

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, 04/05/2014

## Resumo

O aumento das conexões de banda larga à Internet resulta em uma maior exigência de qualidade de serviço dos sistemas de distribuição de mídias ao vivo, elevando-se os custos com largura de banda dos distribuidores de conteúdo. No estado da prática, constata-se que nas principais propostas, como o Denacast/CoolStreaming, prevalece o uso de uma arquitetura híbrida P2P/CDN, objetivando-se disseminar eficientemente o conteúdo multimídia (escala/qualidade). Já no estado da arte, observam-se avanços em novas arquiteturas, como a Rede Centrada no Conteúdo (CCN), que segue um modelo de serviço do tipo *pull* e *cache* dos dados mais acessados nos roteadores. O problema das abordagens práticas é o consequente aumento no consumo de recursos de rede, resultante das estratégias adotadas nos protocolos de aplicação, propostos para suprir limitações das camadas inferiores e dispostos sem qualquer interoperabilidade. Como consequência, aumenta-se o tráfego de datagramas duplicados, limitando-se o desempenho dos sistemas de distribuição de mídias ao vivo – a Internet não foi projetada para este fim. Já em CCN, considera-se um modelo de serviço que não é apropriado para transporte de dados transientes, delegando-se a execução de serviços importantes aos nós clientes, quando deveriam ser prestados pela rede. Com isto, aumenta-se a quantidade de pacotes de controle dependentes da capacidade de transmissão *upstream*, em geral menor se comparada à capacidade de *downstream*, resultando na baixa qualidade dos serviços multimídia. Por estes e outros motivos detalhados neste trabalho, propõe-se o *Global Media Transmission Protocol* (GMTP), apresentado em duas perspectivas: projeto e desempenho. No projeto, propõe-se um protocolo multi-camada para disseminar eficientemente mídias ao vivo através de uma rede de favores constituída entre nós roteadores. As parcerias entre os roteadores são determinadas pelos nós servidores, com base na capacidade dos canais já em uso para disseminar o conteúdo, obtidas e atualizadas periodicamente por meio de um algoritmo de controle de congestionamento assistido pela rede. Já na perspectiva de desempenho, estudou-se o GMTP em confrontos com as propostas supracitadas, avaliando-se as principais métricas de qualidade de serviço. Com base nos resultados obtidos por meio da simulação de um cenário real, demonstra-se que o GMTP obteve um desempenho 61,44 % melhor que o Denacast/CoolStreaming e 36,18 % melhor que o CCN-TV.

## **Abstract**

TBW

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Delimitação . . . . .	2
1.2	Descrição do Problema . . . . .	7
1.3	Objetivos . . . . .	13
1.4	Relevância do Tema e da Tese . . . . .	14
1.5	Estrutura do Documento . . . . .	16
<b>2</b>	<b>Fundamentação</b>	<b>17</b>
2.1	Estruturas para Distribuição de Mídia ao Vivo em P2P . . . . .	18
2.1.1	Estrutura baseada em árvore . . . . .	19
2.1.2	Estrutura baseada em malha . . . . .	21
2.1.3	Estrutura híbrida . . . . .	25
2.2	Sistemas de Transmissão ao Vivo em P2P . . . . .	28
2.2.1	Geração do conteúdo da transmissão . . . . .	31
2.2.2	Armazenamento e consumo de dados . . . . .	32
2.2.3	Estratégia de seleção de <i>chunks</i> . . . . .	33
2.2.4	Realização de parcerias e obtenção de <i>chunks</i> . . . . .	35
2.3	Criptografia de Hash e Assinatura Digital . . . . .	36
2.3.1	Criptografia de hash . . . . .	37
2.3.2	Criptografia assimétrica . . . . .	37
2.3.3	Assinatura e certificação digital . . . . .	37
2.4	Sumário do Capítulo . . . . .	40

<b>3 Trabalhos Relacionados</b>	<b>41</b>
3.1 Visão Geral das Propostas para Distribuição de Mídias ao Vivo . . . . .	42
3.2 Aplicações e Protocolos para Distribuição de Mídias ao Vivo . . . . .	45
3.2.1 Protocolos de adaptação de fluxo baseado em HTTP . . . . .	45
3.2.2 PDTP – <i>Peer Distributed Transfer Protocol</i> . . . . .	50
3.2.3 CPM – <i>Cooperative Peer Assists and Multicast</i> . . . . .	51
3.2.4 HySAC – <i>Hybrid Delivery System with Adaptive Content Management for IPTV Networks</i> . . . . .	53
3.2.5 PPSP/Swift – <i>P2P Streaming Protocol / The Generic Multiparty Transport Protocol</i> . . . . .	55
3.2.6 DONet/CoolStreaming . . . . .	58
3.2.7 CoolStreaming/Denacast . . . . .	62
3.2.8 Outras propostas . . . . .	63
3.3 Redes Centradas na Informação . . . . .	64
3.4 Sumário do Capítulo . . . . .	71
<b>4 Global Media Transmission Protocol (GMTP)</b>	<b>73</b>
4.1 Visão Geral . . . . .	76
4.1.1 Resumo das principais características . . . . .	83
4.1.2 Cabeçalho geral e tipos de pacotes . . . . .	85
4.2 Definições, Relações e Restrições . . . . .	88
4.3 Constituição da Rede de Favores $\eta$ . . . . .	92
4.3.1 Registro de participação de $r_d$ em $\eta$ . . . . .	92
4.3.2 Tabela de recepção de fluxos de dados . . . . .	98
4.3.3 Formação de parcerias . . . . .	100
4.4 Transmissão de $p_x \in P$ através de $\eta$ . . . . .	107
4.4.1 Indexação de conteúdo . . . . .	107
4.4.2 Estabelecimento de conexão entre $c_f$ e $s_a$ para obter $P$ . . . . .	112
4.4.3 Fase 1: primeira requisição a um fluxo de dados $P$ . . . . .	113
4.4.4 Fase 2: próximas requisições para obter $P$ . . . . .	117
4.4.5 Fase 3: busca por mais parceiros $r_q$ para obter $P$ . . . . .	117

4.4.6	Envio e recebimento de $p_x \in P$ em $\eta$	122
4.5	Controle de Congestionamento em $\eta$	125
4.5.1	Controle de congestionamento <i>unicast</i>	125
4.5.2	Controle de congestionamento <i>multicast</i>	139
4.6	Autenticidade de $P$	142
4.6.1	Transmissão e assinatura de autenticidade de $p_x \in P$	144
4.6.2	Verificação de autenticidade de $p_x \in P$	144
4.6.3	Habilitar / desabilitar a validação de pacotes $p_x \in P$	146
4.6.4	Obtenção da chave pública $K_{s_a}^+$ de $s_a$	147
4.7	Outras Considerações	147
4.7.1	Canais de comunicação	148
4.7.2	Procedimentos para desconexão de nós $c_f$ , $l_w$ e $r_d$	149
4.7.3	Eleição de nós $l_w$	150
4.8	Sumário do Capítulo	151
<b>5</b>	<b>Análise de Projeto e do Desempenho do GMTP</b>	<b>155</b>
5.1	Análise do Projeto	156
5.1.1	Projeto e benefícios do GMTP para os sistemas e para as redes	156
5.1.2	Comparativo: GMTP, Denacast/CoolStreaming e CCN	160
5.2	Avaliação de Desempenho	167
5.2.1	Objetivo e hipótese	168
5.2.2	Topologia de rede	168
5.2.3	Definição das variáveis	170
5.2.4	População e amostras	173
5.2.5	Tratamentos	173
5.2.6	Instrumentação	177
5.2.7	Formato da mídia	177
5.3	Resultados e Discussões	178
5.3.1	Atraso de inicialização	179
5.3.2	Índice de continuidade	181
5.3.3	Distorção	184

5.3.4	Sobrecarga de controle . . . . .	188
5.4	Sumário do Capítulo e dos Resultados . . . . .	191
<b>6</b>	<b>Considerações Finais</b>	<b>193</b>
6.1	Conclusões . . . . .	194
6.2	Limitações e Trabalhos Futuros . . . . .	197
6.3	Publicações . . . . .	199
6.4	Resumo das Contribuições . . . . .	200
<b>A</b>	<b>Detalhes dos Experimentos</b>	<b>234</b>
A.1	Largura de Banda e Atraso de Propagação Utilizados na Rede Simulada . .	234
A.2	Distribuição da quantidade de nós clientes nos primeiros 200 s de simulação	236
A.3	Distribuição da quantidade de nós clientes após os 400 s de simulação . .	236
A.4	Quantidade de Ensaios . . . . .	236
A.5	Compilação dos Resultados . . . . .	237
<b>B</b>	<b>Detalhamento dos Tipos de Pacotes do GMTP</b>	<b>242</b>

# **Lista de Símbolos**

- 3WHS – *Three Way Hand Shake*
- ALTO – *Application-Layer Traffic Optimization*
- ALM – *Application Layer Multicast*
- BSD – *Berkley Software Distribution*
- CCID – *Congestion Control IDentifier*
- CCN – *Content Centric Networks*
- CPM – *Cooperative Peer Assists and Multicast*
- DASH – *Dynamic Adaptive Streaming over HTTP*
- DCCP – *Datagram Congestion Control Protocol*
- ECN – *Explicit Congestion Notification*
- GMTP – *Global Media Transport Protocol*
- HLS – *HTTP Live Streaming*
- HDS – *HTTP Dynamic Streaming*
- HySAC – *Hybrid Delivery System with Adaptive Content Management for IPTV Networks*
- IANA – *Internet Assigned Numbers Authority*
- ICN – *Information Centric Networks*
- IETF – *Internet Engineering Task Force*
- IGMP – *Internet Group Management Protocol*
- ITU-T – *International Telecommunication Union – Telecommunication Section*
- NDN – *Named-Data Networks*
- PDTP – *Peer Distributed Transfer Protocol*
- POSIX – *Portable Operating System Interface*
- PPETP – *Peer-to-Peer Epi-Transport Protocol*
- PPSP – *P2P Streaming Protocol*

RCP – *Rate Control Protocol*  
RTO – *Retransmission Timeout*  
RTP – *Real Time Protocol*  
RTSP – *Real Time Streaming Protocol*  
RTT – *Round Trip Time*  
SCTP – *Stream Control Transmission Protocol*  
SIP – *Session Initiation Protocol*  
Swift – *The Generic Multiparty Transport Protocol*  
TCP – *Transport Control Protocol*  
TFRC – *TCP Friendly Rate Control*  
TTL – *Time-To-Live*  
UDP – *User Datagram Protocol*  
URI – *Uniform Resource Identifier*  
VCP – *Variable-Structure Congestion Control Protocol*  
WWW – *World Wide Web*  
XCP – *eXplicit Control Protocol*

# Listas de Figuras

1.1	Esquema de uma arquitetura híbrida P2P/CDN para distribuição de conteúdo multimídia. . . . .	3
1.2	Confronto entre os sistemas Denacast/CoolStreaming e CCN-TV para as métricas distorção e sobrecarga de controle. Transmitiu-se uma mídia do tipo MPEG4-II, com taxa variável de bits de $1\ Mbps$ , através de uma rede simulada com 324 roteadores e diferentes larguras de banda. Mais detalhes sobre este experimento estão disponíveis no Capítulo 5 e no Apêndice A. . . . .	10
2.1	Árvore de <i>multicast</i> em nível da camada de aplicação. . . . .	20
2.2	Manutenção da árvore de <i>multicast</i> em nível da camada de aplicação. . . .	21
2.3	Sistema baseado em múltiplas árvores com dois subfluxos. . . . .	22
2.4	Atividade inicial de um novato - rede P2P baseada em malha. . . . .	23
2.5	Troca de dados na aplicação baseada em malha. . . . .	26
2.6	Modelo de sistema utilizado. . . . .	28
2.7	Mecanismo de consumo da mídia ao vivo. . . . .	34
2.8	Modelo de criptografia assimétrica, chave pública representada por $K^+$ e chave privada representada por $K^-$ . . . . .	38
2.9	Geração de Assinaturas Digitais. . . . .	39
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 têm 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. . . . .	42

---

3.2	Arquitetura do HLS para transmissão de conteúdos multimídia baseado no protocolo HTTP. Figura extraída e adaptada de [1]. . . . .	46
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 [2]. . . . .	47
3.4	Organização dos nós PDTP. Adaptada de [3]. . . . .	51
3.5	Diagrama de sequência do CPM ( <i>Cooperative Peer Assists and Multicast</i> ). Figura extraída de [4]. . . . .	52
3.6	Arquitetura do HySAC ( <i>Hybrid Delivery System with Adaptive Content Management for IPTV Networks</i> ). Figura extraída de [5]. . . . .	54
3.7	Arquitetura e funcionamento do protocolo PPSP/Swift. . . . .	56
3.8	Arquitetura genérica de blocos funcionais do CoolStreaming 1.0 e do 2.0. .	59
3.9	Arquiteturas CDN/P2P utilizadas no Denacast. Figuras extraídas de [6]. .	62
3.10	Evolução das abstrações das redes de comunicação. Figuras de Van Jacobson.	65
3.11	Um exemplo de interação da <i>Content-Centric Network</i> (CCN). . . . .	66
4.1	Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP.	74
4.2	Arquitetura do Protocolo GMTP. . . . .	76
4.3	Tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP. Nessa tela, permite-se que o administrador do roteador configure parâmetros do módulo GMTP-Inter. . . . .	78
4.4	Tipos de nós e modos de conexões do GMTP. . . . .	79
4.5	Rede de sobreposição construída pelo GMTP . . . . .	80
4.6	Fluxograma geral do GMTP, desde a solicitação do cliente por uma mídia ao vivo até receber uma resposta, executando-se ações importantes como registro de participação, controle de congestionamento, eleição de relatores e verificação de autenticidade dos dados. . . . .	81
4.7	Porção fixa do cabeçalho de pacotes GMTP. . . . .	85
4.8	Exemplo de uma tabela de recepção de fluxo mantida por um nó $r_d$ . . . . .	99
4.9	Cenário e passos para seleção de nós (exemplo 1). . . . .	101
4.10	Cenário para seleção de nós por interseção de caminhos $W_v$ . . . . .	102

4.11	Exemplo de rede para o estabelecimento de conexão do GMTP . . . . .	112
4.12	Passos do processo de estabelecimento de conexão do GMTP (Fase 1) . . . . .	113
4.13	Tabela de recepção de fluxos de dados após a Fase 1 . . . . .	115
4.14	Passos do processo de estabelecimento de conexão do GMTP (Fase 2) . . . . .	117
4.15	Fase 3 de conexão do GMTP (Passo 1) . . . . .	118
4.16	Fase 3 de conexão do GMTP (Passo 2) . . . . .	120
4.17	Tabela de recepção de fluxos de dados após a Fase 3 . . . . .	121
4.18	Exemplo da estrutura do <i>buffer</i> de envio e recepção de um nó GMTP com dois ponteiros, um para escrever e outro para ler pacotes $p_x$ . . . . .	123
4.19	Exemplo do mapa de <i>buffer</i> de um nó GMTP com tamanho de 17 $p_x$ . . . . .	123
4.20	Organização do algoritmo de controle de congestionamento no GMTP . . . . .	126
4.21	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 $H$ . . . . .	129
4.22	O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão, porém isto pode limitar alguns clientes 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 todos os nós no caminho $W_v$ recebam o fluxo de dados $P$ a $1\text{ Mb/s}$ . . . . .	132
4.23	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. . . . .	133

---

4.24 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 <i>Poisson</i> e tamanhos do fluxo em distribuição <i>Pareto</i> com média de 30 pacotes (1000 bytes/pacote), <i>shape</i> igual a 1.4, capacidade do <i>enlace</i> 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 [58]. . . . .	135
4.25 Evolução dos números de sequência de fluxos de dados, quando utilizando TCP (Reno), XCP e RCP. O tamanho do fluxo no primeiro gráfico foi de 230 pacotes, e no segundo gráfico foi de 3600 pacotes. Extraído de [58]. . . . .	137
4.26 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 <i>Poisson</i> e tamanhos do fluxo em distribuição Pareto com média de 25 pacotes (1000 bytes/pacote), <i>shape</i> 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 [58]. . . . .	138
5.1 Versão adaptada do <i>backbone</i> da rede GÉANT, com larguras de banda modificadas para ser utilizada no experimento. . . . .	169
5.2 Exemplo de cálculo das variáveis dependentes ST, IC. Nesse caso, o valores ST = 4 s e IC = 70 %. . . . .	172
5.3 Distribuição da quantidade de nós clientes nos primeiros 200 s de simulação. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema. . . . .	175
5.4 Distribuição da quantidade de nós clientes nos primeiros 200 s de simulação. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema. . . . .	176

5.5	Distribuição da quantidade de nós clientes após os 200 s de simulação, com variação a cada 5 s, para o ensaio 1 de todos os tratamentos. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema.	176
5.6	OMNet++ com a execução de um dos ensaios com 1 servidor e 1.500 clientes.	177
5.7	Taxa de bits variáveis dos primeiros 200 s da mídia utilizada no experimento ( <i>Star Wars IV</i> ). . . . .	178
5.8	Resultado dos tratamentos (1 – 18) para a métrica <i>Atraso de Inicialização</i> . .	179
5.9	Resultado dos tratamentos (1 – 18) para a métrica <i>Índice de Continuidade</i> . .	182
5.10	Resultado dos tratamentos (1 – 18) para a métrica <i>Distorção</i> . . . . .	185
5.11	Resultado dos tratamentos (1 – 18) para a métrica <i>Número de Pacotes de Controle</i> . . . . .	188

# **Lista de Tabelas**

2.1	Notações para descrever um sistema P2P de transmissão ao vivo. . . . .	31
2.2	Notações utilizadas para descrever a geração de mídia ao vivo em sistemas P2P. . . . .	32
5.1	Variáveis independentes utilizados no experimento. . . . .	170
5.2	Fatores consideradas no experimento. . . . .	171
5.3	Variáveis dependentes (respostas) consideradas no experimento. . . . .	173
5.4	Tratamentos executados no experimento. . . . .	174
5.5	Propriedades da mídia transmitida. . . . .	177
5.6	Sumário dos ganhos do GMTP sobre o Denacast e o CCN-TV. . . . .	192
A.1	Largura de banda e atraso de propagação utilizados na rede simulada. . . .	234
A.2	Sumário dos valores obtidos para as variáveis dependentes em cada trata- mento do confronto GMTP vs. Denacast/CoolStreaming. . . . .	237
A.3	Sumário dos valores obtidos para as variáveis dependentes em cada trata- mento do confronto GMTP vs. CCN-TV. . . . .	239

# List of Algorithms

1	registerRelay( $s_a$ : PeerServer, $p_x = GMTP\text{-}Request$ ) . . . . .	96
2	onReceiveGMTPRegisterReply( $p_x = GMTP\text{-}Register\text{-}Reply$ ) . . . . .	97
3	handleRegisterParticipation( $r_d$ : PeerRelay, $p_x = GMTP\text{-}Register$ ) . . . . .	105
4	Exemplo de requisição e resposta da lista de nomes dos fluxos de dados $P$ de um distribuidor de conteúdos multimídia. . . . .	109
5	Exemplo de uma mensagem SDP no pacote $GMTP\text{-}MediaDesc$ . . . . .	111
6	respondToClients( $p_x: GMTP\text{-}RequestNotify$ ) . . . . .	116
7	digitalSignPacket( $p_x: GMTP\text{-}Data$ ) . . . . .	144
8	verifyPacketAuthenticity( $P'$ : <b>array of</b> $GMTP\text{-}Data$ ) . . . . .	146

# Capítulo 1

## Introdução

A Internet é um sucesso tecnológico e sua infraestrutura global conecta bilhões de hospedeiros, transmitindo-se uma volumosa quantidade de dados digitais diariamente. 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 [7].

O aumento das conexões de banda larga à Internet resulta em uma maior exigência de qualidade de serviço dos sistemas de distribuição de mídias ao vivo, elevando-se 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 e conseguir transmitir o equivalente a 60 mil anos de vídeos se fossem reproduzidos sequencialmente [8]. Estima-se que este ano (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 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 [7].

Com a evolução da WWW (*World Wide Web*) para a Web 2.0 [9, 10], 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<sup>TM</sup>, PPLive<sup>TM</sup> e o UStream.tv<sup>TM</sup>. 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 conec-

tados à Internet [11]. 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 até março de 2014.

Ao observar o panorama atual dos serviços de distribuição de mídias ao vivo [11–17], o que preocupa é o crescimento acentuado do consumo de recursos computacionais e de rede, resultante das estratégias e protocolos adotados para este fim, cada um com suas próprias soluções [18–23]. Por isto, há uma grande motivação para estudar e propor novas soluções para utilizar eficientemente as redes de computadores, a fim de distribuir mídias ao vivo através da Internet em escala e com qualidade [24, 25].

## 1.1 Delimitação

Apesar de existirem outras estratégias para distribuição de mídias ao vivo, delimita-se este trabalho nos protocolos dos sistemas para distribuição de mídias ao vivo baseados em uma arquitetura de rede híbrida P2P/CDN, ou seja, par-a-par (*Peer to Peer - P2P*) [26–29] e cliente-servidor com suporte de uma rede de distribuição de conteúdos (*Content Delivery Network - CDNs*) [30–32] (Figura 1.1). Isto porque há evidências contundentes [6,33–41] de que se trata da principal escolha dos sistemas mais robustos, conseguindo-se escalabilidade do número de usuários e redução de custos com infraestrutura de rede, por meio das redes P2P (escala); ao passo que facilita-se o gerenciamento e obtém-se maior estabilidade de disponibilização dos serviços, por meio das CDNs (qualidade).

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

---

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

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

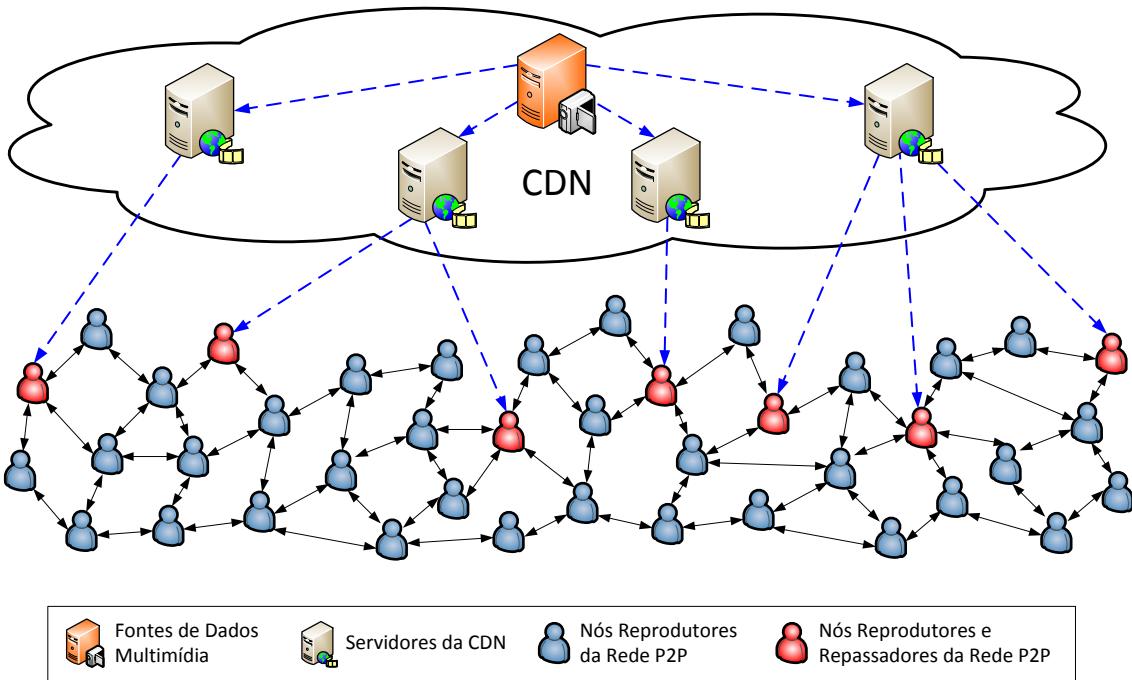


Figura 1.1: Estrutura de rede de uma arquitetura híbrida P2P/CDN para distribuição de conteúdo multimídia.

protocolos de rede, utilizam-se conexões TCP (*Transmission Control Protocol*) [42] para selecionar 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*) [43] para transmitir datagramas contendo as partes da mídia a uma aplicação capaz de reproduzi-las ao usuário final.

Um aspecto importante para esses sistemas é definir de que forma os nós devem escalar 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 mantêm os serviços de distribuição mais estáveis, tornando o sistema menos vulnerável ao dinamismo de participação dos nós de uma redes P2P, também chamado de *churn* [44, 45].

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, estabelecendo 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 às seguintes peculiaridades:

1. o usuário de um potencial nó contribuidor tem que expressar interesse em um determinado 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 satisfação 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 reproduzir do que outros nós. Da mesma forma, não faz sentido manter por muito tempo dados já reproduzidos no *buffer* de reprodução. Os datagramas expiram rapidamente<sup>3</sup> e, nestes casos, as aplicações os descartam após reproduzi-los ao usuário final. Assim, o tamanho necessário para o *buffer* de reprodução deve ser o suficiente para armazenar alguns segundos<sup>4</sup> da mídia e repassá-la aos nós parceiros, requerendo baixa<sup>5</sup> capacidade de armazenamento dos processadores de rede (roteadores, pontos de acesso, estações de trabalho, celulares, *tablets* etc.);
4. os caminhos 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 replicá-lo no ponto de intersecção. Além disso, conhecidos os caminhos e suas capacidades de transmissão corrente (produto largura de banda – atraso),

---

<sup>3</sup>Em aplicações de transmissão multimídia ao vivo, estima-se *jitter* no máximo de 180 ms

<sup>4</sup>Em geral, adota-se entre 5-8 segundos

<sup>5</sup>Em geral, com MPEG-4, utilizam-se alguns kilobits até algumas dezenas de megabits por segundo.

pode-se determinar melhores parcerias, fomentando a cooperação e agrupamento de nós com interesses comuns.

Estas peculiaridades viabilizam uma abordagem muito importante adotada neste trabalho: a participação mais efetiva dos roteadores no processo de distribuição de mídias ao vivo. Isto ocorre ao permitir que os roteadores: (i) constituam uma rede de favores com seleção de nós parceiros (outros roteadores) baseados na capacidade de transmissão dos canais entre os clientes e os servidores; (ii) sirvam como pontes de acesso aos servidores da CDN; e (iii) ajudem na execução do serviço de controle de congestionamento. A principal justificativa para constituir redes de favores entre roteadores é baseada na conjectura de que os roteadores são elementos de rede estáveis com relação a sua disponibilidade se comparados aos sistemas finais, atenuando-se o *churn* das redes em comparação àquelas formadas unicamente por sistemas finais. Como consequência, deve-se permitir que as aplicações forneçam serviços mais estáveis aos usuários. 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  $r$ .

De modo a generalizar essa discussão, 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 o NAT dinâmico (*Dynamic Network Address Translation*) através de UPnP (*Universal Plug and Play*) [46], DNS dinâmico [47], controle de congestionamento, dentre outros. Essa tendência é também sustentada pelas Redes Centradas no Conteúdo (*Content-Centric Networks* – CCN) [48–51].

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 [52–54], 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 [55, 56]. Assim, reduz-se drasticamente as perdas de dados, porque raramente ocorre o transbordo das filas de roteamento; 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 [42, 57].

Neste 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 [52]. Sendo assim, convém também utilizá-los para distribuir mídias ao vivo. Não somente com esta finalidade, mas também permitindo-se a formação de parcerias com base em informações explícitas da capacidade de transmissão de um ou mais enlaces, conhecidos à medida que se distribui os pacotes de dados. No escopo deste trabalho tal abordagem é adotada, destacando-se o *Rate Control Protocol* (RCP) [54, 58, 59], adaptado no GMTP e escolhido em detrimento a outros protocolos da mesma classe, como o *Explicit Control Protocol* (XCP) [53, 60, 61], o *Variable-Structure Congestion Control Protocol* (VCP) [62–66] e o *Congestion Exposure* (ConEx) [67–69]. Em um momento oportuno, discutem-se os principais motivos para tal escolha.

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 [70, 71]. 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 utilizar *multicast* quando controlado por um protocolo capaz de habilitá-lo/desabilitá-lo automaticamente de acordo com a demanda das aplicações, compartilhando-se os fluxos de dados e mantendo-se o princípio da independência dos serviços de rede, ou seja, sem que a aplicação precise se preocupar como isto é feito.

É no contexto de utilização de conceitos e desenvolvimento de protocolos de redes e sistemas distribuídos para transmissão de mídias ao vivo que se insere esse trabalho, motivado pelo grande interesse de pesquisa, indústria e mercado em evoluir o estado da arte. Para estudar e viabilizar uma solução eficiente com base no escopo apresentado, deve-se abordar problemas relacionados à comunicação multi-ponto, escalabilidade quanto ao número de nós

e estratégias otimizadas para utilizar os recursos compartilhados da rede, tendo em vista as métricas que determinam a satisfação de um usuário ao assistir a um evento ao vivo através da Internet.

## 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 [72–86]. 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 [87–89]. Isto não seria um problema para a infraestrutura 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 [90–98]. 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 codificados/decodificados, de acordo com cada emissora de TV, dentre outros. Isto é possível porque todas as TVs simplesmente seguem padrões, independente da sua marca e modelo, caso contrário, somente conseguiriam interpretar o sinal transmitido por um conjunto restrito de emissoras de TV.

Considerando a analogia anterior, os servidores dos atuais sistemas de distribuição multimídia na Internet são as emissoras de TV, ao passo que as aplicações clientes são os aparelhos de TV. A diferença para a analogia supracitada é que na Internet os sistemas não seguem um padrão para transmitir e receber os dados multimídia 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, incluindo alguns proprietários.

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 [81, 84, 99–104]. A duplicação pode ser percebida até em cenários mais simples, quando dois ou mais nós conectados à Internet através do mesmo provedor estão utilizando diferentes sistemas para assistir ao mesmo evento ao vivo. Idealmente, os nós deveriam compartilhar o mesmo fluxo de dados e cada um apresentar o conteúdo da forma definida pela aplicação, desacoplando a forma de transportar os dados da forma como estes são apresentados pela aplicação aos seus respectivos usuários, como no caso do serviço Web (princípio da independência dos serviços).

O grande sucesso do serviço Web, no ponto de vista de protocolo de comunicação, ocorre devido à independência entre a forma que se desacopla o conteúdo a ser transmitido e a forma de comunicação entre o cliente (navegador) e o servidor (web). 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 adequado 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 [29, 81, 86, 99, 100, 102–122].

Atualmente, a visão supracitada (HTTP + TCP) não é empregada na Internet para o caso de distribuição de conteúdos multimídia ao vivo, pois a maioria dos serviços de transporte da Internet se limitam a oferecer recursos que auxiliam aplicações elásticas, não oferecendo

recursos para auxiliar a execução dos sistemas de distribuição de conteúdos multimídia ao vivo – a Internet não foi primariamente projetada para este fim. Por esse motivo, nos últimos anos, propuseram-se diversas soluções de aplicação baseadas em arquiteturas P2P e/ou CDN (P2P/CDN) para distribuição de mídias ao vivo, a fim de mitigar a falta de serviços de rede mais específicos e melhorar a qualidade das transmissões desse tipo de dados. Propuseram-se também protocolos de transporte a fim de suprir as necessidades de serviços de rede das aplicações multimídia, como o *Datagram Congestion Control Protocol*(DCCP) [123, 124]. Apesar de proporcionarem melhorias às aplicações, as soluções existentes ainda são insuficientes para atender à demanda crescente dos serviços de distribuição de mídias ao vivo, de modo que ainda é possível aprimorá-los em escala e em qualidade.

No caso das soluções baseadas em P2P/CDN, utilizam-se *middlewares* que implementam as principais funções necessárias para distribuição de conteúdos multimídia ao vivo. Devido à existência de diferentes opções, cada sistema faz uso de um *middleware* específico (quando o fazem), causando um problema de interoperabilidade entre os sistemas devido à incompatibilidade entre as diferentes propostas. A questão é que os *middlewares* encapsulam e compartilham recursos computacionais complexos para desenvolver aplicações de rede, mas não encapsulam e tampouco compartilham a forma que os nós deveriam se comunicar. Por exemplo, o TCP provê serviços primitivos ao funcionamento do serviço Web (conexão, garantia de entrega, ordenação dos pacotes de dados etc.), ao passo que as aplicações – clientes (navegadores) e servidores (web) processam o conteúdo a fim de apresentá-lo aos usuários finais. Em essência, as minúcias de comunicar diferentes nós em uma rede é responsabilidade da infraestrutura de rede e não das aplicações. A falta de interoperabilidade tem importância crítica para o caso dos atuais sistemas de distribuição de mídias ao vivo: delega-se muitas responsabilidades às aplicações, que passam a ter que tratar problemas que não existiriam se a rede oferecesse um melhor suporte para essa classe de aplicação. Como consequência, as aplicações se tornam soluções limitadas às estimativas sub-ótimas sobre a real disponibilidade dos recursos de rede, uma vez que não existe um protocolo equivalente ao TCP que provê os serviços primitivos ao funcionamento de tais sistemas.

Para instanciar um cenário de desempenho do estado da prática e do estado da arte nesse contexto, observe a Figura 1.2, a respeito de duas proeminentes propostas estudadas no contexto deste trabalho: o Denacast/CoolStreaming [125] e o CCN-TV [126]. Com relação à

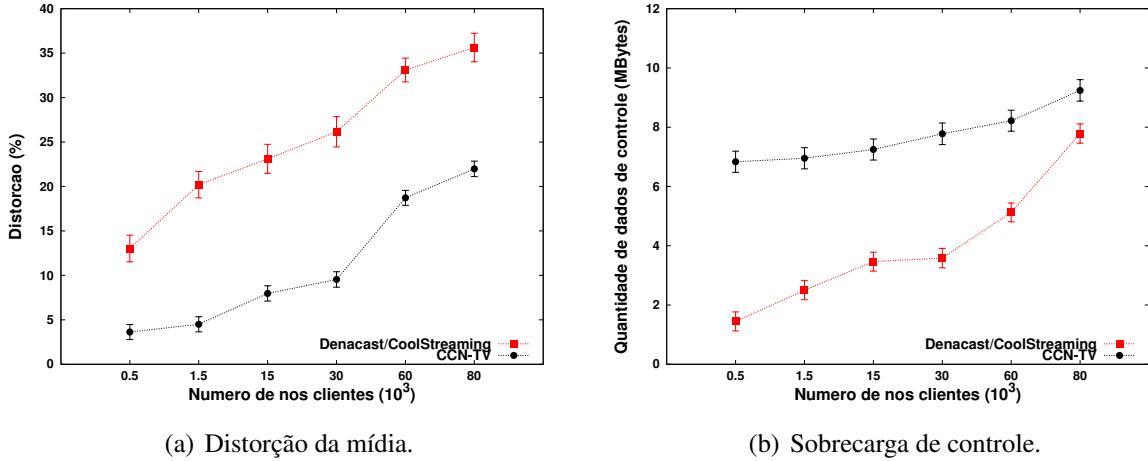


Figura 1.2: Confronto entre os sistemas Denacast/CoolStreaming e CCN-TV para as métricas distorção e sobrecarga de controle. Transmitiu-se uma mídia do tipo MPEG4-II, com taxa variável de bits de  $1 Mbps$ , através de uma rede simulada com 324 roteadores e diferentes larguras de banda. Mais detalhes sobre este experimento estão disponíveis no Capítulo 5 e no Apêndice A.

métrica qualidade da mídia transmitida, ilustrada na Figura 1.2(a), o Denacast/CoolStreaming obtém um desempenho degradante à medida em que aumenta-se o número de clientes interessados pela mesma mídia. Apesar de obter um melhor desempenho se comparado ao Denacast/CoolStreaming, observa-se também um comportamento similar no caso da CCN-TV. Por outro lado, com relação à métrica sobrecarga de controle, ilustrada na Figura 1.2(b), observa-se uma inversão no desempenho dos sistemas estudados, constatando-se que o Denacast/CoolStreaming obtém melhor desempenho com relação ao CCN-TV. O Denacast/-CoolStreaming pertence à categoria das soluções disponíveis na camada de aplicação, ao passo que o CCN-TV pertence à categoria das redes que oferecem serviços de distribuição de conteúdo, no caso a CCN. Enquanto a primeira executa suas próprias estratégias de distribuição de mídia, a segunda utiliza um modelo de serviço não apropriado para este tipo de tráfego de dados, evidenciando-se lacunas para avançar o estado da arte e propor soluções mais otimizadas.

Cenários como estes têm sido observado nos últimos anos, onde os pesquisadores têm apresentado diversas técnicas para otimizar a distribuição de mídias ao vivo na camada de aplicação, quando na verdade o problema é de infraestrutura, não se disponibilizando um protocolo que generalize os serviços comuns dos sistemas de distribuição de mídias ao vivo. Por exemplo, no caso dos *middlewares*, implementam-se funções para tratar aspectos de co-

nexão multi-ponto e diferentes topologias [4, 18, 27, 98, 127–130]; seleção de nós [131–135]; tolerância a desconexões e outros problemas causados pelo *churn* [45, 127, 130, 136–138]; disponibilização de informação de contexto sobre rede para dar suporte à execução dos serviços das aplicações, como localização da rede, medições e custos entre redes para melhorar a qualidade de serviços dos sistemas finais [131, 139–141]; estratégias para incentivos à cooperação entre nós [135, 142–147]; algoritmos para inibir a participação de nós *free-riders* [145, 148–150]; adaptação de fluxo baseado na capacidade de recepção dos nós [1, 2, 151, 152]; e segurança [147, 153–156]. De fato, neste trabalho, questiona-se a real necessidade de se tratar os problemas supostamente resolvidos por essas funções e, nesse ínterim, identifica-se quais desses serviços poderiam ser melhor aproveitados se posicionados e adaptados nas camadas apropriadas da pilha de protocolos TCP/IP.

O fato é que os temas supracitados foram ganhando importância ao longo dos anos porque os serviços de transmissão de mídias ao vivo se popularizaram, ao passo que surgiram mais necessidades de sofisticar as aplicações a fim de melhorar seu desempenho em entregar os fluxos de dados para uma população crescente de sistemas finais, com diferentes tipos de conexão e mobilidade. Apesar da arquitetura da Internet ser modularizada em camadas funcionais, 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 desenvolver uma solução na camada de aplicação do que em outras camadas mais abaixo, não apenas por aspecto de facilidade de implementação de software, mas também de implantação em larga escala na Internet, pois todas as outras camadas estão implementadas do sistema operacional para baixo.

A forma mais básica para distribuir conteúdos e já conhecida há pelo menos 3 décadas é o IP *multicast* [97]. Apesar de melhorar escalabilidade no número de receptores do mesmo conteúdo ao vivo, tal estratégia foi considerada insuficiente devido à complexidade de habilitá-la e mantê-la, principalmente em grande escala [86]. Para contornar a dificuldade em habilitar o modo *multicast* em nível de IP, propôs-se a arquitetura ALM (*Application Layer Multicast*) [157], que é equivalente ao IP *multicast*, porém implementada em nível de aplicação. A diferença entre o IP *multicast* 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 ALM 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 de uso e funcionalidades, incompatíveis entre si [86, 158–163]. Sendo assim, a questão de interoperabilidade dos sistemas de distribuição de mídias ao vivo vem à tona mais uma vez.

Uma possível proposta seria se os protocolos de transporte atuais (TCP, UDP, DCCP, SCTP etc.) pudessem absorver parte da complexidade dos serviços de distribuição multimídia, mas não o fazem porque não foram projetados para atender às necessidades peculiares desse tipo de serviço e alterá-los requereria um grande esforço, ao ponto de culminar em um novo protocolo de rede. Este fato acontece basicamente por dois motivos. O primeiro é um motivo histórico (no caso dos protocolos mais antigos, como o TCP e o UDP), pois na época de suas propostas não se discutia sobre sistemas de distribuição de mídias ao vivo. O segundo motivo é que mesmo os protocolos mais recentes (DCCP, SCTP etc.) não acompanharam a evolução das boas práticas no processo de distribuição de mídias ao vivo, como o uso de arquiteturas P2P/CDN e as diversas funções empregadas nesse contexto.

Para instanciar a discussão anterior, considere o caso do protocolo DCCP, proposto como um protocolo de transporte para transmissão de mídias ao vivo na Internet. 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. Este cenário é motivador para que os desenvolvedores de aplicações continuem tentando resolver na camada de aplicação, de forma recorrente e paliativa, um problema que é de infraestrutura de rede. É evidente que os sistemas atuais podem passar a utilizar o DCCP para transportar os segmentos de mídias ao vivo, mas o DCCP apresenta problemas críticos quanto à escalabilidade devido ao mecanismo de conexão e proposta de controle de congestionamento fim-a-fim e portanto os problemas discutidos até aqui continuariam explícitos na camada de aplicação. Estas limitações do DCCP foram constatadas em estudos preliminares realizados no contexto deste trabalho [164–166].

Apesar do protocolo DCCP ter sido utilizado para evidenciar parte dos problemas que acabara de ser apresentado, esta discussão pode ser generalizada para qualquer outro pro-

tocolo 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 disponíveis através das referências [167–186] e no Capítulo 3 deste documento.

Diante do exposto, remete-se esta pesquisa ao estudo de técnicas e mudança na perspectiva da execução de serviços de rede em camadas inferiores a da aplicação, buscando-se melhorias substanciais na execução dos sistemas de distribuição de mídia ao vivo, permitindo-se que a rede desempenhe um papel mais efetivo nesse processo. Por isto, pretende-se responder a seguinte questão de pesquisa: *como distribuir conteúdos multimídia ao vivo em larga escala, abstraindo-se a complexidade e promovendo-se a interoperabilidade entre os sistemas com base no princípio da independência dos serviços de rede?*

Para responder a questão supracitada, adota-se a seguinte hipótese: *a constituição de uma rede de favores entre roteadores que interceptam, realizam cache temporário e compartilham pacotes de dados tanto em modo multicast quanto em modo unicast, auxiliados por servidores para determinar as parcerias com base em um algoritmo para controle de congestionamento assistido pela rede, resulta na distribuição mais eficiente (escala e qualidade) de mídias ao vivo.*

## 1.3 Objetivos

Neste trabalho, o objetivo é o projeto e a avaliação de um protocolo de rede denominado *Global Media Transmission Protocol* (GMTP), que considera uma arquitetura híbrida P2P/CDN para distribuição de mídias ao vivo através da Internet. Mais especificamente, propor um protocolo que explora a relação entre o escalonamento de recursos computacionais em redes P2P, a estabilidade das redes CDN através da proposição de técnicas (algoritmos) que possam aumentar a eficiência de um sistema através da avaliação das métricas que determinam a qualidade de serviço quando usuários reproduzem uma mídia ao vivo através de uma rede.

Inicialmente, propõe-se dar um passo atrás e repensar *distribuição de mídias ao vivo em uma rede de datagramas IP* com a premissa de que a interoperabilidade dos sistemas é interesse geral de seus fornecedores a fim de permitir que os datagramas referentes a uma mídia ao vivo sejam reproduzidos nas aplicações dos usuários finais o mais rápido possível

e com melhor qualidade, sem preocupações com restrições comerciais.

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

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, revisar as propostas proeminentes nesse contexto, a fim de entender quais outras funções estão presentes nesse tipo de sistema e compará-las ao GMTP;
2. projetar o GMTP e discutir as principais funcionalidades com vista à distribuição de mídias ao vivo. Além disso, implementar o GMTP em um ambiente que proporcione a obtenção de valores para as métricas que determinam a experiência de um usuário ao assistir a um evento ao vivo através de uma rede, permitindo-se estudá-lo por meio de diferentes configurações, entender suas vantagens, limites e os impactos que seus recursos podem gerar tanto sobre os nós quanto sobre a rede;
3. avaliar e discutir o protocolo GMTP em comparação ao estado da arte, discutindo-se sobre suas vantagens e desvantagens.

## 1.4 Relevância do Tema e da Tese

Distribuição de mídias ao vivo é um tema relevante no contexto de redes de computadores devido aos recentes interesses dos usuários por esse serviço, em particular, na Internet, como se discutiu na Seção 1.1.

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 à consolidação de algumas soluções em paralelo ao surgimento de novas técnicas, métodos e/ou paradigmas, como as redes centrada na Informação 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 funções e problemas inerentes a essa classe de aplicação, bem como os cenários considerados na Internet, considerando diferentes topologias de rede e heterogeneidade de nós. 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 em disponibilizar serviços multimídia na Internet.

Nesse contexto, apesar das 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, 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. No que diz respeito à relevância do trabalho na perspectiva de engenharia de software para sistemas distribuídos, considera-se 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 requisito 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 confiável 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 que combinem as características aqui propostas com base no objetivo supracitado, 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 6.3.

