D. Rubenstein, and H. Schulzrinne. The Delay-Friendliness of TCP for Real-Time traffic. *IEEE/ACM Transactions on Networking*, 18(5):1478–1491, October 2010.
- [170] V. Lucas, J. -J Pansiot, D. Grad, and B. Hilt. Fair multicast congestion control (M2C). In *IEEE INFOCOM Workshops 2009*, pages 1–6. IEEE, April 2009.
- [171] Ju-Won Park, Jong Won Kim, and R. P Karrer. TCP-ROME: a Transport-Layer approach to enhance quality of experience for online media streaming. In *16th International Workshop on Quality of Service, 2008. IWQoS 2008*, pages 249–258. IEEE, June 2008.
- [172] Hala ElAarag, Andrew Moedinger, and Chris Hogg. TCP friendly protocols for media streams over heterogeneous wired-wireless networks. *Comput. Commun.*, 31(10):2242–2256, June 2008. ACM ID: 1380037.
- [173] R. Stewart, J. Stone, D. Otis, K. Morneau, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol (SCTP), 9 2007. <http://www.ietf.org/rfc/rfc4960.txt>. Último acesso: 28 de Fevereiro de 2014.
- [174] Lin Ma and Wei Tsang Ooi. Congestion control in distributed media streaming. In *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 1397–1405. IEEE, May 2007.
- [175] Yongxiang Liu, K. N Srijith, L. Jacob, and A. L Ananda. TCP-CM: a transport protocol for TCP-friendly transmission of continuous media. In *Performance, Computing,*

- and Communications Conference, 2002. 21st IEEE International*, pages 83–91. IEEE, 2002.
- [176] Jack Brassil and Henning Schulzrinne. Structuring internet media streams with cueing protocols. *IEEE/ACM Trans. Netw.*, 10(4):466–476, August 2002. ACM ID: 581865.
- [177] A. Banerjea, D. Ferrari, B. A Mah, M. Moran, D. C Verma, and Hui Zhang. The tenet real-time protocol suite: design, implementation, and experiences. *IEEE/ACM Transactions on Networking*, 4(1):1–10, February 1996.
- [178] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock. Host-to-Host Congestion Control for TCP. *Communications Surveys Tutorials, IEEE*, 12(3):304–342, 8 2010.
- [179] He Xu, Suo ping Wang, Ru chuan Wang, and Ping Tan. A survey of peer-to-peer simulators and simulation technology. *Journal of Convergence Information Technology*, 6(5):260–272, 5 2011.
- [180] L.M. de Sales, H.O. Almeida, A. Perkusich, and K. Gorgonio. About encouraging residential users to share upload bandwidth with cdn/p2p live streaming systems. In *International Conference on Consumers Electronics (ICCE), 2013 IEEE*, pages 673–674, Jan.
- [181] Leandro Melo de Sales, Hyggo O. Almeida, Angelo Perkusich, and Marcello Sales Jr. On the Performance of TCP, UDP and DCCP over 802.11g Networks. In *In Proceedings of the SAC 2008 23rd ACM Symposium on Applied Computing Fortaleza, CE*, pages 2074–2080, 1 2008.
- [182] Larry L. Peterson and Bruce S. Davie. *Computer Networks, A System Approach*. Morgan Kaufmann, 5 ed. edition, 3 2011.
- [183] Luiz Fernando Gomes Soares. *Redes de Computadores: das LANs, MANs e WANs às Redes ATM*. Campus, 2 edition, 1995.
- [184] M.F. Bari, S. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu. A survey of naming and routing in information-centric networks. *Communications Magazine, IEEE*, 50(12):44–53, 2012.

