

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

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

Leandro Melo de Sales

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

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

Angelo Perkusich e Hyggo Almeida
(Orientadores)

Campina Grande, Paraíba, Brasil

©Leandro Melo de Sales, 03/03/2014

Resumo

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

Abstract

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

Conteúdo

1	Introdução	1
1.1	Delimitação e Argumentos	4
1.2	Descrição do Problema	9
1.3	Hipótese	17
1.4	Objetivo	18
1.4.1	Objetivos Específicos	18
1.5	Relevância do Tema e da Tese	19
1.6	Resumo das Contribuições	20
1.7	Publicações	21
1.8	Estrutura do Documento	22
2	Fundamentação	23
2.1	Distribuição de Mídia ao Vivo em Arquiteturas P2P	24
2.1.1	Estrutura Baseada em Árvore	24
2.1.2	Estrutura Baseada em Malha	27
2.1.3	Estrutura Híbrida	31
2.2	Sistemas de Transmissão ao Vivo em P2P	34
2.2.1	Geração do Conteúdo da Transmissão	37
2.2.2	Armazenamento e Consumo de Dados	38
2.2.3	Estratégia de Seleção de <i>Chunks</i>	40
2.2.4	Realização de parcerias e obtenção de <i>chunks</i>	41
2.3	Criptografia de Hash e Assinatura Digital	42
2.3.1	Criptografia de Hash	43
2.3.2	Criptografia Assimétrica	43

2.3.3	Assinatura e Certificação Digital	44
2.4	Sumário do Capítulo	46
3	Trabalhos Relacionados	48
3.1	Protocolos Multimídia Padronizados	49
3.1.1	H.323 / ITU-T	49
3.1.2	SIP – <i>Session Initiation Protocol</i>	49
3.1.3	RTP – <i>Real Time Protocol</i>	50
3.1.4	RTSP – <i>Real Time Streaming Protocol</i>	50
3.1.5	ALTO – <i>Application-Layer Traffic Optimization</i>	51
3.1.6	UDP, DCCP e SCTP	51
3.2	Protocolos de Transporte de Dados Multimídia	51
3.2.1	PPETP – <i>Peer-to-Peer Epi-Transport Protocol</i>	51
3.2.2	PPSP/Swift – <i>P2P Streaming Protocol / The Generic Multiparty Transport Protocol</i>	55
3.3	Protocolos de Aplicação para Transmissão de Mídias	57
3.3.1	CoolStreaming	57
3.3.2	Denacast	58
3.3.3	PeerCast	58
3.3.4	<i>HTTP Live Streaming</i>	58
3.3.5	PDTP – <i>Peer Distributed Transfer Protocol</i>	58
3.3.6	CPM – <i>Cooperative Peer Assists and Multicast</i>	59
3.3.7	HySAC – <i>Hybrid Delivery System with Adaptive Content Management for IPTV Networks</i>	61
3.3.8	PULSE – <i>Peer-to-Peer Unstructured Live Streaming Experiment</i> . .	63
3.3.9	SmoothCache – <i>HTTP-Live Streaming Goes Peer-To-Peer</i>	64
3.3.10	Pastry/SplitStream – <i>High-bandwidth content distribution</i>	64
3.3.11	LayeredCast – <i>Hybrid Peer-to-Peer Live Layered Video Streaming Protocol</i>	64
3.3.12	BitTorrent/LEDBAT – <i>Low Extra Delay Background Transport</i> . .	64
3.3.13	Outras propostas	64

3.4	Redes Centradas no Conteúdo	65
3.4.1	CCNx – <i>Content Centric Network Protocol</i>	65
3.4.2	NDN – <i>Named Data Network</i>	65
3.5	Sumário Comparativo	65
3.6	Sumário do Capítulo	66
3.6.1	Preâmbulo ao GMTP	66
4	Global Media Transmission Protocol (GMTP)	70
4.1	Visão Geral	72
4.1.1	Resumo das principais funções	78
4.2	Definições, Relações e Restrições	80
4.3	Constituição da Rede de Favores η	83
4.3.1	Tipos de Pacotes	84
4.3.2	Registro de participação de r_d em η	86
4.3.3	Tabela de Recepção de Fluxos de Dados	90
4.3.4	Formação de parcerias	92
4.4	Transmissão de $p_x \in P$ através de η	98
4.4.1	Indexação de Conteúdo	98
4.4.2	Estabelecimento de conexão entre c_f e s_a para obter P	103
4.4.3	Fase 1: primeira requisição a um fluxo de dados P	103
4.4.4	Fase 2: próximas requisições para obter P	108
4.4.5	Fase 3: busca por mais parceiros r_q para obter P	108
4.4.6	Envio e recebimento de $p_x \in P$ em η	113
4.5	Controle de Congestionamento em η	116
4.5.1	Controle de Congestionamento Unicast	117
4.5.2	Controle de Congestionamento Multicast	127
4.6	Autenticidade de P	130
4.6.1	Transmissão e assinatura de autenticidade de $p_x \in P$	132
4.6.2	Verificação de autenticidade de $p_x \in P$	132
4.6.3	Habilitar / desabilitar validação dos pacotes $p_x \in P$	134
4.6.4	Obtenção da chave pública $K_{s_a}^+$ de s_a	135

4.7	Outras considerações	135
4.7.1	Canais de Comunicação	136
4.7.2	Procedimentos para desconexão de nós c_f , l_w e r_d	137
4.7.3	Eleição de nós l_w	139
4.8	Sumário do Capítulo	139
5	Análise de Desempenho do GMTP	142
5.1	Análise da Escalabilidade do Número de Clientes	144
5.2	Análise da Taxa de Transmissão	146
5.3	Análise do Atraso Fim-a-Fim	147
5.4	Compilação dos Resultados	148
5.5	Sumário do Capítulo	150
6	Considerações Finais	152
6.1	Conclusões	153
6.1.1	Benefícios e Aplicabilidade	153
6.2	Perguntas Frequentes	155
6.3	Trabalhos Futuros	158

Lista de Símbolos

3WHS - *Three Way Hand Shake*

ALM - *Application Layer Multicast*

BSD - *Berkley Software Distribution*

CCID - *Congestion Control IDentifier*

CCN - *Content Centric Networks*

CPM - *Cooperative Peer Assists and Multicast*

DCCP - *Datagram Congestion Control Protocol*

ECN - *Explicit Congestion Notification*

GMTP - *Global Media Transport Protocol*

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*

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*

RTT - *Round Trip Time*

SCTP - *Stream Control Transmission Protocol*

Swift - *The Generic Multiparty Transport Protocol*

TCP - *Transport Control Protocol*

TFRC - *TCP Friendly Rate Control*

TTL - *Time-To-Live*

UDP - *User Datagram Protocol*

VCP - *Variable-Structure Congestion Control Protocol*

XCP - *eXplicit Control Protocol*

Listas de Figuras

1.1	Perfil de Tráfego de Rede da América do Norte durante o horário de pico.	3
1.2	Esquema de uma arquitetura híbrida P2P/CDN para distribuição de conteúdo multimídia.	5
2.1	Árvore de <i>multicast</i> em nível da camada de aplicação.	26
2.2	Manutenção da árvore de <i>multicast</i> em nível da camada de aplicação.	27
2.3	Sistema baseado em múltiplas árvores com dois subfluxos.	28
2.4	Atividade inicial de um novato - rede P2P baseada em malha.	29
2.5	Troca de dados na aplicação baseada em malha.	32
2.6	Modelo de sistema utilizado.	34
2.7	Mecanismo de consumo da mídia ao vivo.	40
2.8	Modelo de criptografia assimétrica, chave pública representada por K^+ e chave privada representada por K^-	44
2.9	Geração de Assinaturas Digitais.	45
3.1	Arquitetura e funcionamento do protocolo PPETP.	52
3.2	Arquitetura e funcionamento do protocolo PPSP/Swift.	56
3.3	Organização dos Nós PDTP.	59
3.4	Diagrama de sequência do CPM (<i>Cooperative Peer Assists and Multicast</i>). .	60
3.5	Arquitetura do HySAC (<i>Hybrid Delivery System with Adaptive Content Management for IPTV Networks</i>).	63
4.1	Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP. .	71
4.2	Arquitetura do Protocolo GMTP.	73

4.3	Tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP. Nessa tela, permitir que o administrador do roteador configure parâmetros do módulo GMTP Inter.	75
4.4	Rede de sobreposição construída pelo GMTP	76
4.5	Tipos de Nós e modos de conexões do GMTP.	77
4.6	Exemplo de uma tabela de recepção de fluxo mantida por um nó r_d	91
4.7	Cenário e passos para seleção de nós (exemplo 1).	93
4.8	Cenário para seleção de nós por interseção de caminhos W_v	94
4.9	Exemplo de rede para o estabelecimento de conexão do GMTP.	104
4.10	Passos do processo de estabelecimento de conexão do GMTP (Fase 1).	105
4.11	Tabela de recepção de fluxos de dados após a Fase 1.	108
4.12	Passos do processo de estabelecimento de conexão do GMTP (Fase 2).	109
4.13	Fase 3 de conexão do GMTP (Passo 1).	111
4.14	Fase 3 de conexão do GMTP (Passo 2).	112
4.15	Tabela de recepção de fluxos de dados após a Fase 3.	112
4.16	Exemplo da estrutura do buffer de envio e recepção de um nó GMTP com dois ponteiros, um para escrever e outro para ler pacotes p_x	114
4.17	Exemplo do mapa de buffer de um nó GMTP com tamanho de 17 p_x	114
4.18	Organização do algoritmo de controle de congestionamento no GMTP.	117
4.19	Cada r_d mantém uma única taxa de transmissão $R(t)$ que é atribuída em todos os pacotes de dados p_x no cabeçalho de qualquer pacote GMTP. A medida que o pacote passa através dos roteadores em um caminho W_v , se a taxa atual $R(t)$ no roteador for menor do que R_p informado no pacote sendo processado, o roteador o sobrescreve com sua taxa atual $R(t)$. Quando o pacote alcançar o nó s_a , este utiliza R_p para transmitir o fluxo de dados, pois trata-se da taxa de transmissão do roteador mais congestionado no caminho W_v	119
4.20	O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão, porém isto pode limitar alguns nós clientes a receberem os pacotes de dados em uma taxa maior.	122

4.21 O GMTP-UCC segmenta o caminho e dessa forma não limita a taxa de transmissão de um fluxo de dados para certos nós capazes de receber em uma taxa de transmissão maior.	123
4.22 No gráfico esquerdo, ilustra-se o tempo médio de duração (quanto tempo leva para completar) de um fluxo versus o tamanho do fluxo utilizando o TCP e o XCP. No gráfico direito, 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 <i>pacote</i>), 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. Os valores de PS foram calculados a partir de expressões analíticas. Extraído de [1]. . .	125
4.23 Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP como resultados de simulações com taxa de entrada de dados em Poisson e tamanhos do fluxo em distribuição Pareto com média de 25 pacotes (1000 bytes <i>pacote</i>), 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 [1].	126
5.1 Quantidade de conexões simultâneas do GMTP e o DCCP	145
5.2 Taxa média de recepção e perda de pacotes com GMTP e DCCP	146
5.3 Taxa de recepção do GMTP, DCCP e TCP	148
5.4 Atraso médio de recepção de pacotes com GMTP e DCCP	149