*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. 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.

## 1.5 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 o protocolo *Global Media Transmission Protocol* (GMTP).
- No Capítulo 5, apresentam-se os resultados sobre o desempenho do protocolo GMTP quando confrontado com duas outras proeminentes soluções disponíveis na literatura.
- Por fim, no Capítulo 6, 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 as principais estruturas adotadas nos sistemas de distribuição de mídias ao vivo e as funções básicas empregadas nesses sistemas. Por fim, como complemento, discute-se brevemente sobre criptografia e assinatura digital, compreendendo alguns conceitos aplicados ao GMTP para evitar ataques de poluição. Caso o leitor se considera familiarizado com esses assuntos, recomenda-se, pelo menos, a leitura do sumário deste capítulo, disponível na Seção 2.4. Caso necessário, sugere-se as seguintes referências complementares:

- Em [187–189], para conceitos básicos sobre comunicação e redes de computadores, modos de transmissão, redes de distribuição de conteúdo e protocolos de rede;
- Em [190], 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;
- Em [50, 69, 191–193], para conceitos sobre Redes Centradas no Conteúdo; e
- Em [97, 153], para mais detalhes sobre os conceitos apresentados neste capítulo.

## 2.1 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 previamente 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 [97].

Recentemente, vários sistemas P2P foram desenvolvidos para prover conteúdo de vídeo ao vivo e sob-demanda na Internet, com baixo custo operacional [84, 86, 100, 101, 194–198]. 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.

### 2.1.1 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*) [137, 199]. 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 [97]. 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 [161], utilizam esta abordagem. Na Figura 2.1, ilustra-se um sistema com quinze nós participantes.

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

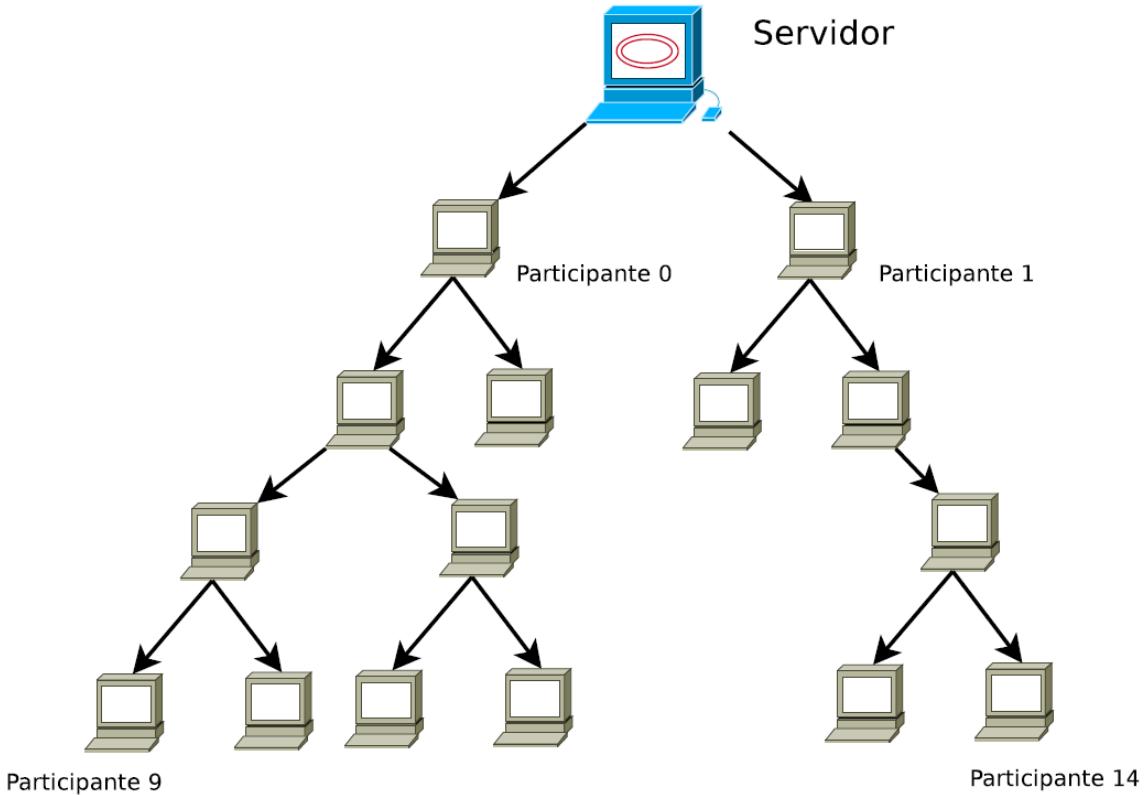


Figura 2.1: Árvore de *multicast* em nível da camada de aplicação.

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.

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 [97]. Vários algoritmos distribuídos abordam e tratam o problema de manutenção e construção da árvore de maneira distribuída [104]. 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 [97, 137, 199].

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

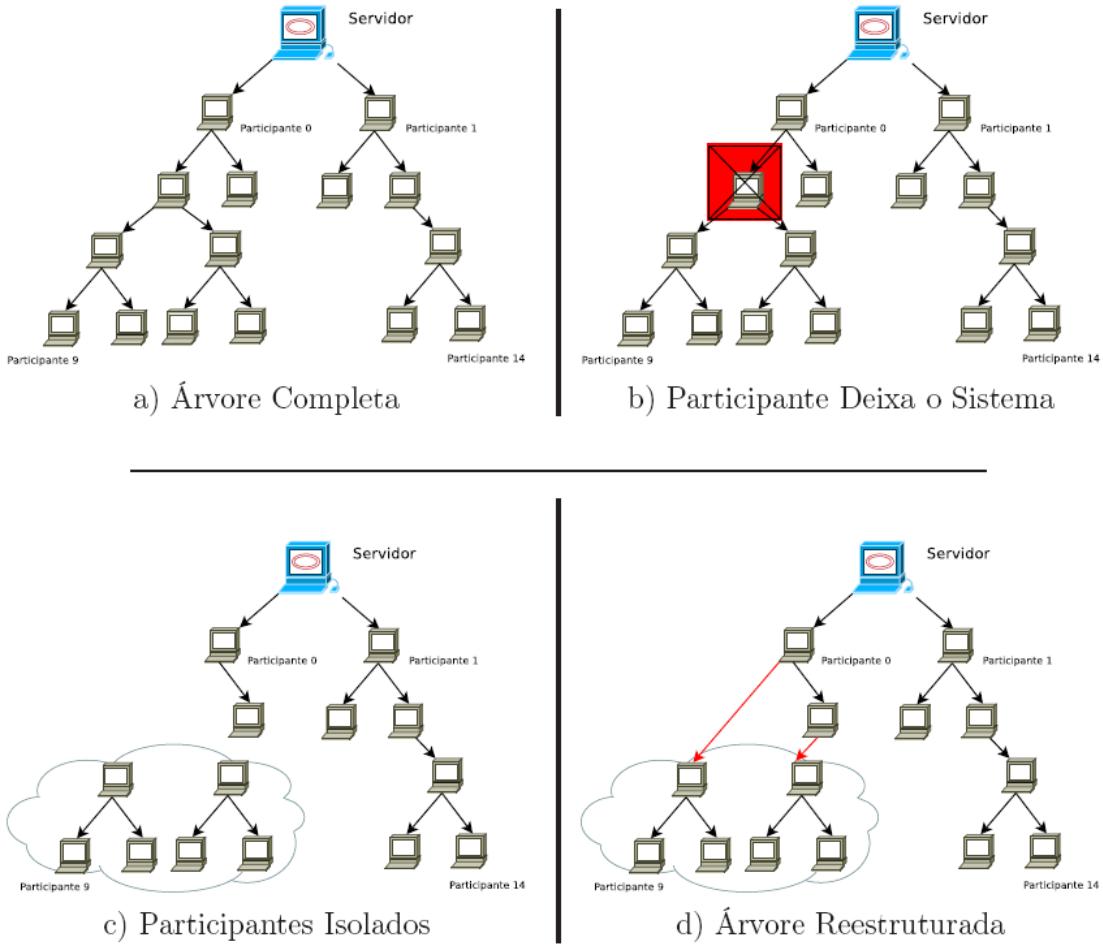


Figura 2.2: Manutenção da árvore de *multicast* em nível da camada de aplicação.

da árvore se torna subestimada. Para lidar com esse problema, foram propostas abordagens baseadas em múltiplas árvores como em [86]. 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.

### 2.1.2 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 dispo-

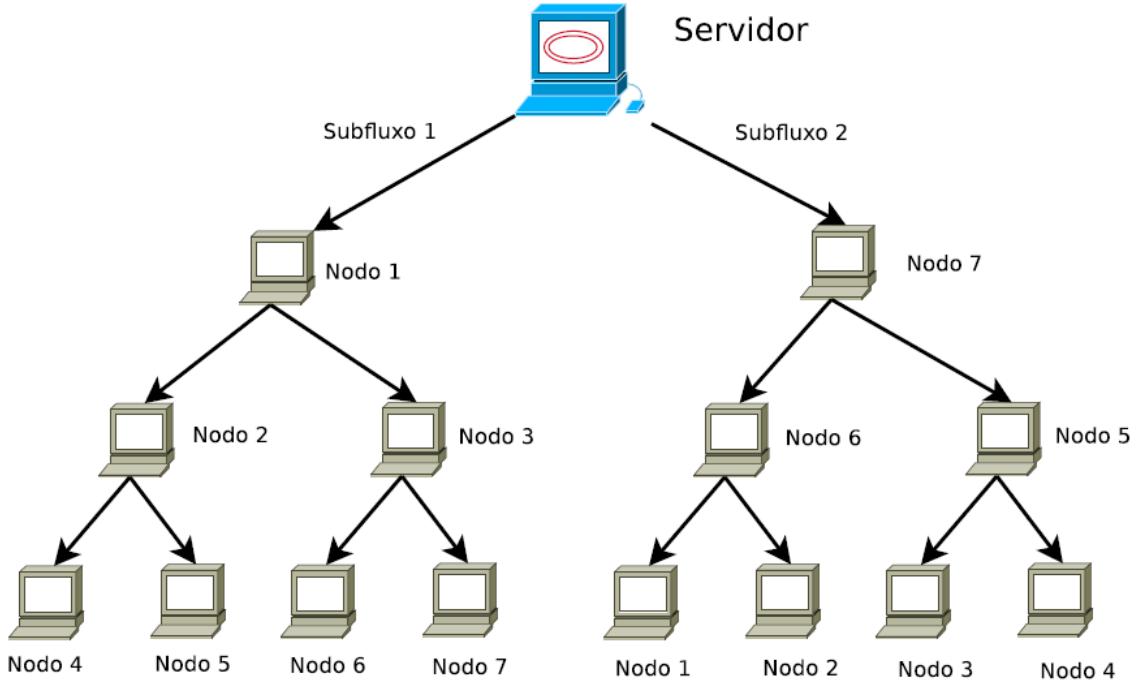


Figura 2.3: Sistema baseado em múltiplas árvores com dois subfluxos.

níveis 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 [83, 101, 198, 200]. Em um sistema desse tipo não existe uma topologia fixa da 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 [199], 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*, *rendevouz* 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.

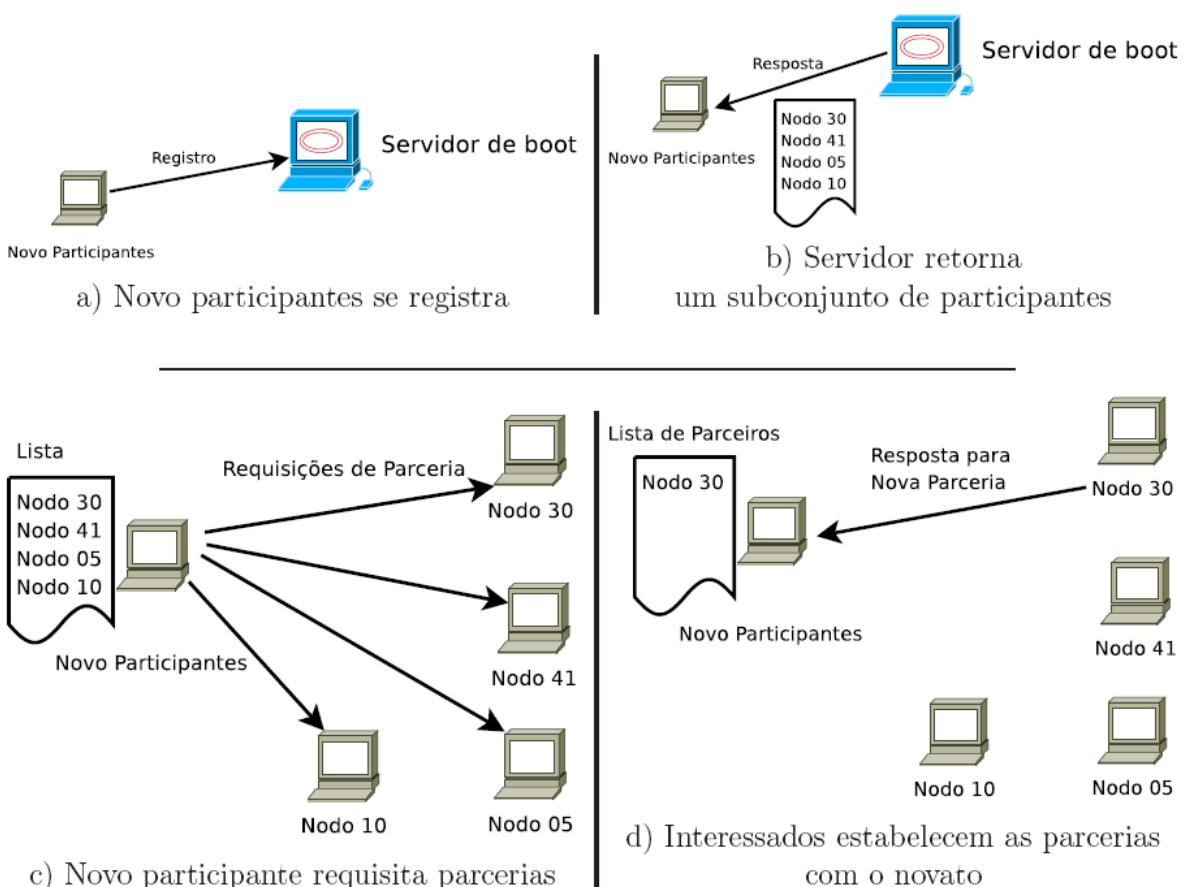


Figura 2.4: Atividade inicial de um novato - rede P2P baseada em malha.

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 [198]. 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 [97]. 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 [97]. Esses *chunks* são numerados em uma sequência temporal, para que os nós participantes possam identificar e executar o vídeo correspondente de forma adequada. 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 [198].

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.

### 2.1.3 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.

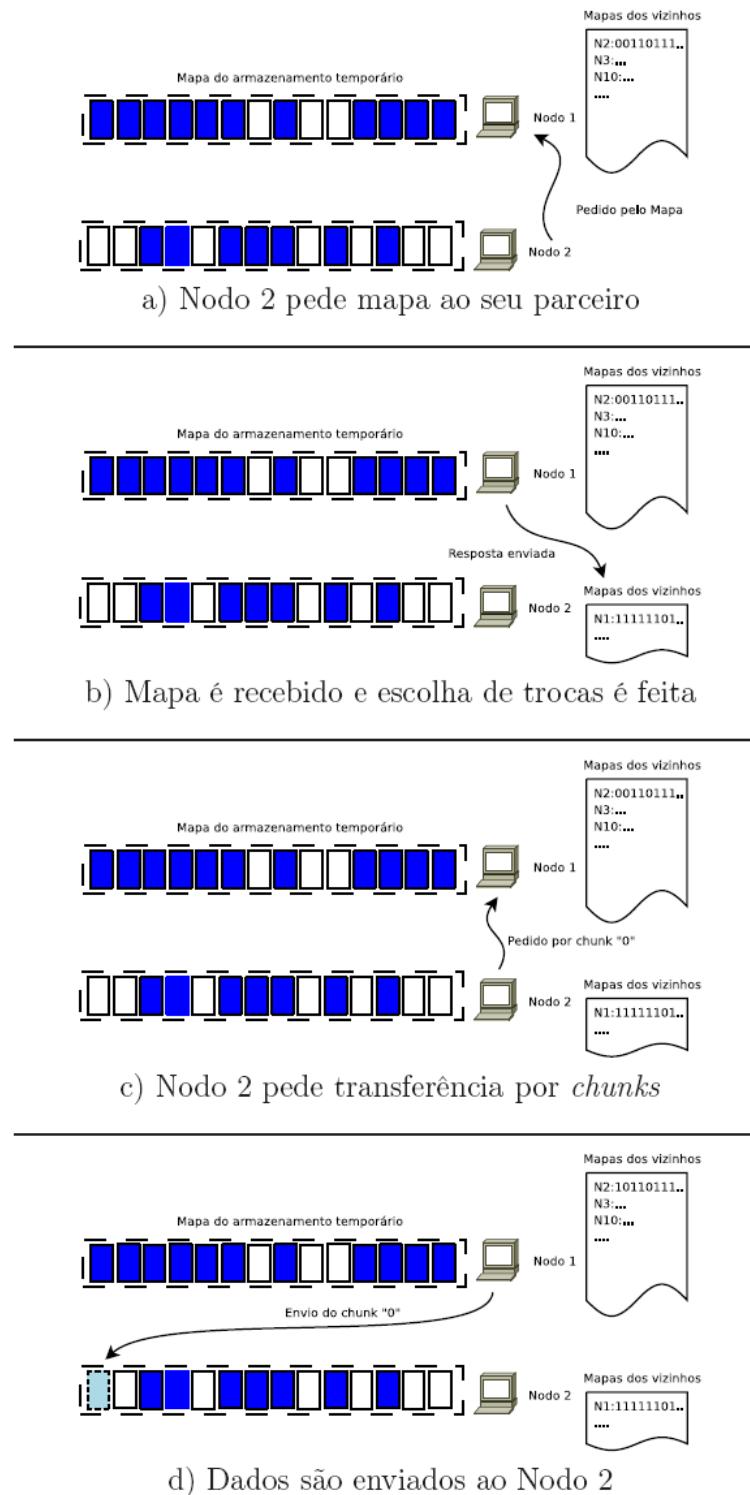


Figura 2.5: Troca de dados na aplicação baseada em malha.

## 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 imprevisibilidade [82].

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 [81] 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 explícito), 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*” [78, 108]. 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 [201] 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 [107], 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 [202], 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.

## 2.2 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 [100], o PPLive [102], o GridMedia [198, 202], Octoshape, e o CoolStreaming [95, 101]. Na Figura 2.6, ilustra-se um cenário exemplo de um sistema de transmissão ao vivo em P2P que será detalhado nessa seção.

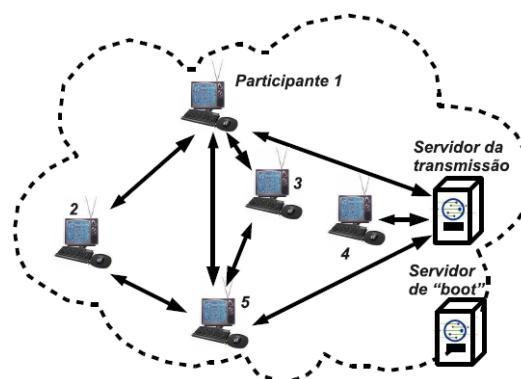


Figura 2.6: Modelo de sistema utilizado.

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 2.1. 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 [99, 202].

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  $B$  [99]. 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 ve-

rifica 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, apresentam-se as notações utilizadas para descrever um sistema P2P de transmissão ao vivo.

Tabela 2.1: Notações para descrever um sistema P2P de transmissão ao vivo.

Notação	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 $p_i$ .
$cm_i[x] = desejada$ , onde $x = [0..m]$	Conteúdo inicial do mapa de <i>chunks</i> de $p_i$ .

### 2.2.1 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,  $cm_s[c_i] = disponivel$ .

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, apresentam-se as notações para descrever a função de geração das partes de uma mídia em transmissão ao vivo baseado em uma arquitetura P2P.

Tabela 2.2: Notações utilizadas para descrever a geração de mídia ao vivo em sistemas P2P.

Notação	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 $p_i$ .

## 2.2.2 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 iniciada 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*.

### 2.2.3 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

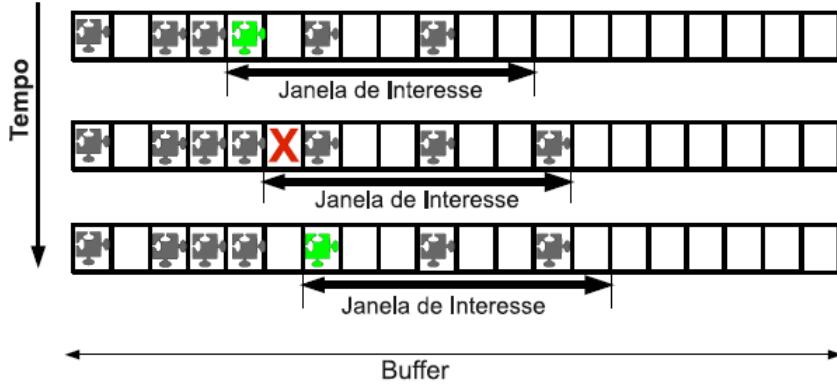


Figura 2.7: Mecanismo de consumo da mídia ao vivo.

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 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 [101]. 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  $S$ .

#### 2.2.4 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.

## 2.3 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 [203], 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.

### 2.3.1 Criptografia de hash

Uma função de *hash* [204]  $H(m)$  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*) [205].

### 2.3.2 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  $K_A^+$ .

### 2.3.3 Assinatura e certificação digital

Uma assinatura digital [206] é um modelo de autenticação de informações digitais tipicamente análoga à assinatura física em papel. Através de uma assinatura digital, pode-se provar

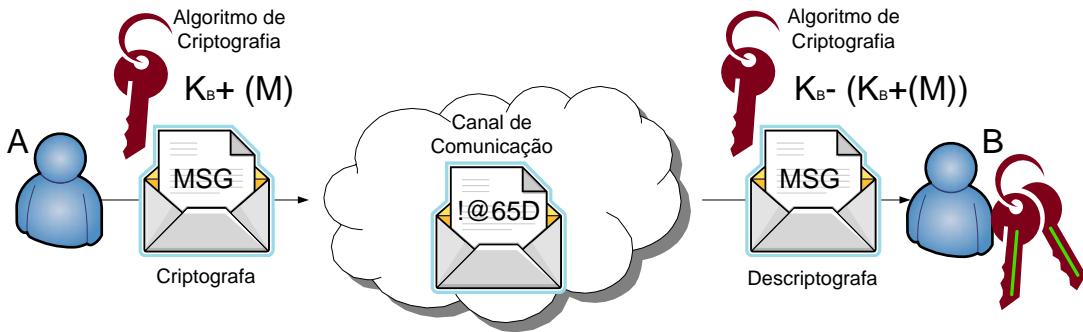


Figura 2.8: Modelo de criptografia assimétrica, chave pública representada por  $K^+$  e chave privada presentada por  $K^-$ .

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 corresponde à chave pública utilizada na verificação; ou
  - Se  $m'_{recebida} \neq m'_{enviada}$ , a a mensagem  $m$  foi alterada, ou seja, a assinatura digital emitida pelo sistema final  $A$  não corresponde à mensagem  $m$ .

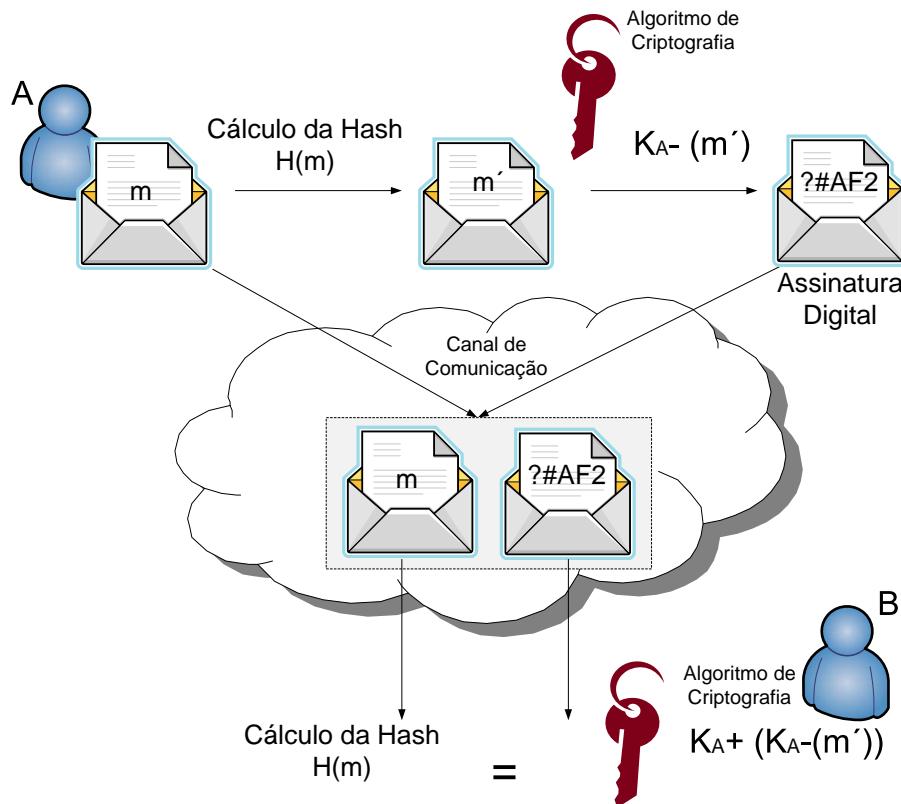


Figura 2.9: Geração de Assinaturas Digitais.

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-repudiação 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 [207] (*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.

## 2.4 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 se baseiam 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 partes 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. No contexto deste trabalho, aplicam-se tais conceitos para criptografar e assinar digitalmente os dados transmitidos por um sistema de transmissão ao vivo a fim de 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, consideram-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 comparados 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, das Seções 3.2.6 e 3.2.7 e da 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.

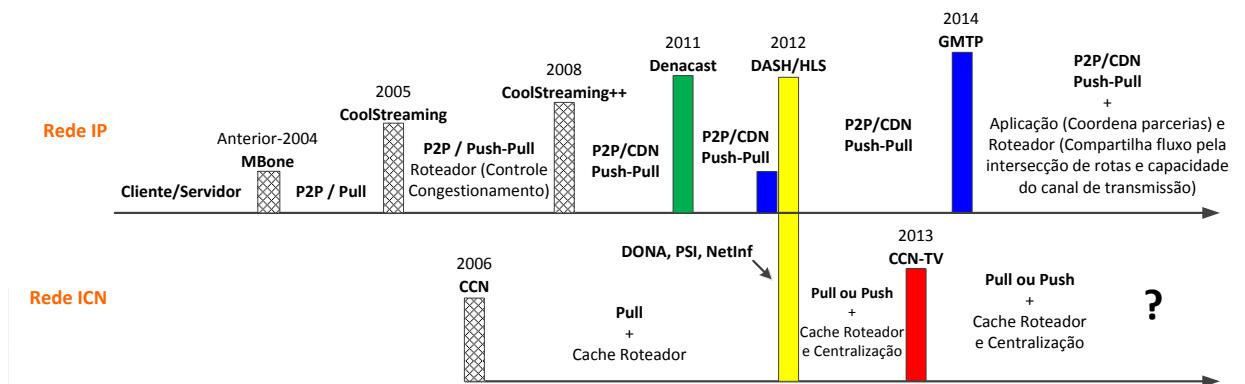


Figura 3.1: Representantes e marcos importantes no contexto de distribuição de mídias ao vivo. Evolução das arquiteturas e modelos de serviços adotados na Internet (incluindo o GMTP). Antes do GMTP, todas as soluções têm uma forte dependência dos sistemas finais para executar suas funções. No GMTP, os 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 as propostas DONet/CoolStreaming [83], Denacast [6, 125] DASH [2], detalhados na Seção 3.2. Já para as Redes ICN, destaca-se a CCN [191], representada inicialmente pelo sistema CCN-TV [126] para o contexto deste trabalho. Apesar do foco da discussão neste capítulo ser nesses trabalhos, outros trabalhos relevantes serão mencionados.

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 clientes. Em 2008, os autores do CoolStreaming propuseram melhorias arquiteturais, sugerindo a mudança do método *pull* para o método *push-pull* de *download* das partes de uma mídia.

Em paralelo ao fato supracitado, entre 2002 e 2005, publicaram-se as primeiras soluções de um protocolo de controle de congestionamento assistido pela rede chamado XCP [53]. 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 [58] e do VCP [62], após a identificação de algumas limitações do XCP [56]. Em 2010, publicou-se o ConEx [68, 69], tendo como principal objetivo permitir que os sistemas finais compartilhem, com os roteadores, informações sobre o congestionamento experimentado pelas aplicações em execução. Desde então, não sugiram novas propostas relevantes nesse contexto, exceto o ALTO (*Application-Layer Traffic Optimization*) [208, 209], um serviço que provê informações de contexto sobre as redes (por exemplo, localização, estrutura e rotas preferenciais). O objetivo do ALTO é modificar os padrões de consumo de recursos de rede e melhorar o desempenho das aplicações e, no contexto dos sistemas de distribuição de mídias ao vivo, seu uso ocorre quando um 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 correspondente. 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 [210]. Isto ocorre porque os clientes decidem com qual frequência transmitir informações sobre seu estado ao servidor, como capacidade de transmissão, número de saltos entre o cliente e o servidor etc. No GMTP, as medições são

feitas pelos roteadores com base no protocolo RCP e os servidores decidem quais parcerias devem ocorrer, de acordo com a capacidade de cada canal utilizado para distribuir o fluxo de dados.

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 [6, 103, 110, 111, 113, 114]. Além disso, atualmente, existem diversas pesquisas para distribuição de mídias ao vivo através do uso do protocolo HTTP, permitindo-se que os 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, destaca-se as soluções híbridas P2P/CDN, com o uso de CDNs comerciais. Até o presente momento, o esforço têm sido 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 [211]. Esta palestra marcou o (re)início das pesquisas nessa 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 [50]. 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*) [192]. Especificamente em 2013, publicou-se o CCN-TV, o primeiro sistema para distribuição de mídias ao vivo em CCN. Na Seção 3.3, discutem-se detalhes sobre CCN e sobre o CCN-TV.

Dentre todas as soluções discutidas, pode-se considerar o CoolStreaming/Denacast e o CCN-TV fundamentalmente trabalhos mais relacionados ao GMTP. Por esse motivo, no Capítulo 5, 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, gerando 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 com legados que contribuiram para a evolução do estado da prática e da arte. Nos últimos 15+, a maior parte das soluções propostas para disseminação de mídias ao vivo são apresentadas como sistemas ou *middlewares*. A seguir, destacaram-se algumas dessas propostas.

### 3.2.1 Protocolos de adaptação de fluxo baseado em HTTP

No contexto dos protocolos de adaptação de fluxo baseado em HTTP, destacam-se o HLS (*HTTP Live Streaming*) [1], HDS (*HTTP Dynamic Streaming*) [212], DASH (*Dynamic Adaptive Streaming over HTTP*) [2, 213] e HSS (*Smooth Streaming*) [214], 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, porém com base no protocolo HTTP.

Uma característica comum entre os protocolos HLS, HDS, HSS e DASH é que a transmissão da mídia ocorre sem a manutenção de uma conexão, já que o HTTP é um protocolo *stateless*. Além disso, os clientes podem escolher entre diferentes modos de codificação mul-

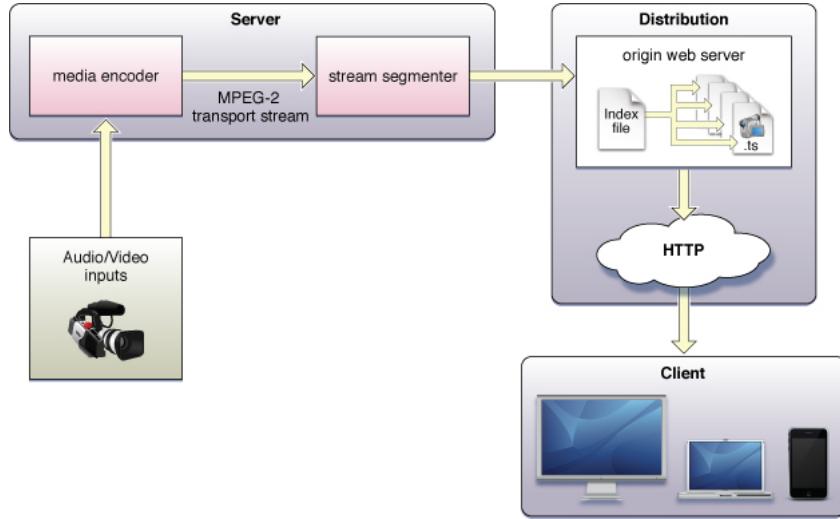


Figura 3.2: Arquitetura do HLS para transmissão de conteúdos multimídia baseado no protocolo HTTP. Figura extraída e adaptada de [1].

timídia, 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*) [215], que aponta para um arquivo contendo uma lista ordenada das URIs para as mídias a serem reproduzidas. Para reproduzir um determinado conteúdo, um cliente primeiro obtém o arquivo contendo a lista de URIs e então obtém e reproduz cada segmento de mídia especificado na lista. Periodicamente, o 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.

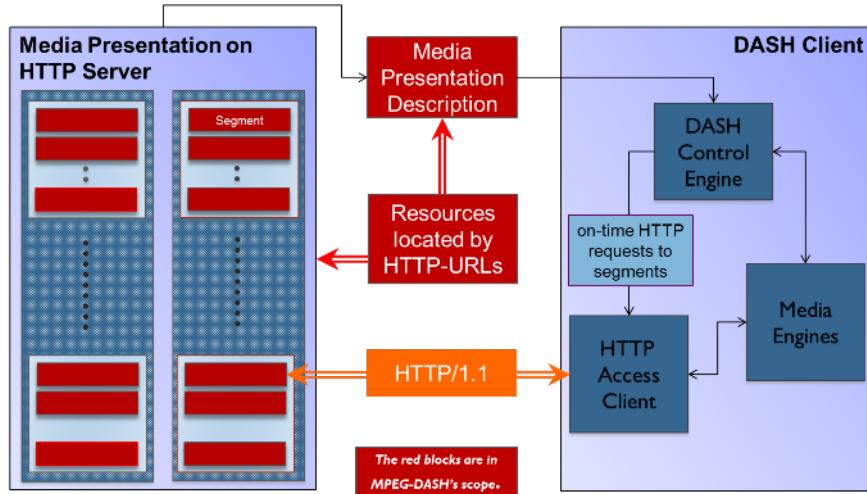


Figura 3.3: Arquitetura do DASH para transmissão de conteúdos multimídia utilizando o protocolo HTTP. Os formatos e as funcionalidades dos blocos vermelhos são definidos pela especificação do DASH. Adaptado de [2].

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 clientes. Isto permite que inúmeros clientes, que suportam diferentes formatos e conectados através de redes com diferentes capacidades de recepção, possam reproduzir o conteúdo de forma adequada. Para que isso seja possível, os 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 Kbps e o servidor anunciar a transmissão de vídeo em 512 Kbps, 720 Kbps, 1 Mbps e

4 Mbps, este deve requisitar o conteúdo codificado em uma taxa de bits de 512 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 anterior sobre a distribuição de mídias ao vivo através da web, consideram-se as seguintes limitações de tal abordagem:

- 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, tratam-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. Nessa rede, o conteúdo pode ser entregue pelo servidor diretamente ao roteador do cliente ou indiretamente, quando o conteúdo é entregue por um roteador parceiro ao roteador do cliente interessado pelo fluxo e assim sucessivamente. Nesse contexto, pode-se construir uma aplicação baseada em DASH e utilizar GMTP para transportar os pacotes de dados, 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 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. Por outro lado, se diminuir a sobrecarga de controle, aumenta-se o tempo de atraso na recepção dos pacotes de dados [101, 108, 216]. Além disso, essa proposta exige uma maior capacidade de *upload* dos cliente, uma vez que estes devem transmitir frequentemente requisições para obter a lista dos próximos segmentos a serem reproduzidos. No caso do GMTP, utiliza-se 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 uma determinada parte de uma mídia e esta ainda não está disponível. Além disso, apesar do método *pull* não necessitar conexão e o HTTP é interessante para este fim, no GMTP, reduz-se o número de conexões ao permitir que os roteadores interceptem os pedidos de múltiplas conexões transmitidas em direção ao 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. No contexto de HTTP, os clientes precisam monitorar sua capacidade de recepção e solicitar ao servidor os segmentos correspondentes. Isto significa que o conteúdo é adaptado de acordo com o estado da rede percebida pelo cliente TCP [217]. 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, utilizam-se informações explícitas sobre a capacidade de transmissão dos canais de transmissão informadas pelos roteadores, regulando-se a taxa de transmissão dos servidores, ao mesmo tempo em que se determinam parcerias entre os roteadores com base em tais informações. Como resultado, pode-se melhorar o desempenho das redes e das aplicações e, consequentemente, melhorar a satisfação do usuário ao visualizar uma mídia ao vivo.

- Soluções baseadas em HTTP devem considerar aspectos relacionados à segurança, descritos na Seção 15 da referência [218]. Os principais são a disponibilização de informações pessoais e privadas e possíveis ataques de interceptação dos seguimentos através do uso de servidores de *cache*. No GMTP, oferece-se um mecanismo de segurança que permite aos 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 [219].

### 3.2.2 PDTP – *Peer Distributed Transfer Protocol*

O protocolo *Peer Distributed Transfer Protocol* (PDTP) [3] 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, ressaltam-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, tornando-o similar aos protocolos como o HTTP e o FTP. Além disso, na proposta do PDTP, previa-se suporte à 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 PDTP 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 que se utiliza um conjunto de nós chamados de *Piece Proxies* (PP). Os PPs fazem *download* e *cache* de *chunks* de arquivos armazenados nos nós *hubs* e então servem estes *chunks* 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.

---

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

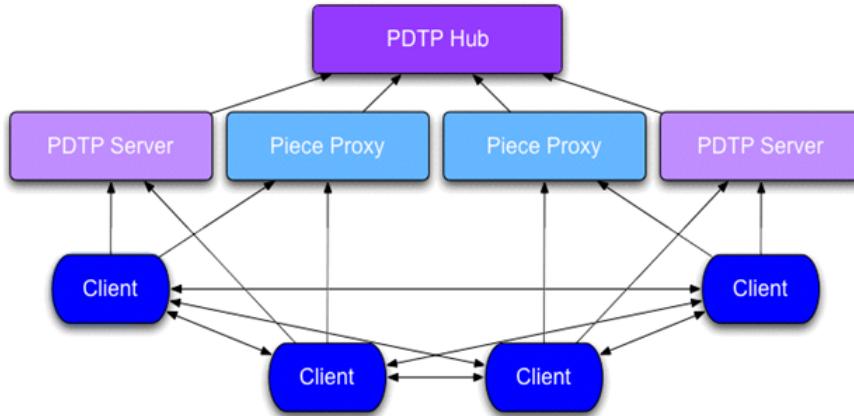


Figura 3.4: Organização dos nós PDTP. Adaptada de [3].

#### **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, podendo desfavorecer os clientes que estão mais distantes. No GMTP, os servidores de uma rede CDN são também os indexadores de conteúdo, bem como auxiliadores no processo de definir as parcerias entre os roteadores.

#### **3.2.3 CPM – Cooperative Peer Assists and Multicast**

No *Cooperative Peer Assists and Multicast* (CPM) [4], propõe-se uma abordagem unificada para prover suporte eficiente de transmissão de vídeos sob demanda, para ser utilizado por provedores de serviços. O CPM é um protocolo de aplicação que suporta transmissão em modo *multicast*, *cache* de dados nos clientes e compartilhamento de dados entre os mesmos, 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.

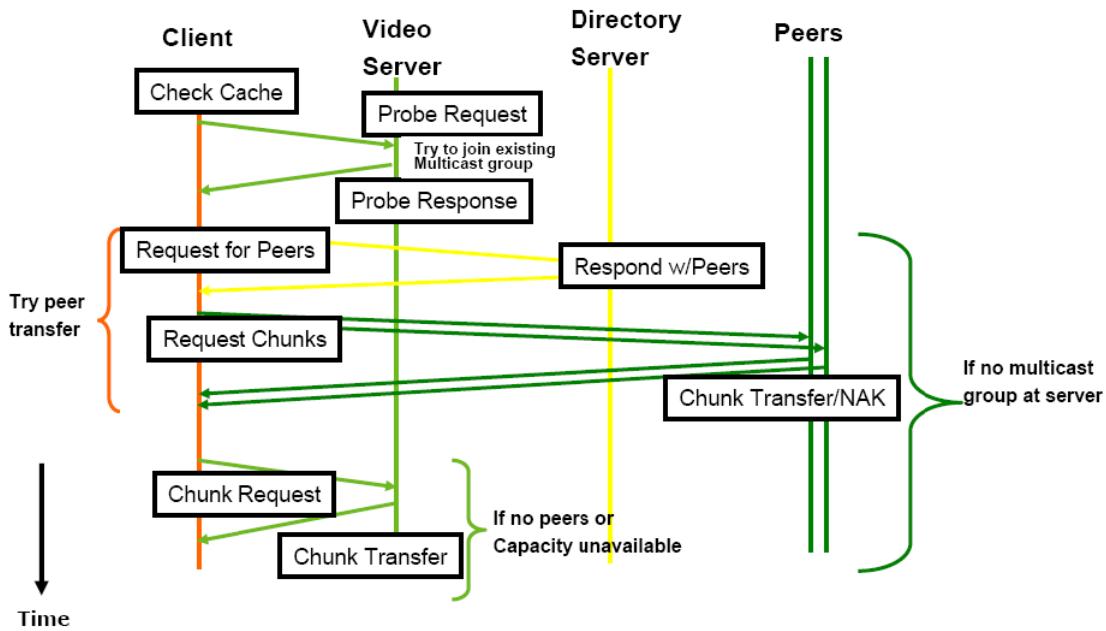


Figura 3.5: Diagrama de sequência do CPM (*Cooperative Peer Assists and Multicast*). Figura extraída de [4].

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 no lado do servidor. O modelo de dados divide o vídeo em *chunks* de tamanhos fixos. Cada *chunk* é identificado por um GUID (*Globally Unique Identifier*), onde um segmento consiste em uma sequência de *chunks* 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 *chunks* 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 *chunks* e transmitidos para o cliente. 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 *chunks*, o que torna o gerenciamento mais complexo devido ao espalhamento dos *chunks* entre os nós participantes da transmissão. Isto pode gerar atrasos na reprodução do conteúdo no cliente devido à necessidade de localizar cada *chunk* individualmente, embora o uso dessa abordagem pode aumentar a velocidade de *download*. No caso do GMTP, realiza-se uma duplicação controlada dos *chunks*, realizada pelos próprios roteadores. Os clientes não têm influência sobre essa função, portanto não precisam gerenciar múltiplas fontes (outros clientes e servidores).
3. O uso de servidor de diretórios para consultar a lista de nós que mantêm o conteúdo multimídia desejado não é uma estratégia apropriada em sistemas de transmissão de mídia ao vivo, pois se trata de uma fonte de atraso para disseminar rapidamente as partes da mídia.

**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) [5], 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 clientes. Como consequência, aumenta-se o consumo de recursos de rede e piora-se a 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 entregá-lo é baseado na popularidade do mesmo. 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.

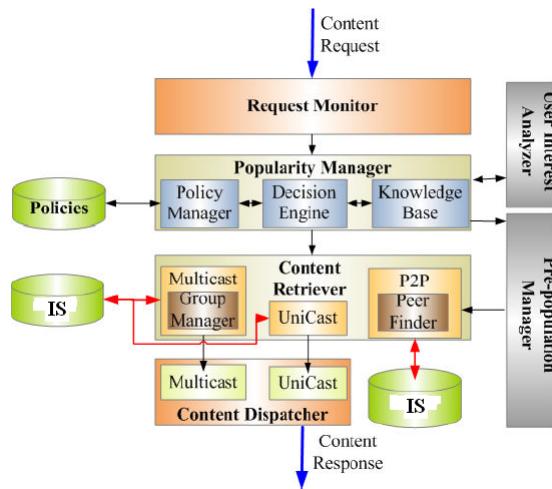


Figura 3.6: Arquitetura do HySAC (*Hybrid Delivery System with Adaptive Content Management for IPTV Networks*). Figura extraída de [5].

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 de se utilizar *multicast* para transmiti-lo. Se a popularidade do vídeo for intermediária, o vídeo será transmitido em modo P2P e se for baixa, transmite-o do servidor diretamente para o cliente em modo *unicast*.