- [185] Hongfeng Xu, Zhen Chen, Rui Chen, and Junwei Cao. Live Streaming with Content Centric Networking. In *Networking and Distributed Computing (ICND), 2012 Third International Conference on*, pages 1–5, Oct.
- [186] Hong Liu and P. Mouchtaris. Voice over IP Signaling: H.323 and Beyond. In *Communications Magazine, IEEE*, volume 28, pages 142 – 148, 10 2000.
- [187] J. Rosenberg, H. Schulzrinne, and G. Camarillo. SIP: Session Initiation Protocol, 6 2002. <http://www.ietf.org/rfc/rfc3261.txt>. Último acesso: 28 de Fevereiro de 2014.
- [188] Long Vu, Indranil Gupta, Klara Nahrstedt, and Jin Liang. Understanding overlay characteristics of a large-scale peer-to-peer IPTV system. *ACM Trans. Multimedia Comput. Commun. Appl.*, 6(4):31:1–31:24, November 2010. ACM ID: 1865115.
- [189] Darshan Purandare and Ratan Guha. An alliance based peering scheme for peer-to-peer live media streaming. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, P2P-TV ’07, pages 340–345, Kyoto, Japan, 2007. ACM. ACM ID: 1326328.
- [190] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and K. W Ross. A measurement study of a Large-Scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.
- [191] Thomas Bonald, Laurent Massoulié, Fabien Mathieu, Diego Perino, and Andrew Twigg. Epidemic live streaming: optimal performance trade-offs. In *Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS ’08, pages 325–336, Annapolis, MD, USA, 2008. ACM. ACM ID: 1375494.
- [192] Meng Zhang, Jian-Guang Luo, Li Zhao, and Shi-Qiang Yang. A peer-to-peer network for live media streaming using a push-pull approach. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA ’05, pages 287–290, Hilton, Singapore, 2005. ACM. ACM ID: 1101206.

- [193] Yi Cui, Yanchuan Cao, Liang Dai, and Yuan Xue. Optimizing P2P streaming throughput under peer churning. *Multimedia Systems*, 15(2):83–99, November 2008.
- [194] N. Magharei, R. Rejaie, and Y. Guo. Mesh or Multiple-Tree: a comparative study of live P2P streaming approaches. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 1424–1432, Anchorage, AK, USA, 2007.
- [195] Xiaojun Hei, Yong Liu, and Keith Ross. IPTV over P2P streaming networks: the mesh-pull approach. *IEEE Communications Magazine*, 46(2):86–92, February 2008.
- [196] Meng Zhang, Qian Zhang, Lifeng Sun, and Shiqiang Yang. Understanding the power of Pull-Based streaming protocol: Can we do better? *IEEE Journal on Selected Areas in Communications*, 25(9):1678–1694, December 2007.
- [197] R. Lo Cigno, A. Russo, and D. Carra. On some fundamental properties of P2P push-/pull protocols. In *Second International Conference on Communications and Electronics, 2008. ICCE 2008*, pages 67–73. IEEE, June 2008.
- [198] 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.
- [199] Li Zhao, Jian-Guang Luo, Meng Zhang, Wen-Jie Fu, Ji Luo, Yi-Fei Zhang, and Shi-Qiang Yang. Gridmedia: A practical Peer-to-Peer based live video streaming system. In *2005 IEEE 7th Workshop on Multimedia Signal Processing*, pages 1–4. IEEE, November 2005.
- [200] Thiago Bruno Melo de Sales. Especificação baseada no padrão upnp para autenticação e autorização de usuários em ambientes de computação pervasiva. Master's thesis, Universidade Federal de Campina Grande, 2010.

- [201] Yih-Chun Hu, Markus Jakobsson, and Adrian Perrig. Efficient constructions for one-way hash chains. In *IN APPLIED CRYPTOGRAPHY AND NETWORK SECURITY (ACNS)*, pages 423–441, 2005.
- [202] Leslie Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11):770–772, 1981.
- [203] Henk Meijer and Selim Akl. Digital signature schemes for computer communication networks. In *SIGCOMM '81: Proceedings of the seventh symposium on Data communications*, pages 37–41, New York, NY, USA, 1981. ACM.
- [204] Presidencia da República Federativa do Brasil. Medida provisória número 2.200-2, Agosto 2001.
- [205] V. Ciancaglini, G. Piro, R. Loti, L.A. Grieco, and L. Liquori. CCN-TV: A Data-centric Approach to Real-Time Video Services. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 982–989, March 2013.
- [206] J. Seedorf and E. Burger. Application-Layer Traffic Optimization (ALTO) Problem Statement, 10 2009. <http://www.ietf.org/rfc/rfc5693.txt>. Último acesso: 28 de Fevereiro de 2014.
- [207] R. Alimi, R. Penno, and Y. Yang. Application-Layer Traffic Optimization (ALTO) Protocol, 1 2014. <http://www.ietf.org/id/draft-ietf-alto-protocol-25.txt>. Último acesso: 28 de Fevereiro de 2014.
- [208] K.D. Teket and M. Sayit. P2p video streaming with alto protocol: A simulation study. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2013 IEEE International Symposium on*, pages 1–6, June 2013.
- [209] Majed Alhaisoni and Antonio Liotta. Characterization of signaling and traffic in Joost. *Peer-to-Peer Networking and Applications*, 2(1):75–83, 2009.