Lista de Tabelas

List of Algorithms

1	registerRelay(s_a : PeerServer, $p_x = GMTP\text{-}Request$)	88
2	onReceiveGMTPRegisterReply($p_x = GMTP\text{-}Register\text{-}Reply$)	89
3	handleRegisterParticipation(r_d : PeerRelay, $p_x = GMTP\text{-}Register$)	97
4	Exemplo de requisição e resposta da lista de nomes dos fluxos de dados P de um distribuidor de conteúdos multimídia.	100
5	Exemplo de uma mensagem SDP no pacote $GMTP\text{-}MediaDesc$	102
6	respondToClients($p_x: GMTP\text{-}RequestNotify$)	107
7	digitalSignPacket($p_x: GMTP\text{-}Data$)	132
8	verifyPacketAuthenticity(P' : array of $GMTP\text{-}Data$)	134

Capítulo 1

Introdução

Nos últimos anos, tem ocorrido uma revolução com relação às formas de criação e transmissão de mídias (áudio e/ou vídeo) na Internet. O número crescente de conexões de banda larga à Internet têm provocado o aumento do consumo e da exigência de qualidade dos conteúdos multimídia, que eleva os custos com largura de banda dos distribuidores de conteúdo. Por exemplo, o YouTube™ registra um custo operacional com largura de banda da ordem de US\$1 milhão por mês para atender cerca de 20 milhões de usuários por dia, para transmitir o equivalente a 60 mil anos de vídeos se fossem reproduzidos sequencialmente [2]. Estima-se que em 2014 o tráfego de video será maior do que foi o tráfego de redes entre pares (P2P) para compartilhamento de arquivos em 2009, correspondendo a 39 % do tráfego de dados total na Internet. Além disso, estima-se que o tráfego de VoIP, vídeo e jogos na Internet atingirá a marca de 40 exabytes por mês, quase 50 % do tráfego de dados total na Internet previsto para 2014 [3]. Segundo um estudo da empresa Cisco™, 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 [4].

Com a evolução da WWW (*World Wide Web*) para a Web 2.0 [5,6], 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™, ou principalmente os casos de transmissões empresariais e residenciais ao vivo, como o PPLive™ e o UStream.tv™. Nesses últimos casos, os sistemas permitem a transmissão de conteúdos ao vivo gerados a partir do computador de um usuário para milhares de outros usuários conectados à Internet [7]. O

sistema 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 canais¹ 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.TV² (*National Aeronautics and Space Administration Television*), que em 2011 migrou todos seus canais ao vivo na Internet para o UStream.tv, com mais de 25 milhões de acessos em fevereiro de 2014.

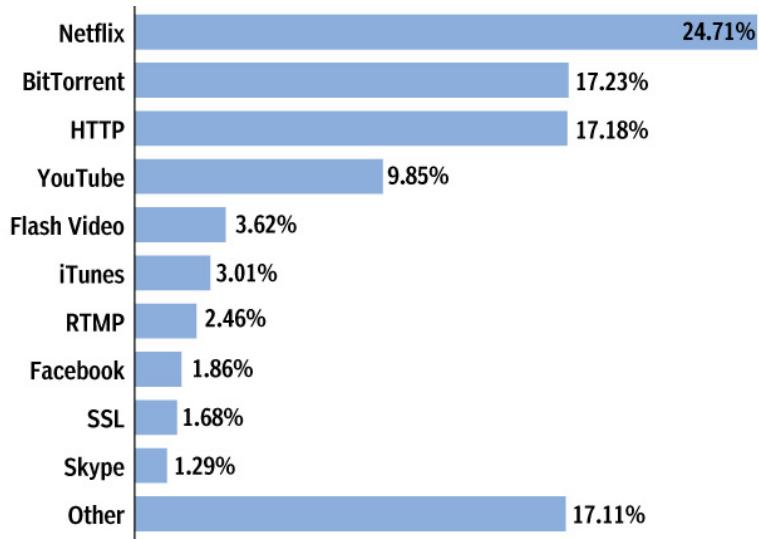
Ao observar essas e outras estatísticas [7–12], o que preocupa é o crescimento acentuado do consumo de recursos computacionais de rede resultante principalmente das estratégias e protocolos de rede adotados para realizar a distribuição dos conteúdos multimídia, cada um com suas próprias soluções [13–18]. No final do primeiro semestre de 2011, um artigo publicado no *Financial Post* chamou atenção ao “*anunciar momentos de grandes congestionamentos na Internet nos próximos anos*” [19,20]. Destacou-se a Netflix como sendo a maior consumidora de banda de rede da América do Norte, respondendo por 24,71 % de todos os dados transferidos durante o horário de pico de uso da Internet por norte-americanos, ultrapassando até mesmo a rede BitTorrent™ de compartilhamento de arquivos, como ilustra-se na Figura 1.1(a). “*A questão é que se não fosse a Netflix, teria sido a Amazon™, se não fosse a Amazon teria sido o Google™, mas o verdadeiro e iminente problema de congestionamento da rede começará quando todos passarem a fazer isto.*”, afirmou Colin Gillis, analista sênior de tecnologia da BGC Partners³.

Essa discussão se generaliza em um relatório publicado pela empresa de consultoria Sandvine, onde se menciona que “*com o rápido sucesso da Netflix, YouTube, UStream e outros, os provedores de Internet em todo o mundo devem se preparar para um futuro em que serviços de multimídia ao vivo será disponibilizado em grandes proporções, podendo ser responsável pela maior fatia do tráfego de dados na Internet*” [20]. De acordo com o que foi publicado nesse relatório, serviços ao vivo de entretenimento foram responsáveis por quase metade (49,2 %) de todo o tráfego de Internet na América do Norte no primeiro trimestre de 2011, como ilustrado-se na Figura 1.1(b). Este número alcançou 60 % no final de 2011 e “*o fato é que o volume de tráfego na Internet cresce exponencialmente e o congestionamento da*

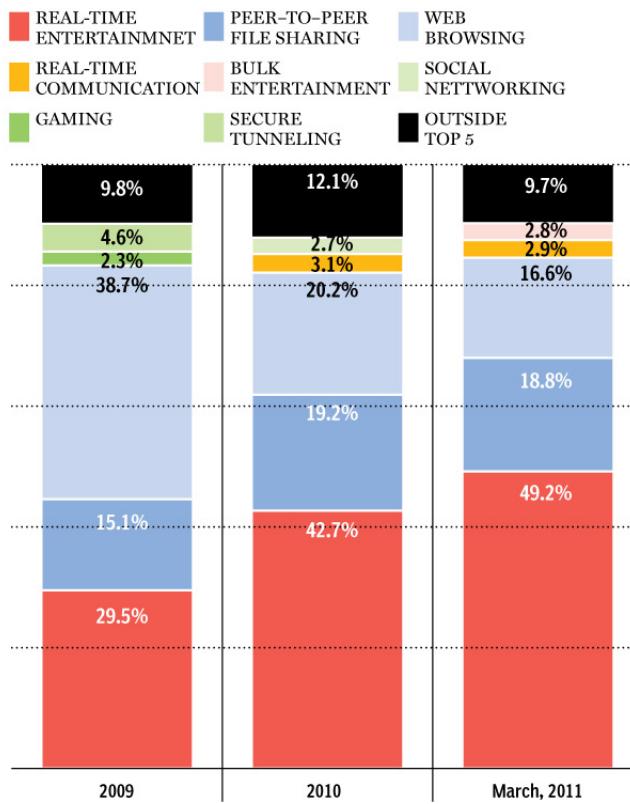
¹<http://www.ustream.tv/decoraheagles>. O nome do canal é *Decorah Eagles*, um acen-tamento de águias, localizado em Decorah, Iowa.

²<http://www.ustream.tv/nasa>

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



(a) Percentual de consumo de dados transmitidos e recebidos por aplicação.



(b) Tráfego de Internet em redes de acesso de banda larga.

Figura 1.1: Perfil de Tráfego de Rede da América do Norte durante o horário de pico (considerando-se redes fixas) [19].

rede tende a piorar. As pessoas sabem o que está acontecendo, mas está acontecendo mais rápido do que elas esperavam.”, comentou Tom Donnelly, vice-presidente da Sandvine.

Nesse contexto, nota-se que o panorama atual dos serviços de distribuição de conteúdos multimídia ao vivo se apresenta em fase de grande expansão, principalmente no que diz respeito ao interesse dos usuários e ao super consumo dos recursos de rede. Por isto, há uma grande motivação para estudar e propor novas soluções para utilizar eficientemente as redes de computadores a fim de melhorar os serviços de distribuição multimídia na Internet [21, 22].

1.1 Delimitação e Argumentos

O ponto de partida no contexto deste trabalho são os protocolos de rede utilizados nos sistemas baseados em uma arquitetura de rede híbrida P2P/CDN, ou seja, par-a-par (*Peer to Peer - P2P*) [23–26] e cliente-servidor com suporte de uma rede de distribuição de conteúdos (*Content Delivery Network - CDNs*) [27–29]. Isto porque há evidências contundentes [30–38] de que se trata da principal escolha dos sistemas mais robustos, conseguindo-se escalabilidade do número de usuários e redução de custos com infra-estrutura de rede, por meio das redes P2P; ao passo que se consegue facilidade no gerenciamento e maior estabilidade de disponibilização dos serviços, por meio das CDNs.

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. Do ponto de vista de protocolos de rede, utiliza-se conexões TCP (*Transmission Control Protocol*) [39] 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*) [40] para transmitir datagramas contendo as partes da mídia e exibi-las ao usuário final. Um aspecto importante para esses sistemas é definir de que forma os nós devem escalar 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, ilustrado na Figura 1.2, os nós estão sempre acessando e disponibilizando recursos uns dos outros, ao mesmo tempo que os servidores da CDN

organizam os nós da rede P2P e mantém os serviços de distribuição mais estáveis, tornando o sistema menos vulnerável ao dinamismo de participação dos nós de uma redes P2P, também conhecido por *churn* [41, 42].