**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 armazenam-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, sem qualquer menção a respeito de algoritmo(s) 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 clientes e o servidor para realizar acesso ao *cache* dos roteadores.

### **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 e que mantêm meta informações sobre os conteúdos transmitidos e uma lista dinâmica de *peers*. Os *trackers* podem ser organizados de forma centralizada ou distribuída, propondo-se dois protocolos base. O primeiro protocolo é executado pelos *trackers*, que cuida das trocas de meta informações entre os *trackers* e os *peers*, tais como a lista dos peers e informações sobre os conteúdos. E o segundo protocolo

é executado pelos *peers*, que controla os anúncios e informações sobre a disponibilidade das partes da mídia em cada *peer*.

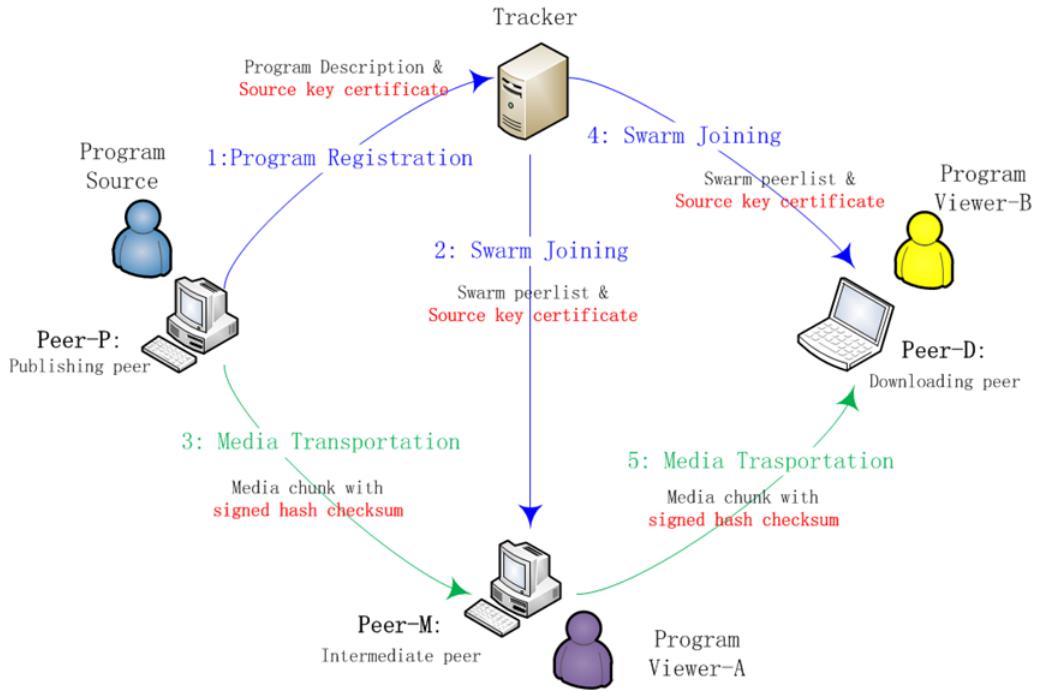


Figura 3.7: Arquitetura e funcionamento do protocolo PPSP/Swift.

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 Multiparty Transport Protocol* (Swift). A responsabilidade do Swift no processo descrito é cuidar do transporte de dados entre os *peers* interessados 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. O Swift transmite os dados entre os *peers* utilizando o protocolo UDP com suporte ao controle de congestionamento chamado de LEDBAT [176, 220], o mesmo empregado no

BitTorrent.

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

Em 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. Como já foi dito, 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 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*. No caso do GMTP, negociam-se dinamicamente os canais *multicast*, de acordo com as demandas locais de cada rede. A comunicação ocorre com a transmissão dos pacotes de dados entre o servidor e o roteador do cliente em modo *unicast* e tal roteador estabelece-se um canal *multicast* através do qual seus clientes receberão as partes da mídia referentes a um evento a vivo. Nesse ínterim, roteadores intermediários entre o cliente e o servidor podem interceptar o pacote de dados e também servir seus clientes locais (se houver demanda).
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 rede, e não na camada de aplicação. Isso limita o uso dos recursos implementados no Swift apenas para aplicações que utilizam sua implementação, a *libswift*. Por ser um protocolo de transporte/rede, o GMTP abstrai as principais funções de distribuição de mídias ao vivo e considera o uso de *multicast* sempre que possível, com suporte aos algoritmos de controle de congestionamento tanto no modo *unicast* quanto no modo *multicast*. Além disso, se duas aplicações independentes utilizarem o protocolo GMTP, estas poderão compartilhar o conteúdo multimídia transmitido por um servidor, sendo tal recurso disponibilizado de forma nativa e transparente à qualquer aplicação – os sistemas finais recebem meta informações sobre o conteúdo multimídia.

### 3.2.6 DONet/CoolStreaming

O CoolStreaming é um sistema para distribuição de mídias ao vivo baseado em uma arquitetura P2P [83, 101, 221]. 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, 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

---

<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.

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.

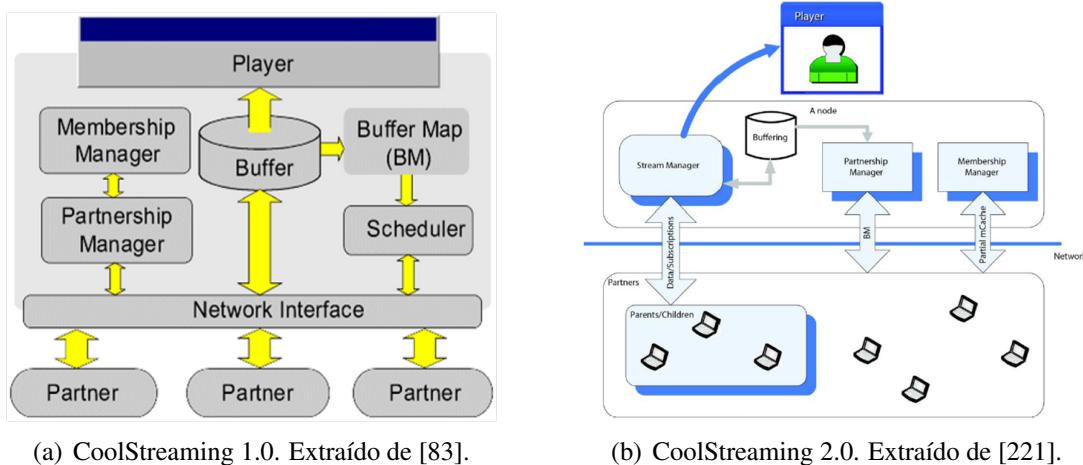


Figura 3.8: Arquitetura genérica de blocos funcionais do CoolStreaming 1.0 e do 2.0.

O sistema CoolStreaming tem versões similares (Figura 3.8). 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 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 cliente. Nesse caso, o nó transmissor divide o vídeo em blocos de tamanhos iguais e marca 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.

Na primeira versão, os blocos de dados eram obtidos sempre utilizando o método *pull*, o que acarretava no aumento do 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. Dado que um nó  $C_1$  está interessado em obter um determinado bloco da mídia, este solicita e recebe de um nó parceiro  $C_2$  o mapa de buffer contendo a disponibilidade dos blocos do vídeo (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-o ao nó  $C_1$  (ou seja, *push*).

Os outros componentes do sistema CoolStreaming são:

- *Membership manager*, que permite aos 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 pelas trocas dos mapa de *buffer* entre os clientes. O algoritmo 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 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 à seleção de nós no sistema CoolStreaming, este 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 em 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, monitora-se 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, sendo  $T_s$  e  $T_p$  duas métricas para especificar a capacidade de *upload* de um

nó  $C_1$  e correspondem aos números de sequência de blocos para um sub-fluxo qualquer  $j$  em  $C_2$ , onde:

- $T_s$ , é o limite do máximo número de sequência permitido entre os últimos blocos de vídeo recebidos por qualquer dois sub-fluxos em  $C_2$ ;
- $T_p$ , 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  $S_i$  em  $C_2$ ;
- $K$ , é o número de sub-fluxos gerados pelo nó transmissor que origina o conteúdo de vídeo.

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

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

Com base nisso, um 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 suficiente de *upload*. 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*.

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

O CoolStreaming foi um dos projetos escolhidos para um estudo comparativo de desempenho frente ao GMTP. Por este motivo, reservou-se uma discussão mais detalhada sobre este assunto no Capítulo 5.

#### 3.2.7 CoolStreaming/Denacast

O Denacast é um sistema para distribuição de mídias ao vivo baseado em um arquitetura híbrida P2P/CDN [6, 125]. 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.

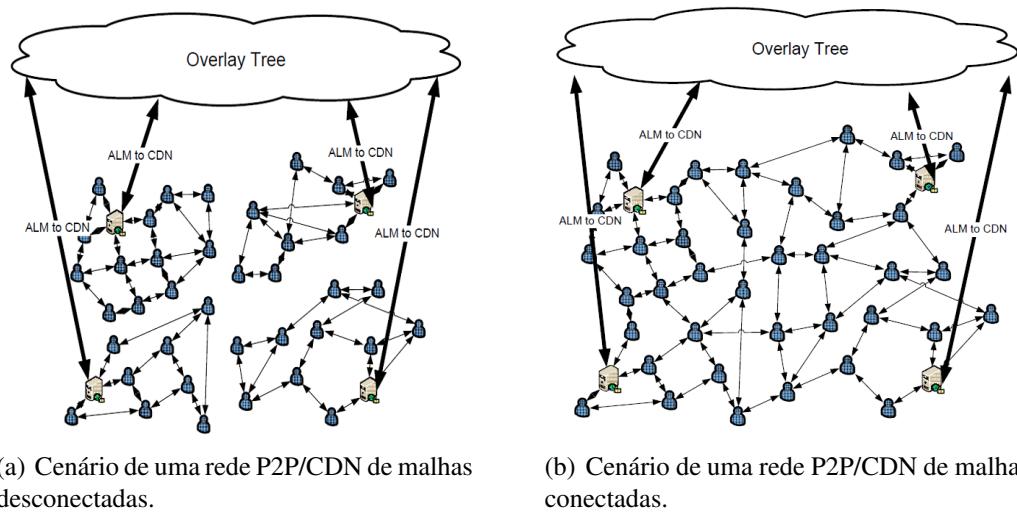


Figura 3.9: Arquiteturas CDN/P2P utilizadas no Denacast. Figuras extraídas de [6].

Como ilustra-se na Figura 3.9, no Denacast, adotam-se duas arquiteturas CDN/P2P. Ambas arquiteturas têm em comum 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)): constituem-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 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, ao passo que 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 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. Isto ocorre quanto a quantidade de nós ativos nas duas redes ultrapassa o valor do número de 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/CoolStreaming e o GMTP para distribuição de mídias ao vivo:**

O Denacast/CoolStreaming foi um dos projetos escolhidos para um estudo comparativo de desempenho frente ao GMTP. Por este motivo, reservou-se uma discussão mais detalhada sobre este assunto no Capítulo 5.

#### **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 neste trabalho, 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, catalogaram-se 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 [81], BEAM/Alliances [29], BitTorrentLIVE [105], DLNA-P2P [106], GridMedia [99, 107, 108], IV5S [109], Joost [110, 111], LayeredCast [112], LiveSky [113], Octoshape [114], OverCast [115], Pastry/SplitStream [86], PeerCast [116], PPPLive [102], PRIME [117], PROMISE [118], PULSE [103], SAMP [119], SmoothCache [120], Sopcast [100], SPPM [121], TURINstream [122] e ZIGZAG [104]. 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 [222], que foram resolvidos por outras proposta [223] ou utilizadas de forma errada [224]. 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*) [48].

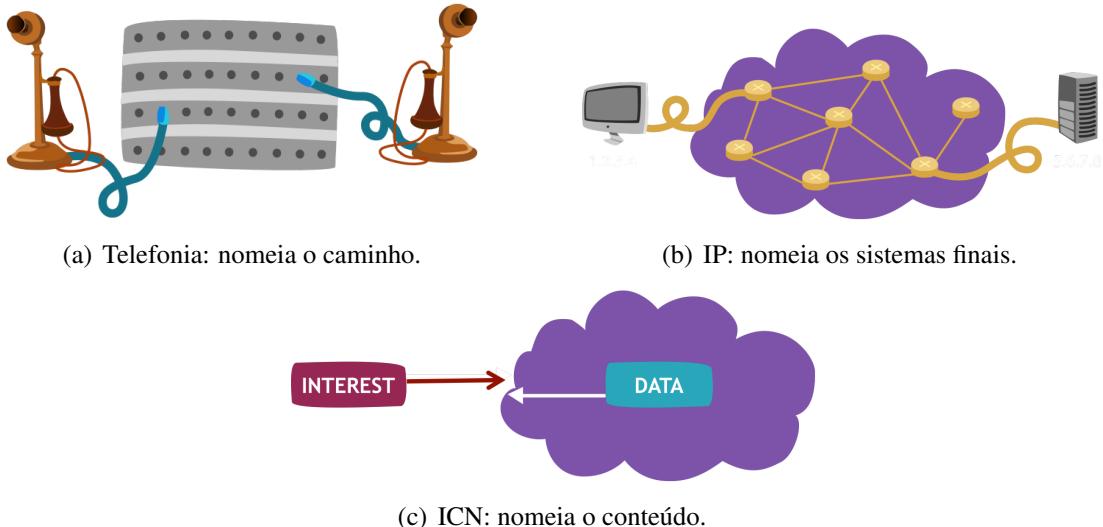


Figura 3.10: Evolução das abstrações das redes de comunicação. Figuras de Van Jacobson.

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 [225]. 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, dar suporte à distribuição de mídias ao vivo usando ICN requer melhorias no projeto arquitetural, como o desenvolvimento de 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 *Content-Centric Networks* (CCN) [191, 193, 226], ao passo que outras propostas ainda estão em estágio embrionário [48–50, 192, 227], como discutiu-se no início deste capítulo (Seção 3.1). Como a CCN apresenta diferenças substanciais em seu modelo de

serviço para distribuir o conteúdo de interesse, com base em funções disponíveis no núcleo da rede, a seguir, detalham-se as atuais aplicações no contexto de distribuição de mídias ao vivo.

Na arquitetura CCN, nomeiam-se os pacotes de conteúdos e defini-se 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 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.

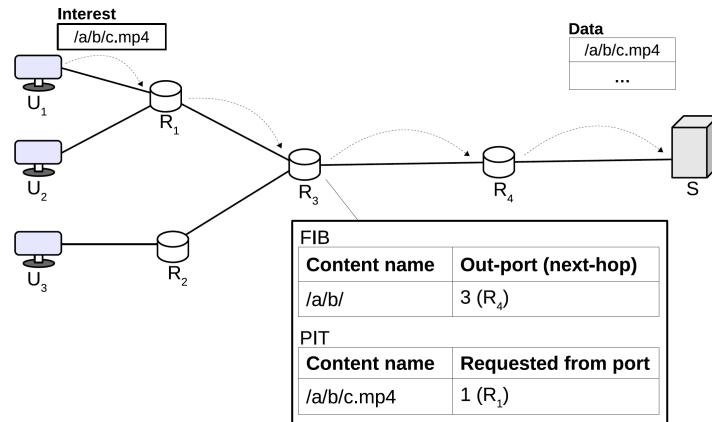


Figura 3.11: Um exemplo de interação da *Content-Centric Network* (CCN).  $U_1$  envia um pacote de interesse por */a/b/c.mp4*, que está localizado em  $S$ . Com as setas, ilustra-se a direção de propagação do pacote de interesse. O pacote de dados segue o caminho inverso. Observam-se também as tabelas FIB e PIT para o roteador  $R_3$ . Figura extraída de [225].

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* que contém os pacotes de dados que já foram repassados pelo respectivo roteador. Ao receber um pacote de interesse, o roteador verifica a existência do referido conteúdo em seu *cache*. 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. Se o pacote de dados ainda não estiver disponível, aguarda-se uma resposta contendo os dados requisitados e, ao recebê-la, deve-se sempre verificar se há uma entrada correspondente na tabela PIT.
2. *Pending Interest Table (PIT)*: o roteador adiciona um pacote de interesse na tabela PIT e o roteia através de uma ou mais interfaces determinadas na FIB, discutida no próximo item. Em seguida, o roteador monitora suas interfaces de entrada verificando a recepção de pacote de dados correspondente ao pacote de interesse transmitido 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 um roteador receber um pacote de dados que não tem uma correspondência para o pacote de interesse na PIT, o pacote simplesmente é descartado. Se anteriormente o roteador estiver recebido múltiplas requisições em diferentes interfaces referente ao pacote de dados que acabara de receber, duplica-se o pacote de dados e o transmite através das interfaces correspondentes. Para isto, o roteador agrupa pacotes de interesses para o mesmo conteúdo e somente o primeiro é repassado para uma ou mais interfaces de acordo com a FIB.
3. *Forwarding Information Base (FIB)*: é uma tabela que associa os prefixos dos nomes de um conteúdo a uma ou mais interfaces de saída. 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 os roteadores correspondentes com essa verificação.

Sendo assim, o CS funciona como um *cache*, enquanto que os pacotes de interesse são repassados de acordo com informações na FIB e os pacotes de dados de acordo com as informações da PIT. Os pacotes de interesse e de dados trafegam através dos nós da rede, portanto múltiplas avaliações em estruturas de dados são realizadas. 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 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 um método de *checksum*, o nó receptor deve retransmitir 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, controlam-se as taxas em que os pacotes de interesse são transmitidos, utilizando-se um modelo baseado em janela deslizante, similar ao utilizado no TCP [228].

Portanto, o princípio da CCN é que uma rede de comunicação deve viabilizar o acesso ao conteúdo de interesse de um usuário (*content-centric approach*), independente da localização física do conteúdo, enfraquecendo a abordagem restrita de organização dos protocolos em camadas funcionais (*host-centric approach*). Sendo assim, substitui-se o esquema de acesso a um conteúdo por meio de endereço IP/porta pelo seu nome, pois se considera que entregar o conteúdo é mais importante do que a forma de obtê-lo [191].

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

O suporte de sistemas de distribuição de mídias ao vivo em CCN tem atraído o interesse da comunidade científica desde suas primeiras versões [126, 193, 229–233]. Em CCN, em vez de um 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 ccn 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 CCN 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 [225].

Atualmente, os trabalhos voltados para transmissão de mídias com base em CCN 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 clientes. O primeiro trabalho de transmissão de mídias ao vivo foi apresentado por Van Jacobson em uma aplicação chamada VoCCN [232]. 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 [229], 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 CCN em comparação do HTTP 1.0 e o 1.1 em rede TCP/IP. Com base nos resultados obtidos, concluiu-se que a CCN 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 CCN. 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 CCN.

No trabalho [234], 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 CCN, 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 [233] também se assemelha ao anterior no que diz respeito ao estudo de CCN 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 CCN.

Já em [193], 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 [126], 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 até 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 mídia ao vivo, pois estas são transientes. 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 o CCN e o GMTP para distribuição de mídias ao vivo:**

O CCN-TV, atualmente o principal representante para distribuição de mídias ao vivo em redes CCN, foi um dos projetos escolhidos para um estudo comparativo de desempenho frente ao GMTP. Por este motivo, reservou-se uma discussão mais detalhada sobre este assunto no Capítulo 5.

## **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. Destacaram-se as propostas concorrentes, contrapondo-se as limitações de cada uma frente ao GMTP, as quais foram individualmente discutidas. Por apresentarem mais semelhança quanto aos aspectos arquiteturais e modelo de serviço para distribuição de mídias ao vivo, reservou-se o Capítulo 5 para uma discussão mais acen-tuada sobre as limitações do Denacast/CoolStreaming e do CCN-TV, considerando-se um foco comparativo no projeto e no desempenho do GMTP. A seguir, descreve-se o protocolo GMTP.

## Capítulo 4

# Global Media Transmission Protocol (GMTP)

O *Global Media Transmission Protocol* (GMTP) atua nas camadas de transporte e de rede (*cross-layer*) da pilha TCP/IP, projetado para operar na Internet em sistemas de distribuição de mídias ao vivo. Trata-se de um protocolo baseado em uma arquitetura de rede híbrida P2P/CDN, através da qual se transmitem pacotes de dados de um ou mais sistemas. Para isto, constituem-se redes de favores formadas por roteadores de rede, que cooperam entre si a fim de entregarem o conteúdo multimídia de interesse comum aos seus clientes. Nesse cenário, os servidores atuam como super nós para os nós da rede P2P, auxiliando-os no envio e recebimento dos fluxos de dados. Uma aplicação cliente reproduz o conteúdo multimídia ao usuário final à medida que recebe pacotes de dados contendo partes da mídia, geradas por um ou mais servidores. Os clientes não, necessariamente, recebem os pacotes de dados diretamente dos servidores, mas podem receberê-los de roteadores já envolvidos na transmissão da mídia. Nesse contexto, os servidores instruem os roteadores a efetivarem as parcerias com outros roteadores que possuem clientes também interessados no mesmo conteúdo multimídia, o que ocorre com base no conhecimento sobre as rotas já utilizadas para disseminar a referida mídia.

As trocas de dados entre os nós GMTP ocorrem por meio do envio e recebimento de partes de uma mídia (*chunks*), transmitidas por diferentes nós da rede. Os roteadores GMTP transmitem os fluxos de dados para outros roteadores em modo *unicast*, ao passo que distribuem os pacotes de dados a seus clientes em modo *multicast*, caracterizando um modo

de transmissão *multi-unicast*. Em ambos os modos de transmissão, o GMTP realiza controle de congestionamento em transmissões sem garantia de entrega e a escolha do modo de transmissão para disseminar um fluxo de dados ocorre sem a influência da aplicação. Nesse contexto, a aplicação precisa simplesmente “sintonizar” sua conexão em um determinado canal *multicast* configurado automaticamente pelo roteador, de acordo com a demanda de seus clientes. 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, facilitando-se seu uso nos atuais e futuros sistemas, onde os clientes se tornam compatíveis para reproduzir um conteúdo independente do servidor que o transmite, tal como acontece no serviço Web.

Com o uso do GMTP, o plano é permitir o estabelecimento de conexões e a cooperação entre diferentes fornecedores de aplicações, obtendo-se os *chunks* a serem reproduzidos nos sistemas finais tão logo quanto possível. Isto significa que o GMTP torna as aplicações interoperáveis, mesmo entre diferentes fornecedores, uma vez que o protocolo desacopla a forma como se transportam os dados da aplicação da forma como se deve exibi-los aos usuários finais (uma decisão meramente de aplicação). Sendo assim, promove-se a integração do GMTP em aplicações existentes, quando se consideram 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 no GMTP, a fim de construir uma rede de favores formada por roteadores, pela execução de quatro grandes etapas:

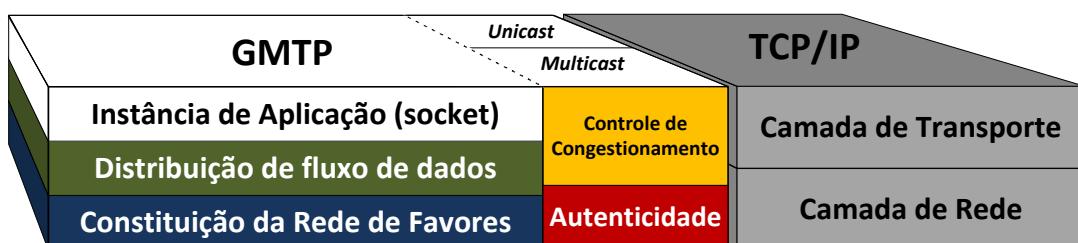


Figura 4.1: Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP.

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 clientes interessados em receber um fluxo de dados de um evento ao vivo, bem como transmitir tal fluxo de dados através da rede de favores 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 favores entre roteadores.
4. *Autenticidade do conteúdo*: verificar a autenticidade do fluxo de dados antes de repassá-los aos clientes, evitando-se ataques de poluição.

Com base nessas etapas, 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 o algoritmo de compartilhamento dados às 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; as influências de tais algoritmos no processo de formação de parcerias; e discussões sobre a escolha do RCP em detrimento ao XCP, o qual foi adaptado para suportar o conceito de sub-fluxo.
- 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 relatores para o funcionamento do algoritmo de controle de congestionamento *multicast*.
- E, por fim, na Seção 4.8, apresenta-se o sumário deste capítulo, elencando-se 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.

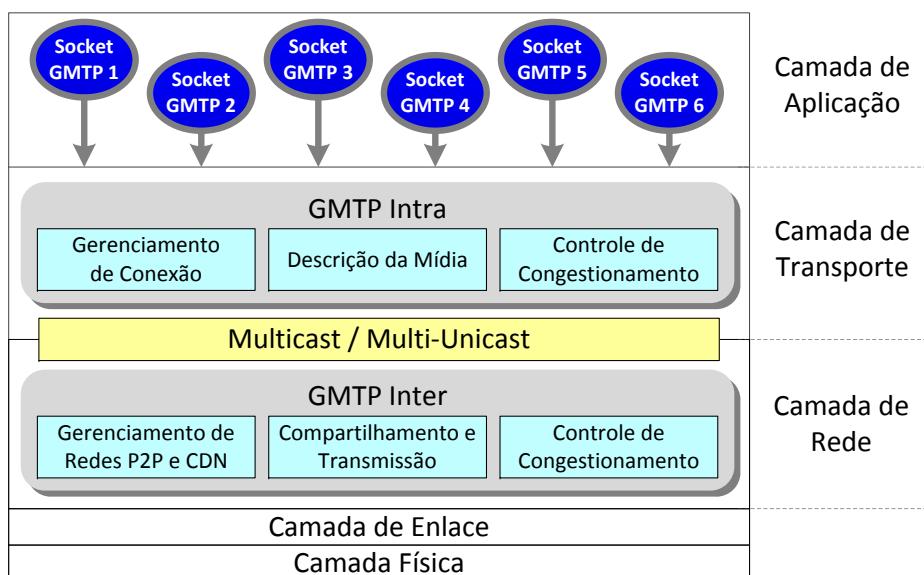


Figura 4.2: Arquitetura do Protocolo GMTP.

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 comprehende a instância do GMTP em execução no sistema operacional do cliente, acessível através de uma

API de *socket* GMTP. Nesse contexto, um *socket* é a representação de uma instância do protocolo GMTP em execução, sendo responsável por gerenciar todas as atividades de comunicação da aplicação correspondentes 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 de datagramas, eleição de 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 obtenha acesso aos seus valores.

- **GMTP-Inter:** constituir redes de favores P2P compostas 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 a fim de constituírem as redes de favores, com base no interesse comum de um determinado conteúdo, aceitando conexões originadas por seus clientes, bem como instruções dos servidores sobre formar parcerias com outros roteadores a fim de disseminar um determinado conteúdo. No contexto de uma conexão, o GMTP-Inter mantém variáveis de estado relacionadas às funções de sua responsabilidade, tais como estabelecimento de conexão com servidores ou entre roteadores; seleção e gerenciamento de roteadores parceiros; eleição de relatores; compartilhamento de fluxos multimídia; e controle de congestionamento assistido pela rede. Além disso, permite-se a definição de parâmetros iniciais de configuração da rede de favores e da integração com servidores de uma ou mais CDNs, como ilustra-se na Figura 4.3. Nesse caso, o usuário administrador de um roteador pode definir os seguintes parâmetros:

- configurações sobre registro de participação em uma ou mais redes CDNs;
- largura de banda máxima (*download* e *upload*) que o roteador está autorizado à compartilhar;
- o período que o roteador está autorizado a participar das redes de favores, em dias e horários;
- quantidade máxima de parcerias que podem ser realizadas e quantidade máxima de fluxos de dados que podem ser compartilhados;

- parâmetros das equações utilizadas para executar os algoritmos de controle de congestionamento; e
- tamanho máximo do *cache* de mídia a ser compartilhado;
- configurações acerca dos certificados digitais disponibilizados pelos servidores, tais como *download* automático e realização de *cache*.

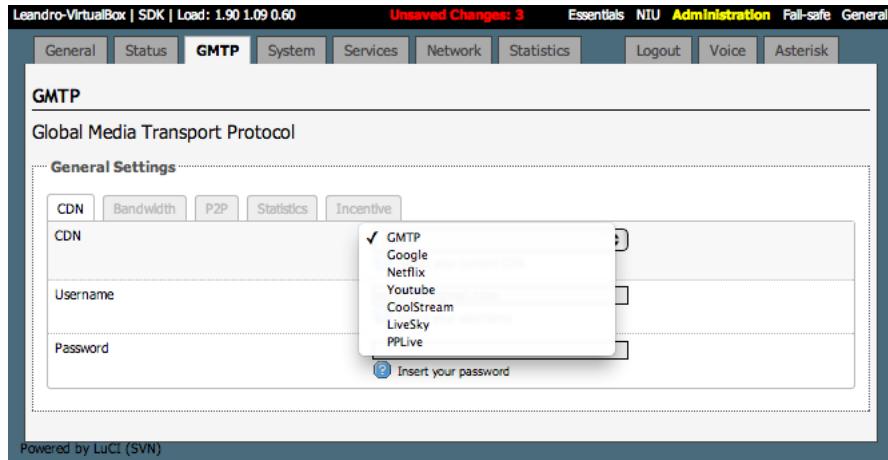


Figura 4.3: Tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP. Nessa tela, permite-se que o administrador do roteador configure parâmetros do módulo GMTP-Inter.

Para viabilizar a disseminação de conteúdos multimídia, cada nó roteador, localizado no caminho entre o servidor da mídia e o cliente interessado em obtê-la, pode repassar os pacotes de dados para seus clientes de acesso direto (um salto) e replicá-los para outros roteadores interessados em recebê-los, 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-se múltiplas conexões ao servidor.

Em um contexto geral, como ilustra-se na Figura 4.4, definem-se 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 em nível de sistema operacional, representando uma aplicação manipulada pelo usuário final. Um *Cliente GMTP* recebe os pacotes de dados e os entrega ao processo de aplicação em execução,

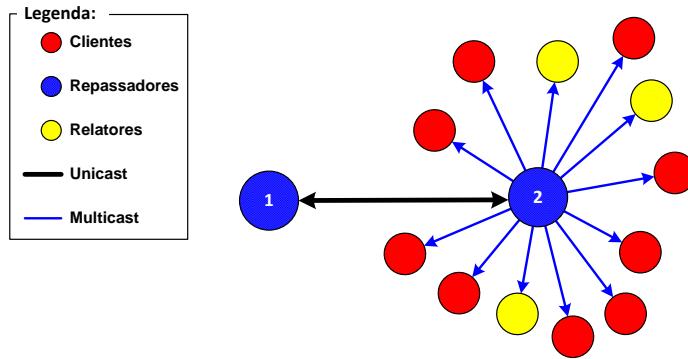


Figura 4.4: Tipos de nós e modos de conexões do GMTP.

sendo que alguns destes clientes, chamados de *Relatores GMTP*, auxiliam na execução do algoritmo de controle de congestionamento em transmissões *multicast*, como descreve-se a seguir.

- *Relator GMTP*: é um *Cliente GMTP* com habilidades de enviar relatórios periódicos ao nó *Repassador GMTP* sobre o estado da transmissão.
- *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 *Repassadores GMTP* até que os pacotes de dados alcancem os *Clientes GMTP*.
- *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 geradora 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 *Repassadores GMTP* que, ao receberem uma resposta correspondente à sua requisição, atendem à demanda de um ou mais *Clientes GMTP*.

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. 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 interfaces 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 [235].

### Fluxo básico de comunicação

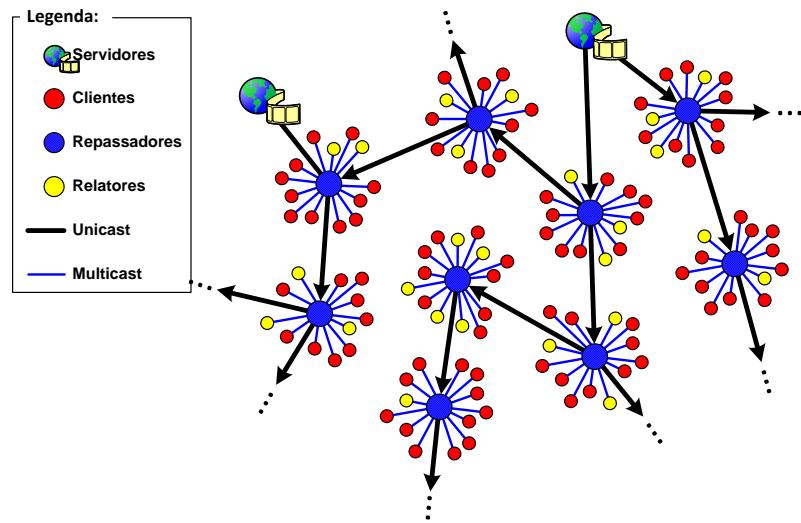


Figura 4.5: Rede de sobreposição construída dinamicamente pelo GMTP com a presença de repassadores e relatores.

Com base na Figura 4.4, observa-se o cenário geral de atuação do protocolo GMTP, onde ilustram-se os clientes interessados em obter o conteúdo de um determinado evento ao vivo. Neste caso, observa-se também um servidor, que está conectado a uma rede CDN e atua como fonte geradora de dados, ao passo que os clientes se conectam aos repassadores. Note que as transmissões entre os repassadores ocorrem em modo *unicast*, o que evita o uso e limitações existentes em protocolos *multicast* tradicionais, como o *Internet Group Management Protocol – (IGMP)* [236, 237].

Como ilustra-se no fluxograma da Figura 4.6, o fluxo de comunicação do GMTP ocorre quando um cliente deseja reproduzir um conteúdo multimídia. Inicia-se a comunicação com

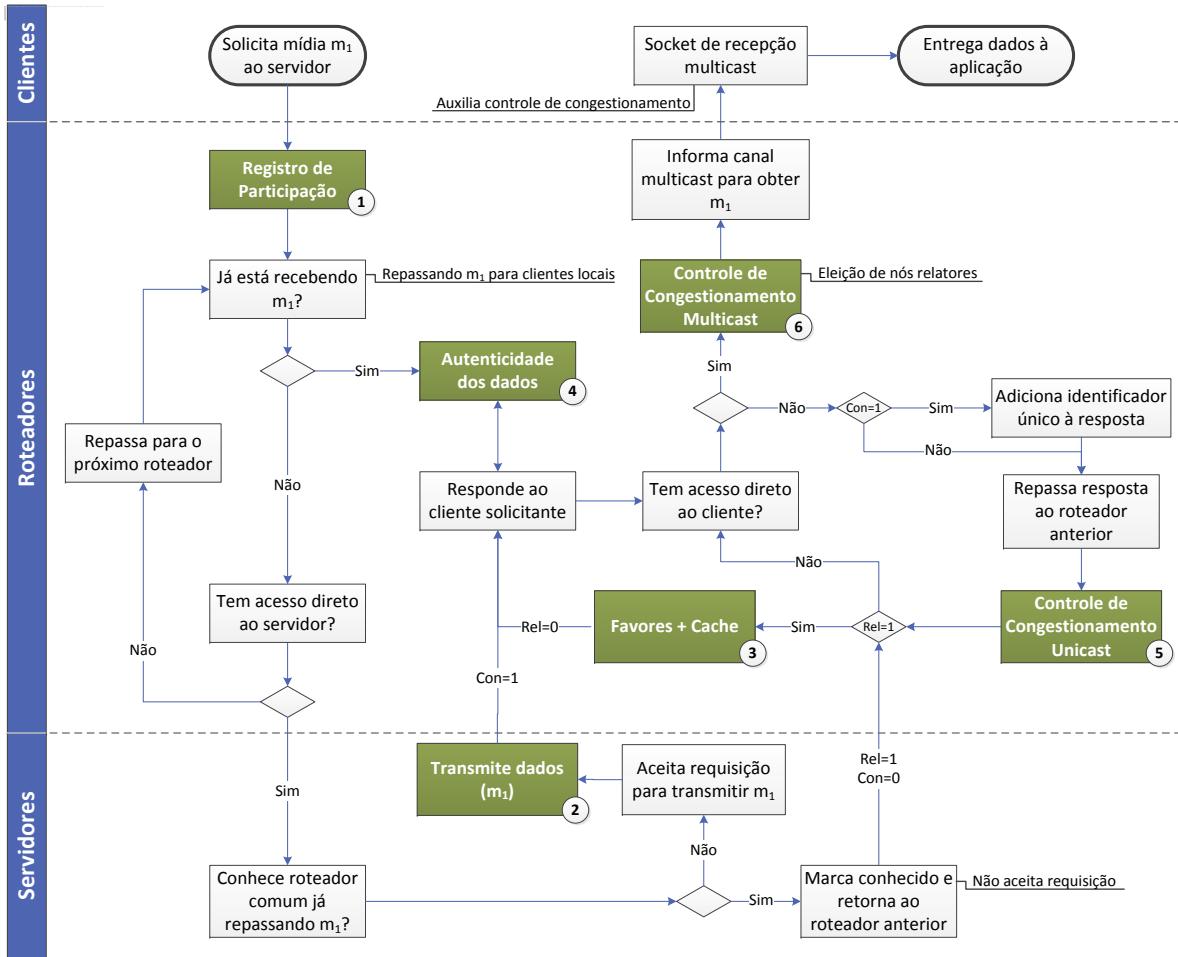


Figura 4.6: Fluxograma geral do GMTP, desde a solicitação do cliente por uma mídia ao vivo até receber uma resposta, executando-se ações importantes como registro de participação, controle de congestionamento, eleição de relatores e verificação de autenticidade dos dados.

um cliente enviando uma requisição em direção ao servidor que está transmitindo o conteúdo multimídia de interesse, como atualmente acontece em qualquer conexão na Internet. Em seguida, um repassador intercepta a requisição durante seu trajeto até o servidor, que é capaz de determinar os melhores parceiros para atendê-la. Em geral, isto ocorre já no roteador de borda do cliente, que funciona como repassador de origem, em um procedimento chamado de registro de participação (Passo 1 do fluxograma da Figura 4.6). Caso o repassador não encontre nenhum repassador parceiro capaz de transmitir a mídia de interesse, a mensagem de requisição alcança o servidor que transmite a mídia correspondente, já que o pedido de conexão é intencionalmente endereçado a este. Em seguida, o servidor decide se aceita o pedido de conexão ou se delega tal requisição a um repassador, com base no conhecimento de quais repassadores já estão encaminhando o conteúdo de interessado para

alguma rede. No primeiro caso, o servidor simplesmente responde ao cliente e inicia a transmissão do fluxo de dados correspondente, o que resulta em conhecer a nova rota utilizada para transmitir o conteúdo, definida pelos roteadores existentes do servidor até ao cliente, nessa direção e ordem (Passo 2). No segundo caso, em vez de aceitar uma nova conexão, o servidor a recusa e determina que um repassador comece a servir o repassador (de origem) do cliente, determinando-se uma parceria entre um repassador já em uso para disseminar o conteúdo de interesse e o repassador que solicitou o registro de participação (Passo 3). Em ambos os casos, sempre o repassador de origem assumirá o controle de uma requisição do cliente, habilitando-se como candidato a parceiro para outros repassadores, quando motivados por requisições originadas pelos seus respectivos clientes. Nesse ínterim, outros passos são executados, como distribuição dos pacotes de dados nas redes locais e verificação da autenticidade do conteúdo no momento do repasse (Passo 4). Especificamente, a verificação da autenticidade se baseia na assinatura digital dos dados transportados nos pacotes e na capacidade que os repassadores têm de validar se o conteúdo foi intencionalmente alterado por usuários maliciosos no trajeto até o cliente, com base no certificado digital emitido pelo servidor – esta ação é opcional e decidida pela aplicação no início da transmissão.

Por fim, nos Passos 5 e 6, executam-se os algoritmos de controle de congestionamento. No Passo 5, executa-se um algoritmo de controle de congestionamento em cada repassador, que calcula sua capacidade de transmissão atual com base apenas no tamanho da fila de repasse e no atraso fim-a-fim marcado em pacote de dados a ser roteado. Em seguida, o repassador compara sua capacidade de transmissão com a menor capacidade de transmissão de todos os repassadores anteriores, transportada em cada pacote de dados. Se sua capacidade de transmissão for menor que a capacidade de transmissão informada no pacote de dados, o repassador o altera informando sua capacidade de transmissão, caso contrário, transmite-o para o próximo repassador. Ao final desse processo, o servidor regulará sua taxa de transmissão com base no repassador com menor capacidade de transmissão, informação contida nos pacotes de dados transmitidos ao servidor. Essa estratégia foi baseada no protocolo RCP [58], porém adaptada no GMTP para suportar o conceito de sub-fluxos, discutido mais adiante neste capítulo. O importante é que além de regular sua taxa de transmissão, o servidor também sugere que os repassadores realizem parcerias entre si com base na capacidade de transmissão de cada repassador, propondo-se portanto um mecanismo explícito de

seleção de nós com base na capacidade de transmissão das rotas utilizadas para distribuir o conteúdo. Já no Passo 6, executa-se um algoritmo de controle de congestionamento adaptado do *TCP-Friendly Rate Control* (TFRC) [169] para fluxo de dados *multicast*, regulando-se a taxa de transmissão na rede local com base nos relatórios transmitidos pelos relatores.

O posicionamento dos repassadores e suas habilidades permitem a redução do número de fluxos de dados na rede correspondentes a um mesmo evento, ao tempo que maximiza a quantidade de clientes interessados em receber o mesmo fluxo (escalabilidade). Por este mesmo motivo, o protocolo GMTP é flexível para permitir que um repassador atue somente encaminhando conteúdos multimídia entre duas ou mais redes distintas, mesmo que este não tenha demandas explícitas dos seus 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 disponíveis de rede. Isto pode ocorrer sem a necessidade de manter um determinado computador ligado e conectado à rede, bastando apenas manter o roteador ligado, diferentemente de outras soluções existentes baseadas em arquitetura P2P ou P2P/CDN, que requer pelo menos um cliente em execução e registrado pela aplicação.

As requisições de conexão podem ser originadas não apenas por clientes para seu respectivo repassador, mas também estas podem ocorrer entre repassadores que, motivados pelos interesses dos seus clientes, formam parcerias entre si. Isto significa que um repassador pode agir como se fosse um servidor, respondendo às requisições originadas por seus clientes ou por outros repassadores, reduzindo o número de conexões nos servidores.

#### 4.1.1 Resumo das principais características

Com base no fluxograma ilustrado na Figura 4.6, enumeram-se as principais características do GMTP com base nos processos macros, especialmente os marcados na cor verde, os quais serão detalhados nas seções subsequentes deste capítulo.

1. Constituição de redes de favores entre roteadores, transparente para a aplicação. Nesse contexto, realiza-se o Registro de participação de um repassador em um servidor. Isto permite que um 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 oferecida aos sistemas finais. Estes assuntos serão retomados na Seção 4.3.

2. 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 servidor. O acesso a uma transmissão de um evento ao vivo ocorre através de um processo de conexão em três-vias (*3WHS*), com a requisição de conexão transmitida ao servidor e podendo ser interceptada por um repassador em seu trajeto até o servidor. Estes assuntos serão retomados na Seção 4.4.
3. Controle de congestionamento para transmissões *unicast* com base na menor largura de banda de um roteador em uma rota entre o cliente e o servidor (GMTP-UCC); e Controle de Congestionamento para transmissões *multicast* com base em relatórios transmitidos pelos clientes (GMTP-MCC). Estes assuntos serão retomados na Seção 4.5.
4. Descoberta de nós parceiros entre redes distintas e negociação de parcerias, com suporte à formação de parcerias baseadas em métricas de rede, tal como a capacidade de transmissão fim-a-fim. Além disso, permite-se distribuir um fluxo de dados em múltiplas taxas de transmissão de acordo com a largura de banda dos repassadores. Para isto, o GMTP segmenta os canais de transmissão (rota entre um servidor e os repassadores) quando existem múltiplos repassadores em uma determinada rota e estes estão interessados em receber o mesmo fluxo de dados. Estes assuntos serão retomados nas Seções 4.3 e 4.5.
5. Verificação de autenticidade dos dados multimídia, por meio do uso de assinaturas digitais disponibilizadas pelos servidores, impedindo assim ataques de poluição de conteúdo. Este assunto será retomado na Seção 4.6.
6. Suporte a funções auxiliares, como eleição de relatores e tolerância à desconexões de nós. Estes assuntos serão retomados na Seção 4.7.

### 4.1.2 Cabeçalho geral e tipos de pacotes

Toda comunicação entre dois ou mais nós GMTP ocorre através da troca de datagramas IP, os quais carregam sinalizações de controle e/ou dados da aplicação. É muito importante entender os conceitos básicos de cada campo do cabeçalho, a fim de compreender os algoritmos empregados no GMTP. Para adoção na Internet, será necessário registrar junto à *Internet Assigned Numbers Authority – IANA*<sup>1</sup> o uso de um código para o campo *Protocolo* do cabeçalho IP referente ao GMTP. Atualmente, está sendo discutida a alocação do código 100 para o GMTP, como pode-se constatar no documento *Protocol Numbers* da IANA, disponível em <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

O cabeçalho do GMTP foi organizado estrategicamente em uma parte fixa e em uma parte variável. A parte fixa é definida em 52 Bytes, como ilustra-se na Figura 4.7.