- [210] 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.
- [211] 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.
- [212] 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.
- [213] Van Jacobson. A New Way to Look at Networking, 8 2006. <http://named-data.net/a-new-way-to-look-at-networking/>. Último acesso: 28 de Fevereiro de 2014.
- [214] Adobe Inc. HTTP Dynamic Streaming, 2 2014. <http://www.adobe.com/br/products/hds-dynamic-streaming.html>. Último acesso: 28 de Fevereiro de 2014.
- [215] 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.
- [216] A. Gouta, C. Hong, D. Hong, A.-M. Kermarrec, and Y. Lelouedec. Large scale analysis of http adaptive streaming in mobile networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–10, June 2013.
- [217] 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: 28 de Fevereiro de 2014.
- [218] 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.

- [219] 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.
- [220] 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.
- [221] 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.
- [222] A. Arcieri. Peer distributed transfer protocol, 7 2004. <https://www.defcon.org/images/defcon-12/dc-12-presentations/Arcieri/draft-arcieri-peer-distributed-transfer-protocol.html>. Último acesso: 28 de Fevereiro de 2014.
- [223] C. Kulatunga, G. Kandavanam, A.I. Rana, S. Balasubramaniam, and D. Botvich. Hy-sac: A hybrid delivery system with adaptive content management for iptv networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5, june 2011.
- [224] 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: 28 de Fevereiro de 2014.
- [225] 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.
- [226] Qingchao Cai, Xiaolu Zhang, and Xuejie Zhang. Streaming live media over bittor-

- rent. In *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, volume 3, pages 44–49, Jan 2009.
- [227] 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.
- [228] 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.
- [229] 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.
- [230] 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.
- [231] Jie Xiong and R.R. Choudhury. Peercast: Improving link layer multicast through cooperative relaying. In *INFOCOM, 2011 Proceedings IEEE*, pages 2939–2947, April 2011.
- [232] 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.
- [233] Mohamed Hefeeda, Ahsan Habib, Boyan Botev, Dongyan Xu, and Bharat Bhargava. Promise: Peer-to-peer media streaming using collectcast. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, MULTIMEDIA ’03, pages 45–54, New York, NY, USA, 2003. ACM.
- [234] 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.

- [235] Roberto Roverso, Sameh El-Ansary, and Seif Haridi. Smoothcache: Http-live streaming goes peer-to-peer. 7290:29–43, 2012.
- [236] 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.
- [237] K Moore. Things that NATs break. Online, 3 2004. <http://web.mit.edu/6.033/2002/wwwdocs/papers/what-nats-break.html>. Último acesso: 28 de Fevereiro de 2014.
- [238] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), 2003. <http://www.ietf.org/rfc/rfc3489.txt>. Último acesso, 28 de Fevereiro de 2014.
- [239] Paul Vixie. What DNS is Not. *Commun. ACM*, 52(12):43–47, December 2009.
- [240] C. Tsilopoulos, G. Xylomenos, and G.C. Polyzos. Are information-centric networks video-ready? In *Packet Video Workshop (PV), 2013 20th International*, pages 1–8, Dec 2013.
- [241] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. Networking named content. *Commun. ACM*, 55(1):117–124, 1 2012.
- [242] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V.A. Siris, and G.C. Polyzos. Caching and mobility support in a publish-subscribe internet architecture. *Communications Magazine, IEEE*, 50(7):52–58, 7 2012.
- [243] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner. An experimental analysis of dynamic adaptive streaming over http in content centric networks. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6, July 2013.