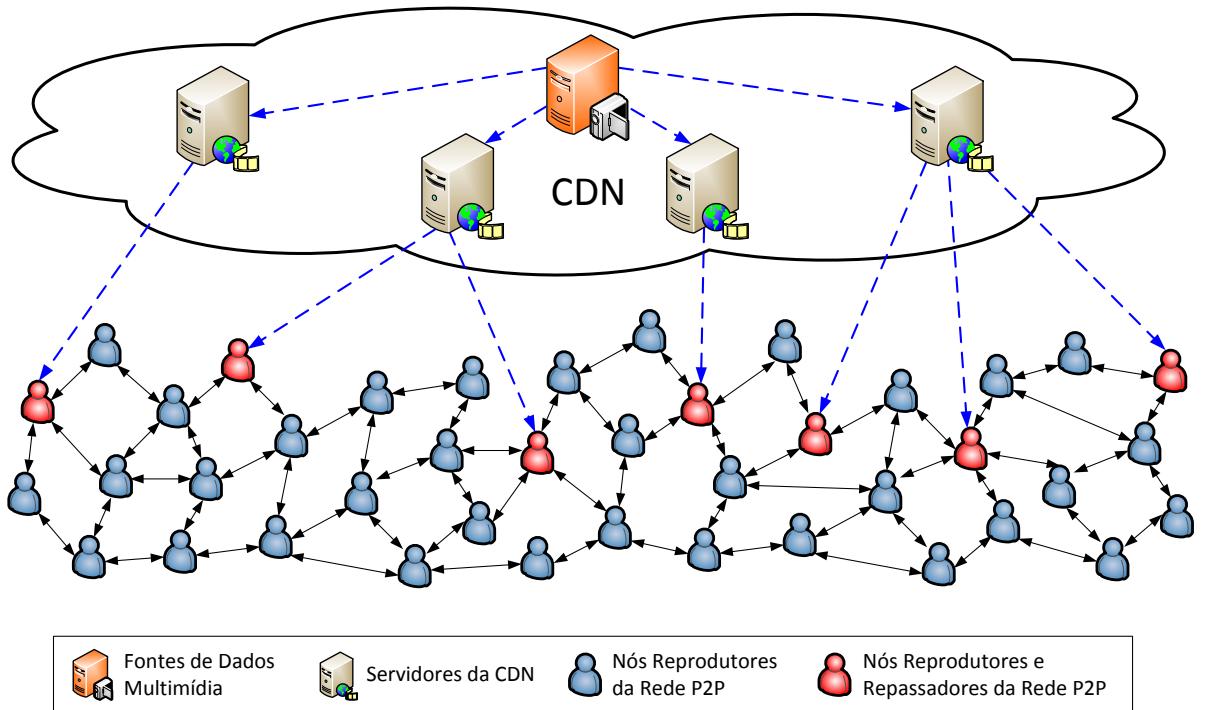


Figura 1.2: Estrutura de rede de uma arquitetura híbrida P2P/CDN para distribuição de conteúdo multimídia.

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 as 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 experiência do usuário ao reproduzir um conteúdo multimídia devido ao aumento do *churn*;
3. o estado do mapa de *buffer* de reprodução de cada nó da rede é semelhante, pois não existe a possibilidade de um nó ter um mapa de *buffer* com muito mais dados para reproduzir do que outros nós. Da mesma forma, não faz sentido manter por muito tempo dados já reproduzidos no *buffer* de reprodução. Os datagramas expiram⁴ rapidamente e, nestes casos, devem ser descartados pelas aplicações dos nós que os recebem. Assim, o tamanho necessário para o *buffer* de reprodução deve ser o suficiente para armazenar alguns segundos⁵ da mídia e repassá-la aos seus nós parceiros, requerendo baixa⁶ capacidade de armazenamento dos processadores de rede (roteadores, pontos de acesso etc.), incluindo dos nós finais (*desktops*, celulares, *tablets* etc.);
4. o caminho dos fluxos de dados transmitidos por um mesmo servidor da CDN s_1 para um conjunto de nós $c_1..c_n$, localizados em redes distintas, são mais previsíveis. Isto possibilita parcerias de melhor qualidade ao levar em consideração que é possível determinar pontos de intersecção das rotas desses fluxos de dados, uma vez que estes convergem para um mesmo servidor s_1 . Nestes casos, é possível manter apenas um fluxo de dados e replica-lo no ponto de intersecção. Isto permite o fomento à cooperação e o melhor agrupamento entre os nós.

Estas peculiaridades viabilizam uma abordagem muito importante adotada neste trabalho: a participação mais efetiva dos roteadores no processo de distribuição do conteúdo multimídia. Isto ocorre ao permitir que os roteadores: (i) constituam uma rede de sobreposição P2P; (ii) selecionem nós parceiros (outros roteadores); (iii) sirvam como pontes de

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

⁵No máximo, 5 segundos

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

acesso aos servidores da CDN; e (iv) ajudem na execução do serviço de controle de congestionamento. Isto porque conjectura-se que os roteadores são elementos de rede estáveis com relação a sua disponibilidade se comparados aos sistemas finais, portanto atenuando o *churn* da rede P2P e assim permitir que as aplicações forneçam serviços mais estáveis aos usuários, mesmo quando um usuário não tem interesse explícito por um determinado conteúdo. Esta decisão também se baseia no fato de que se existir um nó c interessado por um fluxo de dados P , transmitido por um servidor s através de um roteador r , os pacotes de dados de P que deverão ser entregues a c passarão obrigatoriamente por r .

O uso de tal abordagem é justificado por uma tendência em utilizar os roteadores para auxiliar e otimizar os serviços das camadas TCP/IP mais acima, como os serviços de controle de congestionamento, NAT dinâmico (*Dynamic Network Address Translation*) através de UPnP (*Universal Plug and Play*) [43], DNS dinâmico [44], dentre outros. Essa tendência é também sustentada pelas Redes Centradas no Conteúdo (*Content-Centric Networks – CCN*), proposta por Van Jacobson [45–49]. O princípio é que uma rede de comunicação deve viabilizar o acesso ao conteúdo de interesse de um usuário (*content-centric approach*) independente do local físico onde tal conteúdo está armazenado, enfraquecendo a abordagem restrita de organização dos protocolos em camadas funcionais (*host-centric approach*). Em linhas gerais, abstrai-se o esquema de acesso a um recurso pelo endereço IP passando a acessá-lo pelo seu nome, independente da localização do nó gerador do conteúdo de interesse. Isto porque se considera que entregar o conteúdo é mais importante do que a forma de obtê-lo, pois 90 % do tráfego na Internet consistem de dados disseminados a partir de uma fonte para diversos destinos [46].

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 [50–52], 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 [1, 53]. Assim, reduz-se drasticamente as perdas de dados, porque as filas de roteamento não atingem 100 % da sua capacidade; promove-se o compartilhamento equânime do canal

e descarta-se a necessidade de procedimentos de inicialização que subutilizam os canais de transmissão, como a fase de partida lenta do TCP [39, 54].

Nesse contexto, baseando-se em resultados de pesquisas publicados recentemente, fazer uso dos algoritmos de controle de congestionamento assistidos pela rede é a melhor abordagem para realizar controle de congestionamento em transmissões de dados na Internet [50]. Sendo assim, convém também utilizar estes algoritmos para distribuição de mídias ao vivo. No escopo deste trabalho tal abordagem é adotada. Para isto, destaca-se o protocolo *Rate Control Protocol* (RCP) [52, 55, 56], utilizado de forma adaptada para atender às necessidades do protocolo proposto neste trabalho. Outros protocolos desta mesma classe também deve ser mencionado, como o *Explicit Control Protocol* (XCP) [51, 57, 58], o *Variable-Structure Congestion Control Protocol* (VCP) [59–63] e o *Congestion Exposure* (ConEx) [47, 64, 65]. Este último, não é considerado um protocolo para controle de congestionamento propriamente dito, mas auxilia uma fonte transmissora de um fluxo de dados a expor para a rede informações sobre o congestionamento sofrido pelos pacotes anteriormente transmitidos dentro do referido fluxo. Com base nas informações de congestionamento, um algoritmo de congestionamento pode tomar decisões para regular a taxa de transmissão mais apropriada para os sistemas finais.

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

Por fim, argumenta-se neste trabalho que ao mover a execução de alguns serviços para o núcleo da rede, ou pelo menos utilizar a rede para auxiliar neste processo, facilita-se o uso de alguns recursos antes difíceis de serem utilizados, tais como transmissões de fluxos de dados em modo *multicast*. No caso do *multicast*, embora esteja disponível há mais de 20 anos, utilizar tal abordagem largamente na Internet não se mostrou uma opção viável e poucas soluções o adotaram largamente na Internet [66]. Isto ocorre devido às barreiras administrativas que dificultam a manipulação de rotas *multicast*, principalmente por conta da dependência do administrador da rede em ter que habilitar explicitamente este tipo de tráfego de dados para um cenário específico de aplicação. Contudo, é possível utilizá-lo quando controlado por um protocolo capaz para habilitá-lo automaticamente de acordo com a demanda das aplicações,

onde os fluxos poderão ser compartilhados, mantendo-se o princípio da independência dos serviços de rede, ou seja, sem que a aplicação precise se preocupar como isto é feito.

Com esse norte, delimitou-se algumas métricas que determinam a qualidade de serviço para permitir que os usuários assistam conteúdo ao vivo na Internet. Tais métricas são divididas em três grupos, detalhados a seguir:

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

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

1.2 Descrição do Problema

Nos últimos anos, diversos esforços acadêmicos e da indústria foram feitos para disponibilizar sistemas de distribuição de conteúdo ao vivo na Internet [67–81]. 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 [82–84]. Isto não seria um problema para a infra-estrutura de rede e consequentemente para os sistemas finais se essa pulverização não potencializasse duplicações de fluxos de dados da transmissão de um evento destinado a um sub-conjunto de redes contendo nós receptores interessados em tal evento [85–93]. Como consequência, exponencia-se o consumo desnecessário de recursos computacionais e de rede.

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

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

Apesar de existirem protocolos para a Internet que descrevem o conteúdo da mídia transmitida [94–96], atualmente não existe um protocolo de transporte de mídia ao vivo que considere toda a complexidade existente em se utilizar as redes orientadas a datagramas, ponderando-se as recentes soluções empregadas nos sistemas mais modernos e estudados neste trabalho. Em vez disso, cada sistema implementa suas próprias formas de transporte de dados, com a execução de ações complexas implementadas na camada de aplicação, com predominância e variações de uso dos protocolos RTP/RTSP combinado com o UDP.

A transmissão de múltiplos fluxos de dados com o mesmo conteúdo por parte de diferentes sistemas, gera um consumo desnecessário de recursos de rede e, principalmente, a impossibilidade de que dois ou mais nós conectados em sistemas distintos possam cooperar entre si, compartilhando o mesmo fluxo de dados transmitido por um servidor, quando há interesse das partes envolvidas. Por exemplo, é comum encontrar na Internet diversos sistemas de distribuição de conteúdo, cada um com milhares de usuários conectados, porém recebendo fluxos independentes do mesmo jogo de futebol, da corrida de fórmula 1, etc [76, 79, 97–102]. A duplicação pode ser percebida até em cenários mais simples, quando dois ou mais nós conectados à Internet pela mesma rede estão utilizando diferentes sistemas para assistir ao mesmo evento ao vivo. Idealmente, os nós deveriam compartilhar o mesmo fluxo de dados e cada um renderizar o conteúdo da forma definida pela aplicação, desacoplando a forma de transportar os dados da forma como estes são apresentados pela aplicação (princípio da independência dos serviços) aos seus respectivos usuários, como no caso do serviço Web.

O grande sucesso do serviço Web, no ponto de vista de protocolo de comunicação, ocorre devido à independência da forma que se implementa desacopla o conteúdo a ser transmitido e a comunicação entre o cliente (navegador) e o servidor. Independente do modelo e versão do navegador e do servidor web, todos transportam dados utilizando o protocolo TCP, evitando-se assim a pulverização de diferentes formas de transportar os dados de tal serviço – fica apenas a cargo do desenvolvedor de cada aplicação decidir como exibir as informações recebidas. Contudo, percebe-se a inexistência de um protocolo equivalente ao TCP para os sistemas de distribuição de conteúdos multimídia, apesar da grande demanda e sucesso de utilização dos mais variados sistemas de transmissão multimídia disponíveis na Internet. Um protocolo similar ao TCP para o cenário de distribuição de conteúdos multimídia ao vivo poderia evitar as duplicações de transmissão por meio de estratégias colaborativas entre os nós interessados por um mesmo conteúdo, uma vez que não há a necessidade de individualização dos dados transmitidos. Como consequência, seria possível fazer melhor uso dos canais de transmissão ao enviar fluxos de dados apenas quando necessário, restando para as aplicações a responsabilidade de renderizá-los ao seus usuários, como ocorre no serviço Web. Nesse contexto, a arquitetura de serviço P2P [103] combinada com o *Application Layer Multicast - ALM* [104] é uma abordagem interessante que explora os aspectos de comparti-

lhamento de fluxo de dados, mas os desenvolvedores ainda continuam obrigados a imbutir a implementação das diretrizes do P2P/ALMA nas próprias aplicações, em vez de utilizá-las como os desenvolvedores de navegadores web ou de serviços web fazem: delegar ao TCP algumas importantes responsabilidades. O TCP abstrai toda complexidade de algumas responsabilidades da camada de transporte, tal como garantia de entrega de dados, ordenação de segmentos, controle de congestionamento etc. Algumas funções implementadas em soluções P2P/ALMA estão consolidadas e bastante utilizadas nos sistemas de distribuição de conteúdos multimídia ao vivo, cada um a sua maneira e, mesmo que sejam similares, ainda são incompatíveis.