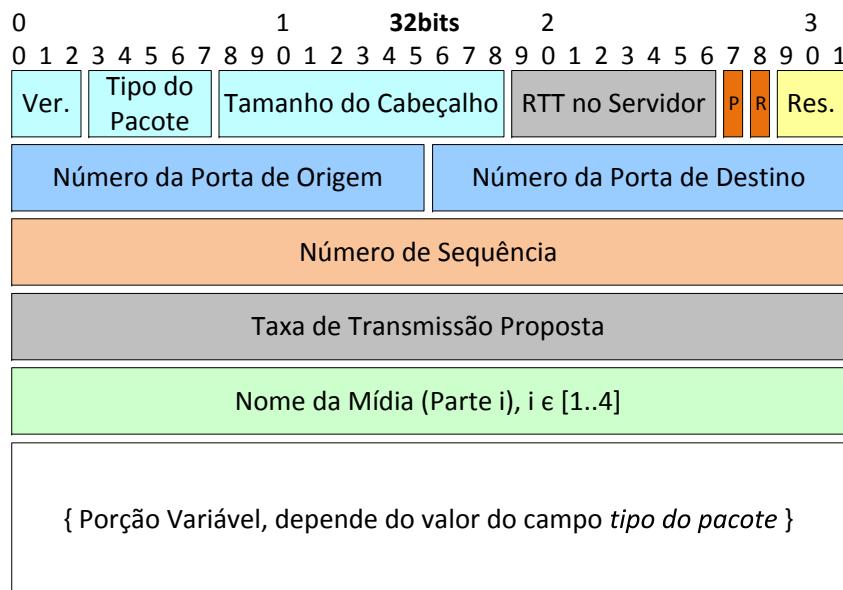


Figura 4.7: Porção fixa do cabeçalho de pacotes GMTP.

A distribuição ocorre da seguinte forma: 3 bits para a *Versão do Protocolo*, 5 bits para o *Tipo de Pacote*, 8 bits para o *Tamanho do Cabeçalho*, 8 bits para o *RTT no Servidor*, 32 bits para os *Números das Portas de Origem e Destino* (16 bits para cada campo), 32 bits para a *Taxa Proposta de Transmissão*, 64 bits para os *Números de Sequência* e de *Reconhecimento* (32 bits para cada campo) e 256 bits para o *Nome do Fluxo de Dados*. É importante entender

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

que os campos *RTT no servidor* e *taxa de transmissão* estão relacionados com o mecanismo de controle de congestionamento adotado no GMTP, ao passo que o campo *nome do fluxo de dados* é uma forma de identificar unicamente a transmissão de um evento ao vivo e, com base nessa informação, formam-se as redes de favores entre os repassadores.

Apesar de parecer uma ordenação simples e aleatória, sem qualquer pretensão além do aspecto organizacional, tanto a ordem quanto o tamanho de cada campo da porção fixa do cabeçalho GMTP foram definidos criteriosamente para i) permitir manutenção (adição, atualização e remoção) das funcionalidades existentes; ii) flexibilizar o uso do protocolo por diferentes tipos de aplicação multimídia e iii) otimizar a leitura e escrita dos campos de cada pacote, buscando-se reduzir o atraso nodal. Por exemplo, o campo *versão* permite uma implantação gradativa do protocolo GMTP, bem como atualizações para novas versões sem, impactar diretamente no funcionamento das versões existentes. Por isso, é importante que o próximo campo seja o *tipo do pacote*, pois dependendo do seu valor, um nó interpretará o restante do cabeçalho (parte variável) de forma diferente. Já o campo *Tamanho do Cabeçalho* ainda favorece a flexibilidade da parte variável do cabeçalho e seu espaço de 11 bits permite informar um cabeçalho com tamanho total de 2047 Bytes.

A estratégia de usar cabeçalho de tamanho variável é flexibilizar as funções do protocolo e transportar dados de controle apenas quando estritamente necessário. Por exemplo, no GMTP permite-se que uma aplicação injete informações na porção variável do cabeçalho para que sejam lidas pelos repassadores localizados entre os sistemas finais. Um exemplo disso é que um servidor pode sinalizar a um repassador que realize parcerias com outros repassadores que já estejam distribuindo um determinado conteúdo. Um outro exemplo é no processo de conexão do GMTP, onde cada repassador no caminho entre o cliente e o servidor adiciona seu identificador no cabeçalho de um pacote GMTP, o que permite aos servidores conhecerem os caminhos sendo utilizados para distribuir a mídia até seus clientes. Conhecendo-se os caminhos pode-se sugerir parcerias de melhor qualidade.

O campo *tipo do pacote* determina quais informações serão encapsuladas em cada tipo de pacote para, posteriormente, serem consumidas pelos nós da rede (clientes, servidores, repassadores e relatores), a fim de executarem as ações definidas no protocolo GMTP e disseminarem os pacotes de dados de um fluxo. Essas ações foram traduzidas em algoritmos, os quais serão detalhados ao longo deste capítulo. Sendo assim, para finalizar este assunto

sobre cabeçalho GMTP, a seguir, apresentam-se brevemente os tipos de pacote utilizados no GMTP.

0. *GMTP-Request*: o cliente envia requisição para obter um fluxo de dados multimídia, com base no nome do fluxo de interesse;
1. *GMTP-RequestNotify*: o 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 repassador confirma o estabelecimento de uma parceria com outro repassador, dado um determinado fluxo de dados;
3. *GMTP-Register*: o repassador registra participação no servidor para funcionar como distribuidor de um fluxo de dados;
4. *GMTP-Register-Reply*: o servidor responde sobre o pedido de registro de participação enviado por um repassador. Além disso, transporta o identificador de todos os repassadores entre o servidor e o cliente;
5. *GMTP-Route-Notify*: contém um caminho (rota) entre o repassador e um servidor. O repassador envia esse tipo de pacote ao servidor, após receber o pacote do tipo *GMTP-Register-Reply*;
6. *GMTP-RelayQuery*: o repassador pode solicitar ao servidor uma lista de possíveis repassadores parceiros;
7. *GMTP-RelayQuery-Reply*: o servidor envia uma resposta ao 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*: em geral, qualquer nó utiliza esse tipo de pacote para confirmar a recepção de um determinado pacote;

10. *GMTP-DataAck*: combinação dos pacotes GMTP-Data e GMTP-Ack (*PiggyBack*);
11. *GMTP-MediaDesc*: descrever informações sobre a mídia sendo transmitida em um determinado fluxo de dados (conexão). Este pacote é gerado pelo servidor e pode ser processado e/ou distribuído pelos repassadores;
12. *GMTP-DataPull-Request*: o repassador envia um pedido para obter o mapa de *buffer* atual de um outro repassador parceiro;
13. *GMTP-DataPull-Response*: resposta ao pedido para obtenção de um mapa de *buffer*;
14. *GMTP-Elect-Request*: o repassador envia para um cliente uma solicitação para que este atue como relator;
15. *GMTP-Elect-Response*: o cliente envia ao repassador uma confirmação de que aceita atuar como relator;
16. *GMTP-Close*: os servidores, repassadores ou clientes solicitam o término de uma conexão;
17. *GMTP-Reset*: determina, incondicionalmente, a finalização de uma conexão;
18. *Reservado*: deste até o identificador 31, tratam-se de valores reservados para uso futuro e os pacotes com esses valores devem ser descartados pelos nós que o processam.

No Apêndice B, apresentam-se detalhes das porções variáveis de cada tipo de pacotes do GMTP, sendo seu teor bastante técnico e portanto dedicado aos leitores interessados em sua implementação.

## 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 [238–241].

1. Seja o conjunto finito dos 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  $r_d \in R$  sobre os roteadores em  $B$ .
3. Seja o conjunto finito dos 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 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  $\varepsilon$ , 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, sendo tal conjunto denominado caminho e representado por  $W$ , por onde se transmitem os pacotes de dados  $p_h \in \mathbb{P}$ , 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  $\varepsilon$ , definida por  $\mathcal{T} = \{Z, P, C_i\}$ , tal que:
    - i. Seja  $P \subset \mathbb{P}$ , o conjunto *parcialmente ordenado (poset)* dos pacotes de dados  $p_x \in P$ , 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 todos os pacotes de dados  $p_h$ ;
    - ii. Seja  $C_i$ , uma função que denota os  $c_f$  que estão conectados (uma relação) 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 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)  $W = \bigcup_{v=1}^j W_v$ , denota todos os caminhos conhecidos por  $s_a$  para distribuir  $P$ , onde  $j \in \mathbb{N}$  e corresponde à quantidade de todos os possíveis caminhos *to set*  $(W_v, \prec)$ . Nesse caso,  $W_v$  denota um dos possíveis caminhos por onde um fluxo de dados  $P$  pode ser transmitido, obrigatoriamente a partir de um nó  $s_a$  até a um nó  $r_1$ , portanto, tem-se 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 nós  $b_e \in B$  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  $W_v^\circ$  e se  $W_v \subset T$ , tal que  $T \subset Z$  e 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  $T = \{t_u \mid \varphi(t_u, P) = 1\}$ , tal que  $u \in \mathbb{N}$  e que:
    - A.  $\varphi$  é 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 *to set*, 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  $(W_v, \prec) = \{t_{u+2}, t_{u+1}, t_u, t_{u-1}, t_{u-2}, \dots, t_2, t_1\}$ ,  $\delta(t_u, W_v) = W_v^\triangleleft = \{t_u, t_{u-1}, t_{u-2}, \dots, t_2, t_1\}$ .
- (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 à 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 incapazes 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  $s_a$ . 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. Em seguida, repassam-se os sinais 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 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 repassador não tiver 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. Em geral, os nós  $r_d$  realizam o registro de participação previamente.

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  $s_a$ . 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 até alcançar  $r_d$ , organizando-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 também solicitarem  $P$ , após o primeiro

repassador pedir, bastando apenas que os próximos nós  $c_f$  “sintonizem” suas interfaces 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  $r_d$ , resumem-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, resumem-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* [242].

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  $r_q$ .

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 [243], discute-se sobre outros aspectos de expiração no tempo que podem ser adaptados para o GMTP.

**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( $P, \text{false}$ );
2 if  $p_x = \text{OK}$  then                                /*  $s_a$  confirmed registration */
3    $W_v \leftarrow \text{getPacketFieldValue}(p_x, \text{'way'})$ ;    /* Gets  $W_v$  in  $p_x$  */
4    $s_a \leftarrow \text{getPacketFieldValue}(p_x, \text{'server'})$ ;    /* Gets  $s_a$  in  $p_x$  */
5    $P \leftarrow \text{getPacketFieldValue}(p_x, \text{'flow'})$ ;        /* 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    $errorCode \leftarrow \text{getPacketFieldValue}(p_x, \text{'error'})$ ;
28   respondToClients(GMTPRequestReply( $errorCode, P$ ));
29
30 end

```

---

**Algoritmo para gerar o identificador de um nó  $r_d$** 

É muito importante entender a motivação e como se define um identificador de um nó  $r_d$ . Como discutiu-se anteriormente, um nó  $s_a$  passa a conhecer um caminho até o cliente, porque cada nó  $r_d$  deve informar seu identificador nos pacotes dos tipos *GMTP-Request-Reply* e *GMTP-Route-Notify*. O identificador deve ser único em toda a rede, pois os servidores devem ser capazes de identificar intersecções entre os repassadores já sendo utilizados para transportar os pacotes de dados da mídia. Ao conhecer as intersecções, pode-se sugerir parcerias entre repassadores com interesses comuns. Sendo assim, a questão é qual poderia ser o identificador único de um repassador. A primeira opção e a mais simples é utilizar o endereço IP do roteador. Porém, um roteador possui múltiplas interfaces de rede e dependendo da interface de saída, um endereço IP específico será utilizado. Ou seja, o endereço IP identificará a interface de rede daquele roteador e não o nó  $r_d$ . Sendo assim, não se pode utilizar o endereço IP como identificador único, caso contrário o nó  $s_a$  poderia interpretar um mesmo nó  $r_d$  como se fosse dois repassadores distintos.

Por este motivo, tomou-se a seguinte decisão: o identificador de um nó  $r_d$  é um código *hash* MD5 da concatenação dos endereços MAC (*Media Access Control*) de todas as interfaces de rede do roteador, seguido do endereço IP da Internet de saída. Sendo assim, o identificador de um nó  $r_d$  terá o tamanho de 160 *bits* para IPv4 e 256 *bits* para IPv6. Como resultado, considerando os cabeçalhos do GMTP<sup>2</sup>, do IPv4 e da camada de enlace<sup>3</sup>, em um datagrama de 1.500 *Bytes*, é possível descobrir caminhos IPv4 de até 72 saltos entre o cliente e o servidor no procedimento de registro de participação, restando 18 *Bytes*. No caso do IPv6, é possível descobrir caminhos de até 44 saltos, restando 30 *Bytes*. Os espaços restantes nos pacotes dos tipos *GMTP-Request-Reply* e *GMTP-Route-Notify* podem ser úteis para transportar outros tipos de dados de controle ou até mesmo dados de aplicação.

**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*

---

<sup>2</sup>Menos 16 *Bytes*, pois os pacotes dos tipos *GMTP-Request-Reply* e *GMTP-Route-Notify* não transportam o nome da mídia.

<sup>3</sup>Considerou-se cabeçalho Ethernet de 48 *bits*.

*Fluxos de Dados*, como ilustra-se na Figura 4.8. 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 $r_q$	Porta de Recepção de P	End. do Canal Multicast	Porta do Canal Multicast
- - - Vazia - - -						

Figura 4.8: 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 descreve-se 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 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 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, discutem-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 nó  $s_a$ , 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平衡amento 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 4.9. 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

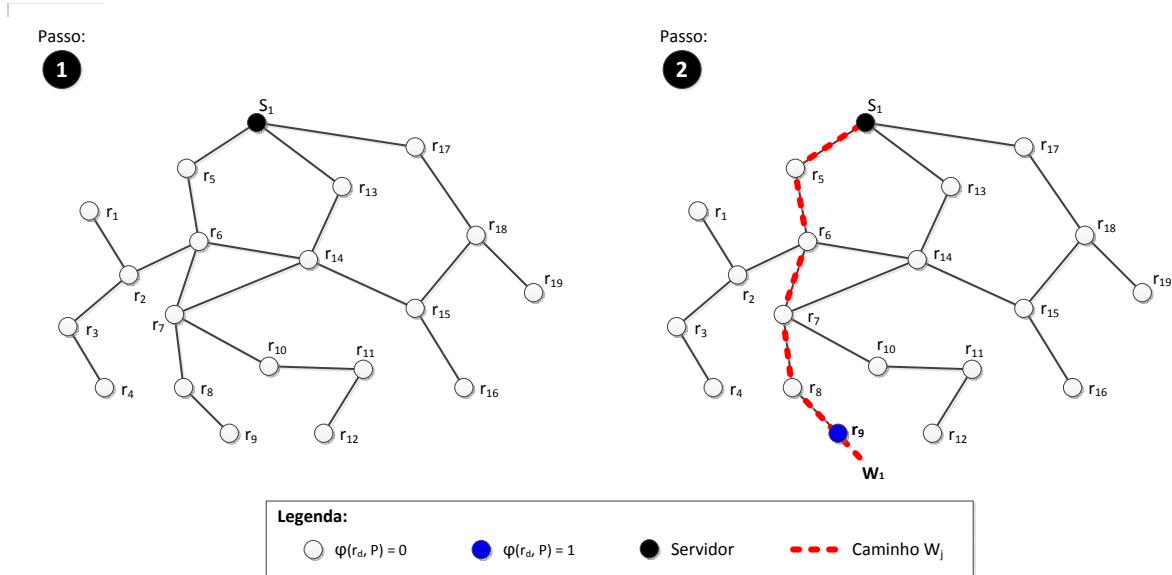
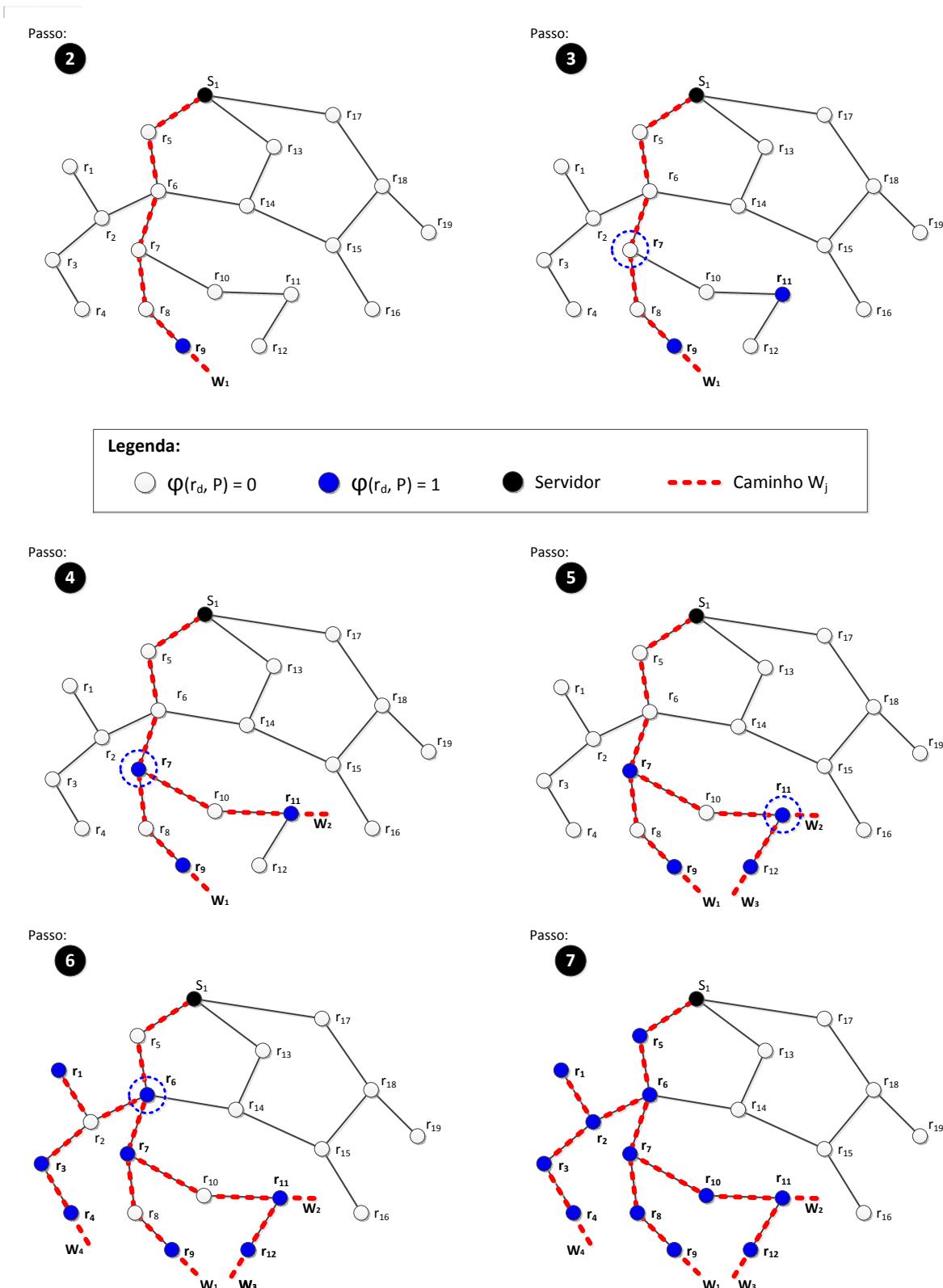


Figura 4.9: Cenário e passos para seleção de nós (exemplo 1).

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 4.10, considera-se a formação de parceria por intersecção do fluxo de dados  $P$ , a partir do Passo 2 da Figura 4.9. 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  $r_d$ , sendo desnecessidade 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 4.10 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 4.10), o nó  $s_1$  descobrirá o caminho  $W_2 = \{r_5, r_6, r_7, r_{10}, r_{11}\}$  (Passo 4). Em seguida, pela intersecção ( $W_1 \cap W_2$ ),

Figura 4.10: Cenário para seleção de nós por interseção de caminhos  $W_v$ .

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 4.10, 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 P (*pass-along*) do cabeçalho GMTP (Figura 4.7) para o pacote do tipo *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 intercepção 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 em nível de implementação, tal decisão está fora do escopo dessa discussão. No caso da implementação do GMTP realizada em simulador e utilizada neste trabalho, definiu-se que para todo registro de participação interceptado, gera-se e transmite-se uma notificação ao nó  $s_a$ .

No Algoritmo 3, resumem-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, permitem-se repasses de pacotes de dados pelo nome do fluxo de dados e não com base no nó que o produz e transmite. Em todo caso, o destino da requisição é sempre o servidor, garantindo-se que, se nenhum repassador interceptar o pedido de registro de participação, com certeza tal pedido alcançará o 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 recebem uma resposta que pode ser ou não oficialmente gerada pelo 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  $W_P \leftarrow \text{getKnownPathsOfFlow}(P)$ ;          /*  $W_P \subset W$  */
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  $W_{r_d}$ . */
```

done  $\leftarrow$  true;

break;

**end**

**end**

**if** done **then**

/\*  $s_a$  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*. \*/

$W_P[\text{length}(W_P)] \leftarrow W_{r_d}$ ;

**return** GMTPRegisterReply( $w_m$ , relay=1);

**end**

**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  $r_d$ . \*/

$W[\text{length}(W)] \leftarrow W_{r_d}$ ;

**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 sobrecarregar 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, apresentaram-se procedimentos que não são adotados em nenhum protocolo de rede pesquisado no estado da arte. Trata-se da primeira solução em que o servidor auxilia os 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 servidor. Diferentemente se comparado às soluções baseadas em 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 à 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 [244]. 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* MD5 no formato UUID (*Universally Unique Identifier*) de 128 bits [245]. 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 nó  $s_a$ , utiliza-se  $SHA1(IP_{s_a} + : + PORTA_{s_a})$ . 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  $SHA1("200.17.113.98:21200") = f8ea01fd-4d71-5d95-89ec-35646e11d7fe$ . 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 dos fluxo de dados que estão repassando para os nós  $c_f \in C_i(r_d)$  e, sendo assim, podem compartilhá-la para outros 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 alterações 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 [246]. 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 infraestrutura 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 (por exemplo, similar ao serviço Web, onde as aplicações servidoras, como o Apache, é independente das aplicações cliente, como Chrome, Firefox, etc). 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 [247]. Com o SDP, permite-se que as aplicações obtenham mais detalhes sobre a mídia, flexibilizando-se o acesso a um determinado conteúdo. Como consequência, as aplicações se preocupam apenas com a decodificação e a reprodução do conteúdo ao usuário, independente de qual sistema remoto que 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 em nível de sistema operacional. A título comparativo, considerando os protocolos baseado em HTTP e os sistemas de distribuição 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 [248] */
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 7, 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 14 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  $c_f \in C_i(r_d)$  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 4.11, 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).

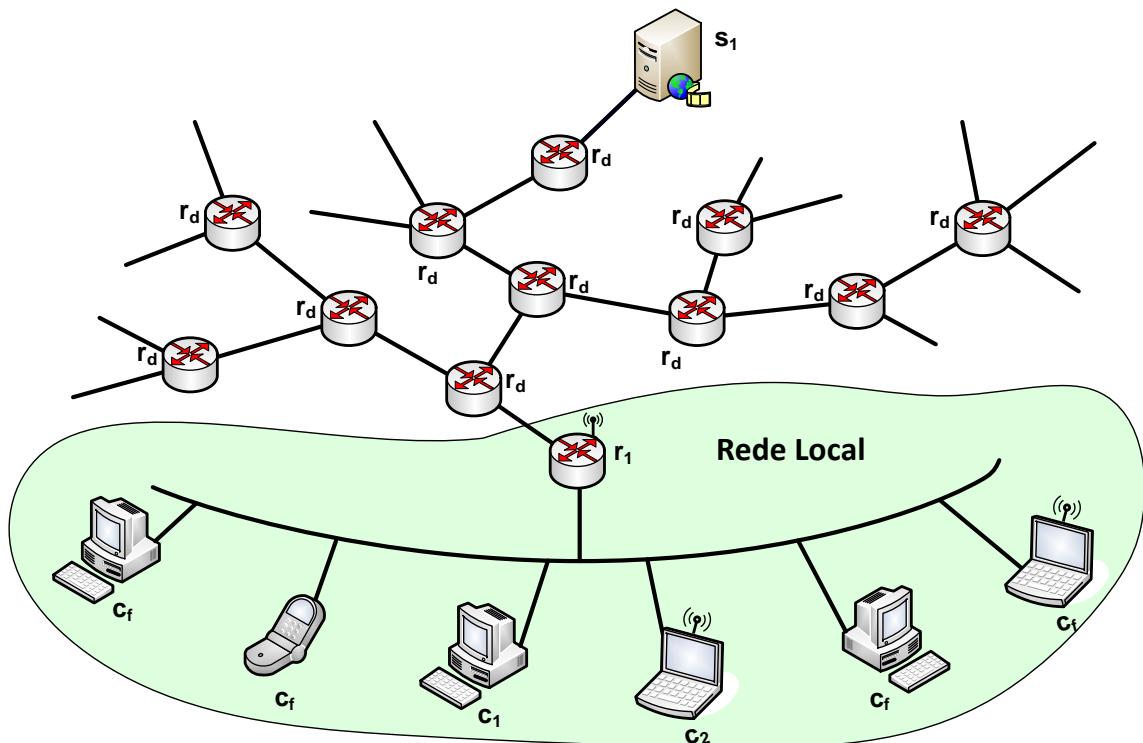


Figura 4.11: Exemplo de rede para o estabelecimento de conexão do GMTP.

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 4.12, onde ilustra-se um exemplo de conexão na Fase 1, considere:

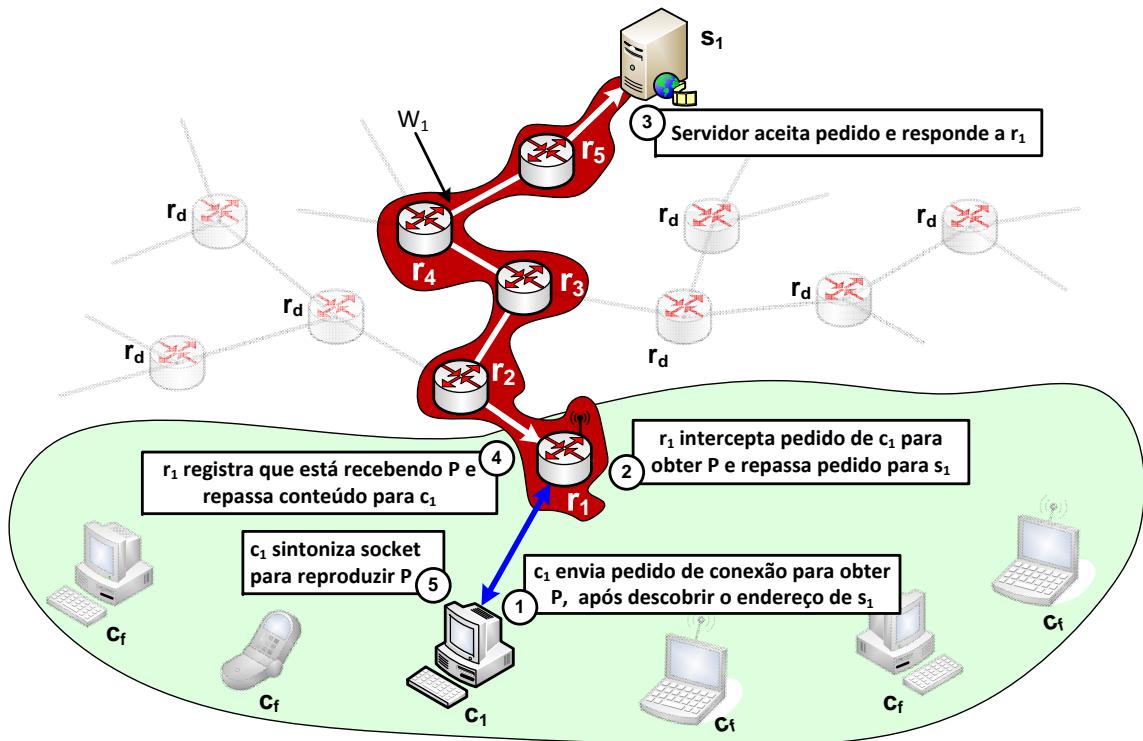


Figura 4.12: Passos do processo de estabelecimento de conexão do GMTP (Fase 1).

- $P$ , um fluxo de dados;
- $s_1$ , o servidor que gera os pacotes de dados  $p_x \in P$ ;
- $r_1$ , o repassador dos nós  $c_f \in C_i(r_1)$ ; e
- $c_1$ , um 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 4.12, 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 4.12), 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 4.12 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 nós  $c_f \in C_i(r_1)$  (Passo 4 da Figura 4.12). 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 4.12).

No Algoritmo 6, resumem-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*. Nota-se 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 4.13. 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 4.13: 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 4.14 (Passo 1 da Figura 4.14). 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  $P$ .

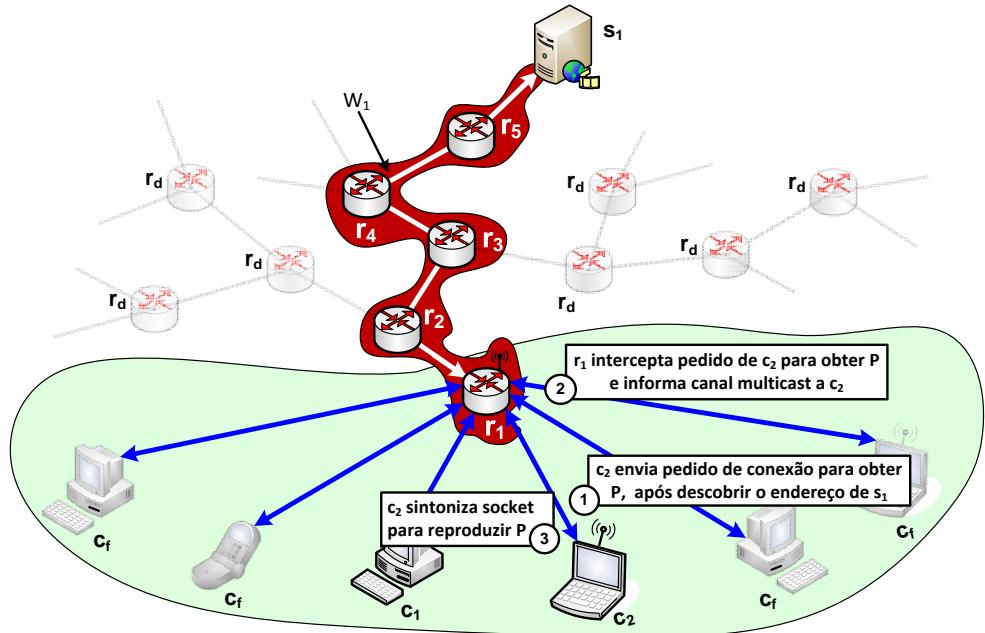


Figura 4.14: Passos do processo de estabelecimento de conexão do GMTP (Fase 2).

#### 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. Para isto, o nó  $s_a$  constrói

uma lista de nós parceiros e envia ao nó  $r_d$ , funcionando como um indexador de nós parceiros  $r_q$ , pré-selecionando parceiros para  $r_d$ . Por exemplo, seja um nó  $r_3$  que esteja recebendo  $P$  originado em um nó  $s_a$ , como ilustra-se na Figura 4.15. 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  $r_3$ . 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:

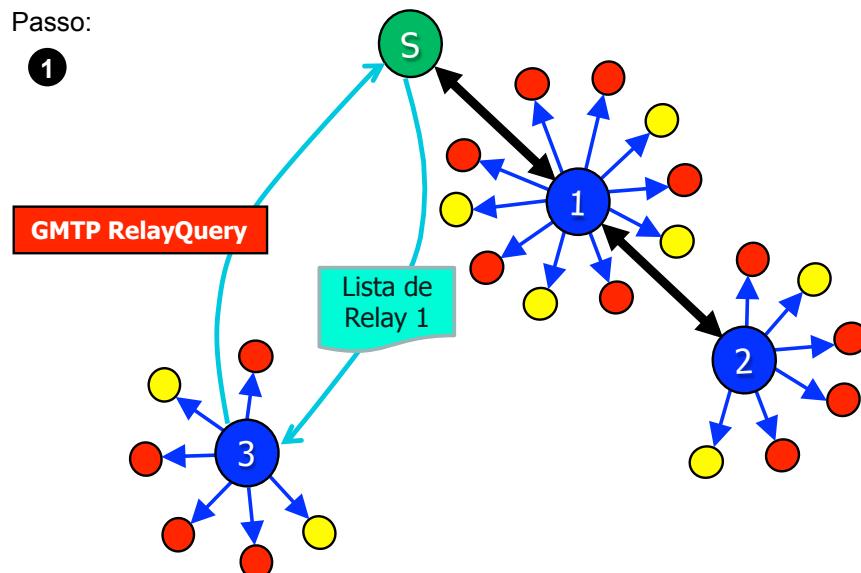


Figura 4.15: Fase 3 de conexão do GMTP (Passo 1).

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  $W_v^*$ , 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. Nota-se que, por exemplo, no CoolStreaming, a escolha aleatória é feita em nível de 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, executam-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 4.16, 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 repassador candidato a parceiro e sua respectiva chave de autorização;

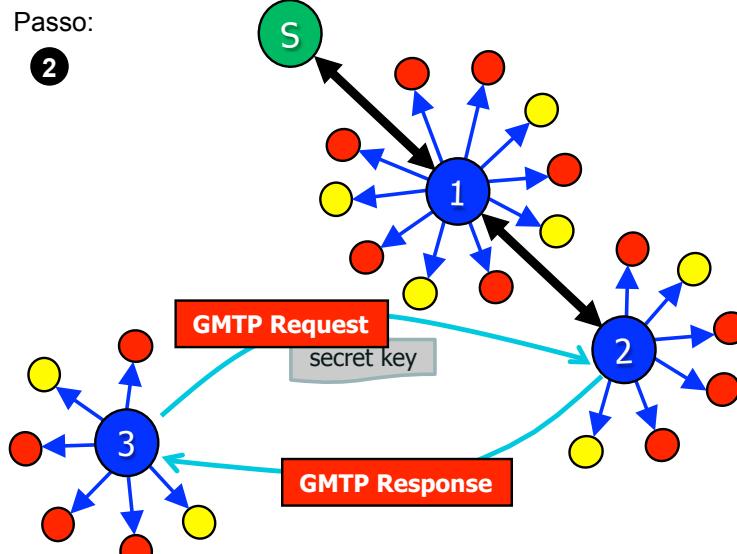


Figura 4.16: Fase 3 de conexão do GMTP (Passo 2).

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.

Com a execução da Fase 3 do processo de conexão do GMTP, pode-se expandir a quantidade de parcerias e se registram tais informações na tabela de recepção de fluxos de dados, tal como ilustra-se Figura 4.17. 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,

#	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 4.17: Tabela de recepção de fluxos de dados após a Fase 3.

mesmo que estas não tenham sido desenvolvidas pela mesma equipe. Esta unificação ajuda no processo de distribuição do fluxo de dados  $P$ , pois 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 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 à 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 5 (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 4.18. 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 estabelecimento 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 pacotes 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*.

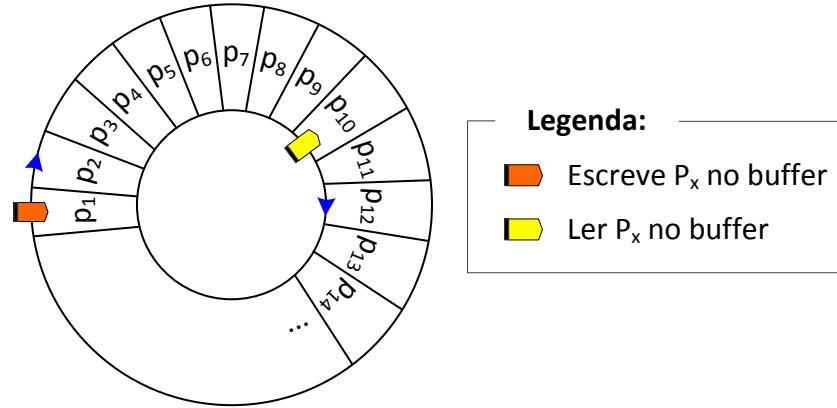


Figura 4.18: Exemplo da estrutura do *buffer* de envio e recepção de um nó GMTP com dois ponteiros, um para escrever e outro para ler pacotes  $p_x$ .

#### 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 4.19, 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.

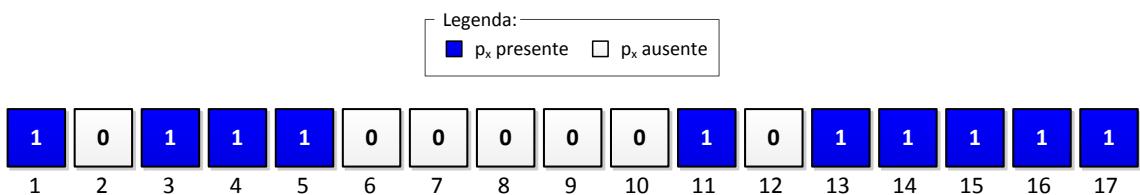


Figura 4.19: Exemplo do mapa de *buffer* de um nó GMTP com tamanho de 17  $p_x$ .

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 4.19, 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 deve-se evitar o método *pull* devido à transitoriedade dos pacotes de dados  $p_x$  (transmissão de dados ao vivo) e um nó  $r_d$  deve apenas realizar 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 4.19, o nó GMTP preenche o pacote do tipo *GMTP-DataPull-Request* com a sequência 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  $p_x$ . 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 4.20, tratam-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 atual em transmissões *multicast*, chamado de *GMTP Multicast Congestion Control* (GMTP-MCC). No GMTP-UCC, utilizado na comunicação entre os nós  $r_d$ , define-se a taxa de transmissão de um nó GMTP com base em uma versão modificada do protocolo RCP [58]. Já em modo de transmissão *multicast*, executa-se uma versão modificada do TFRC [169], com base em relatórios transmitidos por nós  $l_w \in C_i(r_d)$ , 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 [56]. 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

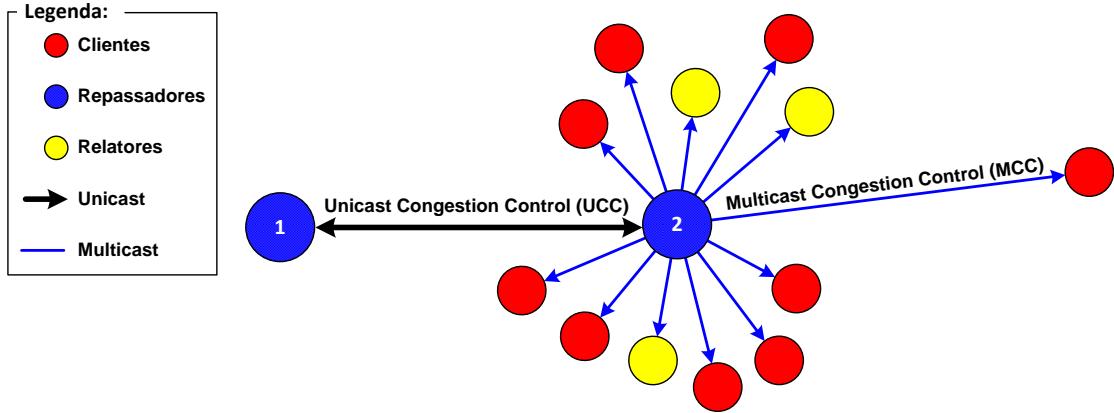


Figura 4.20: Organização do algoritmo de controle de congestionamento no GMTP.

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  $\hat{t}$ .

Com base nesse princípio, Nandita *et. al* [58] argumentaram que para um roteador funcionar de forma equânime e reduzir o tempo de duração de um fluxo (seja de curta ou de longa duração, proporcional à capacidade e independente da topologia da rede), deve-se oferecer a mesma taxa de transmissão para todos os fluxos transmitidos através deste roteador. Com isso, objetiva-se manter o número de pacotes na fila de roteamento perto de zero e evitar que apenas os fluxos mais antigos, ou seja, com a taxa de transmissão mais próxima da equânime, utilizem mais largura de banda se comparado aos fluxos mais recentes (discussões sobre este aspecto mais adiante, ainda nessa seção).

Com base nisso, determinou-se 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<sup>4</sup>  $\hat{N}(\hat{t}) = \frac{C}{R(\hat{t}-h_0)}$ . Além disso, para atualizar  $R(\hat{t})$  com mais frequência do que no intervalo de um RTT ( $h_0$ ), definiu-se  $H = \min(H_{user}, h_0)$  e, para manter a estabilidade do restante da equação, escala-se a mudança agregada por  $\frac{H}{h_0}$ , resultando na Equação 4.2, onde:

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

<sup>4</sup>Mais detalhes sobre a estimativa do valor de  $\hat{N}(\hat{t})$  está disponível na Seção 3.2.1 da referência [58].

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

- $h_0$ , 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.

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

- $H = \min(H_{user}, h_0)$ , sendo  $H_{user}$  um tempo definido pelo usuário (por exemplo, o administrador do roteador), caso seja necessário atualizar  $R(\hat{t})$  em um intervalo de tempo menor que  $h_0$ . Por exemplo, se a fila estiver enchendo, é desnecessário esperar um tempo de  $h_0$  para processar a fila. O valor para  $H$  é definido em *milissegundos*;
- $R(\hat{t} - H)$ , é a última taxa de transmissão medida, em *bytes/milissegundos*;
- $C$ , é a capacidade (largura de banda) do canal.
- $y(\hat{t})$ , é a taxa de tráfego de entrada medida no intervalo entre a última atualização de  $R(\hat{t})$  e o instante  $H$ ;
- $q(\hat{t})$ , 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 nos buffers, 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, por exemplo, pacotes TCP;
- $\alpha$  e  $\beta$ , tal que  $0 < \alpha, \beta \leq 1$ , determinam a estabilidade e o desempenho do algoritmo, respectivamente. Com base em discussões apresentadas em [58], os valores de  $\alpha$  e  $\beta$  dependem do tamanho médio dos fluxos comparado com o produto largura de banda – atraso. Quando o tamanho médio dos fluxos estão próximo do produto largura de banda – atraso (fluxos longos), sugere-se um alto valor para  $\alpha$  e um baixo valor para

$\beta$  (por exemplo:  $\alpha = 0.9$  e  $\beta = 0.1$ ), uma vez que para fluxos longos, prefere-se maximizar a taxa de transmissão de cada fluxo a minimizar o atraso na fila. Por outro lado, para fluxos de curta duração, recomenda-se um valor baixo de  $\alpha$  e um valor alto de  $\beta$  (por exemplo:  $\alpha = 0.1$  e  $\beta = 1$ ), pois esta combinação ajuda manter um baixo atraso de fila. Para um equilíbrio entre a estabilidade e o desempenho, recomendam-se valores de  $\alpha \in (0.4, 0.6)$  e  $\beta = (0.2, 0.6)$ . Para efeito de experimentação do GMTP, utilizou-se  $\alpha = 0.3$  e  $\beta = 0.7$ ;

- $\gamma$ , tal que  $0 < \gamma \leq 1$ , controla o pico de utilização do canal. Por exemplo, se desejar utilizar no máximo 95 % do canal em um certo instante  $\hat{t}$ , basta definir  $\gamma = 0.95$ . Nesse caso, pode-se tratar cenários de reserva de banda para um ou mais fluxos de dados específicos. Para efeito de experimentação do GMTP, utilizou-se  $\gamma = 1$ .

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  $t_2$ . Desta forma, segmenta-se 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 discute a seguir.