- [244] Zhenkai Zhu, Sen Wang, Xu Yang, Van Jacobson, and Lixia Zhang. ACT: Audio Conference Tool over Named Data Networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 68–73, New York, NY, USA, 2011. ACM.
- [245] Christos Tsilopoulos and George Xylomenos. Supporting diverse traffic types in information centric networks. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 13–18, New York, NY, USA, 2011. ACM.
- [246] Van Jacobson, Diana K. Smetters, Nicholas H. Briggs, Michael F. Plass, Paul Stewart, James D. Thornton, and Rebecca L. Braynard. VoCCN: Voice-over Content-centric Networks. In *Proceedings of the 2009 Workshop on Re-architecting the Internet*, ReArch '09, pages 1–6, New York, NY, USA, 2009. ACM.
- [247] Andrea Detti, Bruno Ricci, and Nicola Belfari-Melazzi. Peer-to-peer live adaptive video streaming for information centric cellular networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pages 3583–3588, Sept 2013.
- [248] Yaning Liu, J. Geurts, J.-C. Point, S. Lederer, B. Rainer, C. Muller, C. Timmerer, and H. Hellwagner. Dynamic adaptive streaming over ccn: A caching and overhead analysis. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3629–3633, June 2013.
- [249] S. Bradner. Key words for use in rfc's to indicate requirement levels, 3 1997. <http://www.ietf.org/rfc/rfc2119.txt>. Último acesso: 28 de Fevereiro de 2014.
- [250] Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory (Discrete Mathematics and Its Applications)*. CRC Press, 2 2003. ISBN: 978-1584880905.
- [251] R Séroul. *Programming for Mathematicians*. Springer-Verlag, 2 2000. ISBN: 978-3540664222.

- [252] R. Courant and H. Robbins. *The Algebra of Sets. What Is Mathematics?: An Elementary Approach to Ideas and Methods.* Oxford University Press, 7 1996. ISBN: 978-0195105193.
- [253] K. J. Devlin. *Fundamentals of Contemporary Set Theory.* Springer, 9 1979. ISBN: 978-0387904412.
- [254] R. Braden. Requirements for Internet Hosts - Communication Layers, 10 1989. Último acesso: 28 de Fevereiro de 2014.
- [255] L. Eggert and F. Gont. TCP User Timeout Option, 3 2009. <http://www.ietf.org/rfc/rfc5482.txt>. Último acesso: 28 de Fevereiro de 2014.
- [256] S. Tanwir and H. Perros. A survey of vbr video traffic models. *Communications Surveys Tutorials, IEEE*, 15(4):1778–1802, Fourth 2013.
- [257] P. Leach, M. Mealling, and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace, 7 2005. <http://www.ietf.org/rfc/rfc4122.txt>. Último acesso: 28 de Fevereiro de 2014.
- [258] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic updates in the domain name system (dns update), 4 1997. <http://www.ietf.org/rfc/rfc2136.txt>. Último acesso: 28 de Fevereiro de 2014.
- [259] H. Alvestrand. Tags for the identification of languages, 2 1995. <http://www.ietf.org/rfc/rfc1766.txt>. Último acesso: 28 de Fevereiro de 2014.
- [260] V. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's Retransmission Timer, 6 2011. <http://www.ietf.org/rfc/rfc6298.txt>. Último acesso: 28 de Fevereiro de 2014.
- [261] W. Wolff. *Stochastic Modeling and the Theory of Queues.* PrenticeHall, 1989.
- [262] M. Handley, S. Floyd, J. Padhye, and J. Widmer. Tcp friendly rate control (tfrc): Protocol specification, 1 2003. <http://www.ietf.org/rfc/rfc3448.txt>. Último acesso: 28 de Fevereiro de 2014.

- [263] Eddie Kohler, Mark Handley, and Floyd. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. In *IETF Online RFC*, 3 2006. <http://www.ietf.org/rfc/rfc4341.txt>. Último acesso: 28 de Fevereiro de 2014.
- [264] Eddie Kohler and Mark Handley. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC), 3 2006. <http://www.ietf.org/rfc/rfc4342.txt>. Último acesso: 12/04/2011.
- [265] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, New York, NY, USA, 1998. ACM Press.
- [266] D. Meyer. Administratively scoped ip multicast, 7 1998. <http://www.ietf.org/rfc/rfc2365.txt>. Último acesso: 28 de Fevereiro de 2014.