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

Em geral, nos middlewares para distribuição de conteúdos multimídia ao vivo, implementam-se mecanismos para conexão multi-ponto e topologias [13, 24, 93, 105–109], seleção de nós [110–113], tolerância à desconexões e de outros problemas causados pelo *churn* [42, 105, 109, 114, 115], disponibilização de informação de contexto sobre rede para dar suporte à execução dos serviços das aplicações, como níveis de congestionamento e taxa de transmissão [116–118], estratégias para incentivos à cooperação entre nós [113, 119–124], algoritmos para inibir a participação de nós *free-riders* [122, 125–127] e segurança [124, 128–131]. Temas como estes foram ganhando importância ao longo dos anos porque os serviços de transmissão de mídias ao vivo foi se tornando uma tendência e hoje é uma realidade. Apesar da arquitetura da Internet ser modularizada em camadas funcionais, ao longo dos anos as soluções promovidas no contexto dos referidos temas não foram disponibilizadas estrategicamente nas camadas corretas, mas sim apenas na camada

de aplicação. Isto se explica pelo fato de que é muito mais fácil implementar uma solução na camada de aplicação do que em outras camadas mais abaixo, não apenas por aspecto de facilidade de desenvolvimento, mas também de implantação em larga escala no mercado, pois todas as outras camadas estão implementadas do sistema operacional para baixo.

De forma alternativa, uma maneira eficiente de se estruturar e enviar um fluxo de vídeo a um grupo de usuários na Internet seria a utilização de *multicast* no nível de IP [92]. Em uma sessão de *multicast* IP, uma estrutura de árvore é formada. A fonte de vídeo se torna a raiz desta árvore *multicast* e os clientes recebem o fluxo de vídeo através dos vários nós desta árvore, formada pelos roteadores que suportam *multicast* em nível de IP. Apesar de melhorar a escalabilidade no número de receptores do mesmo conteúdo ao vivo, tal estratégia não é suficiente devido a complexidade de habilitar, principalmente em grande escala, os canais de comunicação *multicast*, pois, para isso, precisa-se de intervenção humana. Para contornar a dificuldade em habilitar o modo *multicast* em nível de IP, a arquitetura ALM, equivalente ao IP *multicast*, porém implementada a nível da camada de aplicação. A diferença entre o *multicast* IP e o ALM é que no primeiro os roteadores duplicam e repassam os datagramas para os próximos roteadores até alcançar os sistemas finais, ao passo que no segundo os sistemas finais executam as ações citadas anteriormente, mas ainda dependem de funções de rede de mais baixo nível, o que pode aumentar o tempo de processamento e a complexidade da aplicação. Existem diversos *middlewares* baseados em ALM e estes implementam suas próprias interfaces e uso e funcionalidades, incompatíveis entre si [81, 132–138]. Apesar de todos esses esforços, questão da pulverização de aplicações mencionada no ínicio dessa seção vem à tona mais uma vez: cada sistema adota suas estratégias de formação da árvore *multicast*, portanto incompatíveis entre si.

Em vez disso, pode-se combinar as duas formas supracitadas em uma solução de protocolo disponível para qualquer aplicação de rede interessada em transmitir e/ou receber um fluxo de dados ao vivo. Ou seja, um protocolo que constitui uma rede de sobreposição formada por roteadores de rede, com suporte automático ao uso de *multicast*, através da qual as aplicações possam trocar dados entre si de forma transparente. Isto significa mover alguns serviços consolidados na camada de aplicação para camadas mais abaixo, abstraindo-se a grande complexidade de se constituir redes P2P, principalmente compatíveis entre si. Nas Seções 2.1 e 2.2, estende-se essa discussão no sentido de entender o funcionamento e os

aspectos arquiteturais das aplicações de distribuição de conteúdos multimídia ao vivo, em redes P2P, bem como os principais problemas com base em alguns cenários de uso.

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

Além do TCP e do UDP, na segunda metade dos anos 2000, outros protocolos mais modernos passaram a apresentar avanços significativos para permitir a transmissão de conteúdos multimídia, como é o caso do DCCP (*Datagram Congestion Control Protocol*) [139, 140]. Apesar de sua eficiência em alguns cenários de transmissão de dados multimídia na Internet, o DCCP possui falhas críticas quando utilizado em larga escala em cenários $1 \rightarrow n$ [141–143]. O fato é que o DCCP é um protocolo orientado à conexão e portanto para cada novo usuário interessado em receber um fluxo multimídia transmitido, uma nova conexão se faz necessária, não havendo a possibilidade de compartilhamento entre diferentes nós interessados pelo mesmo conteúdo, pelo menos de forma automática e transparente para as aplicações de rede. Qualquer protocolo que exija estabelecimento de conexão e realize controle de congestionamento por fluxo de dados apresenta tal limitação, o que gera dois problemas:

1. *excessivo consumo de recursos computacionais*: para cada nova conexão, o nó transmissor deve alocar recursos computacionais (memória e processamento) para tratar cada nova conexão. Em cenários de transmissão multimídia $1 \rightarrow n$, se muitos nós estão conectados em um único servidor, então isto elevará sobremaneira o consumo de recursos computacionais do nó transmissor proporcionalmente à quantidade de nós receptores interessados pelo fluxo multimídia transmitido. Além disso, embora o con-

teúdo transmitido por um nó seja de interesse de muitos outros nós, os fluxos são mantidos independentemente uns dos outros, o que gera duplicações desnecessárias e consequentemente desperdício de recursos de rede. O trabalho mais notável onde discute-se as consequências deste problema em aplicações de rede é o encontrado na referência [92].

2. *a taxa de transmissão de fluxos DCCP individualmente tenderá a 0 (zero):* no caso do DCCP, utiliza-se uma equação matemática para definir a taxa de transmissão durante uma conexão. À medida que mais nós se conectam a um nó transmissor, menor será a taxa de transmissão do nó transmissor para cada um dos nós receptores conectados a ele. Para a rede, esta estratégia é equânime e evita que a mesma entre em colapso de congestionamento, mas para cada fluxo de dados isto é ruim. Este problema tem uma relação estreita com o dilema observado por Garrett Hardin em 1968 e denominado de Tragédia dos Bens Comuns (*Tragedy of the Commons*) [144], presente em diferentes áreas do conhecimento. No caso de protocolos como o DCCP, a tragédia dos bens comuns ocorre porque, à medida que novos fluxos são transmitidos na rede, menor é a taxa de transmissão alocada individual para cada fluxo, a qual pode se tornar insuficiente para a recepção de um fluxo multimídia e, por consequência, nenhum nó receptor reproduzirá o fluxo transmitido pelo nó transmissor, embora todos os fluxos utilizarão o canal de forma equânime.

Desta forma, apesar de os algoritmos de controle de congestionamento buscarem convergir para o caso do melhor global (equidade para com todos os fluxos e assim evitar congestionamento da rede), isto provoca o efeito do caso do pior local (redução da taxa de recepção de cada nó da rede). Este fato pode ser explicado analiticamente utilizando como base a Equação 1.1, que define cada taxa de transmissão X_i para realizar o controle de congestionamento em cada fluxo [145]. Para efeito de estudo, esta equação do protocolo DCCP/CCID3 foi a escolhida, por ser utilizada em vários protocolos de rede, porém equações similares de outros algoritmos para controle de congestionamento poderiam ter sido utilizadas. Nesta equação, define-se:

- X_i é a taxa de transmissão em bytes/segundo;
- s é o tamanho do pacote em bytes;

- R é o RTT (*Round Trip Time*) em segundos;
- p é a taxa de ocorrência de perdas, entre 0 e 1;
- RTO (*Retransmission TimeOut*) é o valor do temporizador de retransmissão do TCP em segundos; e
- b é igual a 1 e representa o número máximo de pacotes confirmados por um único ACK.

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

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

Embora esta seja uma discussão teórica, simulações foram realizadas em busca de evidências mais contundentes de que este fato pode ocorrer na prática. Os resultados e discussão sobre estas simulações são apresentados em [141, 142, 146]. Em resumo, de acordo com os resultados obtidos, apenas fluxos DCCP foram suficientes para causar o problema em questão, sequer fluxos de outros protocolos foram utilizados. Em situações mais realistas, o problema se agrava porque o protocolo DCCP disputará o canal de transmissão não apenas com fluxos DCCP, mas também com fluxos de protocolos como o TCP, UDP, SCTP [147, 148], dentre outros.

Apesar de o protocolo DCCP ter sido utilizado para evidenciar o problema que acabara de ser apresentado, esta discussão pode ser generalizada para qualquer outro protocolo de rede que seja orientado à conexão e que realize controle de congestionamento de fluxos individualmente. A fim de aumentar a representatividade de protocolos nesse contexto, o leitor pode consultar outros trabalhos referenciados em [149–168] e no Capítulo 3 deste documento.

Para contornar os problemas discutidos até aqui, torna-se imprescindível projetar um protocolo de rede com vistas a reduzir a pulverização dos sistemas de distribuição de conteúdo, inspirando-se em casos consistentes ao longo dos anos, como o serviço Web, e que considere a evolução do estado da arte como aliado nesse processo, tais como as funções consolidadas das redes P2P, os novos conceitos das redes centradas no conteúdo e os algoritmos de controle de congestionamento assistidos pela rede, que são mais eficazes se comparados aos algoritmos baseados em perda de pacotes, mudança no atraso e relatórios enviados pelos receptores aos transmissores [149, 152, 153, 155, 156, 169, 170].

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

1.3 Hipótese

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

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

1.4 Objetivo

Neste trabalho, o objetivo é a concepção e a avaliação de um protocolo de rede denominado *Global Media Transmission Protocol* (GMTP), para uso em sistemas que consideram uma arquitetura híbrida P2P e CDN para distribuição de conteúdos multimídia ao vivo. Mais especificamente, propõe-se um protocolo que explora a relação entre o escalonamento de recursos computacionais em redes P2P, a estabilidade das redes CDN e os mecanismos que possam mitigar a complexidade no processo de construção desses sistemas ao mover diversas funções atualmente implementadas nesses sistemas para as camadas de transporte e rede, a fim de utilizar eficientemente os recursos de rede e consequentemente melhorar a qualidade da experiência do usuário ao reproduzir um conteúdo ao vivo na Internet.