O algoritmo adotado no GMTP-UCC, adaptado do RCP, funciona da seguinte forma (acompanhe os passos de acordo com a Figura 4.21):

- 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 pacotes gerado em  $s_a$  de um fluxos 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:

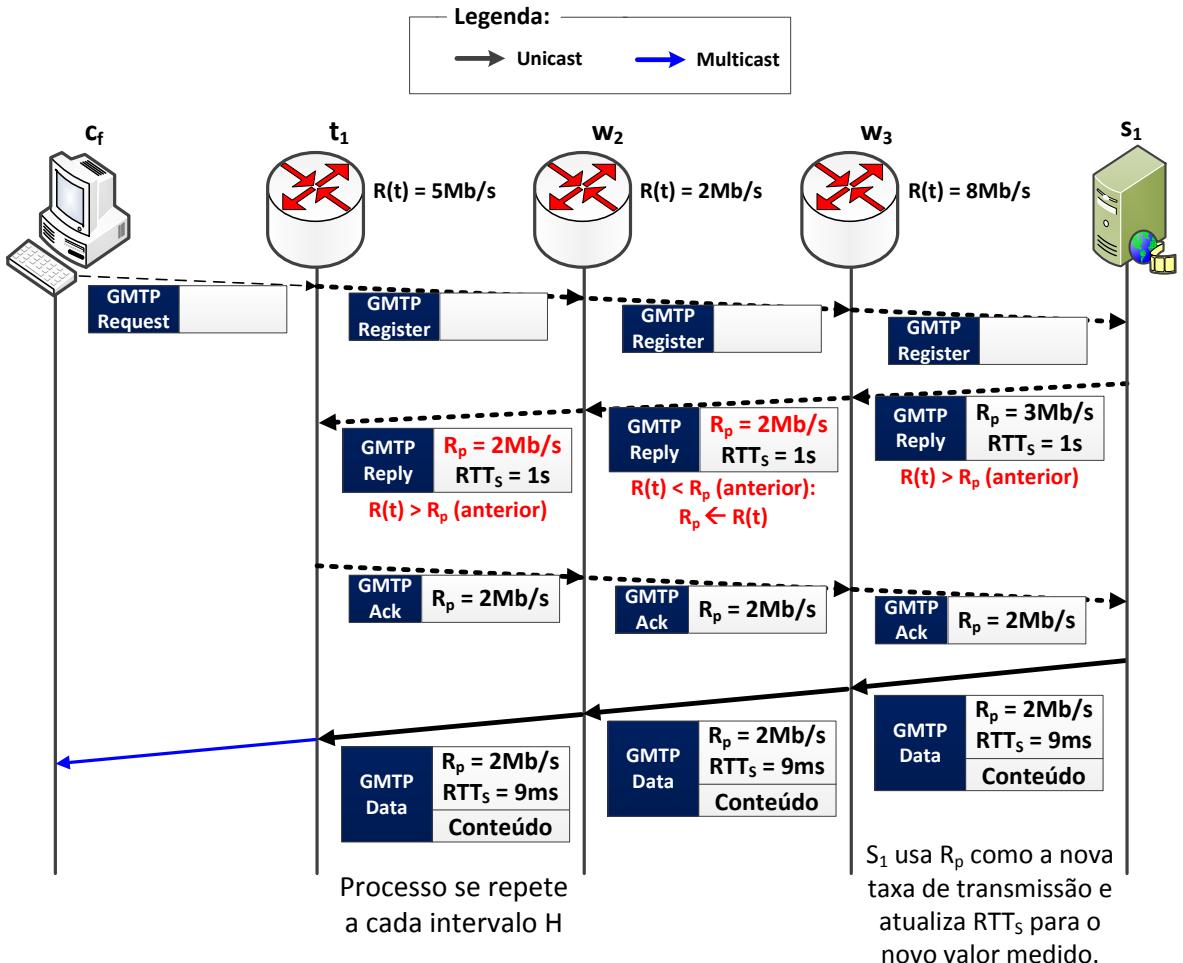


Figura 4.21: Cada  $r_d$  mantém uma única taxa de transmissão  $R(\hat{t})$  que é atribuída no cabeçalho de todos os pacotes transmitidos do nó  $s_a$  aos nós  $w_m \in W_v$ . À medida que o pacote passa em cada  $w_m$ , se a taxa atual  $R(\hat{t})$  no roteador for menor do que  $R_p$ , informada 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  $H$ .

- *taxa de transmissão proposta ( $R_p$ ):* corresponde à taxa de transmissão inicialmente desejada pelo nó  $s_a$  para transmitir os pacotes de dados  $p_x \in P$ . Como o GMTP segmenta o caminho em vários sub-caminhos, o valor de  $R_p$  pode conter a taxa de transmissão de um nó  $w_m \in W_v$  que está posicionado entre dois nós  $t_u, t_{u+1} \in W_v$ , com a menor capacidade de transmissão em um instante  $\hat{t}$ ;
- *RTT na fonte ( $RTT_s$ ):* corresponde ao RTT estimado entre quaisquer nós  $t_u, t_{u+1} \in W_v$ , 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  $c_f \in (C_i(t_u) \cup C_i(t_{u+1}))$ .

*Observação:* no RCP, utiliza-se apenas os sistemas finais (transmissor e receptor) para determinar os valores de  $R_p$  e  $RTT_s$ .

- 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  $s_a$  (Seção 4.3.1). 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 [249]. 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  $h_0$ .
- 4° 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 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 repassa-o 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  $R_p$  como a nova taxa para transmitir os próximos pacotes de dados  $p_x \in P$ , pois se trata da menor taxa de transmissão oferecida ao longo do sub-caminho  $W'_v \subset W_v$ , tal que  $W'_v = \{t_{u-1}, \dots, w_m, w_{m+1}, w_{m+2}, \dots, t_u\}$ .

*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  $w_m \in (W_v \cup T)$ , incluindo o 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.

- 5° A cada instante  $H$ , o 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 $W_v$ :

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 4.22, ilustra-se um cenário de transmissão com apenas os nós  $r_d$  e 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.

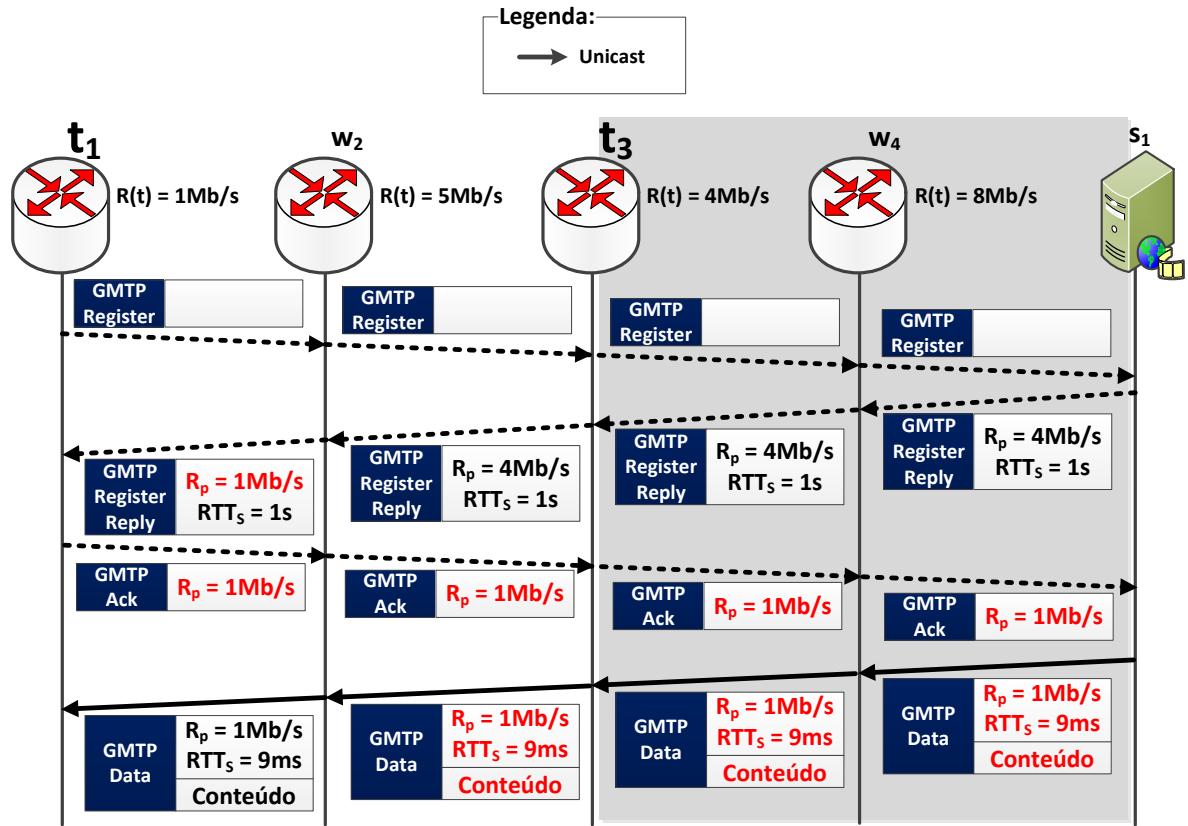


Figura 4.22: O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão, porém isto pode limitar alguns clientes a receberem os pacotes de dados em uma taxa maior. Nesse caso, o nó  $t_3$  tinha capacidade para receber o conteúdo a uma taxa de transmissão de  $4\text{ Mb/s}$ , porém a taxa de máxima relatada por  $t_1$  é de  $1\text{ Mb/s}$ , fazendo com que todos os nós no caminho  $W_v$  recebam o fluxo de dados  $P$  a  $1\text{ Mb/s}$ .

Sendo assim, considerando o mesmo exemplo ilustrado na Figura 4.22, mas adotando essa estratégia de segmentar o caminho  $W_v$ , tal cenário corresponde ao ilustrado na Figura 4.23. Note que, no sub-caminho  $W_1^{\leftarrow} = \{t_3, w_2, t_1\}$ , a taxa de transmissão de  $t_3$  em direção a  $t_1$  será de  $1\text{ Mb/s}$ , ao passo que no sub-caminho  $W_2^{\leftarrow} = \{s_1, w_4, t_3\}$  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

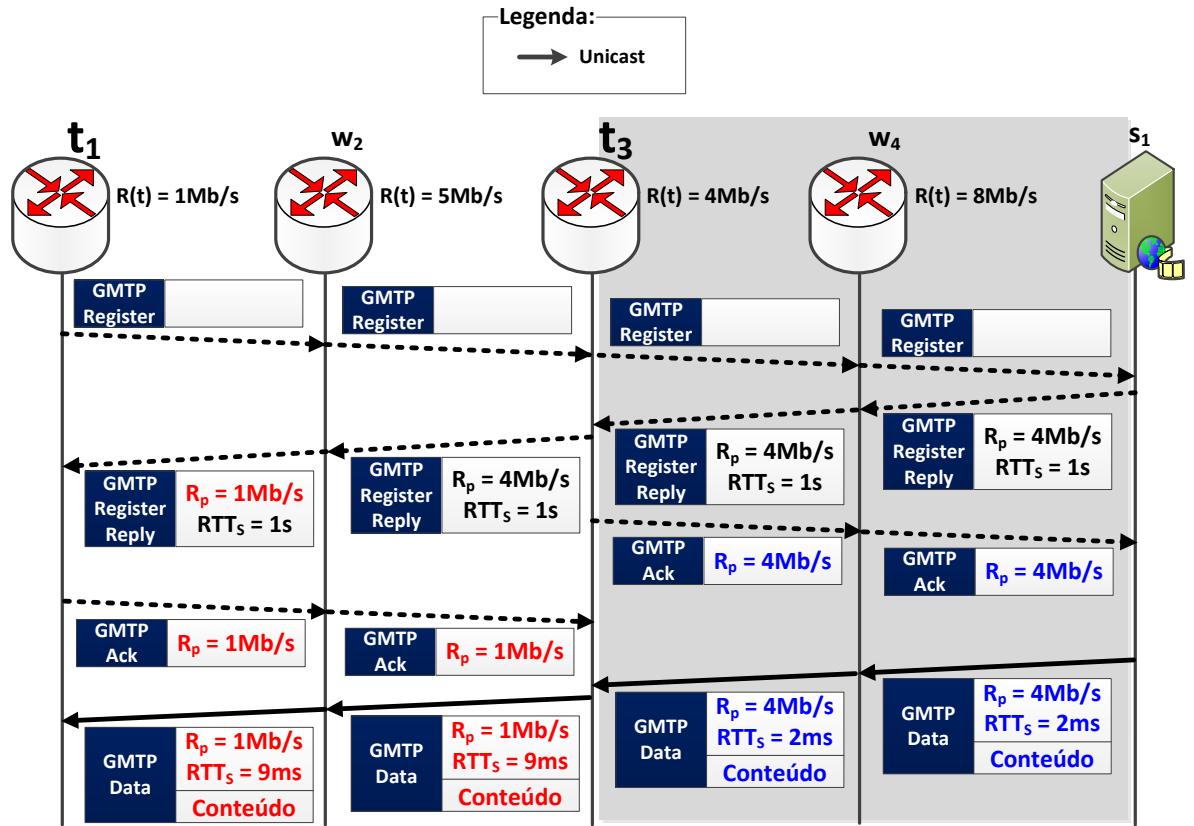


Figura 4.23: O GMTP-UCC segmenta o caminho e dessa forma não limita a taxa de transmissão de um fluxo de dados para certos nós capazes de receberem em uma taxa de transmissão maior.

sugerir nós parceiros  $r_q$  é priorizar os 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 termina um sub-caminho e se inicia o subsequente, obtidos pelo Passo 4b do algoritmo GMTP-UCC.

Após o nó  $s_a$  selecionar um sub-conjunto de nós candidatos  $r_q$  e sugerir ao nó  $r_d$ ,  $r_d$  envia um pedido de interesse de recepção para obter um fluxo de dados  $P$  aos nós  $r_q$ . O nó  $r_d$  escolhe o(s) nó(s)  $r_q$  com base na capacidade de transmissão percebida entre este e cada  $r_q$  (na direção inversa). Nesse caso, cada nó  $r_q$  sugere ao nó  $r_d$  o valor de  $R_p$  atualmente entre  $s_a$  e  $r_q$ . Porém, até o pacote alcançar  $r_d$ , os nós  $r_d$  intermediários podem atualizar o valor de  $R_p$  devido ao Passo 4 do algoritmo GMTP-UCC.

### **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 tentam 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 [56, 58].

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 sucinta, deve-se entender 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 4.24, 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 [250] e mostram que os fluxos poderiam ser finalizados uma ordem de magnitude mais rápida do que o TCP.

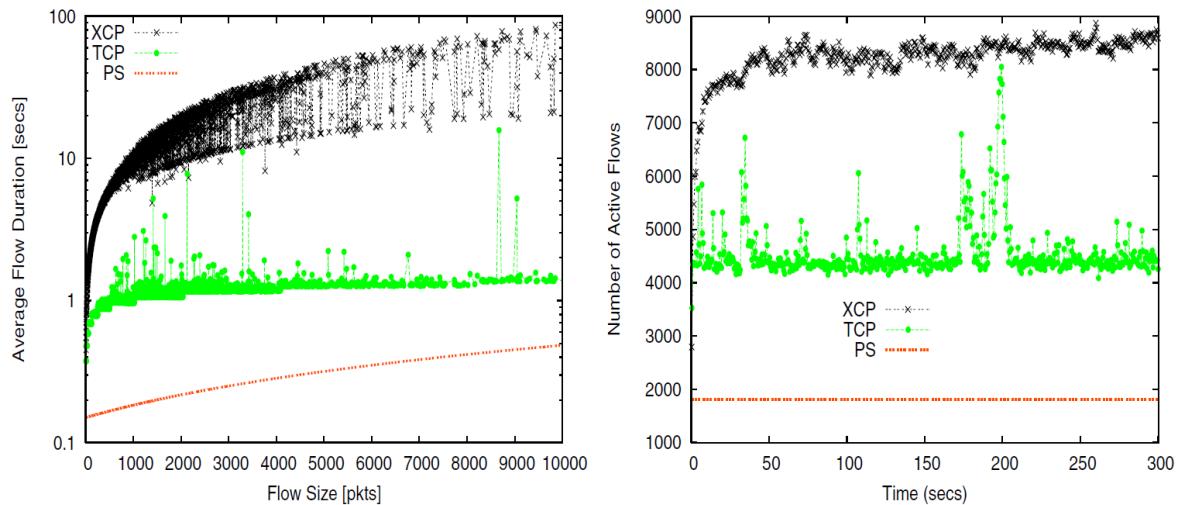


Figura 4.24: No gráfico da esquerda, ilustra-se o tempo médio de duração (quanto tempo leva para finalizar o fluxo) de um fluxo versus o tamanho do fluxo utilizando o TCP e o XCP. No gráfico da direita, ilustra-se o número de fluxos ativos versus o tempo. Ambos os gráficos são resultados de simulações 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 [58].

Com base nos gráficos da Figura 4.24, observa-se que o TCP demora para finalizar os fluxos de dados porque se consomem 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 des-

cartes 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 apenas 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.

Um outro exemplo comparativo e importante entre o RCP, XCP e TCP se observa na Figure 4.25. Nesses gráficos, ilustram-se dois fluxos de tamanhos distintos, um considerado de curta duração (260 pacotes) e outro de longa duração (3600 pacotes). Nota-se que no primeiro gráfico o fluxo do TCP finalizou primeiro (enquanto ainda estava na fase de partida lenta) do que mesmo transmitido usando o XCP. Já no segundo gráfico, observa-se o impacto causado no TCP quando houve uma perda de pacote na fase de partida lenta, forçando-o a entrar na fase de AIMD, retardando a finalização do fluxo. Em ambos os casos, o RCP obteve um melhor desempenho se comparado ao TCP e ao XCP, porque o roteador oferece uma taxa de transmissão inicial muito próxima ao PS, sendo eficiente em rapidamente perceber largura de banda ociosa e que pode ser oferecida aos fluxos.

O XCP é lento em ofertar largura de banda para os fluxos, oferecendo uma baixa taxa de transmissão para os fluxos mais recentes. O XCP gradativamente reduz o tamanho da janela dos fluxos mais antigos e aumenta o tamanho da janela dos fluxos mais recentes, a fim de garantir que não ocorrerá super-utilização da largura de banda disponível. Por isso, gastam-se múltiplos RTTs para fazer com que a maioria dos fluxos alcancem uma taxa de transmissão equânime (que muda à medida que se iniciam novos fluxos na rede), mantendo-se uma baixa ocupação dos *buffers* dos roteadores.

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  $H$

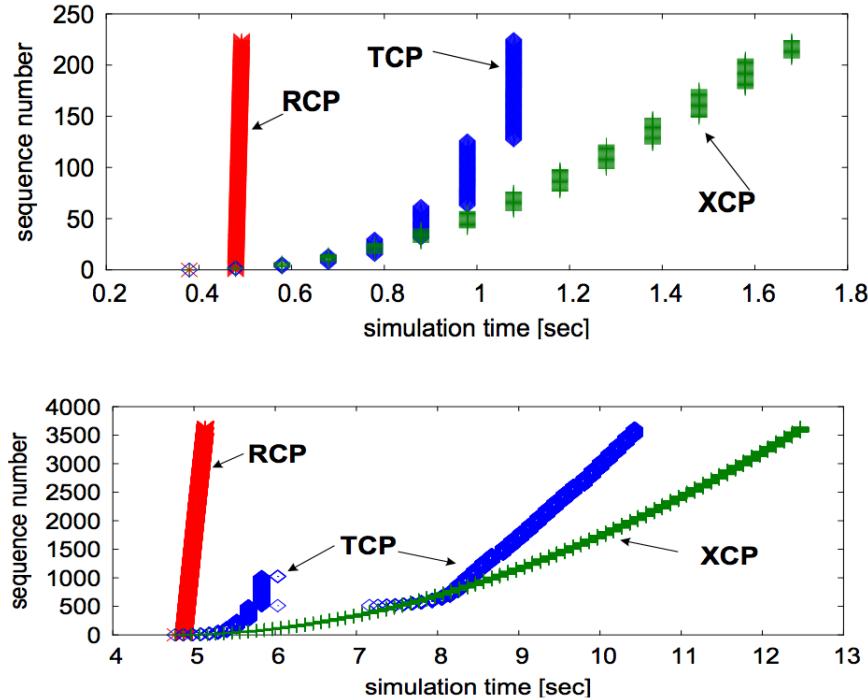


Figura 4.25: Evolução dos números de sequência de fluxos de dados, quando utilizando TCP (Reno), XCP e RCP. O tamanho do fluxo no primeiro gráfico foi de 230 pacotes, e no segundo gráfico foi de 3600 pacotes. Extraído de [58].

(vide Equação 4.2). Isso ocorre ao preço de ocorrer uma super-utilização temporária do canal (quando ocorrer um aumento substancial no número de fluxos no intervalo menor do que  $H$  – *flash crowd*), mas para fluxos de mídias ao vivo pode-se tolerar perdas circunstanciais.

Já ao observar o gráfico da Figura 4.26, 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 à melhor utilização o canal de transmissão (seja quando em altos níveis de utilização quanto de ociosidade). Com base no gráfico, observa-se que, em comparação ao XCP e às 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. Nesse contexto, observa-se que o XCP obteve 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

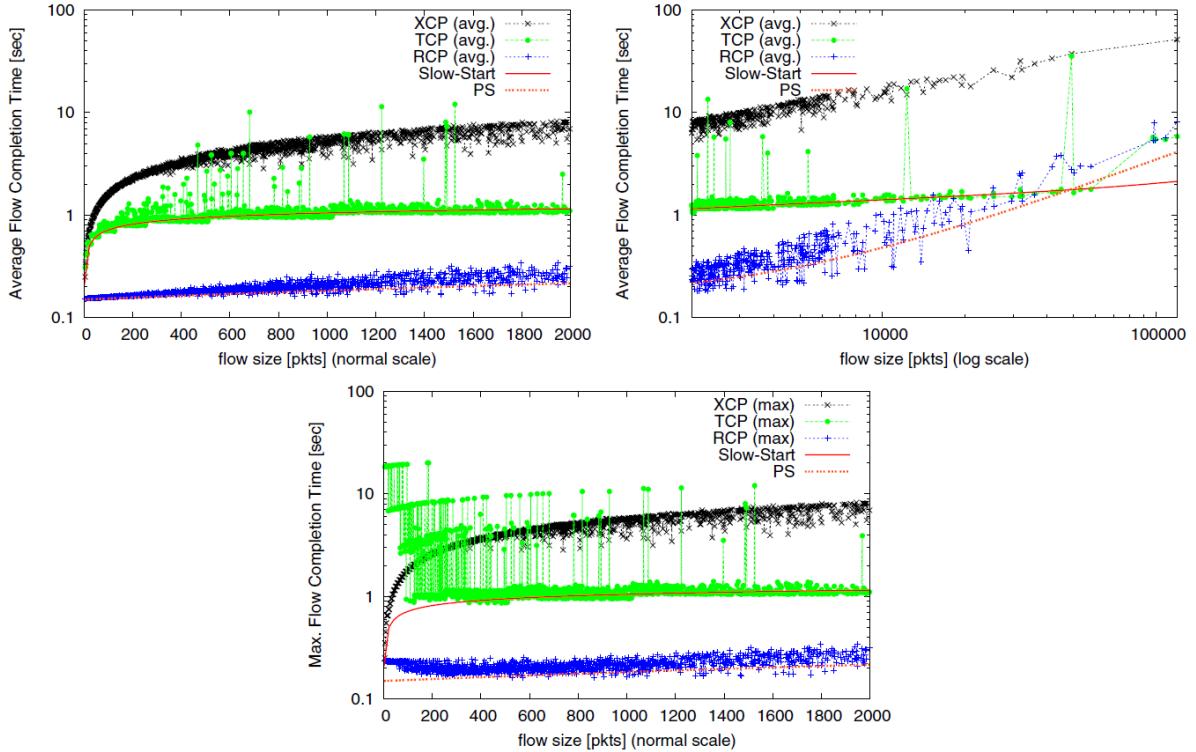


Figura 4.26: Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP como resultados de simulações com taxa de entrada de dados em *Poisson* e tamanhos do fluxo em distribuição Pareto com média de 25 pacotes (1000 bytes/pacote), *shape* igual a 1.2, capacidade 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 [58].

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  $R_{rcp}(t)$ , utilizam-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 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, evitando-se também sobrecarregar os nós  $s_a$  através da função do GMTP de registro de participação.

Apesar das observações anteriores, não é foco desse trabalho aprofundar as discussões acerca do desempenho do XCP comparado ao RCP, mas para não deixar a impressão de que o XCP sempre é ruim comparado ao TCP, ressalta-se que também há cenários onde o XCP tem um desempenho superior ao TCP, incluindo as variantes de *High Speed TCP*. É possível observar um melhor desempenho do XCP quando se aumenta o tamanho médio de duração de um fluxo para se aproximar ao produto largura da banda – atraso. Nesses casos, aproxima-se de um regime onde existem fluxos consumindo praticamente toda a largura de banda disponível de um canal, com o valor equânime da taxa de transmissão alocada individualmente para cada fluxo. Nesse tipo de cenário, o TCP-SACK apresenta vários problemas de desempenho, que foram aprimorados pelo HS-TCP [58]. Em todo caso, sabe-se que em cenários reais, o tamanho dos fluxos variam, existindo fluxos de curta e longa duração multiplexados a todo instante em um roteador.

#### 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 [251]). O TFRC é um mecanismo para controle de congestionamento de fluxos *unicast* que tenta prever a taxa de transmissão de um fluxo TCP e utilizá-la em protocolos diferentes do TCP [169]. 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 [252, 253]. 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 [254],  $R(s, p)$  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 confirmações* (*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 confirmações*, determinou-se que apenas alguns nós  $c_f$  são obrigados a enviar tais relatórios ao nó  $r_d$ . Estes nós são chamados de relatores e representados por  $l_w$ . No GMTP-MCC, a versão original do TFRC foi alterada e funciona da seguinte forma:

- 1° O nó  $r_d$  executa um algoritmo de eleição de nós  $l_w \in C_i(r_d)$ . Na Seção 4.7.3, descreve-se o procedimento para eleger os nós  $l_w$ .
- 2° 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;
- 3° 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 [251, 254];
- 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° Repetem-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 aritimética 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 aritimética 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 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 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  $w_m$ , definiram-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  $s_a$  gera os pacotes de dados para que seja possível verificar sua autenticidade.

### 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 2.3.

---

#### **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  $data \leftarrow \text{getPacketFieldValue}(p_x, 'data');$
- 2  $hashValue \leftarrow \text{hash}(data);$
- 3  $signature \leftarrow \text{encrypt}(K_{s_a}^-, hashValue);$
- 4  $\text{setPacketFieldValue}(p_x, 'signature', signature);$
- 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  $p_x$ . Salienta-se 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  $\varphi(w_m, P) = 1$ .

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*, 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 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 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 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 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 [255], 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 repassadores ao iniciar uma instância do protocolo GMTP. Na prática, trata-se de um *socket* que os repassadores formam as devidas parcerias para transmitir os fluxos de dados uns para os outros e, posteriormente, serem disseminados em modo *multicast* pelos respectivos repassadores aos seus clientes.

Do ponto de vista de roteamento, todo 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 cliente, o 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 cliente, se o fluxo de dados de interesse do nó cliente especificado no pacote *GMTP-Request*

já estiver sendo recebido por tal repassador.

### Canal de repasse *multicast*

O canal de repasse *multicast* é utilizado por um 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 repassador, para transmitir os datagramas para todos os seus clientes interessados por um evento ao vivo.

O *socket de repasse multicast* deve ser criado quando um 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 [255]. 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 nó  $l_w$  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 em 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 período 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 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, uma aplicação 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 referentes à mídia em questão, ao passo que os roteadores intermediários interceptam os mesmos pacotes de dados quando seus clientes locais também têm interesse pela mesma mídia. Isso permite o conceito de sub-fluxo.

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 detecta pontos comuns nas rotas e, em vez de aceitar um novo pedido de conexão de um nó solicitante em obter o fluxo de dados, recusa-o e sugere ao nó solicitante uma lista de roteadores candidatos a parceiros, de modo a evitar múltiplas conexões em direção ao servidor. Um roteador pode também solicitar explicitamente uma lista de candidatos a parceiros a fim de obter mais rapidamente os pacotes de dados e, secundariamente, conhecer outros roteadores e contatá-los em caso de desconexões dos 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, partindo da mesma rede. O GMTP executa três fases de conexão. A primeira fase de conexão ocorre quando o primeiro nó em uma rede local solicita receber um determinado fluxo de dados; a segunda fase ocorre através do compartilhamento dos pacotes de dados de um fluxo em modo *multicast*; e a terceira fase permite que os roteadores aumentem suas parcerias.

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. Além disso, o algoritmo GMTP-UCC pode auxiliar os nós receptores a selecionarem os melhores parceiros de acordo com a capacidade de transmissão do canal entre os receptores e os transmissores. 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, tornando o GMTP um protocolo TCP-Friendly.

Por fim, discutiram-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 se permite os repassadores validarem se tal conteúdo não foi alterado ao longo do caminho entre o servidor e os diversos clientes. Além disso, discutiram-se outros aspectos relacionados ao GMTP, como as ações realizadas na desconexão dos repassadores, relatores e clientes, bem como a eleição de 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 podem ser aprimoradas (ou ainda, aquelas que fazem sentido manter na camada de aplicação). Com esta visão, decidiu-se reduzir as responsabilidades dos 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 mul-

timídia 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 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). A partir do uso do GMTP, bastará posicionar um roteador GMTP entre a rede IP e a rede celular para atender à demanda de todos os nós móveis.

# **Capítulo 5**

## **Análise de Projeto e do Desempenho do GMTP**

Neste capítulo, apresentam-se a análise de projeto e a avaliação de desempenho do protocolo GMTP com relação às propostas Denacast/CoolStreaming [6, 125] e o CCN-TV [126], selecionadas devido à similaridade arquitetural para distribuição de mídias ao vivo. Tratam-se de propostas que adotam a maioria das estratégias disponíveis no estado da prática/arte para o fim que se discute, referenciadas na literatura e acessíveis para estudos comparativos. Nesse contexto, considera-se que as avaliações são equânimis, pois o Denacast estende o funcionamento do CoolStreaming para dar suporte a uma estrutura P2P/CDN, o CCN-TV faz uso de uma infraestrutura que considera efetivamente o suporte da rede para otimizar o acesso aos dados e o GMTP, que essencialmente explora uma estratégia híbrida em relação a estes concorrentes.

Inicialmente, na Seção 5.1, analisa-se o projeto GMTP no tocante aos seus benefícios e funcionalidades, bem como uma comparação de projeto (arquitetura e modelo de serviço) frente ao Denacast/CoolStreaming (a partir deste ponto chamado apenas de Denacast) e o CCN-TV. Em seguida, apresenta-se a fase de avaliação do GMTP. Na Seção 5.2, apresenta-se a metodologia de avaliação de desempenho do GMTP. Na Seção 5.3, estabelecem-se confrontos entre o GMTP e os sistemas supracitados, apresentando-se os resultados e discussões mais relevantes. Por fim, na Seção 5.4, apresentam-se o resumo dos principais resultados obtidos na análise de desempenho e o sumário deste capítulo.

## 5.1 Análise do Projeto

Nesta seção, objetiva-se justificar a proposta do GMTP não apenas como um protocolo que reduz a complexidade das aplicações e promove a interoperabilidade entre os sistemas, mas também suscitar que um conjunto mínimo de serviços, quando organizado de forma mais adequada, promove melhorias substanciais no projeto das aplicações e principalmente no uso dos recursos de rede.

Nesse contexto, organizou-se esta seção em duas partes. Primeiramente, analisam-se as funções do GMTP e seus consequentes benefícios e, em seguida, apresenta-se uma comparação de projeto entre o GMTP, o Denacast e o CCN-TV.

### 5.1.1 Projeto e benefícios do GMTP para os sistemas e para as redes

Como já apresentado na problemática deste trabalho (Seção 1.2), os desenvolvedores de sistemas de distribuição de mídias ao vivo enfrentam problemas que deveriam ser solucionados nas camadas inferiores da pilha TCP/IP e, por tentarem resolver na camada de aplicação, geram problemas adicionais.

Por exemplo, ao tentarem implementar na aplicação recursos para conexão multi-ponto (em geral P2P) e tratamentos de diferentes topologias, passa a ser necessário selecionar nós parceiros e tolerar falhas, o que por si só já geram outros problemas e que impactam negativamente na qualidade de serviço das aplicações. A necessidade de selecionar nós parceiros acarreta na necessidade de definir estratégias para atenuar o *churn* da rede, o que requer que as aplicações definam estratégias utilizem o histórico de disponibilidade e oferta de serviço dos candidatos a parceiros, o que acarreta na necessidade de ter que aferir e tomar decisões para ordenar os nós com base em critérios pré-definidos, mas sem garantias de obter uma melhor qualidade de serviço.

Como consequência das ações anteriores, demanda-se a obtenção de informação de contexto sobre os nós (localização, perfil do usuário, estimativas e medições de custos entre nós, etc). Devido ao dinamismo das redes, gerado pela tentativa de implementar conexão multi-ponto, faz-se necessário incentivar à cooperação entre nós a fim de mandá-los mais tempo conectados. Como consequência, deve-se explorar e diversas técnicas para este fim, como as baseadas em modelos econômicos, cada uma com seus próprios desafios. Pelo modelo

cooperativista implementado nas aplicações P2P, tornou-se necessário identificar e inibir a participação de nós *free-riders*, o que remete à necessidade de incentivos. Devido à complexidade de estimar a real capacidade de transmissão dos nós, tornou-se também necessário o uso de técnicas para monitorar a capacidade de recepção dos nós clientes a fim de adaptar os fluxos de dados com base em tal informação, sendo também necessário oferecer canais de transmissão com diferentes níveis de qualidade da mídia. Por fim, devido à cooperação direta entre os clientes, facilita-se a implementação de ataques de poluição e, portanto, fez-se necessário o emprego de soluções para impedi-los, ou ao menos validar o conteúdo antes de sua reprodução ao usuário final.

Ao observar os problemas resultantes da tentativa suprir as limitações da rede, a seguir, enumeram-se cinco principais benefícios que podem ser constatados ao utilizar o GMTP, tanto no contexto dos sistemas de distribuição de mídias ao vivo quanto no contexto do consumo dos recursos de rede, relacionando-os com os recursos atualmente empregados nos sistemas de distribuição de mídias ao vivo.

1. *Otimização arquitetural:* antes do GMTP, os desenvolvedores de sistemas multimídia eram obrigados a considerar limitações de arquitetura de rede em seus projetos de software. Por exemplo, a lógica para distribuição da mídia era toda embutida na aplicação, aumentando sua complexidade de manutenção. O GMTP abstrai a complexidade de distribuir mídias ao vivo com base, por exemplo, no gerenciamento transparente de canais *multicast* (adicionar e remover) sem a influência da aplicação, inclusive entre diferentes domínios administrativos. Além disso, abstraem-se outros aspectos importantes para a aplicação, como o controle de congestionamento, antes embutidos na aplicação por falta de uma proposta mais adequada.
2. *Interoperabilidade entre sistemas:* O GMTP unifica a forma que os sistemas finais transmitem e receber mídias ao vivo, uma vez que todo esse processo ocorre na camada de transporte e sem qualquer influência da aplicação. Isto significa que os sistemas podem cooperar entre si sem necessariamente deter conhecimentos arquiteturais e de interfaces de aplicação (API) uns dos outros. O plano é entregar as partes da mídia o mais rápido quanto possível, utilizando-se os melhores pares e independente dos sistemas. Para tornar isto possível, estabeleceram-se no GMTP funções como descrição

da mídia e mecanismos que promovem a cooperação entre os nós, sem necessitar manter controle sobre os nós que mais contribuem e aqueles que apenas se aproveitam dos recursos dos outros (*free-riders*). Nesse ínterim, realizam-se outras funcionalidades importantes tanto para melhorar o serviço oferecido às aplicações quanto ao consumo dos recursos de rede. Por exemplo, o mecanismo de controle de congestionamento empregado no GMTP funciona de forma colaborativa, tanto em *unicast* quanto *multicast*, sem a influência da aplicação, que não pode injetar informações falsas sobre sua percepção do estado da rede.

3. *Facilidade na integração*: os atuais e novos sistemas de distribuição de mídias ao vivo poderão integrar o GMTP de forma simples, uma vez que se manteve o uso da API tradicional de *sockets*. Para os sistemas existentes, muitas das funcionalidades, como as citadas no início desta seção não serão mais necessárias, podendo-se reduzir sobremaneira a complexidade da aplicação e consequentemente de manutenção. Para os sistemas novos, permite-se uma prototipação mais rápida e com menor chances de erros, evitando-se a super utilização dos recursos de rede devido ao emprego de boas práticas de engenharia de software para distribuição de mídias ao vivo e desacopladas da aplicação.
4. *Desacoplamento e Extensibilidade*: a organização do GMTP em dois módulos, GMTP-Inter e GMTP-Intra, permite o desacoplamento entre o que é função de rede e de aplicação, respectivamente. Isto evita que as funções de rede, como controle de congestionamento e distribuição de pacotes de dados sofram influência da aplicação. Apesar dessa separação, ainda sim permite-se a troca de informações entre os sistemas finais e a rede, mas que sejam somente pertinentes à distribuição de mídias (*Cross-Layer*). Além disso, possibilita-se a extensão do GMTP em casos de requisitos mais avançados da aplicação, quando não disponível no GMTP. Por exemplo, um desenvolvedor poderá promover alterações no GMTP-Intra e/ou GMTP-Inter a fim de atender suas necessidades sem afetar o GMTP-Inter e a aplicação em execução no sistema final. Uma fabricante de um roteador pode promover mudanças no GMTP-Inter (por exemplo, melhorar a estratégia de manipulação do *cache* ou de verificação de pacotes) e disponibilizar uma nova versão da *firmware* sem necessariamente ter que atualizar o

GMTP-Intra em todos os sistemas finais. As decisões tomadas principalmente na definição do cabeçalho do GMTP oferecem esse tipo de flexibilidade. Nesse contexto, é possível, por exemplo alterar o formato de descrição de uma mídia (função do GMTP-Intra) sem precisar alterar o GMTP-Inter, permitindo-se atualizações gradativas dos sistemas finais para uma nova versão do GMTP-Intra, sem afetar o funcionamento da aplicação, devido à abstração da API de sockets. O fato é que a proposta do GMTP é uma infraestrutura que permitirá a evolução do protocolo sem afetar a camada de aplicação, permitindo-se a adição de outras estratégias de distribuição de mídias ao vivo que possam surgir futuramente e o principal, que estarão disponíveis automaticamente para uso em qualquer aplicação.

5. *Eliminação de recursos paliativos e consequentes:* quando se delega muitas responsabilidades à aplicação, há uma tendência de desordem, de fragmentação de funcionalidades e consequentemente de um baixo aproveitamento dos recursos de rede, devido à liberdade que as aplicações passam a ter de promover seus próprios recursos da forma que seu desenvolvedor desejar. Por exemplo, em redes puramente P2P é sempre complexo manter uma estrutura sem nós *free-riders*, apesar de existirem algoritmos para reduzir o efeito dessa prática na rede. Com o uso do GMTP pode-se reduzir drasticamente os nós *free-riders*, pois quando um nó requisita um fluxo de dados automaticamente o GMTP poderá compartilhá-lo com outros nós parceiros. Um outro exemplo é o tratamento para conexão multi-ponto e tolerância às falhas. O mecanismo de conexão multi-ponto é resolvido no GMTP pela interceptação de requisições e sugestão de parcerias entre nós roteadores. Com relação às falhas, mesmo um nó roteador seja desconectado da rede seus parceiros não se tornarão órfão, pois o nó repassador GMTP pode obter o conteúdo multimídia a partir de múltiplos nós repassadores, o que reduz drasticamente problemas causados pelo *churn*. Enfim, atualmente outros problemas são tratados na camada de aplicação e que, com o uso do GMTP, tornam-se obsoletos são: a necessidade de obter informações de contexto sobre os nós, tais como localização, perfil de usuário e custos; uso de mecanismos de incentivos para permitir a cooperação; adaptação de fluxos de dados com base na capacidade dos nós receptores; controle de congestionamento, detecção de nós *free-riders* e de ataques de poluição.

### 5.1.2 Comparativo: GMTP, Denacast/CoolStreaming e CCN

A seguir, apresentam-se detalhes de comparação entre o GMTP e os seus dois principais concorrentes: Denacast e CCN. O Denacast pode ser considerado o estado da prática, o principal representante dos sistemas para distribuição de mídias ao vivo e que consideram uma arquitetura P2P/CDN. Já a CCN pode ser considerada uma proposta do estado da arte, onde o foco da discussão passa a ser mais arquitetural, com um cunho comparativo e menos relacionado com a camada de aplicação.

#### GMTP vs. Denacast:

1. No CoolStreaming, a rede de favores 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, efetivam-se parcerias independente da posição do nó na rede de favores, 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 favores 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) e o nó transmissor (servidor), são autorizados a interceptar o pedido de conexão e responder ao nó cliente como se fossem 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.
2. O conceito de sub-fluxo empregado no CoolStreaming adiciona complexidade à solução sem necessariamente resultado em melhor desempenho. Em [221], 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, a métrica de índice de continuidade de capacidade de *upload* não melhoram, 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, com interceptação de requisições. O método *pull* é utilizado somente em se deseja obter um determinado conteúdo prestado a ser reproduzido e que ainda não foi recebido via *push*. Essa estratégia reduz a quantidade de requisições ao servidor e o atraso para começar a reprodução de uma mídia ao usuário final (apenas o primeiro usuário perceberá um atraso maior do que os demais).

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. Por exemplo, no CoolStreaming e sistemas similares, se o usuário fechar o aplicativo pode resultar em interrupções temporárias na reprodução da mídia por parte de outros nós da rede, impactando no índice de continuidade.
4. Um nó recém integrado à rede DONet pode levar muito tempo (em alguns casos 20 segundos) para obter os primeiros blocos da mídia a fim de reproduzi-lo ao usuário final. Isto porque, ao se conectar 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 da mídia. Por exemplo, se o novo nó requisitar um bloco de vídeo muito antigo, pode ser que tal bloco não esteja mais disponível, já que o nó cliente (suposto parceiro) remove o bloco após sua reprodução. Por outro lado, se o nó requisitar um bloco da mídia muito recente, pode ser que nenhum de seus nós parceiros o tenha disponível, aumentando-se o tempo de espera. No GMTP, situações como essas não ocorrem porque se utiliza, por padrão, o método *push* e, além disso, as últimas partes da mídia (mais atuais) já podem estar disponíveis no roteador, quando outros clientes já estão conectados e recebendo

o fluxo de dados correspondente à mídia de interesse.

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 da mídia, 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, devido à dinâmica da rede. Um nó parceiro que é identificado com um nó ruim, no instante seguinte pode ser que o cenário mude e o referido nó passe a ser a melhor parceria, porém a desconexão entre estes já pode ter sido efetivada. Para realizar essa avaliação, cada nó monitora 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, apresentadas na Seção 3.2.6, tomindo-se então uma decisão. Essa estratégia é muito complexa quando em sistemas de distribuição multimídia em larga escala, pois se exige o monitoramento constante dos *buffers* dos sub-fluxos, o que implica em exaustivas trocas de mapa de *buffer*, que aumenta proporcionalmente com o aumento da quantidade de nós parceiros, tal como se discutiu na Seção 3.2.6. A consequência 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 nada disso é necessário, aliás, faz-se de uma forma bastante diferente. Utiliza-se o próprio algoritmo de controle de congestionamento (GMTP-Inter), que expõe aos nós clientes e aos nós parceiros e/ou servidores o nível de utilização do canal a cada instante definido com base no RTT, como discutiu-se na Seção 4.5.1. Com isto, permite-se a formação de parcerias sem precisar exigir que as aplicações monitorem o estado da rede. Dessa forma, tanto um nó receptor quanto o nó transmissor conhecem a capacidade máxima de transmissão no canal que separa ambos e, assim, o nó transmissor ajusta sua taxa de transmissão em direção ao nó receptor de acordo com a capacidade de transmissão disponível em um certo instante.
6. No ponto de vista da rede P2P, as considerações sobre o sistema Denacast são similares ao caso do CoolStreaming. Com relação à arquitetura geral do Denacast, propõe-se

uma melhor organização da rede de sobreposição devido ao uso de servidores de uma rede CDN. Isto permite 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, comparando-o ao CoolStreaming [6, 125].