1.4.1 Objetivos Específicos

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

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

qualidade da experiência do usuário, suas vantagens e desvantagens;

1.5 Relevância do Tema e da Tese

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

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

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

No que diz respeito à relevância do trabalho na perspectiva de engenharia de software para sistemas distribuídos, o autor considera indispensáveis três principais requisitos que estão sendo contemplados e que reforçam a relevância da tese. Estes requisitos servem como motivação para a realização das atividades que foram desenvolvidas ao longo deste trabalho.

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

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

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

1.6 Resumo das Contribuições

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

1. a concepção e descrição detalhada do protocolo introduzido. Neste caso, apresenta-se a elaboração do projeto de um protocolo para distribuição de conteúdos multimídia ao

vivo, que considera a comunicação mais efetiva entre as duas camadas mais importantes da pilha TCP/IP, a de transporte e a de rede. Além disso, traz-se à tona os recentes avanços da área de estudo, testando-se o uso dos novos conceitos das redes centradas no conteúdo aplicados ao processo de distribuição de conteúdos e combinados com os recentes avanços dos algoritmos de controle de congestionamento assistido pela rede. O uso do protocolo introduzido contribuirá para a padronização da forma como os desenvolvedores implementam as principais funcionalidades dessas aplicações e, sobretudo, permitindo que estas sejam implementadas de tal forma a promover o reúso em soluções existentes, utilizando-se dos recursos de rede de forma mais eficiente.

2. a implementação de artefatos de software que permitem a reprodução do ambiente de estudo, tais como implementações no simulador de rede [171] OMNet++ e NS2, bem como um protótipo de implementação no núcleo do sistema operacional Linux;
3. os resultados e discussões aprofundadas acerca do protocolo introduzido frente a outra proeminente solução denominada Denacast e CCNx. O primeiro é um sistema baseado no serviço Coolstreaming com suporte a uma arquitetura P2P/CDN, ao passo que o segundo é um protocolo que segue as diretrizes das redes centradas no conteúdo; e
4. os encaminhamentos da pesquisa sobre futuros avanços do estado da arte, tais como as limitações do protocolo proposto e uma breve discussão de como estas poderão ser aprimoradas.

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

1.7 Publicações

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

2. *Sharing Resources with Live Streaming Systems to Improve Users' QoE on Consumer Electronic Devices.* IEEE Transaction on Consumers Eletronics (JCR 1.087).
3. *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 [172].
4. *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 [141].
5. *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 [142].
6. *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 [146].
7. *On the Performance of TCP, UDP and DCCP over 802.11g Networks.* 23rd ACM Symposium on Applied Computing. Qualis A-1 (Computação). 2008. Referência [173].

1.8 Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

TBD

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 funções dos sistemas de distribuição de mídia ao vivo em arquiteturas P2P, o funcionamento de um sistema de distribuição de conteúdos multimídia ao vivo e os conceitos básicos sobre criptografia e assinatura digital. Se o leitor se considera familiarizado com esses assuntos, recomenda-se, pelo menos, a leitura do sumário deste capítulo, disponível na Seção 2.4. Com relação aos conceitos sobre os protocolos de rede e outros temas como controle de congestionamento e modos de transmissão, estes não são abordados neste capítulo por serem assuntos consolidados e vastamente disponíveis na literatura. Apesar dessas omissões, recomenda-se (re)visitar as seguintes referências:

- [103, 174, 175], para conceitos básicos sobre comunicação e redes de computadores, modos de transmissão, redes de distribuição de conteúdo e protocolos de rede;
- [143], 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;
- [45–49], para conceitos sobre Redes Centradas no Conteúdo; e
- [92, 128], para mais detalhes sobre os conceitos apresentados neste capítulo.

2.1 Distribuição de Mídia ao Vivo em Arquiteturas 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. De forma contrária, 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. Essa é a classe de transmissão de conteúdo tratado neste trabalho.

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 [92].

Recentemente, vários sistemas P2P foram desenvolvidos para prover conteúdo de vídeo ao vivo e sob-demanda na Internet, com baixo custo operacional [79, 81, 98, 99, 176–180]. 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*) [181, 182]. 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 [92]. 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 [136], 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 fluxo de vídeo deve ser reconstruída o mais rapidamente possível. Na Figura 2.2, ilustra-se um cenário em que um nó deixar o sistema de vídeo e a árvore de *multicast* ao nível de aplicação, que deve ser reconstruída.

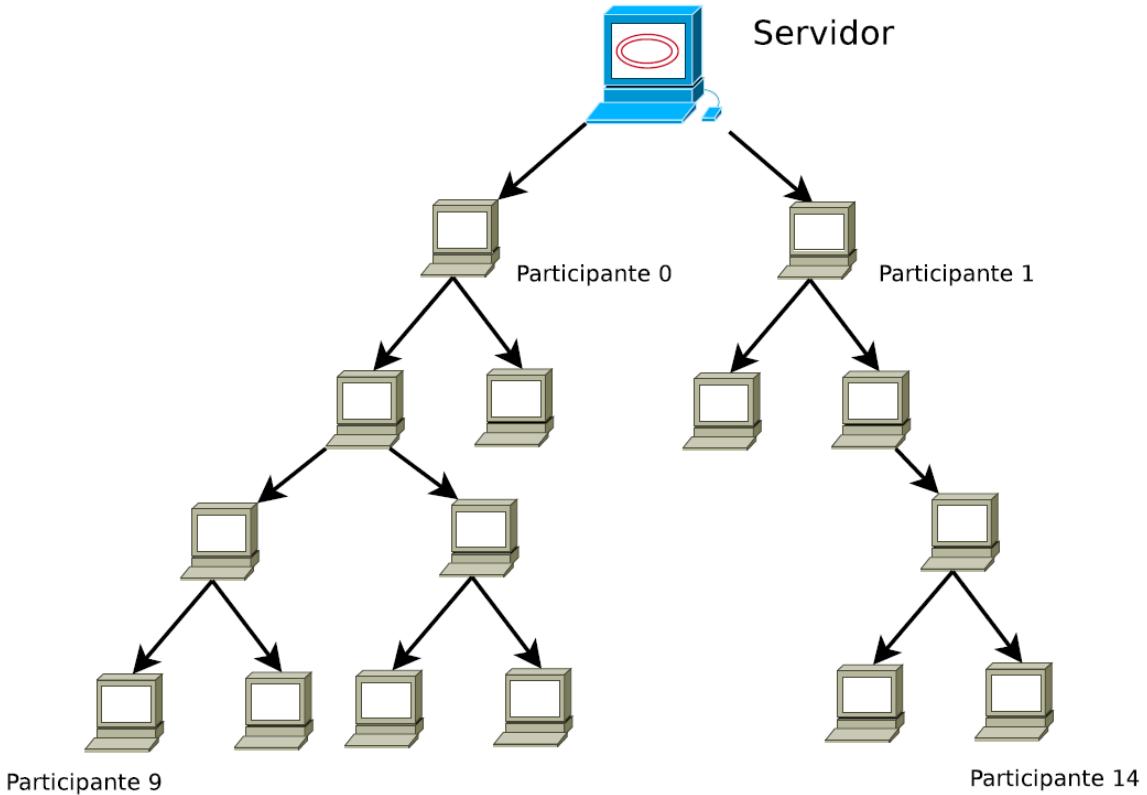


Figura 2.1: Árvore de *multicast* em nível da camada de aplicação.

A construção e manutenção da árvore de envio de fluxo P2P pode ser realizada de maneira centralizada ou descentralizada. Em uma abordagem centralizada, um servidor controla a construção da árvore e sua recuperação. Para grandes sistemas de envio de vídeo, uma abordagem centralizada pode se tornar um gargalo e um ponto de falha [92]. Vários algoritmos distribuídos abordam e tratam o problema de manutenção e construção da árvore de maneira distribuída [102]. 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 [92, 181, 182].

Outro problema encontrado ao se usar uma árvore simples é que os nós, que estão na folha da árvore, acabam por não contribuir com o sistema. Assim a utilização de banda não é totalmente aproveitada. Uma vez que existe um grande número de nós folhas, a capacidade da árvore se torna subestimada. Para lidar com esse problema, foram propostas abordagens baseadas em múltiplas árvores como em [81]. Nesta abordagem, um servidor divide o fluxo de vídeo em vários subfluxos e para cada um destes, uma árvore *multicast* ao nível de apli-

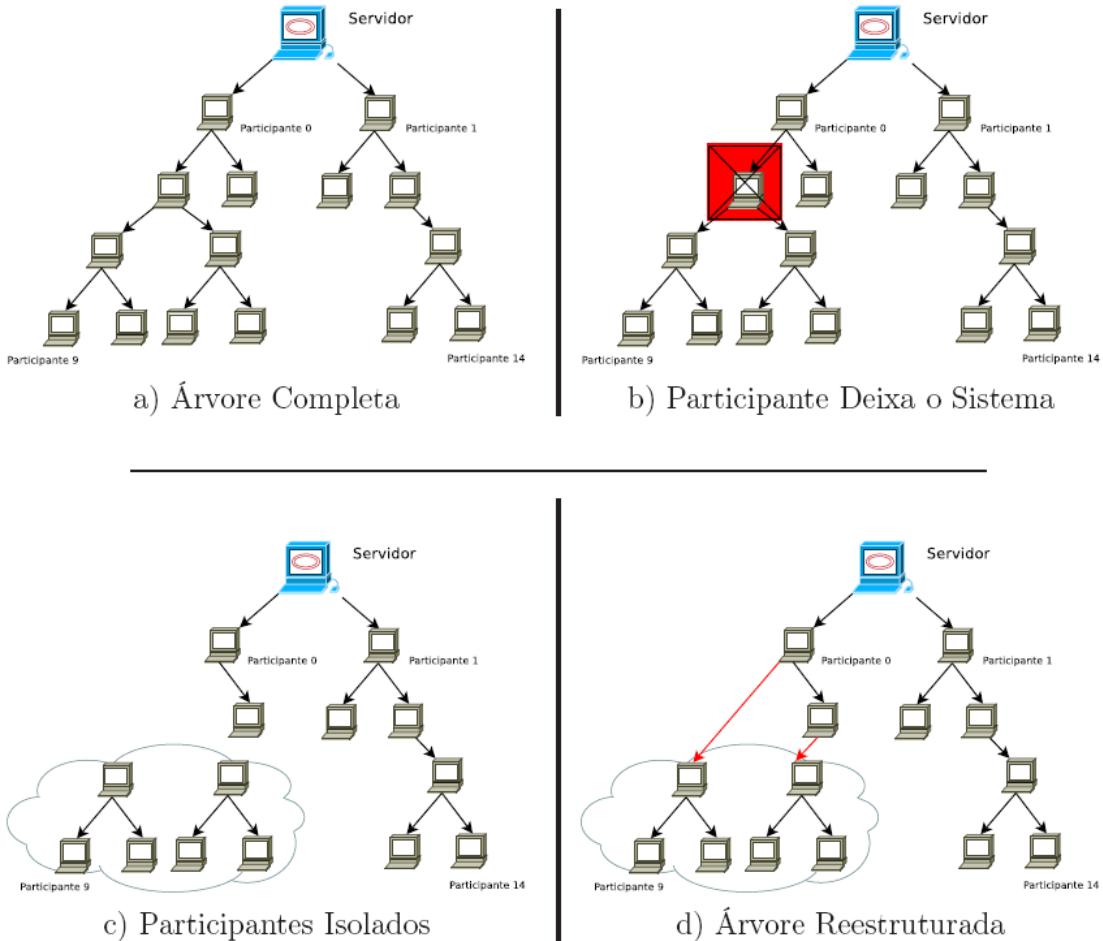


Figura 2.2: Manutenção da árvore de *multicast* em nível da camada de aplicação.