### **GMTP vs. CCN:**

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 infraestrutura que sirva para fluxos de dados elásticos e inelásticos. Em geral, na rede CCN, constatou-se com base na revisão da literatura uma tendência a considerar uma arquitetura mais genérica possível para permitir diferentes padrões de tráfego, mas isso não é trivial – certos tipos de tráfego, multimídia por exemplo, requer tratamento peculiar da aplicação e principalmente da rede. Por este motivo, no GMTP, decidiu-se partir dos requisitos específicos dos sistemas de distribuição de mídias ao vivo e questionar sobre quais funções utilizadas nesses sistemas podem ser generalizadas para a aplicação e controladas pela rede e/ou abstraída pelo sistema operacional. Como resultado do GMTP, adiciona-se 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. Estrategicamente, adotou-se a premissa de que é mais importante entregar os pacotes de dados às aplicações de rede o mais rapidamente possível, mesmo que para isso tivesse que enfraquecer possíveis restrições comerciais da mídia sendo transportada. Nesse último caso, se houver restrições comerciais da aplicação, pode-se simplesmente optar por implementar um mecanismo proprietário de codificação e autenticação. Isto pode ser feito diretamente na aplicação, estendendo-se o GMTP-Intra.
2. Como consequência do item anterior, a rede CCN 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 se utiliza uma estrutura de *buffer* circular, substituindo-se as partes da mídia à medida que se recebem as partes subsequentes, sem a necessidade do roteador transmitir, a todo instante, o equivalente ao que foi denominado de pacote de interesse, considerando-se nomenclaturas básicas das redes CCN. Em CCN, um dos grandes desafios é definir quanto tempo um dado deve permanecer em *cache*, já que nem sempre os pacotes contém dados transientes. Isto impacta diretamente nos sistemas de mídia ao vivo, uma vez que um pacote pode expirar e continuar sendo mantido em *cache*. Isto motiva outro desafio: determinar se um dado em *cache* ainda é válido, principalmente em aplicações elásticas. No caso do GMTP, decidiu-se continuar usando endereçamento IP com o objetivo de que a sua adoção seja mais simples, considerando-se a estratégia de apenas instruir os roteadores a realizar uma política mais sofisticada de repasse, quando os pacotes entram na fila de roteamento, aproveitando-se a oportunidade de também repassá-lo através de rotas adicionais, de acordo com a demanda.

3. Para executar as aplicações de rede existentes na Internet sobre uma rede CCN, fazem-se necessárias maiores alterações nos sistemas de transmissão de mídias ao vivo. Será necessário que os atuais sistemas mudem a lógica de requisição das partes da mídia, sendo mandatório 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. Em CCN há uma inversão na forma que uma aplicação obtém os pacotes de dados. Isto porque propõe-se que o nó cliente controle praticamente todos as funções de transporte de dados, como controle de perda/erro, controle de congestionamento e de fluxo. No GMTP, realiza-se uma forma implícita de nomear os fluxos, levando-se em consideração a estratégia atual de acesso baseada em endereço IP e porta. Gerar um código único baseado nesses dois parâmetros evita a necessidade de ter que alterar as aplicações para especificar o nome do conteúdo a ser acessado. Além disso, o controle das funções não é delegado aos nós clientes, como ocorre nas redes CCN, principalmente em se considerando distribuição de mídias ao vivo. Pelo contrário, os nós finais (cliente e servidor) passam a ter uma participação mínima no processo de distribuir os pacotes de dados e o GMTP extrapola o limite da construção de uma rede P2P, pois

também promove serviços que evitam a ocorrência de requisitos e problemas consequentes da estratégia de uma rede P2P apenas na camada de aplicação, como os citados no início dessa seção. Enfim, mantem-se o GMTP no âmbito do paradigma das redes IP, sem exigir grandes mudanças no núcleo da rede, executando-se as funções de compartilhamento de um mesmo fluxo de dados através da formação de parcerias entre os roteadores, instruídos pelos nós servidores, que detém o efetivo conhecimento da demanda por um certo conteúdo.

4. Nas redes CCN, o mecanismo que determina a interceptação de um pacote de interesse é limitado à verificação local se o fluxo de dados já está sendo recebido ou não, sem auxílio de qualquer outro nó. Por exemplo, o nó servidor tem um papel coadjuvante, apenas respondendo 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 dos pacotes de dados 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 e já sendo utilizadas para transmitir o respectivo fluxo. De fato, há uma troca de serviços, pois o roteador também auxilia o servidor informando qual deve ser sua taxa ideal de transmissão, evitando-se super utilização do canal e permitindo que, se necessário, o servidor realize adaptações no conteúdo.
5. Nas redes CCN, ainda não está claramente definido o uso de soluções mais otimizadas de controle de congestionamento, pelo contrário, as propostas ainda estão no âmbito de algoritmos tradicionais baseado em janelas deslizantes ou similares. Em diversas pesquisas recentes, demonstrou-se que utilizar os roteadores para realizarem controle de congestionamento melhora sobremaneira o desempenho da rede e das aplicações. No GMTP, optou-se por utilizar tal abordagem, estendendo-a para permitir que o nó servidor sugira aos roteadores quais parcerias devem ser efetivadas, baseando-se na capacidade atual de transmissão dos canais conhecidos. Além disso, definiu-se um mecanismo para segmentar um caminho entre o cliente e o servidor com base na capacidade de transmissão dos nós intermediários. Em nenhum outra proposta disponível no estado da arte isto foi feito. Já nas CCNs, as pesquisas estão concentradas em fazer com o que o nó servidor disponibilize múltiplos fluxos de dados codificados em dife-

rentes taxas de bits, onde os nós clientes devem monitorar sua capacidade de recepção e requisitar o fluxo de dados mais apropriado. O problema é que realizar essa função apenas monitorando perdas de dados e atraso não é uma solução efetiva, principalmente devido à dinâmica de entrada e saída dos nós na rede (*churn*).

6. Em CCN, 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 proporcional à quantidade de pacotes de dados, aumentando a ocupação dos *buffers* dos roteadores. Isto porque os nós precisam continuamente enviar pacotes de interesse para obter os próximos pacotes de dados, consequentemente os pacotes de interesse sempre disputarão o canal de transmissão com os pacotes de dados. Para sistemas de distribuição de mídias ao vivo isto pode ser crítico. Note que as requisições estão sujeitas às condições de rede do canal de *upload* (do nó receptor ao nó transmissor, já que o nó cliente controla os serviços de transporte), aumentando-se a probabilidade de perdas de pacotes de interesse, consequentemente da recepção dos respectivos 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. A perda de um pacote de interesse aumenta o atraso em receber os pacotes de dados, pois os nós clientes devem retransmitir os pacotes de interesse, impactando diretamente em métricas como índice de continuidade e distorção do conteúdo. O índice de continuidade pode aumentar porque a perda de um pacote de interesse pode gerar várias perdas de pacotes de dados. No caso da distorção do conteúdo, um nó cliente pode até receber um pacote de dados (depois de retransmitir uma ou mais vezes os pacotes de interesse), porém o atraso de recepção pode ser alto suficiente ao ponto de não fazer mais sentido reproduzi-lo ao usuário final. Sendo assim, mesmo em cenários onde o canal de *download* não esteja congestionado, os nós receptores também podem experimentar uma baixa qualidade de serviço devido às perdas de pacotes de interesse. No GMTP, manteve-se o conceito de estabelecimento de conexão, apesar de uma versão adaptada dos modos tradicionais, então uma vez estabelecida uma conexão, os nós clientes não precisam transmitir novas solicitações a todo instante a fim de obter os próximos pacotes de dados (*push*).

7. Ainda no contexto do item anterior, quando ocorre perda ou o atraso dos pacotes de interesse em CCN, a rede não será capaz de transmitir os pacotes de dados em tempo hábil ou nunca conseguir transmiti-los, uma vez que os mesmos podem expirar no *cache* dos roteadores e portanto se tornarem inacessíveis. Nesse contexto, surgem alguns dilemas: qual é o melhor momento de desistir de requisitar um pacote de dados e requisitar o próximo. Como saber se o próximo pacote de dados foi gerado ou se a transmissão já terminou. Atualmente, existe um estudo sobre como introduzir uma função chamada de agregação de pacotes de interesse [225]. 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 do conteúdo comprometida, impactando na métrica Índice de Continuidade.

Como base no resumo comparativo apresentado nesta seção entre os projetos GMTP, Denacast e CCN, a seguir, concentram-se as discussões em uma avaliação de desempenho do GMTP em um cenário de transmissão de uma mídia ao vivo.

## 5.2 Avaliação de Desempenho

Para realizar a comparação entre os sistemas supracitados, definiu-se a modalidade experimental em um ambiente de simulação de rede. Através da definição de uma topologia de rede que se aproxima do mundo real, das variáveis independentes e dos fatores, mediram-se e analisaram-se as principais métricas (variáveis dependentes) que determinam a satisfação do usuário ao assistir a um evento através de um sistema de distribuição de mídias ao vivo. Para isto, realizou-se um estudo detalhado do comportamento do GMTP, observando-o em diferentes configurações de rede, a fim de determinar suas vantagens, limites e os impactos que seus recursos podem gerar tanto sobre os nós quanto sobre a rede.

A seguir, apresentam-se detalhes do projeto experimental executado, organizado em objetivo e hipótese, topologia de rede, variáveis e fatores, população e amostras, tratamentos,

instrumentação e formato da mídia.

### 5.2.1 Objetivo e hipótese

O objetivo do experimento foi avaliar o desempenho do GMTP com vista à hipótese enunciada de que a constituição de uma rede de favores entre roteadores que interceptam, realizam *cache* temporário e compartilham pacotes de dados tanto em modo *multicast* quanto em modo *unicast*, auxiliados por nós servidores para determinar as parcerias com base em um algoritmo para controle de congestionamento assistido pela rede, resulta na distribuição mais eficiente (escala e qualidade) de mídias ao vivo.

Como apresentou-se no Capítulo 4, o GMTP é uma instância dessa hipótese e, sendo assim, para validá-la, confrontou-se o GMTP com o Denacast e com o CCN-TV, obtendo-se valores objetivos para as métricas que determinam a qualidade de serviço dos sistemas de distribuição de mídias ao vivo.

### 5.2.2 Topologia de rede

Primeiramente definiu-se a topologia da rede, como ilustra-se na Figura 5.1. Simulou-se uma versão adaptada da atual rede GÉANT<sup>1</sup>, composta por 27 roteadores. Na Seção A.1, apresentam-se as configurações de cada roteador e seus enlaces no tocante à largura de banda e atraso de propagação, com base na legenda da Figura 5.1.

Com relação à conectividade dos nós clientes à rede, simularam-se redes locais através das quais os nós clientes estabeleceram conexões com os servidores. Para isto, foram geradas 27 sub-grafos aleatórios de um grafo completo, com 12 vértices (nós roteadores residenciais, por exemplo) e estabeleceu-se uma aresta entre dois vértices (enlace entre roteadores) da seguinte forma: iterou-se os 12 roteadores dois-a-dois, sorteando-se um número  $x \in [0, 1]$ . Com base no valor de  $x$ , decidiu-se com probabilidade de 60 % estabelecer um enlace entre os dois roteadores correspondentes à cada iteração. Neste contexto, assegurou-se o estabelecimento de pelo menos um enlace para cada roteador e determinou-se uma largura de banda de *download* de 1 Mbps e atraso de propagação de 1 ms entre todos os enlaces dessas

<sup>1</sup>Rede GÉANT é a rede de pesquisa e educação pan-europeia, que interliga as Redes Nacionais de Pesquisa e Educação da Europa (NRENs) atualmente com 41 roteadores. Para mais informações, acesse: <http://www.geant.net/>.

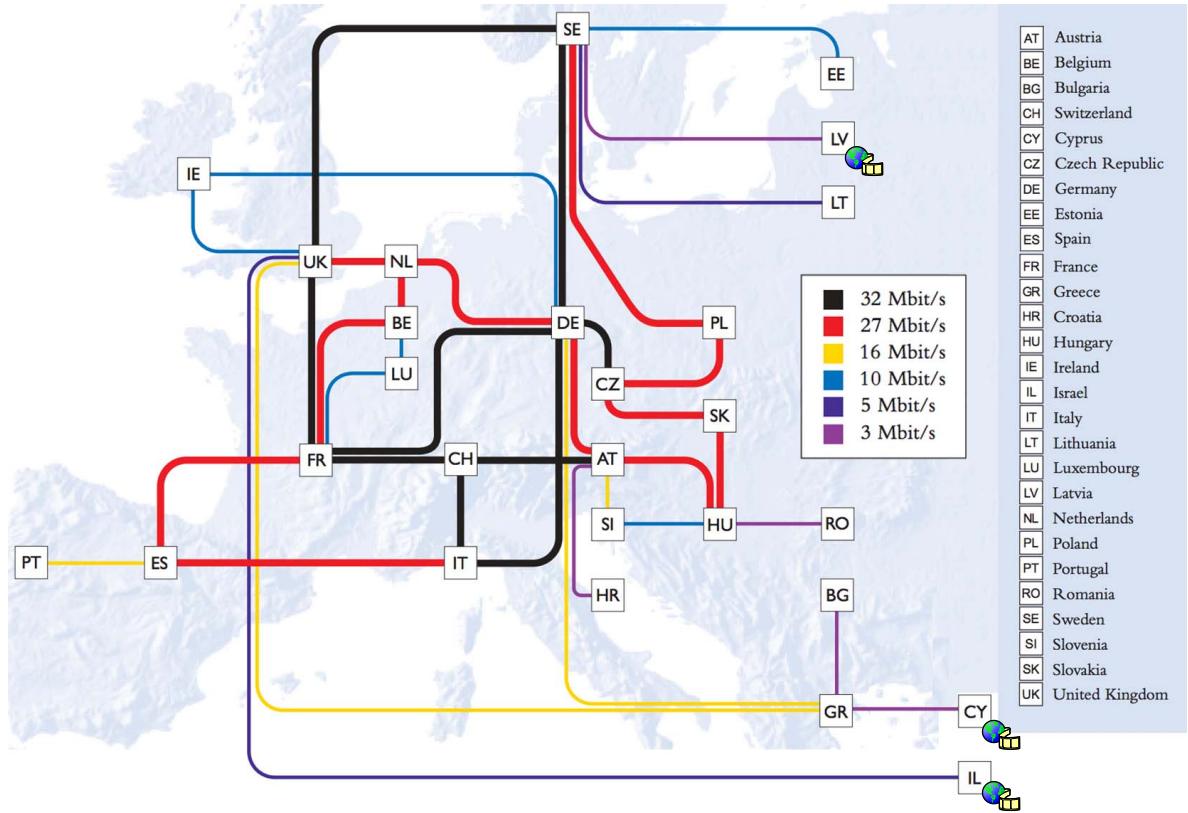


Figura 5.1: Versão adaptada do *backbone* da rede GÉANT, com larguras de banda modificadas para ser utilizada no experimento.

redes locais geradas. Em seguida, para cada roteador do *backbone* ilustrado na Figura 5.1, estabeleceu-se um enlace de *download* de 1 Mbps com um dos roteadores de uma das redes locais geradas (escolhida aleatoriamente dentre as 27).

Por fim, os nós clientes foram distribuídos nas redes locais de forma sequencial, um para cada roteador até distribuir todos os clientes nas redes locais. Por exemplo, supondo-se um tratamento com 500 nós clientes, o nó cliente 1 foi conectado ao nó roteador 2, o nó cliente 2 foi conectado ao nó roteador 2 e assim sucessivamente até atingir o 27º roteador (conectando-se o nó cliente 27), recomeçando-se a contagem do número de roteadores até atingir o número máximo de nós clientes determinado no respectivo tratamento. Desta forma, todos os sistemas foram submetidos às mesmas condições de distribuição dos nós cliente.

Como resultado dessa estratégia, simulou-se uma rede constituída por 324 roteadores, utilizada na execução de todos os tratamentos do experimento. Com isto, garantiu-se que todos os sistemas avaliados fossem submetidos à mesma topologia de rede e alocação dos nós clientes.

### 5.2.3 Definição das variáveis

As variáveis foram definidas em 3 categorias: independentes, fatores e dependentes.

#### Variáveis independentes:

Na Tabela 5.1, apresentam-se as variáveis independentes utilizadas no experimento, com base na topologia da rede apresentada anteriormente.

Tabela 5.1: Variáveis independentes utilizados no experimento.

Parâmetros	Valores
Dinâmica da rede ( <i>churn</i> ) <sup>1</sup>	RandomChurn
Número total de roteadores	324
Número de nós roteadores ( <i>backbone</i> )	27
Número de nós roteadores nas LANs	12
Largura de banda entre LANs	2 Mbps
Atraso de propagação das redes locais	1 s
Atraso máximo aceitável para reprodução	5 s
Tempo de simulação de cada ensaio	900 s
Tamanho do <i>buffer</i> (roteadores)	100 pacotes (em média, 7s da mídia)
Tamanho máximo do pacote de dados	1500 Bytes
Taxa de upload dos roteadores das LANs	512 Kbps, 1 Mbps, 2 Mbps, 3 Mbps
Perda de pacote nos roteadores <sup>2</sup>	3 % de probabilidade de perder 5 % da fila
Tipo da mídia <sup>3</sup>	MPEG-4 Part II

<sup>1</sup> Para mais detalhes, consultar Seção 5.2.5.

<sup>2</sup> Em caso de perda dos 5 % dos pacotes na fila do roteador, os pacotes são escolhidos aleatoriamente, função acionada com o *churn*.

<sup>3</sup> Para mais detalhes, consultar Seção 5.2.7.

Salienta-se que foram alocadas baixas larguras de banda de *upload* e heterogêneas para os nós clientes se comparadas à taxa média de bits da mídia transmitida nos ensaios. Isto significa que todos tem uma baixa capacidade de compartilhar conteúdos com outros nós clientes parceiros, portanto transmitir pacotes de dados na direção *upstream*, ou seja, ao servidor, pode tornar o desempenho da aplicação bastante reduzido, quando necessário transmitir pacotes de controle nessa direção. Cenários como estes são típicos na Internet.

#### Fatores:

Na Tabela 5.2, apresentam-se os fatores considerados no experimento. Os tratamentos foram determinados pelo produto cartesiano desses fatores.

Tabela 5.2: Fatores consideradas no experimento.

Fatores	Valores
Número de nós servidores	1, 3, 5
Número de nós clientes <sup>1</sup>	500; 1.500; 15.000; 30.000; 60.000; 80.000

<sup>1</sup> Todos os clientes solicitaram a mídia nos primeiros 200 s de cada ensaio. Mais detalhes na Seção 5.2.5.

### Variáveis dependentes:

As principais métricas para medir um sistema de distribuição de mídias ao vivo podem ser organizadas em três categorias [40, 256], apresentadas a seguir.

1. *Qualidade de serviço à aplicação*: avaliam-se o atraso para iniciar a reprodução da mídia após um cliente requisitá-la ao servidor (ST); o índice de continuidade (IC); e a distorção do conteúdo em comparação ao original (DI). Como ilustra-se na Figura 5.2, a variável ST é o tempo transcorrido entre um nó cliente requisitar a mídia, receber os primeiros pacotes de dados até ser capaz de produzi-los. O valor da variável IC corresponde à razão entre o número de pacotes de dados da mídia entregues ao nó cliente antes do momento de reproduzi-los e o número total de pacotes de dados transmitidos. Por fim, o valor da variável DI corresponde à razão entre o número de quadros com erros e o número total de quadros disponíveis para serem entregues em um certo período. Para efeito de cálculo das variáveis IC e DI, considerou-se apenas o período em que cada nó cliente permaneceu conectado à rede. Por exemplo (Figura 5.2), se um nó cliente 1 se conectou no instante 14 s, começou a receber os pacotes de dados reproduzíveis no instante 19 s ( $ST=4$ ) e foi desconectado no instante 49 s, considerou-se apenas os pacotes de dados entre os instantes 19 s e 48 s para calcular os valores de CI e DI do nó cliente 1. Além disso, podem ocorrer erros nos quadros de vídeo por dois motivos:

- (a) *Atraso de chegada*: o nó cliente recebe pacote(s) de dados que corresponde a um quadro de vídeo correto, porém não é possível reproduzi-lo por ter chegado após o instante certo para a sua reprodução);
- (b) *Devido às perdas de dependência*: o nó cliente recebe pacotes de dados contendo partes de um quadro de vídeo, porém não é possível decodificá-lo devido

à dependência de outro(s) quadro(s) indisponíveis, por exemplo, a perda de um quadro do tipo I em um GoP para o tipo de compressão MPEG.

Sendo assim, pode-se afirmar que a métrica de distorção está intimamente relacionada à métrica Índice de Continuidade. A distorção representa a qualidade da mídia reproduzida ao usuário final, avaliando-se falhas nos quadros do vídeo recebido em relação ao original. Já o índice de continuidade é uma métrica que contabiliza o tempo em que os nós clientes não conseguiram sequer receber pacotes de dados da mídia, ou seja, interrupção completa da reprodução do conteúdo multimídia. Portanto, a interrupção na reprodução do vídeo resulta em 100 % de distorção dos quadros durante o referido período. A computação da distorção foi simples: todo pacote de dados contém o instante, duração e a quantidade de dados que devem ser reproduzidos, calculando-se a distorção com base na quantidade de dados que deveriam ter sido reproduzidos e o instante em que o pacote de dados alcançou o nó cliente. Por exemplo, se um pacote de dados que contém 1000 Bytes a ser reproduzido entre o instante 10 s e o instante 15 s alcançou o nó cliente no instante 12 s, considerou-se uma distorção de 40 %, pois teoricamente a cada segundo 200 Bytes de dados deveriam ter sido reproduzidos, porém 400 Bytes chegaram expirados.

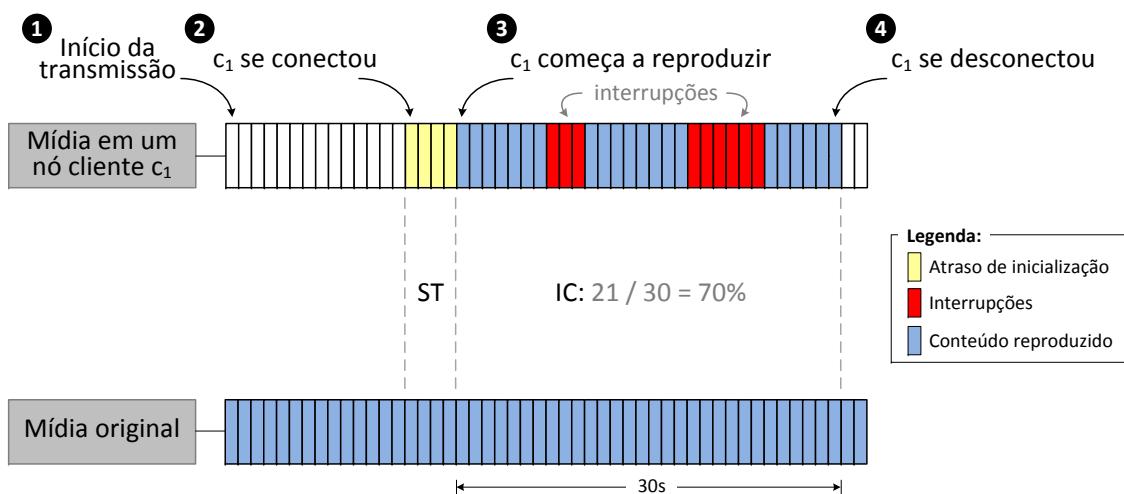


Figura 5.2: Exemplo de cálculo das variáveis dependentes ST, IC. Nesse caso, o valores ST = 4 s e IC = 70 %.

2. *Sobrecarga de controle*: avalia-se a quantidade de pacotes de controle (PC) transmi-

tidos por um protocolo durante o tempo de simulação (contagem dos pacotes que não transportam dados da mídia), considerando-se 0,5 cada pacote de *piggyback*.

Com base nessas métricas, determinou-se as variáveis dependentes, apresentadas na Tabela 5.3.

**Tabela 5.3:** Variáveis dependentes (respostas) consideradas no experimento.

Variáveis dependentes	Símbolo
Atraso de inicialização do fluxo	ST
Índice de continuidade (%)	IC
Distorção do vídeo (%)	DI
Número de pacotes de controle	PC

#### 5.2.4 População e amostras

Constituiu-se a população por dados coletados durante a execução dos ensaios de acordo com às variáveis dependentes apresentadas na Tabela 5.3, com amostras coletadas a cada segundos. Como a duração de cada ensaio foi de 900 s, coletou-se 900 amostras e o valor final de cada variável dependente em cada ensaio foi determinado pela média aritmética das respectivas amostras.

#### 5.2.5 Tratamentos

Na Tabela 5.4, apresentam-se os tratamentos considerados no experimento, definidos com base na combinação dos fatores apresentados na Tabela 5.2. Definiu-se como as unidades experimentais o GMTP, o Denacast e o CCN-TV, comparados em confrontos dois-a-dois, fixando-se o GMTP, em execuções não simultâneas. Nesse contexto, executaram-se 3963 ensaios distribuídos em 18 tratamentos, 1321 ensaios para cada sistema estudado. Na coluna  $n_t$ , apresenta-se a quantidade de repetições de cada tratamento. Além disso, os nós servidores foram dispostos nas redes com menor taxa de transmissão e/ou maior atraso de programação, que correspondem aos roteadores IL, LV e CY, ilustrados na Figura 5.1. Para consultar valores exatos de largura de banda e atraso de propagação, consulte a Tabela A.1.

Com relação a execução de cada tratamento, executaram-se 50 ensaios iniciais de cada sistema estudado, obtendo-se assim 50 amostras para cada variável dependente. Em se-

Tabela 5.4: Tratamentos executados no experimento.

Trat. #	Número de nós servidores (conectado(s) a)	Número de nós clientes	$n_t$
1		500	58
2		1.500	61
3		15.000	56
4	1 (IL)	30.000	51
5		60.000	84
6		80.000	51
7		500	112
8		1.500	79
9		15.000	73
10	2 (IL, LV)	30.000	88
11		60.000	95
12		80.000	86
13		500	96
14		1.500	107
15		15.000	53
16	3 (IL, LV, CY)	30.000	56
17		60.000	62
18		80.000	53

guida, calculou-se a média dessas amostras e, para realizar comparações com 95 % de certeza, calculou-se a quantidade total de ensaios ( $n_t$ ) de cada tratamento a fim de atingir este nível de confiança. Para isto, calculou-se a quantidade de ensaios necessários para obter 95 % de nível de confiança com base em duas médias (das amostras iniciais)  $\mu_1, \mu_2$  de cada variável dependente, fixando-se  $\mu_1$  como a média das variáveis dependentes do GMTP e  $\mu_2$  a média ou do Denacast ou do CCN-TV. Sendo assim, a quantidade total de ensaios de cada tratamento foi determinado por  $n_t = \max(n_{ST}, n_{IC}, n_{DI}, n_{PC}) + 1$ , onde os valores  $n_{ST}, n_{IC}, n_{DI}, n_{PC}$  foram obtidos através da inequação de proporcionalidade para comparar duas alternativas [257]. Por exemplo, se para o confronto GMTP vs. Denacast obteve-se  $n_t = \max(52, 67, 93, 64) + 1$  e no confronto GMTP vs. CCN-TV obteve-se  $n_t = \max(55, 71, 75, 56) + 1$ , assumiu-se  $n_t = 94$ . Ou seja, repetiu-se 94 vezes o mesmo tratamento para todos os sistemas estudados. Na Seção A.4, apresentam-se mais detalhes sobre o cálculo de  $n_t$ .

Por fim, para a execução de cada ensaio para todos os tratamentos, independente do sistema a ser executado, determinou-se o seguinte:

1. Configurou-se todos os nós clientes para enviar a requisição da mídia a um servidor escolhido aleatoriamente (distribuição uniforme), de modo que se conectaram os nós aos mesmos servidores. Sendo assim, garantiu-se que todos os sistemas avaliados foram submetidos às mesmas quantidades de requisições ao(s) nó(s) servidor(es).
2. Definiu-se a taxa de *upload* de cada nó cliente com base em uma escolha aleatória entre as seguintes opções: 512 Kbps, 1 Mbps, 2 Mbps e 5 Mbps. No gráfico da Figura 5.3, representa-se a distribuição das taxas de *upload* pela quantidade de nós. Dessa forma, assegurou-se que os nós clientes foram submetidos as mesmas capacidades de transmissão, independente do sistema avaliado.

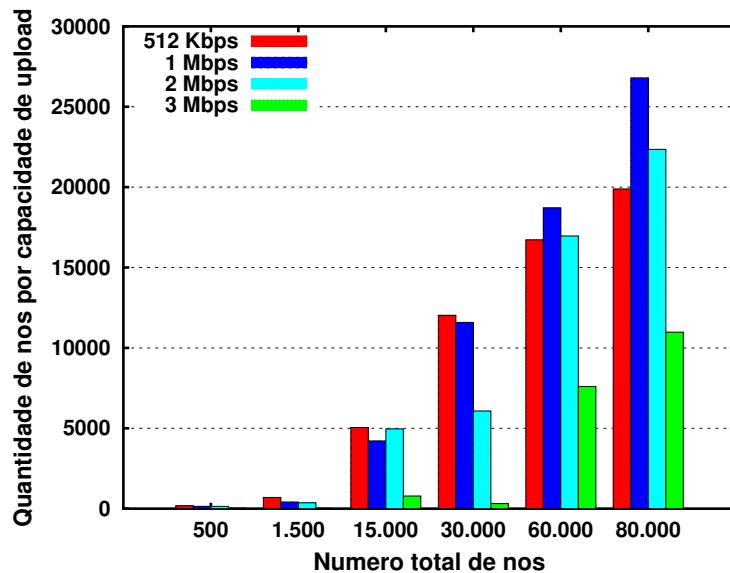


Figura 5.3: Distribuição da quantidade de nós clientes nos primeiros 200 s de simulação. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema.

3. Todos os nós clientes requisitaram a mídia nos primeiros 200 s. Por exemplo, no gráfico da Figura 5.4, representa-se a distribuição de conexão à rede para o ensaio 1 de todos os tratamentos representada. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema. Dessa forma, garantiu-se que os sistemas foram submetidos às mesmas condições iniciais de conexão. Na Seção A.2, discute-se mais detalhes sobre essa distribuição.
4. A função de *churn* foi acionada no instante 400 s. Por exemplo, no gráfico da Figura 5.5, representa-se a distribuição de conexões e desconexões para o ensaio 1 de

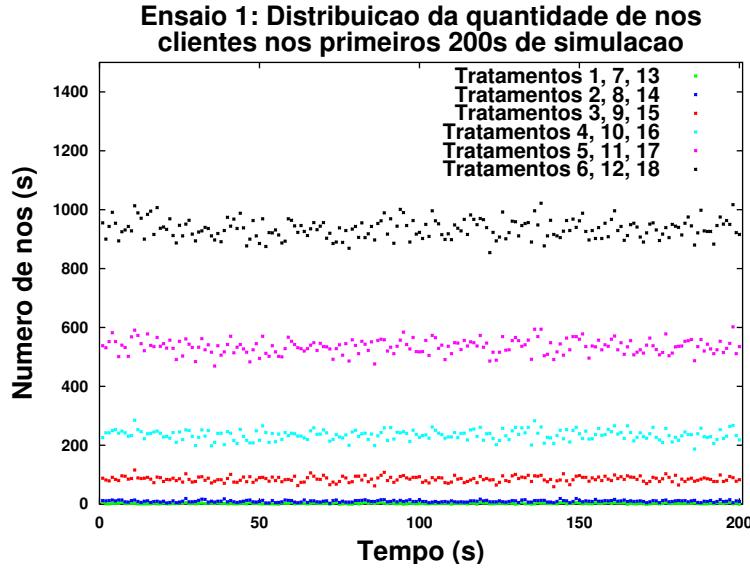


Figura 5.4: Distribuição da quantidade de nós clientes nos primeiros 200 s de simulação. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema.

todos os tratamentos. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema. Com isto, garantiu-se que todas os sistemas avaliados foram submetidos às mesmas condições de dinâmica da rede. Na Seção A.3, discute-se mais detalhes sobre essa distribuição.

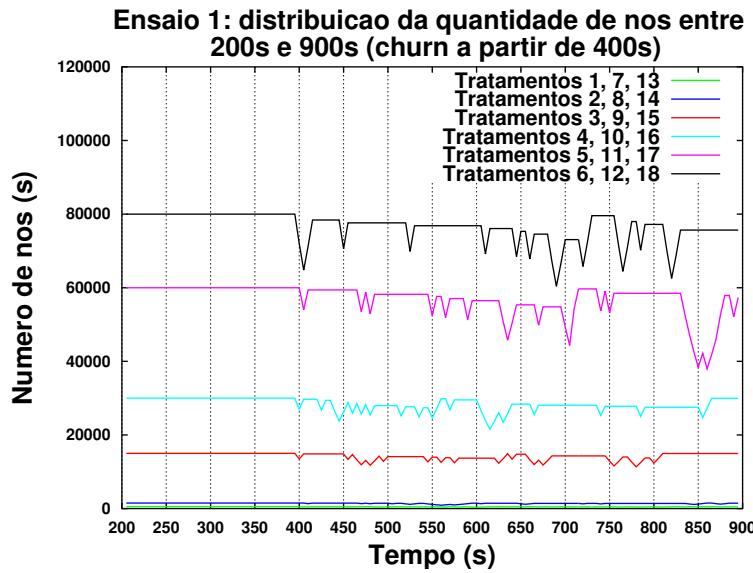


Figura 5.5: Distribuição da quantidade de nós clientes após os 200 s de simulação, com variação a cada 5 s, para o ensaio 1 de todos os tratamentos. Cada ensaio teve uma distribuição diferente, mas igual para a execução de cada sistema.

## 5.2.6 Instrumentação

Com relação à instrumentação, utilizou-se OMNet++ [258, 259], um arcabouço para construção de simuladores de rede. Nesse contexto, utilizaram-se dois simuladores: o OverSim [260] e o CCN-Sim [261]. No OverSim, utilizaram-se as implementações do sistema Denacast [262] e a do GMTP [263] (implementado no contexto deste trabalho), ao passo que no CCN-Sim, utilizou-se a implementação do CCN-TV [264]. Na Figura 5.6, ilustra-se a tela principal do OMNet++ da execução de um dos ensaios com 1 servidor e 1.500 clientes.

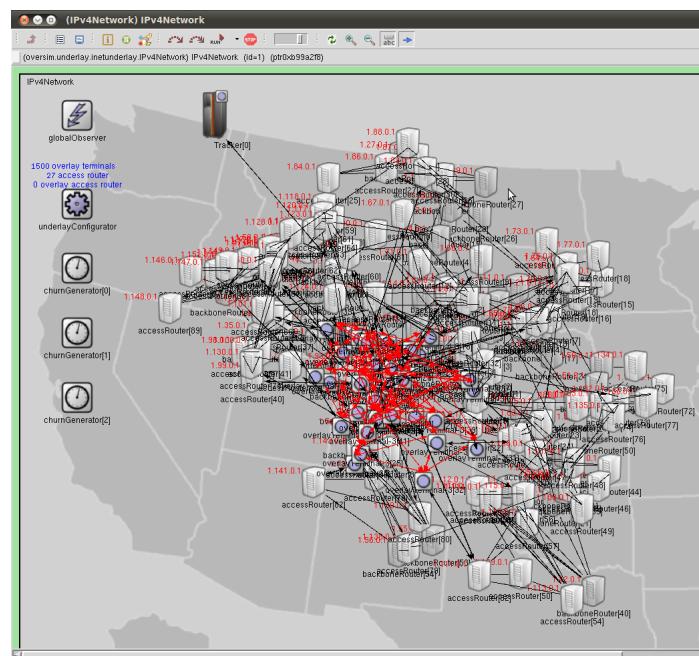


Figura 5.6: OMNet++ com a execução de um dos ensaios com 1 servidor e 1.500 clientes.

### **5.2.7 Formato da mídia**

Na Tabela 5.5, apresentam-se as propriedades da mídia utilizada no experimento [265].

Tabela 5.5: Propriedades da mídia transmitida.

Propriedades	Valores
Mídia sintetizada	<i>Star Wars IV</i>
Codec do vídeo	MPEG-4 Part II
Número de quadros	<i>25fps</i>
Número de quadros em GoP	12
Média VBR	<i>2 Mbps</i>
Tamanho total da mídia	<i>122,61 MB (1-900 s)</i>

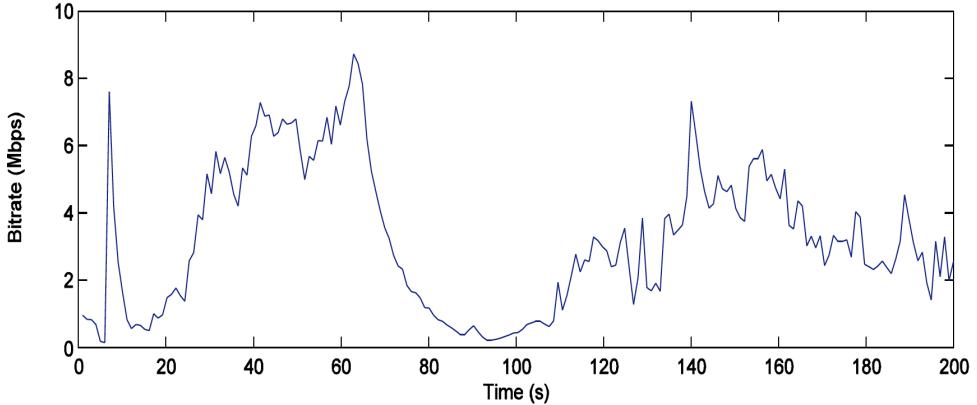


Figura 5.7: Taxa de bits variáveis dos primeiros 200 s da mídia utilizada no experimento (*Star Wars IV*).

Com base na metodologia apresentada nesta seção, coletaram-se as amostras para as variáveis dependentes e realizou-se uma análise dos dados a fim de provar a hipótese enunciada na Seção 5.2.1.

## 5.3 Resultados e Discussões

Nesta seção, apresentam-se os resultados e discussões dos confrontos GMTP vs. Denacast e GMTP vs. CCN-TV. A discussão foi organizada de acordo com as métricas apresentadas na Seção 5.2.3, ou seja, atraso de inicialização, índice de continuidade, distorção da mídia recebida por cada nó cliente e sobrecarga de controle. Em todas as figuras apresentadas a seguir, ilustram-se gráficos onde se observam as evoluções dos sistemas estudados para 1, 2 e 3 nós servidores pela quantidade total de nós clientes, a saber, 500, 1.500, 15.000, 30.000, 60.000 e 80.000. As medições exatas para as métricas estudadas e os intervalos de confiança ilustrados nos gráficos estão disponíveis na Seção A.5.

Em geral, observa-se superioridade de desempenho do GMTP frente aos outros sistemas estudados, apesar de empates técnicos entre o GMTP e o CCN-TV em alguns tratamentos, como discute-se a seguir, especialmente até 30.000 nós clientes e 1 e 3 nós servidores. No caso do Denacast, observa-se claramente que as métricas estudadas foram muito ruins em comparação ao GMTP e ao CCN-TV. Nesse contexto, constatou-se que o Denacast se apresentou como linha base no experimento executado, servindo-se para entender os limites do estado da prática em distribuição de mídias ao vivo, uma vez que se trata de uma proposta significativamente difundida na Internet. De fato, parte dos resultados apresentados sobre o

Denacast neste trabalho confirmam os resultados apresentados por seus autores em [6], com a diferença de que aqui se trabalhou com uma quantidade de nós clientes quase 60 vezes mais se comparado à referência citada. Enfim, as disputas mais acirradas ocorreram no confronto GMTP vs. CCN-TV, exceto com relação à sobrecarga de controle, onde o CCN-TV obteve um desempenho inferior ao Denacast.

### 5.3.1 Atraso de inicialização

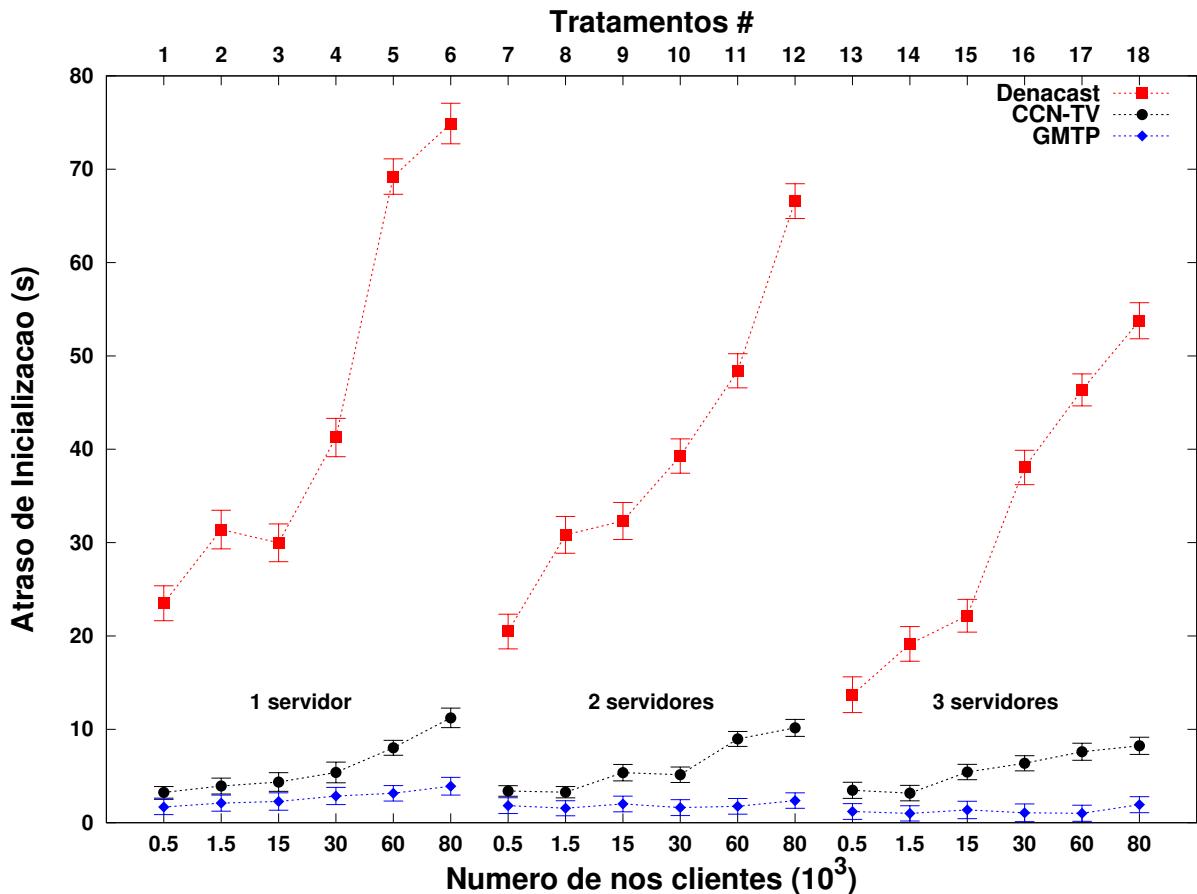


Figura 5.8: Resultado dos tratamentos (1 – 18) para a métrica *Atraso de Inicialização*.

Na Figura 5.8, observa-se uma tendência de crescimento linear com relação ao atraso de inicialização em todos os tratamentos executados com o GMTP, à medida que se aumenta o número de nós clientes, com a curva mais acentuada nos tratamentos com apenas 1 nó servidor (tratamentos 1-6), sendo suavizada com a evolução dos tratamentos. Com o aumento do número de nós servidores, observam-se melhorias no desempenho dos três sistemas estudados, constatando-se uma maior dependência de nós servidores por parte do Denacast.

No caso do GMTP e do CCN-TV, constataram-se melhorias à medida que foram adicionados mais nós servidores, porém com uma menor variação proporcional ao número de nós clientes. Por exemplo, no tratamento com 3 nós servidores, dobrando-se o número de nós clientes de 30.000 para 60.000, o desempenho do GMTP foi praticamente invariável. O que contribui para este fato é que quando há mais nós clientes requisitando a mesma mídia, aumentam-se as chances de um nó cliente obter os pacotes de dados a partir dos *caches* dos nós repassadores, reduzindo-se substancialmente a média para o atraso de inicialização.

Para o caso do CCN-TV, apesar de ser possível constatar um comportamento similar, o CCN-TV teve um desempenho pior se comparado ao GMTP, especialmente a partir de 15.000 nós clientes. A razão para essa redução do desempenho por parte do CCN-TV com relação ao GMTP é que as redes CCNs utilizam uma estratégia de acesso ao conteúdo exclusivamente através do seu nome, indexado em uma tabela chamada FIB e compartilhada entre os roteadores da rede, como discutiu-se na Seção 3.3. Como não há um nó servidor de referência no processo de requisição de um conteúdo, não tem como garantir que uma requisição rapidamente alcançará o nó servidor [225]. No caso do GMTP, transmitem-se as requisições para os nós servidores e se houver um nó roteador já repassando o conteúdo de interesse para outros nós clientes, tal roteador intercepta a requisição e responde ao nó cliente requisitante, o que reduz os pedidos de conexão ao nó servidor e reduz a média do atraso de inicialização. Caso a interceptação não ocorra, o processo de registro de participação ocorre, mapeando-se as rotas até os nós servidores a fim de executar a função de seleção de nós parceiros baseada nas intersecções de rotas, tal como explicado na Seção 4.3.3. Obviamente que para as primeiras requisições transmitidas pelos nós clientes aos servidores GMTP tem um custo de atraso de inicialização maior, porém a média do atraso de inicialização reduz drasticamente, devido ao aumento das possibilidades de intersecções de rotas e portanto o aumento das parcerias entre nós roteadores nas próximas requisições para obter a mesma mídia.