cação 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 disponíveis momentaneamente. Um nó participante se conecta a um subconjunto de outros nós participantes do sistema e, periodicamente, todos trocam informações entre si. Os dados são buscados nos nós participantes que já os têm. Como um nó participante tem múltiplos vizi-

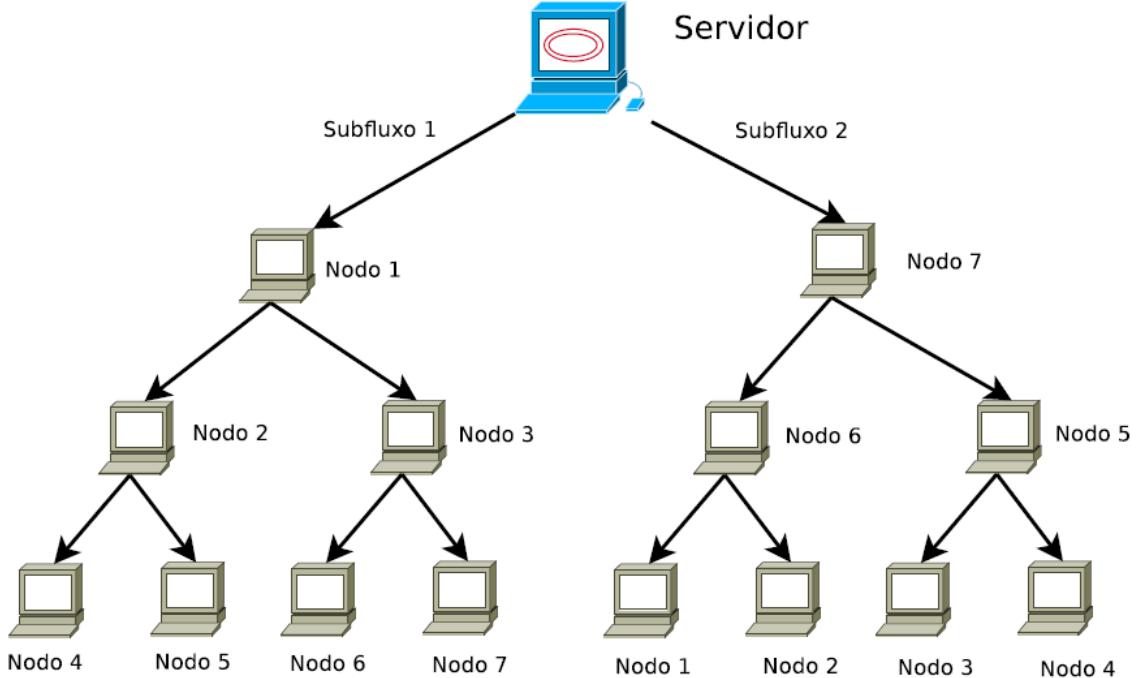


Figura 2.3: Sistema baseado em múltiplas árvores com dois subfluxos.

nhos 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 [78, 99, 180, 183]. 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 [182], 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.

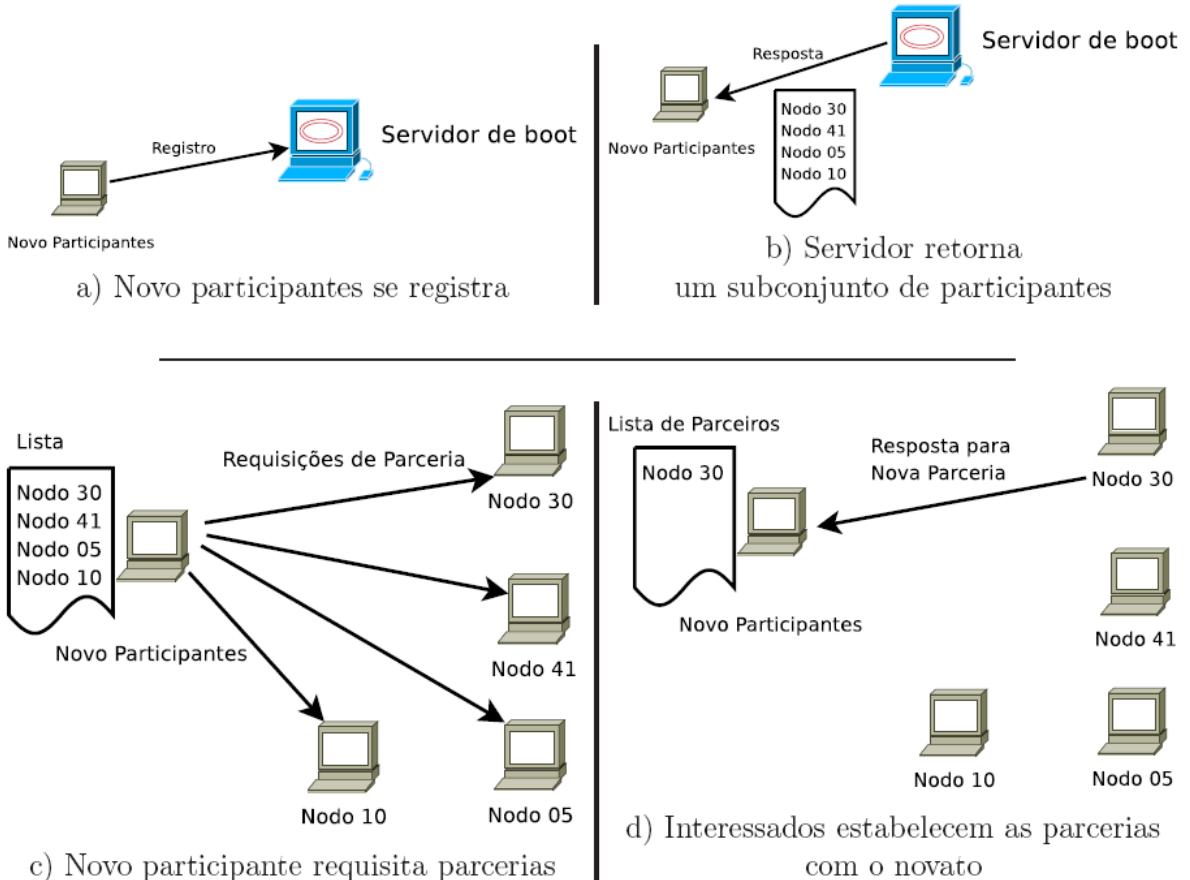


Figura 2.4: Atividade inicial de um novato - rede P2P baseada em malha.

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 [180]. 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 [92]. 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 [92]. Esses *chunks* são numerados em uma sequência temporal, para que os nós participantes possam identificar e executar o vídeo correspondente de forma apropriada. Os pedaços do fluxo são disseminados a partir do servidor para diversos participantes da rede, que os disseminam para seus companheiros, e assim por diante. Como os *chunks* tomam diferentes caminhos para atingir os diversos pontos da rede, estes chegam a um usuário fora de ordem e, para uma execução contínua do vídeo, os nós participantes guardam os *chunks* em um armazenamento temporário de memória, onde são ordenados antes de sua apresentação. Dependendo do tipo de aplicação, o armazenamento pode variar de segundos a minutos. Em uma sessão de vídeo ao vivo, que é o período em que um fluxo de mídia é transmitido, a sequência de identificação dos *chunks* cresce enquanto o vídeo é disseminado.

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

transferências dos dados [180].

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.

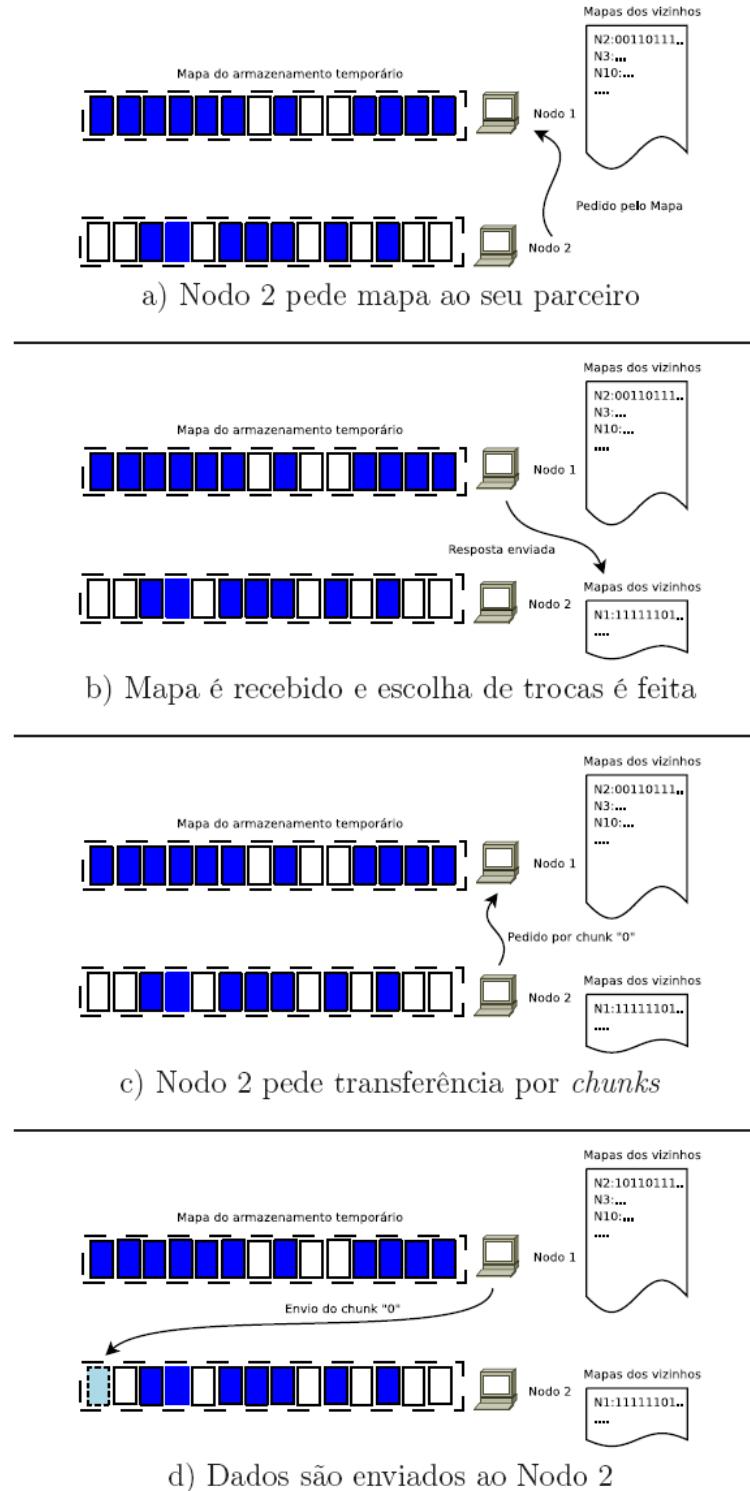
Na Figura 2.5, ilustra-se a troca de *chunks* em uma aplicação com estrutura baseada em malha. Por esta figura, o nó 2 gera seu mapa, indicando quais *chunks* tem disponível em seu armazenamento temporário. O nó 2 então troca este mapa com o nó participante 1 e, como resposta, o nó 1 envia também o seu 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.