Para o caso do Denacast, um nó recém integrado à rede DONet pode levar muito tempo para obter os primeiros blocos de vídeo e assim iniciar a reprodução do conteúdo ao usuário final. Como pode-se observar através do gráfico ilustrado na Figura 5.8, para obter os primeiros pacotes de dados multimídia, no tratamento 1, um nó cliente Denacast gastou, em média, 23,51 s, 23,51 s no tratamento 13 (melhor caso) e 74,89 s no tratamento 6 (pior caso).

Isto acontece 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 disponível, 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*, então há um estabelecimento de conexão (registro de participação) para em seguida receber os pacotes de dados. Como dito, um registro de participação pode ser interceptado e portanto reduz-se o métrica atraso de inicialização.

Para sumarizar os resultados apresentados com relação à métrica atraso de inicialização, o GMTP obteve um desempenho de 93,23 % melhor que o Denacast e 46,79 % melhor que o CCN-TV. Para obter este valor, calculou-se a média ponderada das diferenças dos intervalos de confiança para cada tratamento (considerando-se o limite superior do GMTP e o limite inferior dos outros sistemas confrontados). Atribuíram-se pesos entre 100 e 0,125, de acordo com a dificuldade do tratamento com relação aos fatores considerados no experimento. Por exemplo, atribuiu-se peso 100 ao tratamento 6 por se apresentar com a maior quantidade de nós clientes (80.000) e com a menor quantidade de nós servidores (apenas 1); o tratamento 5 recebeu peso 75 porque tem a mesma quantidade de nós servidores, mas 75 % da quantidade total de nós clientes com relação ao tratamento 6 (60.000). Considerando esse raciocínio, o tratamento 12 recebeu peso 33,33, pois apesar de também se apresentar com 80.000 nós clientes, utilizaram-se 3 nós servidores; e assim sucessivamente até o tratamento 18, que recebeu peso 0,125, por ser o mais simples, com apenas 500 nós clientes e 3 nós servidores. Este mesmo procedimento foi utilizado para sumarizar das demais métricas apresentadas a seguir.

### 5.3.2 Índice de continuidade

Apesar de importante, a métrica atraso de inicialização cobre apenas os instantes iniciais dos ensaios. Diferentemente disso, a métrica índice de continuidade é ainda mais importante e reveladora, pois mapeia o desempenho dos sistemas estudados durante todo o ensaio, a partir do instante em que os nós clientes começam a receber o primeiro pacote de dados, além de

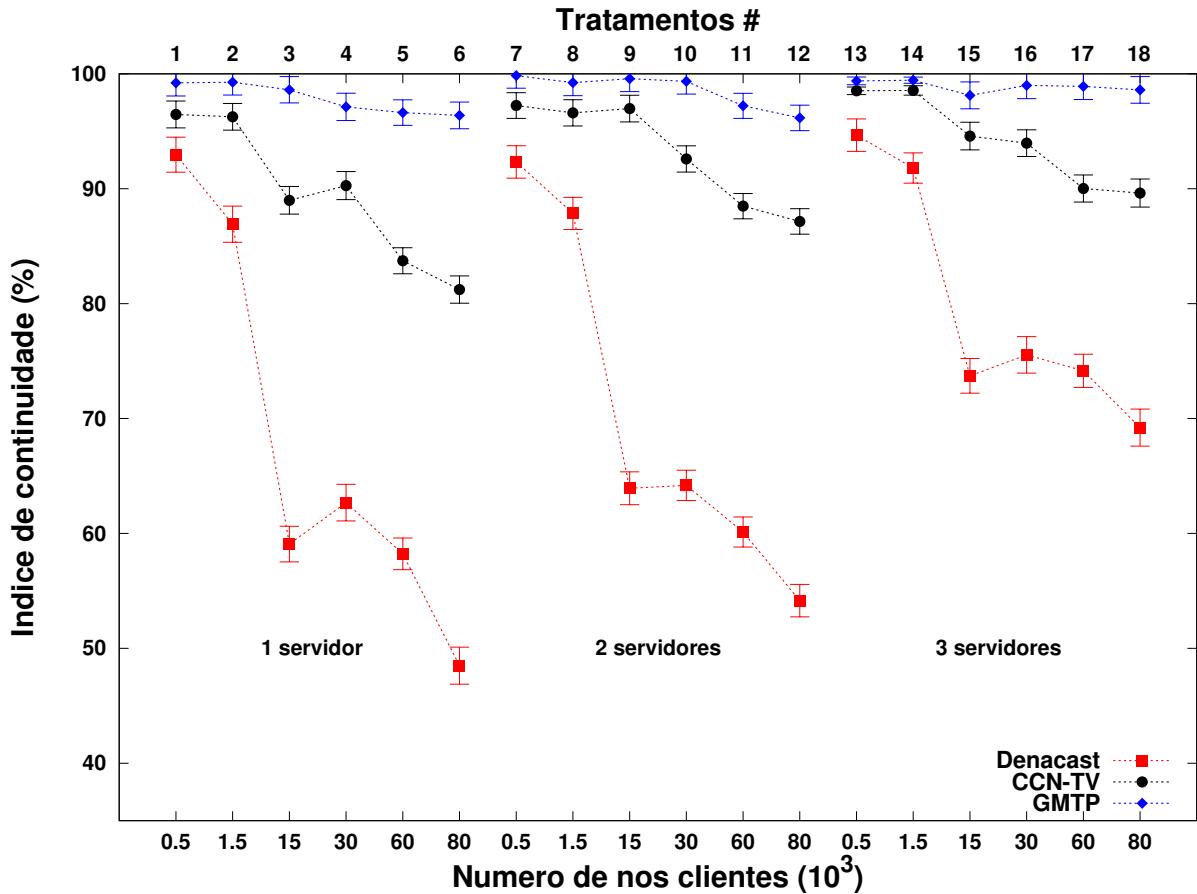


Figura 5.9: Resultado dos tratamentos (1 – 18) para a métrica *Índice de Continuidade*.

contabilizar as variações das métricas devido à dinâmica da rede, acionada no instante 400 s. Sendo assim, avaliar a métrica índice de continuidade permite entender o impacto na qualidade de experiência do usuário final com relação à quantidade e duração das interrupções nas reproduções da mídia em cada nó cliente. Especificamente, os tratamentos 13 e 14 foi uma disputa bastante acirrada entre o GMTP vs. CCN-TV, sendo necessárias mais repetições dos respectivos tratamentos (consultar Tabela 5.4 para maiores detalhes).

Na Figura 5.9, apresenta-se a evolução da métrica índice de continuidade dos sistemas estudados para cada tratamento executado. Observa-se novamente melhor desempenho do GMTP frente ao Denacast e o CCN-TV, com a iminência de empates técnicos no confronto GMTP vs. CCN-TV quando se utilizaram até 30.000 nós clientes.

No caso do GMTP, o desempenho foi melhor porque com o aumento do número de nós clientes requisitando a mesma mídia, aumentam-se também as chances de outros nós clientes obterem as partes da mídia a partir de um *cache* de roteador e não diretamente do servidor.

Este fenômeno também ocorreu com o CCN-TV, devido à estratégia de *cache* de conteúdo nos roteadores, porém não supera o GMTP por dois motivos.

1. em uma rede CCN, deve-se propagar e/ou atualizar a FIB nos roteadores da rede à medida que novos pacotes de dados multimídia se tornam disponíveis nos nós servidores. O problema é que essa estratégia demanda tempo, já que os dados de uma mídia ao vivo são gerados instantes antes de sua transmissão. Então, primeiramente, em vez do servidor transmitir imediatamente o conteúdo gerado, este deve primeiro transmitir meta-dados sobre o conteúdo que está disponível. A consequência crítica disso é que os pacotes de dados relacionados à mídia chegam aos nós clientes mais atrasados ou sejam declarados como perdidos pelos nós clientes, devido à expiração de produção, impactando diretamente no índice de continuidade.
2. Um dos grandes apelos das redes CCNs é a não necessidade de se manter estados de conexões nos nós servidores, transferindo-se as principais responsabilidades do transporte e controle de recursos para os nós clientes. Como discutiu-se na Seção 5.1.2, essa estratégia é mais eficiente em cenários de aplicação de conteúdo já armazenado, mas considerando-se mídias ao vivo o desempenho das CCNs reduz consideravelmente. Como as CCNs utilizam um modelo de serviço do tipo *pull*, os nós clientes precisam continuamente transmitir pacotes de interesse para continuar recebendo o conteúdo multimídia, como discutiu-se nas Seções 3.3. Com resultado, aumenta-se a quantidade de pacotes de controle na rede (pacotes de interesse), os quais são submetidos às taxas de transmissões de *upload* (em geral menores do que as de *download*), aumentando-se as chances de perdas desses pacotes ou significativos atraso de entrega, o que consequentemente atrasa a resposta de pacotes de dados da mídia.

Dessa forma, observa-se uma vantagem do GMTP em se utilizar o modelo de serviço *push/pull* em comparação às redes CCNs (*pull*) para distribuir mídias ao vivo, pelo menos no que diz respeito a referida métrica.

Outra observação importante é que a adição de mais nós servidores aumenta o número de rotas com roteadores realizando *cache* dos pacotes de dados da mídia e, nesse caso, o GMTP tem uma forma mais simples e eficiente de se aproveitar desse fato. Como discutiu-se na Seção 4.5, no GMTP se utiliza uma estratégia de seleção de nós parceiros baseada na

capacidade do canal de transmissão entre os nós roteadores conhecidos, resultando no escalonamento de melhores parcerias entre os nós roteadores, reduzindo-se o tempo de entrega dos pacotes de dados e consequentemente melhorando-se a métrica índice de continuidade.

De forma similar para a métrica atraso de inicialização, o Denacast obteve o pior resultado se comparados aos outros dois sistemas para a métrica índice de continuidade. Nesse caso, observam-se curvas acentuadas de redução dessa métrica, apesar de leves melhorias (se considerar a proporcionalidade do número de nós clientes) quando se aumentou o número de nós servidores e também quando se dobrou o número de nós clientes de 15.000 para 30.000. Observa-se que, quando se aumentou o número de nós clientes de 1.500 para 15.000, ou seja, dez vezes mais, houve uma redução considerável no desempenho do Denacast para a métrica índice de continuidade. Apesar das melhorias observadas entre os tratamentos 9 e 10 e os tratamentos 15 e 16, mesmo aumentando-se a quantidade de nós servidores o Denacast não obteve um desempenho satisfatório. Com isto, pode-se concluir que o Denacast pode apresentar problemas de escalabilidade a partir de uma certa quantidade de nós clientes e com a dinâmica da rede.

Ainda no contexto do Denacast, seu principal limitante é a estratégia de sub-fluxos, pois se adiciona complexidade à solução sem necessariamente resultar em melhor desempenho quanto à escala de distribuição da mídia. Em [221], os autores do CoolStreaming discutem que aumentar a quantidade de número de sub-fluxos não necessariamente melhoraram algumas métricas, como o índice de continuidade. Em um experimento em larga escala em uma rede real, com pico de 40.000 nós clientes e utilizando-se 24 servidores auxiliares (que funcionaram apenas como nós transmissores), os autores observaram que a partir de 12 sub-fluxos, a métrica índice de continuidade não evolui positivamente.

Para sumarizar os resultados apresentados com relação à métrica índice de continuidade, o GMTP obteve um desempenho de 37,88 % melhor que o Denacast e 9,14 % melhor que o CCN-TV.

### 5.3.3 Distorção

Uma outra métrica fundamental para medir a qualidade de experiência dos usuários finais é chamada de distorção, ou seja, uma comparação da mídia recebida pelos nós clientes com relação à mídia original transmitida pelo nó servidor. Basicamente, as distorções podem

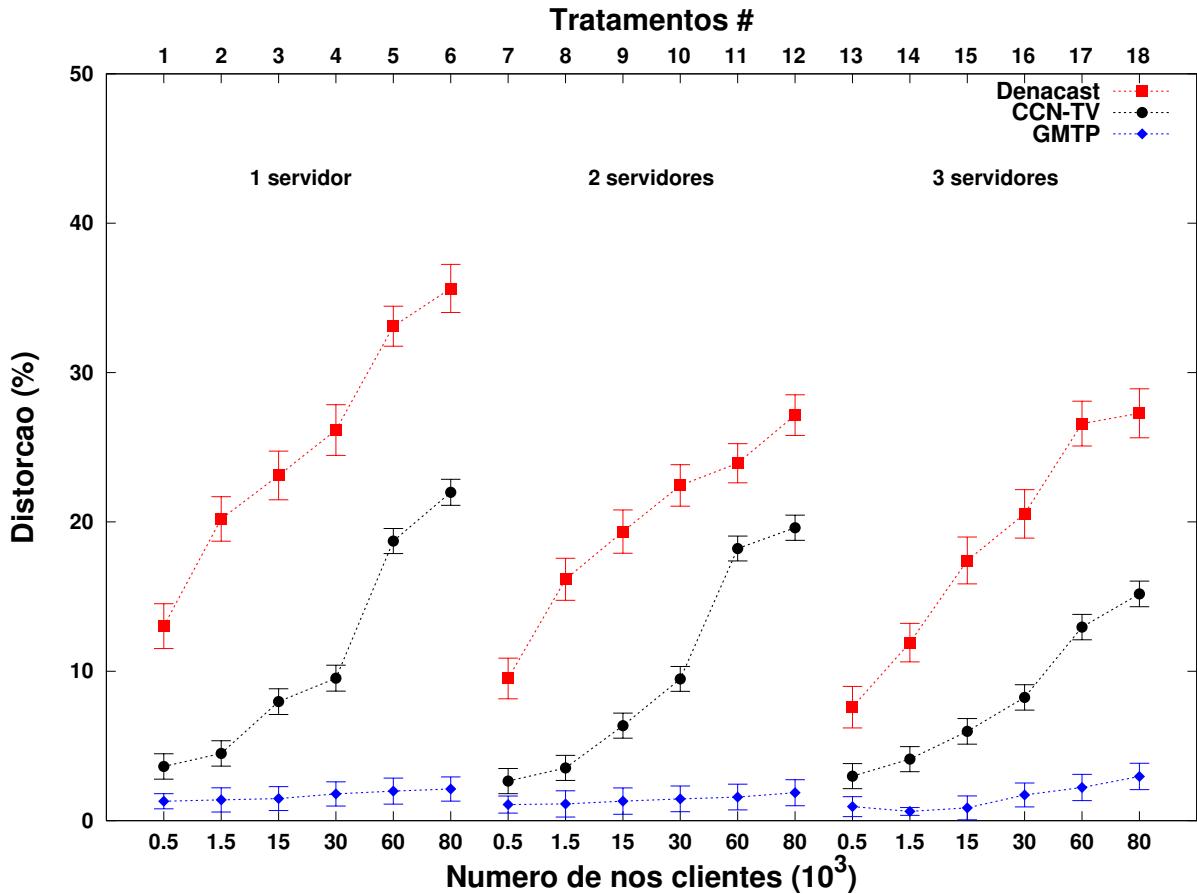


Figura 5.10: Resultado dos tratamentos (1 – 18) para a métrica *Distorção*.

ocorrer devido à chegada atrasada dos pacotes de dados ou por perdas de dependência, como discutiu-se na Seção 5.2.3. Apesar das perdas de pacotes de dados (por exemplo, perdas de quadros MPEG do tipo P ou B), ainda sim pode ser possível reproduzir certas partes do vídeo, porém com possíveis defeitos visuais. Sendo assim, se um ou mais pacotes de dados contiverem quadros do vídeo do tipo I, isto implica em interrupção da produção da mídia, impactando negativamente no índice de continuidade. Mas se os pacotes de dados contiverem quadros do vídeo do tipo P e/ou B, o impacto é na qualidade visual da mídia.

Na Figura 5.10, apresenta-se a evolução dos sistemas estudados para a métrica distorção. Evidencia-se que o GMTP obteve melhor desempenho se comparado ao Denacast e ao CCN-TV. A seguir, apresentam-se as principais justificativas desse resultado.

Os principais motivos que fizeram o GMTP obter melhor desempenho frente ao CCN-TV para a métrica índice de continuidade se estende ao caso da distorção da mídia, com a diferença que ainda foi possível reproduzir parte de um pacote em atraso. O modelo de

serviço *push/pull*, a estratégia de escalonamento de parcerias baseada na capacidade atual de transmissão dos canais e a baixa necessidade de troca de pacotes de controle influenciam diretamente na qualidade da mídia recebida pelos nós clientes. Pode-se afirmar que o CCN-TV perdeu mais pacotes de dados se comparado ao GMTP também devido às estratégias de controle de congestionamento baseadas em janela deslizantes – o controle ocorre não somente no envio dos pacotes dos dados por parte dos nós servidores, mas no envio dos pacotes de interesse por parte dos nós clientes.

Ainda no contexto do desempenho das CCNs, observa-se no gráfico da Figura 5.10 que a curva correspondente ao CCN-TV evidencia um desempenho similar ao GMTP nos tratamentos com poucos nós clientes, degradando-se a medida que o número de nós clientes aumenta. Um dos motivos para esse desempenho é que o algoritmo de controle de congestionamento empregado nas CCNs limita a quota de transmissão de pacotes de interesse de acordo com a largura de banda disponível, impedindo o aumento da janela e portanto a capacidade que o nó cliente tem de transmitir múltiplos pacotes de interesse. No contexto das redes CCN, transmitir múltiplos pacotes de interesse não significa que os pacotes de dados correspondentes chegaram mais rapidamente. O motivo do baixo desempenho do CCN-TV no contexto da qualidade da mídia, explica-se pelo fato de que, quando um pacote de dados a ser reproduzido não chega ao nó cliente devido à perda, a única forma de tentar obtê-lo é retransmitindo um pacote de interesse. O problema é decidir em que instante se deve retransmitir um pacote de interesse para que seja suficiente tentar receber os pacotes de dados correspondentes. Além disso, as operações de *cache* do CCN-TV não oferecem benefícios significativo se os nós clientes transmitirem mais pacotes de interesse para obter um mesmo dado. A principal razão para isso é que as partes da mídia armazenadas nos *caches* dos roteadores perdem sua efetividade depois que expiram. Sendo assim, com diversas tentativas de envio de pacotes de interesse, as respostas contendo os pacotes de dados podem até alcançar os nós clientes interessados, porém há riscos de não ser possível reproduzi-los a tempo.

Além disso, salienta-se que o uso de *cache* em CCN para o caso de distribuição de mídias ao vivo pode garantir apenas uma pequena redução na taxa de perda de pacotes. De fato, pelos estudo apresentado em [126], os autores constataram que a função de *cache* das CCNs, na presença de fluxos de dados de aplicações de tempo real, não representa uma função importante para tal tipo de rede. Argumenta-se que o uso da PIT apenas ajuda a reduzir

uma rajada de requisições ao nó servidor para obter um determinado conteúdo de interesse comum. Sendo assim, o que mais influencia na redução das distorções da mídia em CCN, é a capacidade de transmissão e recepção dos nós clientes. Observa-se que os dados apresentados aqui foram obtidos sem o uso de qualquer tráfego de segundo plano, portanto, o desempenho das redes CCNs pode ser ainda pior em um cenário real de acesso à rede por parte dos usuários finais. Isto porque, em geral, costumam-se realizar acessos a diferentes serviços de rede simultaneamente, o que pode reduzir a capacidade de transmissão de cada fluxo de dados partindo de um mesmo sistema final.

Enfim, com relação ao CCN-TV, suas funções de controle de congestionamento e *cache* são limitadas se comparadas à forma como estas duas funções são desempenhadas pelo GMTP. Desta forma, aumenta-se significativamente a distorção do conteúdo recebido pelos nós clientes, pois as respostas contendo pacotes de dados solicitados não alcançam os nós servidores em tempo hábil, apesar do pacote de interesse alcançar o nó roteador que mantém os dados da mídia. Considerando-se as devidas proporções de cenário adotado, os resultados apresentados sobre a distorção da mídia confirmam os resultados apresentados por seus autores em [126].

Com relação ao Denacast, um fator limitante para o aumento na distorção da mídia está relacionado com a estratégia de escalonamento de nós clientes parceiros. Nesse contexto, considera-se apenas o nó que detém um determinado conteúdo de interesse e sua capacidade de *upload* para escalar os nós clientes parceiros, o que ocorre com base em uma escolha aleatória. O problema é que, devido à dinâmica da rede, a capacidade de *upload* dos nós podem mudar com frequência, resultando em seleção de nós parceiros de baixa qualidade, o que impacta diretamente na qualidade da mídia recebida pelos nós clientes. Enfim, considerando-se as devidas proporções de cenário adotado, os resultados obtidos para o Denacast confirmam os resultados apresentados por seus autores em [6].

Para sumarizar os resultados apresentados com relação à métrica distorção, o GMTP obteve um desempenho de 25,26 % melhor que o Denacast e 13,03 % melhor que o CCN-TV, considerando-se os limites mínimos e máximo do GMTP e dos seus oponentes, respectivamente.

### 5.3.4 Sobrecarga de controle

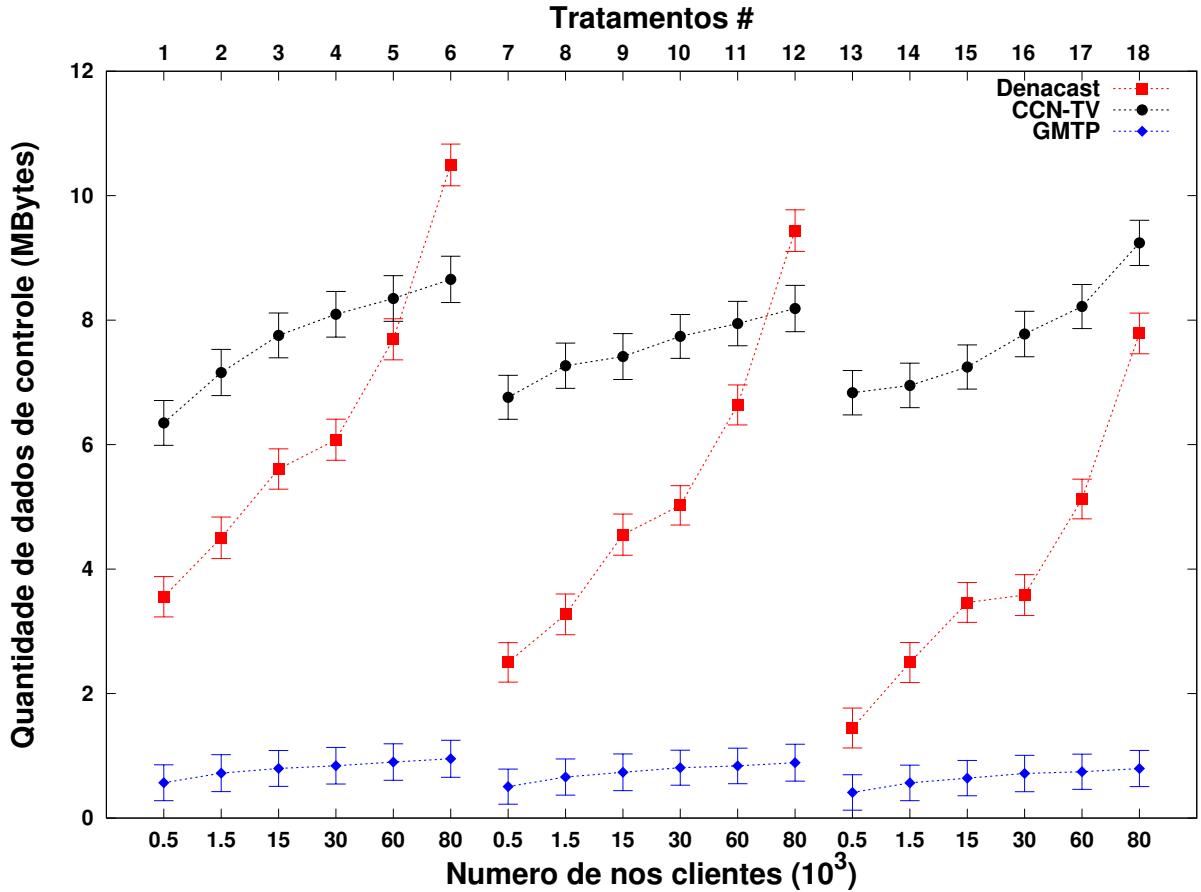


Figura 5.11: Resultado dos tratamentos (1 – 18) para a métrica *Número de Pacotes de Controle*.

Diferentemente das métricas anteriores, que são relacionadas à qualidade de serviço oferecida à aplicação, a sobrecarga de controle é uma relação de proporção entre a quantidade de pacotes de controle (sem dados de aplicação) e a quantidade de dados da aplicação que precisam ser transmitidos. Nesse contexto, um sistema *A* é melhor que um sistema *B* se *A* transmite menos pacotes de controle que *B* para disseminar a mesma quantidade de dados da aplicação. Dessa forma, mede-se a quantidade de pacotes de controle transmitido e contabiliza-se a quantidade de dados de controle transmitidos através da rede, com base no tamanho do cabeçalho de cada tipo de pacote. Como discutiu-se na Seção 4.1.2, o pacote de controle do GMTP tem tamanho de 32 Bytes (Seção 4.1.2); o da CCN 40 Bytes [266] e do Denacast 36 Bytes, considerando-se os cabeçalhos da aplicação e do TCP [221] para esse último caso. Como no GMTP emprega-se um método de cabeçalho variável, seu valor para

a métrica sobrecarga de controle foi obtido com base nos tipos dos pacotes transmitidos ao longo do ensaio, os quais estão detalhados no Apêndice B.

Neste contexto, na Figura 5.11, apresenta-se a evolução dos sistemas estudados para a métrica sobrecarga de controle. Observam-se empates técnicos entre o CCN-TV e o Denacast em alguns tratamentos, como o tratamento 5 e o 18, com apenas um tratamento onde o CCN-TV foi melhor que o Denacast, ao passo que o GMTP obteve o melhor desempenho entre os três sistemas estudados.

O GMTP transmite uma menor quantidade de dados de controle porque, além de possuir um cabeçalho fixo menor que os seus concorrentes, somente aumenta a quantidade de dados de controle quando necessário. Por exemplo, o processo de registro de participação é o instante em que se transmitem mais dados de controle, porém este procedimento ocorre somente no começo da transmissão. Além disso, alguns pacotes transmitidos para registro de participação são interceptados pelos nós roteadores, reduzindo-se antecipadamente o tráfego de controle. Para a topologia de rede utilizada no experimento, o tamanho máximo que um pacote GMTP atingiu foi de 240 *Bytes*<sup>2</sup>, porém somente 27 pacotes de controle desse tipo foram transmitidos em cada ensaio, dos quais 12 foram interceptados antes de atingir o tamanho de 240 *Bytes*. Sendo assim, considerando o GMTP, quando se utilizam pacotes de controle maiores do que tamanho fixo, a quantidade é bastante reduzida (algumas dezenas). Outros motivos para o melhor desempenho do GMTP frente a seus concorrentes estão relacionados aos métodos empregados para distribuir o conteúdo multimídia, discutidos a seguir.

No caso do CCN-TV, seu pior desempenho se deve às muitas retransmissões de pacotes de controle, devido às perdas no canal de *upstream*. Como em redes CCN os nós clientes precisam transmitir pacotes de interesse periodicamente, aumentam-se as chances de perdes-los, principalmente em condições de congestionamento intenso na rede. Por exemplo, no tratamento 6, considerando-se os valores das métricas anteriores, para alcançar tal desempenho, os nós dos clientes CCN-TV tiveram que transmitir 19,8 % da quantidade total de dados da aplicação, ou seja, foram necessários transmitir 24,28 *MB* de pacotes de interesse para um total de 122,61 *MB* da mídia. Considerando-se os resultados de todos os tratamentos apresentados em 5.11 e as devidas proporções de cenário, os resultados apresentados aqui

---

<sup>2</sup>Para os pacotes dos tipos *GMTP-Request-Reply* e *GMTP-Route-Notify*.

confirmam os resultados apresentados pelos autores do CCN-TV em [126].

Além dos motivos apresentados anteriormente, na prática, uma rede CCN pode aumentar significativamente a sobrecarga de controle porque precisa verificar a expiração do *cache*. No caso do GMTP, as partes da mídia armazenadas nos *caches* dos roteadores são substituídas à medida que novos pacotes de dados são gerados pelos nós servidores e chegam aos nós roteadores. Além disso, no GMTP, propõe-se que seja possível uma aplicação informar quanto de espaço em *buffer* é necessário para transmitir a mídia, ao passo que cada roteador no caminho aloca o tamanho solicitado, desde que não ultrapasse um valor limite determinado pelo usuário administrador do roteador.

No caso do Denacast, observa-se uma maior dependência com relação à quantidade de nós servidores disponíveis e o número de nós cliente interessados pela mídia, apesar de seu desempenho ter sido melhor que o CCN-TV. Quanto maior é o número de nós clientes, trocam-se mais dados de controle, apesar de se constatar uma discreta diminuição do crescimento da curva, à medida que se adicionaram mais nós servidores na rede. Esse desempenho do Denacast ocorre em virtude de algumas estratégias adotadas em seu protocolo. A primeira é o mecanismo de sub-fluxo, exigindo-se que os nós clientes troquem mais pacotes de controle para transmitir os mapas de *buffer*. A segunda está relacionada aos mapas de *buffer* que, apesar de poderem ser compartilhados entre os nós clientes, cada compartilhamento requer uma conexão TCP, o que eleva sobremaneira a quantidade de conexão e consequentemente a quantidade de dados de controle, à medida que se aumenta o número de nós clientes. E, por fim, a terceira, que está relacionada à manutenção da rede *mesh* constituída pela CoolStreaming para distribuir os dados multimídia. Os nós clientes no CoolStreaming devem frequentemente se comunicar com seus pares atuais e candidatos, a fim de manter a rede de sobreposição atualizada. Se isto não for feito, os nós clientes não poderão encontrar rapidamente novos nós parceiros em caso de finalização de uma das suas parcerias atuais [267].

Atualmente, existem pesquisas relacionadas a reduzir a quantidade de pacotes de interesse em transmissões de dados através das redes CCNs. A questão primordial em se tratando de mídias ao vivo é que os nomes dos conteúdos são gerados instantes antes de sua transmissão; então, mesmo que se utilize uma estratégia de indexação sequencial dos próximos pacotes de dados, ainda assim se faz necessário que os nós clientes continuem transmitindo os pacotes de interesse periodicamente.

Para sumarizar os resultados apresentados com relação à métrica sobrecarga de controle, o GMTP obteve um desempenho de 177,05 % melhor que o Denacast e 165,94 % melhor que o CCN-TV. Esses valores expressivos também evidenciam a importância de se propor uma solução nas camadas inferiores da pilha TCP/IP, podendo melhorar sobremaneira o consumo da rede com demasiados dados de controle, muitas vezes transmitidos pelas aplicações para suprir limitações de serviços que deveriam estar disponíveis nas camadas corretas. Além disso, o valor bastante expressivo de ganho obtido pelo GMTP frente ao CCN-TV demonstra a importância de se propor um protocolo exclusivo para distribuição de fluxos de dados multimídia, devido à peculiaridade de requisitos.

## 5.4 Sumário do Capítulo e dos Resultados

Neste capítulo, apresentaram-se os resultados obtidos neste trabalho com base em confrontos do GMTP frente ao Denacast (estado da prática) e ao CCN-TV (estado da arte), seguindo duas perspectivas: projeto e desempenho.

Na análise de projeto, discutiu-se brevemente sobre o projeto GMTP, contrapondo-se suas funcionalidades com as das respectivas propostas citadas. Já na análise de desempenho, estudaram-se as referidas propostas sobre a perspectiva da qualidade de serviço (atraso de inicialização, índice de continuidade e distorção) oferecida à aplicação e a sobrecarga de controle necessária para distribuir um conteúdo multimídia com requisitos de tempo real. Neste caso, apresentou-se a metodologia empregada no experimento realizado e em seguida estudou-se o desempenho dos sistemas com base nos valores aferidos para as principais métricas. A topologia de rede e os parâmetros determinados condizem com cenários da realidade, o que culminou em testes mais reais para entender os limites dos sistemas estudados, constatando-se que em níveis críticos de congestionamento da rede e com um número elevado de nós clientes, as soluções disponibilizadas na camada de aplicação, como o Denacast, têm um desempenho significativamente inferior àquelas baseadas no núcleo da rede, como é o caso do CCN-TV e do GMTP.

De fato, as disputas mais acirradas ocorreram entre o GMTP e o CCN-TV, evidenciando-se que a arquitetura e decisões de projeto, como o modelo de serviço utilizado, são aspectos que podem fazer diferenças significativas no desempenho dos sistemas para distribuir mídias

Tabela 5.6: Sumário dos ganhos do GMTP sobre o Denacast e o CCN-TV.

<b>Métricas</b>	<b>Ganhos do GMTP por Confrontos</b>	
	GMTP vs. Denacast	GMTP vs. CCN-TV
Atraso de Inicialização	93,22 %	46,79 %
Índice de Continuidade	37,88 %	9,14 %
Distorção	25,26 %	13,03 %
Sobrecarga de Controle	177,05 %	165,94 %
<b>Média ponderada (Pesos<sup>1</sup>: 2, 3, 3, 1)</b>	<b>61,44 %</b>	<b>36,18 %</b>

<sup>1</sup> Para obter os referidos valores, calculou-se a média ponderada com pesos (2, 3, 3, 1) para as métricas atraso de inicialização, índice de continuidade, distorção e sobrecarga de controle, respectivamente. Esses pesos foram definidos com base na importância de cada métrica no processo de distribuição de mídias ao vivo.

ao vivo. Nesse contexto, sumarizam-se na Tabela 5.6, os ganhos do GMTP sobre o Denacast e o CCN-TV discutidos ao longo deste capítulo. Considerando-se os resultados para cada uma das métricas, com base em simulação de um cenário real de rede, reservando-se suas devidas proporções, pode-se afirmar que o GMTP apresenta um melhor desempenho se comparado às soluções do estado da prática e uma proposta do estado da arte.

Sendo assim, como o GMTP é uma instância da hipótese enunciada e perseguida neste trabalho, pode-se afirmar com 95 % de nível de confiança que seu uso resulta na distribuição mais eficiente de mídias ao vivo em comparação com o estado da prática e da arte. Especificamente, melhora-se o desempenho na distribuição de mídias ao vivo em 61,44 %, comparando o GMTP aos sistemas baseados em técnicas de distribuição de mídias ao vivo adotadas no estado da prática, ao passo que melhora-se em 36,18 % o desempenho dos sistemas GMTP em comparação às técnicas de distribuição de conteúdo com suporte da rede instanciadas com base nos conceitos da rede CCN, uma das propostas mais importantes do estado da arte para distribuição de conteúdos através da Internet.

# Capítulo 6

## Considerações Finais

Neste trabalho, apresentou-se um novo protocolo de distribuição de mídias ao vivo chamado *Global Media Transmission Protocol* (GMTP). O GMTP atua nas camadas de transporte e de rede (*cross-layer*) da pilha TCP/IP, seguindo uma arquitetura de rede híbrida P2P/CDN, através da qual transmitem-se pacotes de dados de um ou mais sistemas. Para isto, constituem-se redes de favores formadas por roteadores, que cooperam entre si a fim de entregarem o conteúdo multimídia de interesse comum aos seus clientes. As parcerias entre os roteadores são determinadas pelos servidores, com base na capacidade dos canais já em uso para disseminar o conteúdo, obtidas e atualizadas periodicamente por meio de um algoritmo de controle de congestionamento assistido pela rede. Nesse contexto, apresentou-se o GMTP em duas perspectivas: projeto e desempenho.

Na perspectiva do projeto, avaliaram-se os principais requisitos de aplicação e as técnicas de distribuição de mídias ao vivo utilizadas no estado da prática e da arte, ajustando-as de forma a permitir a interoperabilidade dos sistemas a fim de fazer melhor uso dos recursos compartilhados de rede. Aliado aos ajustes realizados, definiram-se estratégias como o registro de participação; o compartilhamento de conteúdo por meio da intersecção de rotas; a generalização da descrição da mídia; a definição das parcerias entre os roteadores com base na largura de banda disponível dos canais em uso para distribuir o conteúdo multimídia; a eleição de nós para auxiliar no controle de congestionamento; e funções para validação dos pacotes de dados. Com essas estratégias, sustenta-se a proposta de que o uso do GMTP mitigará a construção de sistemas interoperáveis, separando-se a forma de transportar os dados multimídia e delegando-se à aplicação a responsabilidade de reproduzir o conteúdo aos

usuários finais. Além disso, a forma de expor os serviços do GMTP à aplicação possibilita a fácil migração das aplicações existentes.

Na perspectiva do desempenho, estudou-se o GMTP em confrontos com as propostas Denacast/CoolStreaming e CCN-TV, avaliando-se as principais métricas de qualidade de serviço: atraso de inicialização, índice de continuidade, qualidade da mídia recebida pelos nós clientes e sobrecarga de controle. Com base nos resultados obtidos por meio da simulação de um cenário real, demonstrou-se que o GMTP obteve um desempenho 61,44 % melhor que o Denacast/CoolStreaming e 36,18 % melhor que o CCN-TV.

## 6.1 Conclusões

Com base nas limitações das propostas para distribuição de mídias ao vivo e seus problemas apresentados na Seção 1.2 e no Capítulo 3, as principais conclusões obtidas neste trabalho são enumeradas a seguir.

1. O problema da interoperabilidade entre os sistemas de distribuição de mídias ao vivo pode ser resolvido através da abstração e da otimização das principais técnicas para distribuição de mídias ao vivo, propondo-se novos métodos de conectividade multiponto e compartilhamento de conteúdo, tudo disponibilizado na camada de transporte e rede, sem sofrer a influência da camada de aplicação. Como consequência, conclui-se também que:
  - (a) o conceito de socket P2P empregado no GMTP permite o uso transparente do modo de transmissão multicast por parte da aplicação, compartilhando-se os pacotes de dados entre múltiplos sistemas interessados por um mesmo conteúdo, o que reduz o consumo de recursos de rede e melhora a qualidade de serviço oferecida às aplicações;
  - (b) diversas funções para distribuição de mídias ao vivo realizadas pelas aplicações se tornam obsoletas (ou foram movidas e adaptadas em uma camada mais apropriada), tais como seleção de clientes parceiros; tolerância à desconexões; estratégias para atenuar o *churn*; obtenção e gerenciamento de informações que

auxiliam os algoritmos de seleção de nós com base no contexto (localização, perfil do usuário, estimativas e medições de custos entre nós, etc.); algoritmos de incentivo à cooperação e para inibir a participação de nós *free-riders*; algoritmos para monitorar a capacidade de recepção dos clientes e adaptação de fluxo com base em tal informação, além do gerenciamento de canais de transmissão com diferentes níveis de qualidade da mídia; e algoritmos para avaliar e/ou inibir ataques de poluição;

- (c) como consequência do item anterior, o desenvolvimento de sistemas se torna mais eficiente (mais rápido e com menor chances de erros) devido ao emprego de funções de software relacionadas à distribuição de mídias ao vivo desacopladas da aplicação. Esta decisão evita o uso de diferentes bibliotecas de software que, apesar de facilitar o desenvolvimento, não resolve o problema da interoperabilidade discutido anteriormente e são incapazes de usar toda a capacidade de transmissão que a rede pode oferecer.
2. As soluções de aplicação são incomparáveis às soluções baseadas no núcleo da rede para distribuição de mídias ao vivo, por isso observou-se confrontos mais acentuado entre o GMTP e o CCN-TV. Neste contexto, conclui-se que o GMTP obteve um desempenho superior a CCN pelos seguintes motivos:
- (a) O modelo de serviço *push/pull* empregado no GMTP é mais apropriado para distribuir conteúdos ao vivo, ou seja, para sistemas que tenham como um dos principais requisitos a entrega do conteúdo em um curto espaço de tempo e que não necessite de entrega confiável. A principal consequência positiva desse modelo de serviço é que os sistemas finais interessados em obter um determinado conteúdo não precisam transmitir periodicamente requisições para continuar recebendo um determinado conteúdo, como ocorre na rede CCN.
  - (b) Apesar dos pesquisadores que propuseram o CCN alegarem que não é preciso um sistema final conhecer a localização de um determinado conteúdo de interesse (endereço IP do servidor, por exemplo), neste trabalho, conclui-se o contrário. Ou seja, ainda convém manter a identificação do sistema final de destino, visto

que se pode otimizar métricas importantes, como o atraso de inicialização, como demonstrou-se neste trabalho.

- (c) Devido ao modelo de serviço empregado na rede CCN, os nós clientes não conseguem saber se ainda é possível recuperar um determinado conteúdo ou não. Isto impacta diretamente na tomada de decisão se ainda faz sentido retransmitir um pacote de interesse correspondente a um pacote de dados a ser reproduzido ou se não faz mais sentido aguardar sua chegada. Além disso, os pacotes de interesse são dependentes da capacidade de *upload* do cliente, o que pode reduzir a quantidade de pacotes a serem recebidos por cliente, mesmo havendo largura de banda de *download* disponível e suficiente.
3. Apesar do custo computacional de fazer com que todos os roteadores alterem a requisição para adicionar seu identificador a fim de conhecer as rotas de rede utilizadas para disseminar um determinado conteúdo, consegue-se:
- (a) reduzir a quantidade de conexão ao nó servidor;
  - (b) entregar os pacotes de dados mais rapidamente aos sistemas finais por meio da formação de parcerias por intersecção;
  - (c) melhorar a toma de decisão para propor parcerias entre os roteadores, com base em informações mais precisas sobre a capacidade de transmissão do canal.
4. Apesar de constatações em trabalhos anteriores de que os algoritmos de controle de congestionamento assistidos pela rede melhoram o desempenho da rede nas transmissões de dados, neste trabalho, reforçou-se não somente este fato, mas também conclui-se que a aplicabilidade desse tipo de algoritmo pode servir para melhorar o desempenho no processo de distribuir de mídias ao vivo, utilizando-se informações mais acuradas para a formação de parcerias. Especificamente, utilizar informações explícitas sobre o congestionamento da rede para permitir um servidor determinar parcerias com base nas rotas já em uso para disseminar um conteúdo multimídia, melhoram-se métricas importantes, como o índice de continuidade e a qualidade do conteúdo entregue aos sistemas finais. Nesse contexto, utilizar o RCP e adaptá-lo para o cenário

de distribuição de mídias ao vivo foi uma escolha adequada, pois se trata de uma proposta invariante à distribuição do tamanho do fluxo. Isto é, independente do tamanho do fluxos de dados, se é de curta ou de longa duração, o RCP compartilha o canal de forma mais eficiente porque emula o PS independente de quais e quantos são os fluxos, estabilizando-se a taxa de transmissão dos sistemas finais mais rapidamente, o que ocorre a cada intervalo médio de RTT e de acordo com a demanda real em um determinado instante. Além disso, o RCP aliado ao conceito de sub-fluxo proposto neste trabalho, é simples e praticável, pois a taxa de transmissão ofertada é definida sem o conhecimento do número exato de fluxos, baseando-se somente no tamanho da fila, no tráfego agregado de entrada e no RTT do tráfego passando através do roteador.

5. Com base em revisões da literatura e nos resultados apresentados, não convém utilizar CCN para distribuição de mídias ao vivo. Nesse contexto, as propostas existentes ainda são imaturas devido às incertezas do projeto arquitetural e remoção de informações importantes, como os endereços dos sistemas finais. Especificamente, mudar de um espaço de endereçamento de um bilhão de endereços IPs para pelo menos um trilhão de nomes de conteúdos pode causar problemas críticos de desempenho devido à necessidade de ter que processar tamanha quantidade de dados [225]. No caso do GMTP, buscou-se dividir com os servidores a sobrecarga de processamento nodal relacionada às atividades de distribuição de conteúdo, sendo portanto uma abordagem híbrida rede/hospedeiro. Isto ocorre quando a rede gera informações sobre o tráfego de dados e os hospedeiros as utilizam para tomar decisões no processo de constituir e manter a rede de favores, enfraquecendo a relação um-para-um de conectividade ao permitir que os roteadores realizem *cache* dos pacotes de dados para agirem como se fossem os próprios hospedeiros.

## 6.2 Limitações e Trabalhos Futuros

Nessa seção, apresentam-se as limitações e os trabalhos futuros relacionados ao GMTP.

1. Devido à limitação do tamanho dos datagramas IP em 1500 *Bytes*, o funcionamento correto do GMTP é limitado em topologias de rede em que a distância entre dois sis-

temas finais não pode ultrapassar 72 saltos para IPv4 e 44 saltos para IPv6. Apesar dessa limitação, estudos anteriores podem ser úteis para estimar a distância máxima entre dois sistemas finais quaisquer conectados à Internet [268, 269] e investigar alternativas para superar essa limitação do GMTP. Nesse contexto, pode-se fazer uso de fragmentação na camada IP.

2. Sugere-se investigar a possibilidade de reduzir a frequência em que um roteador adiciona seu identificador no pacote de requisição de uma mídia. Esta ação pode aumentar o atraso de processamento nodal e impactar negativamente na qualidade de serviço. Como linha de investigação, pode-se estudar a possibilidade de reduzir a frequência de escrita nos pacotes de requisição com base em um *cache* dos últimos destinos registrados. Por exemplo, quando um pacote de requisição for transmitido pela primeira vez a partir de uma origem a um determinado destino, o roteador adiciona seu identificador no pacote, armazena a identificação da origem e do destino e determina um período em que não adicionará mais seu identificador para este caso.
3. A decisão de utilizar tamanho de cabeçalho variável no GMTP reduz a sobrecarga de controle. Porém, sugere-se investigação sobre o uso de tal abordagem e o aumento do processamento nodal. Nesse contexto, pode-se propor algoritmos mais eficientes para um processamento mais rápido apenas rearranjando os campos de cada tipo de pacote atualmente definido no GMTP ou mesmo melhorando as funções existentes a fim de requerem menos informações de controle por pacote.
4. A função de sub-fluxo empregado no GMTP-UCC (Seção 4.5.1) permite que roteadores com maior capacidade de recepção continuem recebendo um fluxo multimídia em uma taxa possível, mesmo que outros roteadores no caminho até um determinado cliente possua uma capacidade inferior de recepção em um determinado instante. Para permitir esse conceito de sub-fluxo, necessita-se medir o RTT entre dois roteadores que estão repassando um determinado fluxo de dados multimídia. Dessa forma, deve-se aumentar a capacidade de processamento dos roteadores para que possam controlar múltiplos cronômetros. Nesse contexto, sugere-se investigar se o desempenho do GMTP pode reduzir à medida que múltiplos fluxos de dados são transmitidos na rede.

5. Conduzir estudos que explorem outras topologias de rede, incluindo redes móveis, bem como se tráfego de dados em segundo plano pode revelar outros benefícios do uso do GMTP, bem como outras limitações.
6. Implementar o protocolo GMTP e uma aplicação de referência em um sistema operacional e em um roteador. Nesse caso, sugere-se também avaliar o grau de complexidade de adaptar um sistema existente para utilizar o GMTP.
7. Submeter à IETF um documento de RFC sobre o GMTP.

## 6.3 Publicações

1. *GMTP: A Cross-layer Optimized Protocol for Large Scale Distribution of Live Multi-media Content over the Internet.* Referência: [270]<sup>1</sup>
2. *Generalized Connection and Incentives for Supporting CE Devices in P2P/CDN Live Streaming Systems.* Referência: [271]<sup>2</sup>
3. *Global Media Transmission Protocol (GMTP).* I Workshop Pré-IETF do XXXIV Congresso da Sociedade Brasileira de Computação. Referência: [272]<sup>3</sup>
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: [273].
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: [164].
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: [165].

---

<sup>1</sup>A ser submetido para o IEEE/ACM Transactions on Networking (JCR 2.014).

<sup>2</sup>Em processo de revisão para o IEEE Transaction on Consumers Electronics (JCR 1.087).

<sup>3</sup>Em processo de revisão.

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: [166].
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: [274].

## 6.4 Resumo das Contribuições

1. O projeto e a avaliação de um protocolo para distribuição de conteúdos multimídia ao vivo baseado em uma rede de favores constituída entre roteadores.
2. Estudos e discussões sobre a escolha e modificação do *Rate Control Protocol* (RCP) [58] para uso no GMTP, em detrimento às outras opções, como o XCP [53].
3. Os resultados e discussões acerca do GMTP frente a duas proeminentes propostas: o Denacast/CoolStreaming [6] e o CCN-TV [126].
4. A revisão da literatura sobre as principais técnicas para distribuição de mídias ao vivo, bem como as propostas para este fim, trazendo-se à tona os recentes avanços da área de estudo, como os conceitos das redes centradas na informação [48] e os algoritmos de controle de congestionamento assistido pela rede [52].
5. A implementação de artefatos de software que permitem a reprodução do ambiente de estudo, tais como implementação do GMTP no simulador de rede OMNet++ [258,259] e a metodologia detalhada do experimento.
6. As limitações do GMTP e os encaminhamentos da pesquisa sobre futuros avanços do estado da arte.

Ademais, destaca-se a importância e originalidade do presente trabalho por trazer à tona um problema recorrente e uma proposta de solução do uso de um protocolo de transporte e rede, exclusivamente projetado para a distribuição de mídias ao vivo na Internet. Até então,

distribuir conteúdos multimídia na Internet se resumia em sistemas baseados em protocolos tradicionais (TCP, UDP, DCCP, SCTP) ou outros proprietários, em geral, disponíveis na camada de aplicação.

# Bibliografia

- [1] R. Pantos and W. May. HTTP Live Streaming, 10 2013. <http://tools.ietf.org/html/draft-pantos-http-live-streaming-12>. Último acesso: 2 de Maio de 2014.
- [2] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *MultiMedia, IEEE*, 18(4):62–67, April 2011.
- [3] 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: 2 de Maio de 2014.
- [4] 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.
- [5] C. Kulatunga, G. Kandavanam, A.I. Rana, S. Balasubramaniam, and D. Botvich. Hysac: A hybrid delivery system with adaptive content management for iptv networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5, june 2011.
- [6] 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.
- [7] 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.
- [8] Yonghong Tian, J. Srivastava, Tiejun Huang, and N. Contractor. Social multimedia computing. *Computer*, 43(8):27–36, Aug.
- [9] San Murugesan. Understanding Web 2.0. *IT Professional*, 9(4):34–41, 2009.
- [10] Kwei-Jay Lin. Building Web 2.0. *Computer*, 40(5):101–102, 5 2007.
- [11] 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.
- [12] 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).
- [13] 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: 2 de Maio de 2014.
- [14] 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: 2 de Maio de 2014.
- [15] Eric Franchi. Live streaming continues momentum with march madness, 3 2009. <http://www.mediapost.com/publications/article/101750/#axzz200qSKoNN>. Último acesso: 2 de Maio de 2014.
- [16] 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: 2 de Maio de 2014.

- [17] Liz Gannes. The Obama Inauguration Live Stream Stats, 1 2009. <http://gigaom.com/2009/01/20/the-obama-inauguration-live-stream-stats/>. Último acesso: 2 de Maio de 2014.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.

- [26] Jin Li. On peer-to-peer (p2p) content delivery. *Peer-to-Peer Networking and Applications*, 1(1):45–63, 2008.
- [27] 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.
- [28] 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.
- [29] Darshan Purandare and R. Guha. An Alliance Based Peering Scheme for P2P Live Media Streaming. *Multimedia, IEEE Transactions on*, 9(8):1633–1644, 2007.
- [30] A. Vakali and G. Pallis. Content delivery networks: status and trends. *Internet Computing, IEEE*, 7(6):68–74, nov.-dec. 2003.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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.

- [36] 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.
- [37] 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.
- [38] 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.
- [39] Kideok Cho, Hakyoung 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.
- [40] 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, 12 2009.
- [41] 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.
- [42] 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.
- [43] Douglas E. Comer. *Interligação em Redes com TCP/IP: Princípios, Protocolos e Arquitetura*. ELSEVIER, 2 edition, 9 2004.
- [44] 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.

- [45] 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, 5.
- [46] 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.
- [47] 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, 5 2003.
- [48] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katzaros, and G. Polyzos. A Survey of Information-Centric Networking Research. *Communications Surveys Tutorials, IEEE*, PP(99):1–26, 10 2013.
- [49] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A Survey of Information-Centric Networking. *Communications Magazine, IEEE*, 50(7):26–36, 7 2012.
- [50] 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.
- [51] 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.
- [52] 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, 2001.
- [53] Dina Katabi. *Decoupling congestion control and bandwidth allocation policy with application to high bandwidth-delay product networks*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2003.

- [54] 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, 8 2007.
- [55] 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.
- [56] 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.
- [57] 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.
- [58] Nandita Dukkipati. *Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly*. PhD thesis, Stanford University, Stanford, CA, USA, 2008. AAI3292347.
- [59] 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.
- [60] 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.
- [61] 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.
- [62] Yong Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit is Enough. *Networking, IEEE/ACM Transactions on*, 16(6):1281–1294, 12 2008.

- [63] 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.
- [64] 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.
- [65] 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.
- [66] 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.
- [67] 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: 2 de Maio de 2014.
- [68] Marcelo Bagnulo and Nandita Dukkipati. Congestion exposure group, 6 2010. <https://datatracker.ietf.org/doc/charter-ietf-conex/>. Último acesso: 2 de Maio de 2014.
- [69] Philip Eardley, Michalis Kanakakis, Alexandros Kostopoulos, Tapio Levä, Ken Richardson, and Henna Warma. The Future Internet. In John Domingue, Alex Galis, Anastasius Gavras, Theodore Zahariadis, and Dave Lambert, editors, *The Future Internet*, chapter Deployment and Adoption of Future Internet Protocols, pages 133–144. Springer-Verlag, Berlin, Heidelberg, 2011.

- [70] 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.
- [71] K.C. Almeroth. The evolution of Multicast: From the MBone to Interdomain Multicast to Internet2 Deployment. *Network, IEEE*, 14(1):10–20, Jan 2000.
- [72] 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.
- [73] 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.
- [74] R Rodrigues and P Druschel. Peer-to-peer streaming systems. *Communications of the ACM*, 53(10):3–39, 2010.
- [75] 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.
- [76] 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.
- [77] 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.
- [78] 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.

- [79] 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.
- [80] 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.
- [81] 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.
- [82] 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.
- [83] 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.
- [84] 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.
- [85] 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.
- [86] 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.
- [87] 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.
- [88] 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.
- [89] 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.
- [90] Fabio Pianese. A survey of p2p data-driven live streaming systems. *Streaming Media Architectures, Techniques, and Applications: Recent Advances*, 1:295–310, 10 2011.
- [91] Chuan Wu, Baochun Li, and Shuqiao Zhao. Diagnosing Network-Wide P2P Live Streaming Inefficiencies. In *INFOCOM 2009, IEEE*, pages 2731–2735, 4.
- [92] 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.
- [93] 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.
- [94] 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.

- [95] 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.
- [96] 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, 10 2008. IEEE Computer Society.
- [97] Yang Guo, Chao Liang, and Yong Liu. A Survey on Peer-to-Peer Video Streaming Systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, 2008.
- [98] E. Setton, P. Baccichet, and B. Girod. Peer-to-Peer Live Multicast: A Video Perspective. *Proceedings of the IEEE*, 96(1):25–38, 1 2008.
- [99] 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.
- [100] 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.
- [101] 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.
- [102] 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.
- [103] 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, 4 2006.

- [104] 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.
- [105] Qingchao Cai, Xiaolu Zhang, and Xuejie Zhang. Streaming live media over bittorrent. In *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, volume 3, pages 44–49, Jan 2009.
- [106] 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, 6 2010.
- [107] 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.
- [108] 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.
- [109] 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.
- [110] Majed Alhaisoni and Antonio Liotta. Characterization of signaling and traffic in Joost. *Peer-to-Peer Networking and Applications*, 2(1):75–83, 2009.
- [111] 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, 4 2008.
- [112] M. Moshref, R. Motamed, 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.
- [113] 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.
- [114] 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.
- [115] 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.
- [116] Jie Xiong and R.R. Choudhury. PeerCast: Improving link layer multicast through cooperative relaying. In *INFOCOM, 2011 Proceedings IEEE*, pages 2939–2947, 4 2011.
- [117] 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.
- [118] 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.
- [119] 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.
- [120] Roberto Roverso, Sameh El-Ansary, and Seif Haridi. Smoothcache: Http-live streaming goes peer-to-peer. *NETWORKING 2012*, 7290:29–43, 2012.

- [121] P. Baccichet, Jeonghun Noh, E. Setton, and B. Girod. Content-Aware P2P Video Streaming with Low Latency. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 400–403, 7 2007.
- [122] 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.
- [123] 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.
- [124] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP), 3 2006. <http://www.ietf.org/rfc/rfc4340.txt>. Último acesso: 2 de Maio de 2014.
- [125] Anahita Fellah Jahromi. Temporal Scalable Live Video Streaming over Hybrid CDN-P2P Architecture. *International Journal of Computer Applications*, 46(17):14–20, May 2012. Published by Foundation of Computer Science, New York, USA.
- [126] 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.
- [127] 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, 6.
- [128] 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.
- [129] 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.
- [130] 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.
- [131] 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.
- [132] 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.
- [133] 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.
- [134] 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.
- [135] 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.
- [136] Constantinos Vassilakis and Ioannis Stavrakakis. Minimizing node churn in peer-to-peer streaming. *Comput. Commun.*, 33(14):1598–1614, September 2010.
- [137] Yi Cui, Yanchuan Cao, Liang Dai, and Yuan Xue. Optimizing P2P streaming throughput under peer churning. *Multimedia Systems*, 15(2):83–99, November 2008.
- [138] 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, 9.

- [139] 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, 10 2011.
- [140] 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, 9 2009.
- [141] Harsha V. Madhyastha, Ethan Katz-Bassett, Thomas Anderson, Arvind Krishnamurthy, 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.
- [142] 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.
- [143] 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.
- [144] 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.
- [145] 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.
- [146] 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.

- [147] 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.
- [148] 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.
- [149] 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.
- [150] S. Sanghavi and B. Hajek. A new mechanism for the free-rider problem. *Automatic Control, IEEE Transactions on*, 53(5):1176–1183, June.
- [151] 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.
- [152] 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.
- [153] 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>.
- [154] 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.
- [155] 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.

- [156] 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.
- [157] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable Application Layer Multicast. *SIGCOMM Comput. Commun. Rev.*, 32(4):205–217, 8 2002.
- [158] 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, 8 2003.
- [159] 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.
- [160] 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, 1 2004.
- [161] 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.
- [162] Minseok Kwon and Sonia Fahmy. Path-aware Overlay Multicast. *Comput. Netw.*, 47(1):23–45, January 2005.
- [163] 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 212–221, 12 2005.
- [164] 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, 4 2012.

- [165] 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.
- [166] Leandro Melo de Sales, Hyggo Oliveira, Angelo Perkusich, and Rafael A. Silva. Distribuição de Conteúdos Multimídia em Tempo Real com Transporte de Fluxos Controlados e Não Confiável 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).
- [167] 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.
- [168] 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: 2 de Maio de 2014.
- [169] 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.
- [170] 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.
- [171] Sally Floyd. Highspeed tcp for large congestion windows, 12 2003. <http://www.ietf.org/rfc/rfc3649.txt>. Último acesso: 2 de Maio de 2014.
- [172] 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.
- [173] Cheng Peng Fu and Soung C. Liew. TCP Veno: TCP Enhancement for Transmission

- Over Wireless Access Networks. *IEEE Journal On Selected Areas In Communications*, 21(2):216–228, 2 2003.
- [174] 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.
- [175] 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.
- [176] 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.
- [177] 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.
- [178] 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.
- [179] 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.
- [180] 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.
- [181] 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.

- [182] R. Stewart, J. Stone, D. Otis, K. Morneau, H. Schwarzauer, 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: 2 de Maio de 2014.
- [183] 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.
- [184] 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.
- [185] 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.
- [186] 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.
- [187] Larry L. Peterson and Bruce S. Davie. *Computer Networks, A System Approach*. Morgan Kaufmann, 5 ed. edition, 3 2011.
- [188] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Addison Wesley, trad. 3 ed. edition, 2006.
- [189] Luiz Fernando Gomes Soares. *Redes de Computadores: das LANs, MANs e WANs às Redes ATM*. Campus, 2 edition, 1995.
- [190] 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.
- [191] C. Severance. Van Jacobson: Content-Centric Networking. *Computer*, 46(1):11–13, 2013.

- [192] 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.
- [193] Hongfeng Xu, Zhen Chen, Rui Chen, and Junwei Cao. Live Streaming with Content Centric Networking. In *Networking and Distributed Computing (ICNDC), 2012 Third International Conference on*, pages 1–5, Oct.
- [194] 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.
- [195] 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.
- [196] 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.
- [197] 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.
- [198] 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.
- [199] 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.
- [200] 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.
- [201] 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.
- [202] 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.
- [203] 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.
- [204] 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.
- [205] Leslie Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11):770–772, 1981.
- [206] 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.
- [207] Presidência da República Federativa do Brasil. Medida provisória n. 2.200-2, Agosto 2001.
- [208] J. Seedorf and E. Burger. Application-Layer Traffic Optimization (ALTO) Problem Statement, 10 2009. <http://www.ietf.org/rfc/rfc5693.txt>. Último acesso: 2 de Maio de 2014.

- [209] 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: 2 de Maio de 2014.
- [210] 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, 6 2013.
- [211] Van Jacobson. A New Way to Look at Networking, 8 2006. <http://named-data.net/a-new-way-to-look-at-networking/>. Último acesso: 2 de Maio de 2014.
- [212] Adobe Inc. HTTP Dynamic Streaming, 2 2014. <http://www.adobe.com/br/products/hds-dynamic-streaming.html>. Último acesso: 2 de Maio de 2014.
- [213] 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.
- [214] 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, 6 2013.
- [215] 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: 2 de Maio de 2014.
- [216] 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.
- [217] 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.
- [218] 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.
- [219] L. Tawalbeh, M. Mowafi, and W. Aljoby. Use of elliptic curve cryptography for multimedia encryption. *Information Security, IET*, 7(2):67–74, June 2013.
- [220] 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: 2 de Maio de 2014.
- [221] 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.
- [222] K Moore. Things that NATs break. Online, 3 2004. <http://web.mit.edu/6.033/2002/wwwdocs/papers/what-nats-break.html>. Último acesso: 2 de Maio de 2014.
- [223] 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, 2 de Maio de 2014.
- [224] Paul Vixie. What DNS is Not. *Commun. ACM*, 52(12):43–47, December 2009.
- [225] 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.
- [226] 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.

- [227] 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.
- [228] 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.
- [229] 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.
- [230] 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.
- [231] 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.
- [232] 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.
- [233] 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.
- [234] 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.
- [235] S. Bradner. Keywords for Use in RFCs to Indicate Requirement Levels, 3 1997. <http://www.ietf.org/rfc/rfc2119.txt>. Último acesso: 2 de Maio de 2014.
- [236] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3, 10 2002. <http://www.ietf.org/rfc/rfc3376.txt>. Último acesso: 2 de Maio de 2014.
- [237] B. Fenner, H. He, B. Haberman, and H. Sandick. Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying"), 8 2006. <http://www.ietf.org/rfc/rfc4605.txt>. Último acesso: 2 de Maio de 2014.
- [238] Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory (Discrete Mathematics and Its Applications)*. CRC Press, 2 2003. ISBN: 978-1584880905.
- [239] R Sroul. *Programming for Mathematicians*. Springer-Verlag, 2 2000. ISBN: 978-3540664222.
- [240] 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.
- [241] K. J. Devlin. *Fundamentals of Contemporary Set Theory*. Springer, 9 1979. ISBN: 978-0387904412.
- [242] R. Braden. Requirements for Internet Hosts - Communication Layers, 10 1989. Último acesso: 2 de Maio de 2014.
- [243] L. Eggert and F. Gont. TCP User Timeout Option, 3 2009. <http://www.ietf.org/rfc/rfc5482.txt>. Último acesso: 2 de Maio de 2014.
- [244] S. Tanwir and H. Perros. A Survey of VBR Video Traffic Models. *Communications Surveys Tutorials, IEEE*, 15(4):1778–1802, Fourth 2013.

- [245] 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: 2 de Maio de 2014.
- [246] 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: 2 de Maio de 2014.
- [247] M. Handley and V. Jacobson. SDP: Session Description Protocol, 4 1998. <http://www.ietf.org/rfc/rfc2327.txt>. Último acesso: 2 de Maio de 2014.
- [248] H. Alvestrand. Tags for the identification of languages, 2 1995. <http://www.ietf.org/rfc/rfc1766.txt>. Último acesso: 2 de Maio de 2014.
- [249] 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: 2 de Maio de 2014.
- [250] W. Wolff. *Stochastic Modeling and the Theory of Queues*. PrenticeHall, 1989.
- [251] 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: 2 de Maio de 2014.
- [252] 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: 2 de Maio de 2014.
- [253] 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.
- [254] 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, archi-*

- lectures, and protocols for computer communication*, pages 303–314, New York, NY, USA, 1998. ACM Press.
- [255] D. Meyer. Administratively Scoped IP Multicast, 7 1998. <http://www.ietf.org/rfc/rfc2365.txt>. Último acesso: 2 de Maio de 2014.
- [256] Xiaojun Hei, Yong Liu, and K.W. Ross. Inferring Network-Wide Quality in P2P Live Streaming Systems. *Selected Areas in Communications, IEEE Journal on*, 25(9):1640–1654, December 2007.
- [257] Raj Jan. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Inc, 1 edition, 3 1991.
- [258] 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.
- [259] Andras Varga. OMNeT++: Extensible, Modular, Component-based C++ Simulation Library and Framework for Building Network Simulators, 3 2014. <http://www.omnetpp.org/>. Último acesso: 2 de Maio de 2014.
- [260] Stephan Krause Ingmar Baumgart, Bernhard Heep. OverSim: the Overlay Simulation Framework, 3 2014. <http://www.oversim.org/>. Último acesso: 2 de Maio de 2014.
- [261] Giuseppe Rossini, Raffele Chiocchetti, Andrea Araldo, and Dario Rossi. CCN-Sim: Scalable Chunk-level Simulator of Content Centric Networks (CCN), 3 2014. <http://perso.telecom-paristech.fr/~drossi/index.php?n=Software.CcnSim>. Último acesso: 2 de Maio de 2014.
- [262] S. M Y Seyyedi and B. Akbari. Denacast: A P2P/CDN Video Streaming System in OverSim, 3 2014. <http://www.denacast.org/>. Último acesso: 2 de Maio de 2014.
- [263] Leandro Melo de Sales. Global Media Transmission Protocol (GMTP), 3 2014. <http://www.gmtp-protocol.org/>. Último acesso: 2 de Maio de 2014.

- [264] Vincenzo Ciancaglini, Giuseppe Piro, Riccardo Loti, Luigi Alfredo Griecoy, and Luigi Liquori. CCN-TV: Data-Centric Approach to Real-Time Video Services, 3 2014. <http://telematics.poliba.it/index.php/en/ccn-tv>. Último acesso: 2 de Maio de 2014.
- [265] F. H P Fitzek and M. Reisslein. MPEG-4 and H.263 Video Traces for Network Performance Evaluation. *Network, IEEE*, 15(6):40–54, 11 2001.
- [266] Diego Perino and Matteo Varvello. A reality check for content centric networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN ’11, pages 44–49, New York, NY, USA, 2011. ACM.
- [267] Xuemin Shen, Heather Yu, John Buford, and Mursalin Akon. *Handbook of Peer-to-Peer Networking*. Springer Publishing Company, Incorporated, 1st edition, 2009.
- [268] P. Francis, S. Jamin, Cheng Jin, Yixin Jin, D. Raz, Y. Shavitt, and Lixia Zhang. ID-Maps: a global Internet host distance estimation service. *Networking, IEEE/ACM Transactions on*, 9(5):525–540, Oct 2001.
- [269] T.S.E. Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 170–179 vol.1, 2002.
- [270] L.M. de Sales, H.O. Almeida, and A. Perkusich. GMTP: A Crossing-layer Optimized Protocol for Large Scale Distribution of Live Multimedia Content over the Internet. Technical report, Universidade Federal de Campina Grande, 1 2014. A ser submetido para o IEEE/ACM Transactions on Networking (JCR 2.014).
- [271] L.M. de Sales, H.O. Almeida, and A. Perkusich. Generalized Connection and Incentives for Supporting CE Devices in P2P/CDN Live Streaming Systems. *Consumer Electronics, IEEE Transactions on*, 2014. Em processo de revisão no IEEE Transaction on Consumers Electronics (JCR 1.087).
- [272] L.M. de Sales, H.O. Almeida, and A. Perkusich. Global Media Transmission Protocol (GMTP). Technical report, Universidade Federal de Campina Grande, 1 2014.

- Em revisão – I Workshop Pré-IETF, XXXIV Congresso da Sociedade Brasileira de Computação (CSBC 2014).
- [273] 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, 1 2013.
- [274] 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.

# Apêndice A

## Detalhes dos Experimentos

Neste apêndice, apresentam-se alguns detalhes relacionados aos experimentos realizados.

### A.1 Largura de Banda e Atraso de Propagação Utilizados na Rede Simulada

Na Tabela A.1, apresenta-se a largura de banda e o atraso de propagação da rede utilizada no experimento apresentado no Capítulo 5.

Tabela A.1: Largura de banda e atraso de propagação utilizados na rede simulada.

#	Roteador 1	Roteador 2	Largura de Banda	Atraso de Propagação
1	DE	SE	32 Mbps	6.34 ms
2	DE	CZ	32 Mbps	2.10 ms
3	DE	FR	32 Mbps	11.39 ms
4	DE	IT	32 Mbps	11.93 ms
5	DE	NL	27 Mbps	5.23 ms
6	DE	AT	27 Mbps	4.71 ms
7	DE	GR	16 Mbps	21.69 ms
8	DE	IE	10 Mbps	13.69 ms
9	SE	UK	32 Mbps	14.59 ms

Continuação na próxima página

**Tabela A.1 – continuação da página anterior**

#	Roteador 1	Roteador 2	Largura de Banda	Atraso de Propagação
10	SE	EE	10 Mbps	8.78 ms
11	SE	LV	3 Mbps	9.71 ms
12	SE	LT	5 Mbps	10.78 ms
13	SE	PL	27 Mbps	6.49 ms
14	UK	FR	32 Mbps	5.51 ms
15	UK	NL	27 Mbps	1.29 ms
16	UK	IE	10 Mbps	3.97 ms
17	UK	IL	5 Mbps	36.09 ms
18	UK	GR	10 Mbps	31.25 ms
19	FR	CH	32 Mbps	3.84 ms
20	FR	BE	27 Mbps	4.56 ms
21	FR	LU	10 Mbps	3.34 ms
22	CH	AT	32 Mbps	4.01 ms
23	CH	IT	32 Mbps	2.14 ms
24	IT	ES	27 Mbps	7.54 ms
25	NL	BE	27 Mbps	1.18 ms
26	BE	LU	10 Mbps	1.16 ms
27	PL	CZ	27 Mbps	3.64 ms
28	CZ	SK	27 Mbps	2.21 ms
29	SK	HU	27 Mbps	2.13 ms
30	AT	HU	27 Mbps	3.68 ms
31	AT	HR	3 Mbps	3.88 ms
32	AT	SI	16 Mbps	1.18 ms
33	HU	SI	10 Mbps	2.18 ms
34	HU	RO	3 Mbps	2.18 ms
35	ES	PT	16 Mbps	2.57 ms
36	CY	GR	3 Mbps	2.57 ms
37	BG	GR	3 Mbps	2.57 ms

## A.2 Distribuição da quantidade de nós clientes nos primeiros 200 s de simulação

Para definir o instante de cada requisição do cliente definiu-se o seguinte: gerou-se uma quantidade de números inteiros entre 0 – 200 igual à quantidade de nós clientes determinado para um certo tratamento. Por exemplo, no Tratamento 1, sortearam-se 500 números inteiros aleatórios entre 0 – 200 e somou-se o número de ocorrências de cada número sorteado. Durante a execução dos ensaios, adicionou-se à rede a quantidade de nós correspondente ao número de ocorrências em cada instante da simulação. Por exemplo, se o número 199 foi sorteado 10 vezes, no instante 199 da simulação foram adicionados 10 nós clientes que imediatamente requisitaram a mídia ao servidor.

## A.3 Distribuição da quantidade de nós clientes após os 400 s de simulação

Para definir o churn da rede definiu-se a seguinte estratégia: a cada intervalo de 5 s, um número  $x \in [0, 1]$  foi gerado. Dependendo do valor de  $x$ , 10 % dos nós clientes foram mantidos ou removidos com uma probabilidade de 0.8. Ou seja, os nós foram mantidos conectados à rede com uma probabilidade de 80 % e removidos com uma probabilidade de 20 %.

## A.4 Quantidade de Ensaios

No experimento realizado para comparar o GMTP com o Denacast/CoolStreaming e o CCN-TV, apresentado no Capítulo 5, fez-se necessário determinar a quantidade de repetição de um tratamento para que fosse possível comparar, com 95 % de certeza, os valores obtidos para as variáveis dependentes nos confrontos estabelecidos. Para isto, cada valor de  $n$  que representou uma variável dependente (por exemplo,  $n_{ST}$ ) foi calculado através da Inequação A.1, onde  $\mu_1$  correspondeu à média obtida para uma variável dependente através da execução do GMTP e,  $\mu_2$ , a média obtida para a mesma variável dependente através da execução do outro sistema confrontado.

$$\mu_1 \pm 1.96 \sqrt{\frac{0,995\mu_1}{n}} \leq \mu_2 \pm 1.96 \sqrt{\frac{0,995\mu_2}{n}} \quad (\text{A.1})$$

## A.5 Compilação dos Resultados

Na Tabela A.2, apresentam-se os resultados do confronto GMTP vs. Denacast/CoolStreaming e na Tabela A.3 os resultados do confronto GMTP vs. CCN-TV.

Tabela A.2: Sumário dos valores obtidos para as variáveis dependentes em cada tratamento do confronto GMTP vs. Denacast/CoolStreaming.

Trat. #	Atraso de inicialização (s)	Índice de continuidade (%)	Distorção do vídeo (%)	Pacotes de controle
<b>GMTP</b> <b>1</b> <b>Denacast</b>	1.68 (0.87 – 2.49)	99.21 (98.07 – 100.35)	1.30 (0.79 – 1.81)	995.43 (984.13 – 1006.73)
	23.51 (21.64 – 25.38)	92.96 (91.44 – 94.48)	13.02 (11.52 – 14.52)	1783.20 (1752.46 – 1813.94)
<b>GMTP</b> <b>2</b> <b>Denacast</b>	2.10 (1.24 – 2.96)	99.27 (98.15 – 100.39)	1.39 (0.58 – 2.20)	2933.60 (2901.79 – 2965.41)
	31.40 (29.33 – 33.47)	86.91 (85.34 – 88.48)	20.20 (18.71 – 21.69)	4727.92 (4657.82 – 4798.02)
<b>GMTP</b> <b>3</b> <b>Denacast</b>	2.27 (1.34 – 3.20)	98.61 (97.46 – 99.76)	1.48 (0.68 – 2.28)	28501.28 (28142.15 – 28860.41)
	29.98 (27.96 – 32.00)	59.07 (57.52 – 60.62)	23.11 (21.48 – 24.74)	45553.58 (44676.17 – 46430.99)
<b>GMTP</b> <b>4</b> <b>Denacast</b>	2.86 (1.94 – 3.78)	97.12 (95.93 – 98.31)	1.79 (0.98 – 2.60)	57945.70 (57236.01 – 58655.39)
	41.26 (39.21 – 43.31)	62.68 (61.09 – 64.27)	26.15 (24.45 – 27.85)	110374.53 (108605.37 – 112143.69)
<b>GMTP</b> <b>5</b> <b>Denacast</b>	3.15 (2.32 – 3.98)	96.62 (95.51 – 97.73)	1.98 (1.11 – 2.85)	129665.79 (128857.79 – 130473.79)
	69.21 (67.31 – 71.11)	58.22 (56.84 – 59.60)	33.10 (31.76 – 34.44)	230712.40 (228746.89 – 232677.91)
<b>GMTP</b> <b>6</b>	3.91 (2.96 – 4.86)	96.38 (95.22 – 97.54)	2.12 (1.31 – 2.93)	165415.93 (163357.60 – 167474.26)

Continuação na próxima página

**Tabela A.2 – continuação da página anterior**

<b>Trat. #</b>	<b>Atraso de inicialização (s)</b>	<b>Índice de continuidade (%)</b>	<b>Distorção do vídeo (%)</b>	<b>Pacotes de controle</b>
<b>Denacast</b>	74.89 (72.72 – 77.06)	48.49 (46.88 – 50.10)	35.63 (34.02 – 37.24)	270844.47 (266531.30 – 275157.64)
<b>GMTP</b> <b>7</b>	1.83 (0.98 – 2.68)	99.84 (98.75 – 100.93)	1.08 (0.51 – 1.65)	976.31 (969.03 – 983.59)
	20.48 (18.62 – 22.34)	92.33 (90.92 – 93.74)	9.52 (8.16 – 10.88)	1870.31 (1849.28 – 1891.34)
<b>GMTP</b> <b>8</b>	1.55 (0.74 – 2.36)	99.23 (98.10 – 100.36)	1.12 (0.24 – 2.00)	3086.69 (3062.69 – 3110.69)
	30.83 (28.86 – 32.80)	87.85 (86.45 – 89.25)	16.16 (14.75 – 17.57)	5304.01 (5246.36 – 5361.66)
<b>GMTP</b> <b>9</b>	2.01 (1.17 – 2.85)	99.57 (98.45 – 100.69)	1.31 (0.43 – 2.19)	31688.33 (31438.66 – 31938.00)
	32.32 (30.34 – 34.30)	63.93 (62.50 – 65.36)	19.35 (17.90 – 20.80)	49290.56 (48707.40 – 49873.72)
<b>GMTP</b> <b>10</b>	1.62 (0.77 – 2.47)	99.34 (98.24 – 100.44)	1.46 (0.60 – 2.32)	61527.51 (61058.00 – 61997.02)
	39.27 (37.43 – 41.11)	64.18 (62.86 – 65.50)	22.44 (21.05 – 23.83)	100287.74 (99052.38 – 101523.10)
<b>GMTP</b> <b>11</b>	1.76 (0.92 – 2.60)	97.21 (96.12 – 98.30)	1.58 (0.72 – 2.44)	116935.79 (116149.32 – 117722.26)
	48.41 (46.58 – 50.24)	60.12 (58.81 – 61.43)	23.93 (22.62 – 25.24)	217567.43 (215776.75 – 219358.11)
<b>GMTP</b> <b>12</b>	2.37 (1.54 – 3.20)	96.16 (95.05 – 97.27)	1.87 (1.00 – 2.74)	157364.71 (156138.00 – 158591.42)
	66.58 (64.72 – 68.44)	54.15 (52.74 – 55.56)	27.15 (25.79 – 28.51)	286721.13 (284008.35 – 289433.91)
<b>GMTP</b> <b>13</b>	1.20 (0.35 – 2.05)	99.38 (99.04 – 99.72)	0.94 (0.27 – 1.61)	988.12 (979.79 – 996.45)
	13.71 (11.80 – 15.62)	94.66 (93.25 – 96.07)	7.60 (6.21 – 8.99)	1869.09 (1848.65 – 1889.53)
<b>GMTP</b> <b>14</b>	1.00 (0.18 – 1.82)	99.44 (99.17 – 99.71)	0.62 (0.36 – 0.88)	2967.48 (2947.59 – 2987.37)
	19.15	91.80	11.92	5639.91

Continuação na próxima página

**Tabela A.2 – continuação da página anterior**

<b>Trat. #</b>	<b>Atraso de inicialização (s)</b>	<b>Índice de continuidade (%)</b>	<b>Distorção do vídeo (%)</b>	<b>Pacotes de controle</b>
	(17.29 – 21.01)	(90.48 – 93.12)	(10.63 – 13.21)	(5590.83 – 5688.99)
<b>GMTP 15</b> <b>Denicast</b>	1.37 (0.44 – 2.30)	98.12 (96.95 – 99.29)	0.86 (0.06 – 1.66)	30011.68 (29664.55 – 30358.81)
	22.17 (20.42 – 23.92)	73.71 (72.20 – 75.22)	17.42 (15.85 – 18.99)	49257.29 (48410.32 – 50104.26)
<b>GMTP 16</b> <b>Denicast</b>	1.06 (0.11 – 2.01)	98.99 (97.84 – 100.14)	1.72 (0.92 – 2.52)	61390.66 (60777.47 – 62003.85)
	38.05 (36.21 – 39.89)	75.54 (73.95 – 77.13)	20.54 (18.92 – 22.16)	103460.84 (101813.30 – 105108.38)
<b>GMTP 17</b> <b>Denicast</b>	1.01 (0.15 – 1.87)	98.90 (97.76 – 100.04)	2.22 (1.34 – 3.10)	125339.24 (124097.93 – 126580.55)
	46.36 (44.65 – 48.07)	74.15 (72.71 – 75.59)	26.58 (25.08 – 28.08)	202064.30 (199066.55 – 205062.05)
<b>GMTP 18</b> <b>Denicast</b>	1.93 (1.07 – 2.79)	98.60 (97.44 – 99.76)	2.96 (2.08 – 3.84)	149991.96 (148027.77 – 151956.15)
	53.77 (51.84 – 55.70)	69.21 (67.60 – 70.82)	27.27 (25.63 – 28.91)	272874.40 (268310.63 – 277438.17)

Tabela A.3: Sumário dos valores obtidos para as variáveis dependentes em cada tratamento do confronto GMTP vs. CCN-TV.

<b>Trat. #</b>	<b>Atraso de inicialização (s)</b>	<b>Índice de continuidade (%)</b>	<b>Distorção do vídeo (%)</b>	<b>Pacotes de controle</b>
<b>GMTP 1</b> <b>CCN-TV</b>	1.68 (0.87 – 2.49)	99.21 (98.07 – 100.35)	1.30 (0.79 – 1.81)	995.43 (984.13 – 1006.73)
	3.25 (2.63 – 3.87)	96.46 (95.29 – 97.63)	3.63 (2.78 – 4.48)	4554.12 (4463.57 – 4644.67)
<b>GMTP 2</b> <b>CCN-TV</b>	2.10 (1.24 – 2.96)	99.27 (98.15 – 100.39)	1.39 (0.58 – 2.20)	2933.60 (2901.79 – 2965.41)
	3.93 (3.08 – 4.78)	96.26 (95.10 – 97.42)	4.50 (3.65 – 5.35)	14032.32 (13765.55 – 14299.09)

Continuação na próxima página

**Tabela A.3 – continuação da página anterior**

<b>Trat. #</b>	<b>Atraso de inicialização (s)</b>	<b>Índice de continuidade (%)</b>	<b>Distorção do vídeo (%)</b>	<b>Pacotes de controle</b>
<b>GMTP 3</b>	2.27 (1.34 – 3.20)	98.61 (97.46 – 99.76)	1.48 (0.68 – 2.28)	28501.28 (28142.15 – 28860.41)
	4.36 (3.36 – 5.36)	88.99 (87.79 – 90.19)	7.97 (7.11 – 8.83)	144365.38 (141357.78 – 147372.98)
<b>GMTP 4</b>	2.86 (1.94 – 3.78)	97.12 (95.93 – 98.31)	1.79 (0.98 – 2.60)	57945.70 (57236.01 – 58655.39)
	5.38 (4.27 – 6.49)	90.27 (89.05 – 91.49)	9.54 (8.67 – 10.41)	83287.32 (77035.85 – 89538.79)
<b>GMTP 5</b>	3.15 (2.32 – 3.98)	96.62 (95.51 – 97.73)	1.98 (1.11 – 2.85)	129665.79 (128857.79 – 130473.79)
	8.02 (7.22 – 8.82)	83.73 (82.60 – 84.86)	18.72 (17.88 – 19.56)	8.37 (–8122.27 – 8139.01)
<b>GMTP 6</b>	3.91 (2.96 – 4.86)	96.38 (95.22 – 97.54)	2.12 (1.31 – 2.93)	165415.93 (163357.60 – 167474.26)
	11.23 (10.18 – 12.28)	81.22 (80.03 – 82.41)	21.98 (21.11 – 22.85)	242.37 (–16271.82 – 16756.56)
<b>GMTP 7</b>	1.83 (0.98 – 2.68)	99.84 (98.75 – 100.93)	1.08 (0.51 – 1.65)	976.31 (969.03 – 983.59)
	3.40 (2.83 – 3.97)	97.24 (96.12 – 98.36)	2.65 (1.81 – 3.49)	4570.20 (4508.93 – 4631.47)
<b>GMTP 8</b>	1.55 (0.74 – 2.36)	99.23 (98.10 – 100.36)	1.12 (0.24 – 2.00)	3086.69 (3062.69 – 3110.69)
	3.27 (2.66 – 3.88)	96.60 (95.46 – 97.74)	3.53 (2.69 – 4.37)	14228.89 (14033.99 – 14423.79)
<b>GMTP 9</b>	2.01 (1.17 – 2.85)	99.57 (98.45 – 100.69)	1.31 (0.43 – 2.19)	31688.33 (31438.66 – 31938.00)
	5.36 (4.48 – 6.24)	96.97 (95.81 – 98.13)	6.36 (5.52 – 7.20)	138290.07 (136192.79 – 140387.35)
<b>GMTP 10</b>	1.62 (0.77 – 2.47)	99.34 (98.24 – 100.44)	1.46 (0.60 – 2.32)	61527.51 (61058.00 – 61997.02)
	5.14 (4.31 – 5.97)	92.59 (91.45 – 93.73)	9.49 (8.66 – 10.32)	271113.55 (267364.06 – 274863.04)
<b>GMTP</b>	1.76	97.21	1.58	116935.79

Continuação na próxima página

Tabela A.3 – continuação da página anterior

<b>Trat. #</b>	<b>Atraso de inicialização (s)</b>	<b>Índice de continuidade (%)</b>	<b>Distorção do vídeo (%)</b>	<b>Pacotes de controle</b>
<b>11 CCN-TV</b>	(0.92 – 2.60)	(96.12 – 98.30)	(0.72 – 2.44)	(116149.32 – 117722.26)
	8.97	88.48	18.22	515098.83
	(8.17 – 9.77)	(87.38 – 89.58)	(17.39 – 19.05)	(508231.11 – 521966.55)
<b>GMTP 12 CCN-TV</b>	2.37	96.16	1.87	157364.71
	(1.54 – 3.20)	(95.05 – 97.27)	(1.00 – 2.74)	(156138.00 – 158591.42)
	10.16	87.15	19.61	796329.49
<b>GMTP 13 CCN-TV</b>	(9.25 – 11.07)	(86.04 – 88.26)	(18.77 – 20.45)	(785538.00 – 807120.98)
	1.20	99.38	0.94	988.12
	(0.35 – 2.05)	(99.04 – 99.72)	(0.27 – 1.61)	(979.79 – 996.45)
<b>GMTP 14 CCN-TV</b>	3.47	98.52	2.98	4332.17
	(2.61 – 4.33)	(98.19 – 98.85)	(2.14 – 3.82)	(4260.99 – 4403.35)
	1.00	99.44	0.62	2967.48
<b>GMTP 15 CCN-TV</b>	(0.18 – 1.82)	(99.17 – 99.71)	(0.36 – 0.88)	(2947.59 – 2987.37)
	3.17	98.55	4.12	13460.06
	(2.34 – 4.00)	(98.14 – 98.96)	(3.28 – 4.96)	(13289.52 – 13630.60)
<b>GMTP 16 CCN-TV</b>	1.37	98.12	0.86	30011.68
	(0.44 – 2.30)	(96.95 – 99.29)	(0.06 – 1.66)	(29664.55 – 30358.81)
	5.43	94.58	5.98	132808.98
<b>GMTP 17 CCN-TV</b>	(4.61 – 6.25)	(93.38 – 95.78)	(5.12 – 6.84)	(129592.34 – 136025.62)
	1.06	98.99	1.72	61390.66
	(0.11 – 2.01)	(97.84 – 100.14)	(0.92 – 2.52)	(60777.47 – 62003.85)
<b>GMTP 18 CCN-TV</b>	6.37	93.96	8.25	259228.45
	(5.56 – 7.18)	(92.80 – 95.12)	(7.40 – 9.10)	(253833.38 – 264623.52)
	1.01	98.90	2.22	125339.24
<b>GMTP 18 CCN-TV</b>	(0.15 – 1.87)	(97.76 – 100.04)	(1.34 – 3.10)	(124097.93 – 126580.55)
	7.60	90.02	12.96	586015.86
	(6.69 – 8.51)	(88.84 – 91.20)	(12.11 – 13.81)	(575627.27 – 596404.45)
<b>GMTP 18 CCN-TV</b>	1.93	98.60	2.96	149991.96
	(1.07 – 2.79)	(97.44 – 99.76)	(2.08 – 3.84)	(148027.77 – 151956.15)
	8.24	89.62	15.18	788436.46
<b>GMTP 18 CCN-TV</b>	(7.32 – 9.16)	(88.40 – 90.84)	(14.32 – 16.04)	(775712.37 – 801160.55)

## **Apêndice B**

### **Detalhamento dos Tipos de Pacotes do GMTP**