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



hierarquias, entretanto, a recepção dos dados da transmissão ao vivo está sujeita a atrasos e imprevisibilidade [77].

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

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

Híbrido por Encaminhamento Automático / Pedidos Explícitos (*Push-Pull*)

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

Existem alguns mecanismos propostos com a combinação do “*push-pull*” [73, 184]. 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 [185] 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 [186], 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 [187], 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 [98], o PPLive [100], o GridMedia [180, 187], Octoshape, e o CoolStreaming [90, 99]. Na Figura 2.6, ilustra-se um cenário exemplo de um sistema de transmissão ao vivo em P2P que será detalhado nessa seção.

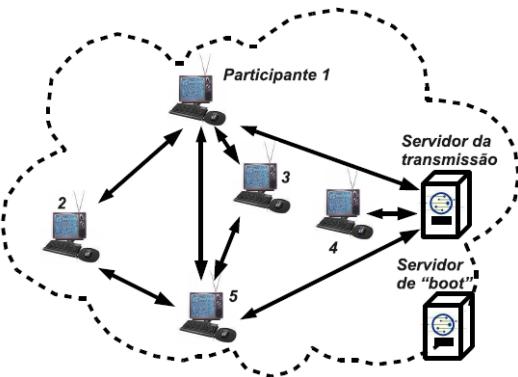


Figura 2.6: Modelo de sistema utilizado.

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 [97, 187].

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

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

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

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

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

Periodicamente, p_i verifica quais *chunks* c_t são necessários ($cm_i[c_t] = desejada$) e verifica entre seus parceiros p_j , com $p_j \in LP_i$, quais possuem o *chunk* c_t com $cm_j[c_t] = disponivel$. O parceiro p_j que contém o *chunk* c_t e que possui maior disponibilidade de recursos é escolhido por p_i para a realização do pedido de *chunk*. Quando p_i recebe o *chunk* c_t de p_j , p_i marca $cm_i[c_t] = disponivel$. Os processos de escolha do *chunk* a ser requisitado

e a escolha do parceiro serão detalhados nas seções seguintes. Na Tabela 2.1, apresenta-se um resumo dos elementos utilizados para descrever um sistema P2P de transmissão ao vivo.

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

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

2.2.1 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, $cm_s[c_t] = \text{disponível}$.

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

Nas aplicações mais populares, o servidor S gera os dados da mídia contínua ao vivo a uma taxa aproximada de 6 *chunks* por segundo. Normalmente, um vídeo é codificado a

aproximadamente 300 Kbps e assim, cada *chunk* tem cerca de 6 KB de dados.

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

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

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

2.2.2 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 inicializada com conteúdo vazio e, à medida que p_i recebe os *chunks* de seus parceiros, cada posição vai sendo preenchida. Nesse esquema de armazenamento temporário, o nó p_i tem

por objetivo manter o *buffer* B_i preenchido de forma que se garanta uma contínua exibição da mídia ao vivo, mesmo se este perder conexão temporariamente com seus parceiros.

Inicialmente, o *buffer* B_i pode ser representado por uma estrutura do tipo “*buffer circular*”. Dois processos trabalham em conjunto para manter o *buffer* B_i preenchido e a execução da mídia constante (produtor/consumidor). Assim, a posição de B_i a ser consumida é $B_i[0]$ e a posição mais recentemente a ser preenchida será $B_i[b]$ (b é a última posição de um *buffer* com pelo menos $b + 1$ posições).

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

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

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

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

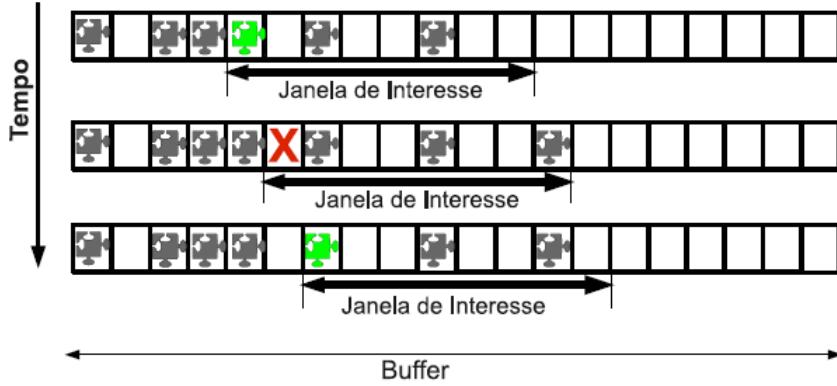


Figura 2.7: Mecanismo de consumo da mídia ao vivo.

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 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 [99]. 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 acabou 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 LP_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_j 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 [188], 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* [189] $H(m)$ 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*) [190].

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^+ .



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

2.3.3 Assinatura e Certificação Digital

Uma assinatura digital [191] é um modelo de autenticação de informações digitais tipicamente análoga à assinatura física em papel. Através de uma assinatura digital, pode-se provar 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:

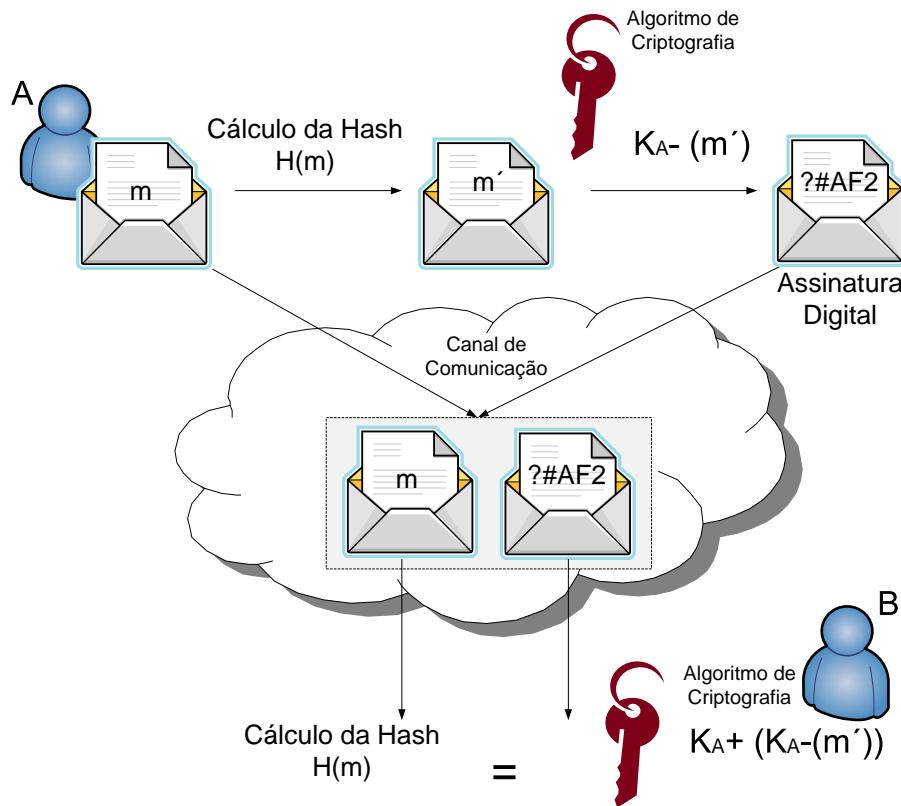


Figura 2.9: Geração de Assinaturas Digitais.

- Se $m'_{recebida} = m'_{enviada}$, a mensagem m está íntegra, ou seja, a mensagem m não sofreu qualquer alteração. Sendo assim, a assinatura está correta, pois foi gerada pela chave privada correspondente à chave pública utilizada na verificação; ou
- Se $m'_{recebida} \neq m'_{enviada}$, a a mensagem m foi alterada, ou seja, a assinatura digital emitida pelo sistema final A não corresponde à mensagem m .

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

Os mecanismos oferecidos pela assinatura digital é uma forma eficaz de garantir a au-

toria de documentos eletrônicos. Em agosto de 2001, a Medida Provisória da República do Brasil, número 2.200 [192] (*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 considerados são baseados em uma arquitetura em malha e sem organização rígida dos nós do sistema. Os nós podem entrar e sair a qualquer instante e realizam parcerias com um subconjunto de outros nós. Os parceiros trocam informações entre si para colaborar uns com os outros na obtenção de uma mídia transmitida ao vivo.

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

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

Caso mais de um parceiro possa contribuir com o dado necessário, um nó deve escolher

a qual fará a solicitação. O mecanismo adotado se baseia na disponibilidade de recursos de cada parceiro, que será escolhido de acordo com a maior quantidade de recursos disponíveis, como por exemplo, banda de rede, processamento, memória etc.

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

Capítulo 3

Trabalhos Relacionados

=====

Usar na apresentação: <http://www.cs.helsinki.fi/u/jakangas/Teaching/P2P/P2P-02-Systems.pdf>

5343509, 6121327, 4378429, <https://peercdn.com/> - só para conteúdos estáticos, mais uma para fragmentar Is there a future for mesh-based live video streaming? (Set/2008) - analisar pra ver se rola...

PODE PEGAR ESSE MATERIAL PARA MENCIONAR AS QUESTÕES DE SEGURANÇA E PORQUE O GMTP TEM

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

=====

Neste capítulo apresenta-se uma avaliação crítica acerca de um conjunto de trabalhos sobre protocolos de transporte para distribuição de conteúdos em aplicações caracterizadas por um nó transmissor e muitos nós receptores. Trabalhos que não apresentam uma proposta de protocolos de transporte de dados multimídia, mas que são considerados proeminentes na literatura também são apresentados e analisados por apresentarem alguma similaridade com a proposta do GMTP.

A compilação dos trabalhos a seguir foi realizada baseando-se em informações obtidas e adaptadas de publicações encontradas em diversas fontes (revistas, conferências, livros, teses

e dissertações etc.) disponíveis na literatura.

Na Seção 3.1 apresenta-se uma breve discussão acerca de protocolos multimídia padronizado. Em seguida, na Seção 3.2 discute-se o estado da arte acerca de protocolos de transporte, dando-se ênfase às propostas de protocolos de transporte que utilizam uma abordagem P2P. Por fim, na Seção 3.5 apresenta-se um sumário comparativo sobre os protocolos apresentados e uma breve discussão sobre a proposta diferenciada promovida no GMTP.

3.1 Protocolos Multimídia Padronizados

Alguns protocolos padronizados pela IETF foram especificados, implementados e disponibilizado publicamente. Em geral, os protocolos dessa categoria permitem a criação, encerramento e controle de conferências, como videoconferência e uma transmissão ponto-a-ponto de um conteúdo de audio/vídeo em tempo real. Nesta perspectiva, a seguir, apresentam-se os protocolos de rede que merecem destaque, principalmente por serem empregado na maioria das aplicações multimídia existentes.

3.1.1 H.323 / ITU-T

O processo de desenvolvimento de padrões abertos iniciou com um grupo de estudos do ITU-T (*International Telecommunication Union – Telecommunication Section*), em 1996. O ITU-T especificou o padrão H.323 [193], que estabelece uma arquitetura de comunicação destinada ao controle de conferências de voz, vídeo e dados sobre redes TCP/IP. O H.323 se tornou largamente utilizado, uma vez que, a época da especificação de sua segunda versão, em 1998, não havia qualquer padrão aberto e aceito pelo mercado capaz de atender às aplicações multimídia. Embora as transmissões multimídia em tempo real por *multicast* já estivessem sendo realizadas no Mbone [66] há algum tempo, não havia ainda qualquer padrão aberto para controle de conferências multimídia.

3.1.2 SIP – *Session Initiation Protocol*

No período equivalente a maturação do H.323, a IETF iniciou o desenvolvimento de uma arquitetura de comunicação mais flexível e poderosa que o H.323. Alicerçada sobre o proto-

colo SIP (*Session Initiation Protocol*) [194], a arquitetura SIP teve logo potencial reconhecido, uma vez que supria todos os pontos fracos da arquitetura H.323, como a demora no estabelecimento de conexão (canais H.225 e H.245) e a complexidade de operação.

3.1.3 RTP – *Real Time Protocol*

Devido à natureza da pilha de protocolos TCP/IP, voltada à transmissão de dados pelo paradigma do “melhor esforço”, sem qualquer cumprimento a requisitos de tempo, novos protocolos foram propostos para as necessidades de transmissão de dados em tempo real. O protocolo em destaque neste sentido é o RTP (*Real Time Protocol*) [95], um protocolo de transporte para aplicações de tempo real. O RTP atualmente constitui a base das transmissões multimídias em tempo real na Internet, sendo o padrão adotado em praticamente todas as soluções multimídia baseadas em IP. O RTP oferece um mecanismo para que as aplicações tenham controle sobre número de seqüência e marcas de tempo de cada pacote de dados transmitido, não oferecendo qualquer mecanismo de retransmissão de informações perdidas.

3.1.4 RTSP – *Real Time Streaming Protocol*

Em paralelo ao RTP, as aplicações multimídia utilizam o *Real Time Streaming Protocol* (RTSP) [96]. O RTSP é um protocolo a nível de aplicação desenvolvido pela IETF para controle na transferência de dados com propriedades de tempo real. Tal protocolo torna possível a transferência, sob demanda, de dados em tempo real como áudio e vídeo. O RTSP serve para estabelecer e controlar um único ou vários fluxos sincronizados de mídias contínuas pertencentes a uma apresentação.

O conjunto de fluxos a ser controlado é definido por uma descrição de apresentação, normalmente um arquivo, que pode ser obtido por um cliente usando HTTP ou outros meios, armazenado em um local diferente do servidor de mídia. Uma descrição de apresentação pode conter informações sobre um ou mais fluxos que compõe a apresentação, como endereços de rede e informações sobre o conteúdo da apresentação, além de parâmetros que tornam possível ao cliente escolher a combinação mais apropriada das mídias.

3.1.5 ALTO – Application-Layer Traffic Optimization

Preencher...

<https://datatracker.ietf.org/wg/alto/> <http://tools.ietf.org/html/rfc5693>

3.1.6 UDP, DCCP e SCTP

Tanto o RTP quanto o RTSP são propostas desenvolvidas para suprir as carências encontradas em protocolos de transporte tradicionais da Internet, como o UDP e o TCP. Apesar de novos protocolos de transporte padronizados pela IETF, tais como o DCCP e o SCTP (*Stream Control Transmission Protocol*) [147, 195], o uso dos protocolos RTP/RTSP em conjunto com o protocolo UDP é predominante nas aplicações multimídia encontradas na Internet.

As especificações e detalhes de funcionamento dos protocolos citados nesta seção estão disponíveis abertamente na literatura. Por serem documentos extensos e de conhecimento já bastante difundido, neste trabalho decidiu-se omitida discussões detalhadas sobre cada um deles, dando-se ênfase em protocolos cujas propostas se aproximam ao GMTP.

3.2 Protocolos de Transporte de Dados Multimídia

Apesar da existência de protocolos padronizados para a transmissão de dados na Internet, diversos grupos de pesquisa têm investido no desenvolvimento de novos protocolos ou na extensão dos existentes para o transporte de dados multimídia pela Internet. A seguir, apresentam-se os trabalhos representativos ao estado da arte neste sentido.

3.2.1 PPETP – Peer-to-Peer Epi-Transport Protocol

O *Peer-to-Peer Epi-Transport Protocol* (PPETP) é um protocolo distribuído que utiliza uma abordagem P2P para transmissão de mídias em tempo real, sendo proposto para operar em redes com nós heterogêneos [196].

O PPETP é um protocolo que constrói uma rede de sobreposição com suporte a transmissão em modo multicast. Em tal protocolo, propõe-se uma solução de fácil integração às aplicações multimídia existentes por meio de uma biblioteca de programação similar, porém não integrada, à *Socket* BSD. O PPETP tem como principal aplicação os sistemas de

transmissão de vídeo executados por usuários residenciais, geralmente conectados através de uma tecnologia xDSL, fornecendo uma visão de um protocolo de transporte multicast ao desenvolvedor da aplicação, embora o PPETP é executado na camada de aplicação.

O PPETP utiliza uma abordagem de transmissão do tipo *push* onde os nós iniciam e finalizam as conexões utilizando pacotes de controle e trocam dados em modo unicast. Ao desejar receber um fluxo de dados, um nó B envia um pedido de conexão ao servidor PPETP (Figura 3.1). Em seguida, o servidor responde com uma informação que determina qual nó o cliente B deve solicitar os dados da transmissão, além de solicitar que o nó B obtenha um arquivo de configuração dos parâmetros de conexão em um servidor de configuração chamado de *starting point*. O servidor de configuração pode ser um nó diferente do servidor gerador do fluxo de dados multimídia. O nó B então requisita os dados ao nó determinado pelo servidor. No PPETP utiliza-se o protocolo UDP para transmissão de dados por padrão, embora permita-se o uso de outros protocolos, como o TCP e possivelmente o DCCP.

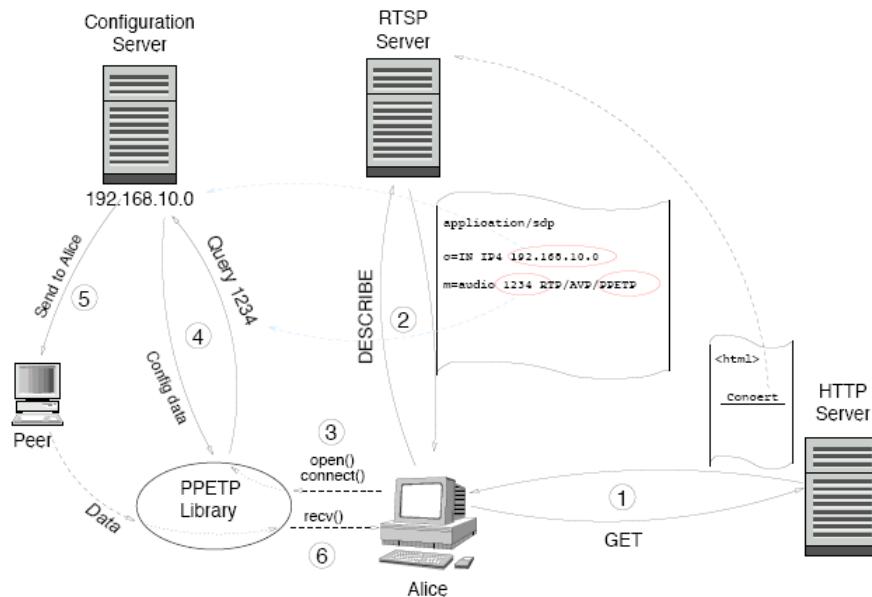


Figura 3.1: Arquitetura e funcionamento do protocolo PPETP.

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

consigam contribuir com o sistema.

No PPETP utiliza-se uma função de redução do tamanho do pacote que é parametrizável e diversas funções podem ser utilizadas através de uma arquitetura de componentes. Atualmente existem duas funções, a de *Vandermonde* e a básica (sem redução). A função de *Vandermonde* reduzido cada pacote por um fator R e então o pacote é repassado para outros nós. Cada nó recebe um conjunto de pacotes reduzidos, reconstrói o conteúdo do pacote, entrega-o para a aplicação e repete o procedimento de redução, repassando-o para outros nós interessados pelo conteúdo.

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

Considerações sobre o trabalho

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

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

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

2. O fluxo de dados de controle é centralizado no servidor. Isto significa que para que um nó A comece a receber um fluxo de dados de um outro nó, primeiro o nó A precisa solicitar ao servidor o acesso ao conteúdo para em seguida efetivamente começar a recebê-lo. No GMTP isto não acontece, pois qualquer nó pode funcionar como servidor, o que ocorre de forma transparente para a aplicação.
3. O PPETP atualmente utiliza o protocolo UDP que, como já discutido, possui diversas desvantagens para a aplicação e para a rede, principalmente em situação de congestionamento na rede. O GMTP é um protocolo de transporte e portanto não necessita de nenhum outro protocolo da sua própria camada. Além disso, o GMTP possui um arcabouço para adicionar novos algoritmos de controle de congestionamento, tanto para as transmissões em modo unicast quanto para as transmissões em modo multicast.
4. O PPETP não suporta transmissão de dados em modo multicast. No PPETP os dados são transmitidos entre os nós em modo unicast, apesar dos autores mencionarem que o protocolo funciona em modo multicast, pelo menos no ponto de vista da aplicação. O termo multicast empregado nesse contexto é apenas para dar a idéia que poucos fluxos são transmitidos a partir do nó transmissor, mas que todos os nós receptores os recebem através da rede sobreposição criada pelo PPETP. No GMTP utiliza-se um mecanismo híbrido de transmissão: sempre que possível usa-se o modo multicast, caso contrário usa-se o modo unicast.
5. No PPETP alguns mecanismos bastante utilizados em sistemas de transmissão de mídias em tempo real, como o de descoberta de nós, deve ser implementado na camada de aplicação. Apesar dessa abordagem do PPETP funcionar por ser flexível para a camada de aplicação, a mesma limita o uso desses mecanismos à própria aplicação, impedindo que outras aplicações façam uso dos mesmos. No GMTP procurou-se adicionar tal funcionalidade dentro do próprio protocolo, permitindo-se o reúso desses mecanismos em diferentes aplicações. Desta forma, é possível que um algoritmo para descoberta de nós seja implementado no GMTP, em forma de componente, e qualquer outra aplicação reutilizar tal mecanismo. Com isto, o GMTP permite a interoperabilidade entre diferentes aplicações a nível de camada de transporte.

6. Para que o PPETP funcione efetivamente nas aplicações serão necessárias diversas alterações em protocolos da camada de aplicação, como no RTSP e no SDP (*Session Description Protocol*). Todas as modificações necessárias estão listadas no documento disponível na referência [196].

3.2.2 PPSP/Swift – P2P Streaming Protocol / The Generic Multiparty Transport Protocol

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

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

O funcionamento básico do PPSP ocorre da seguinte forma 3.2. Um nó transmissor *Peer-P* notifica ao tracker a transmissão realizada por ele (passo 1). O *tracker* então transmite uma mensagem para os *Peer-M* e *Peer-D* interessados em receber o conteúdo multimídia para se juntarem ao grupo (passos 2 e 4). O *Peer-P* transmite o conteúdo para o *Peer-M*, que repassa para o *Peer-D*. Em seguida, outros *peers* se registram como clientes interessados o *tracker*

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

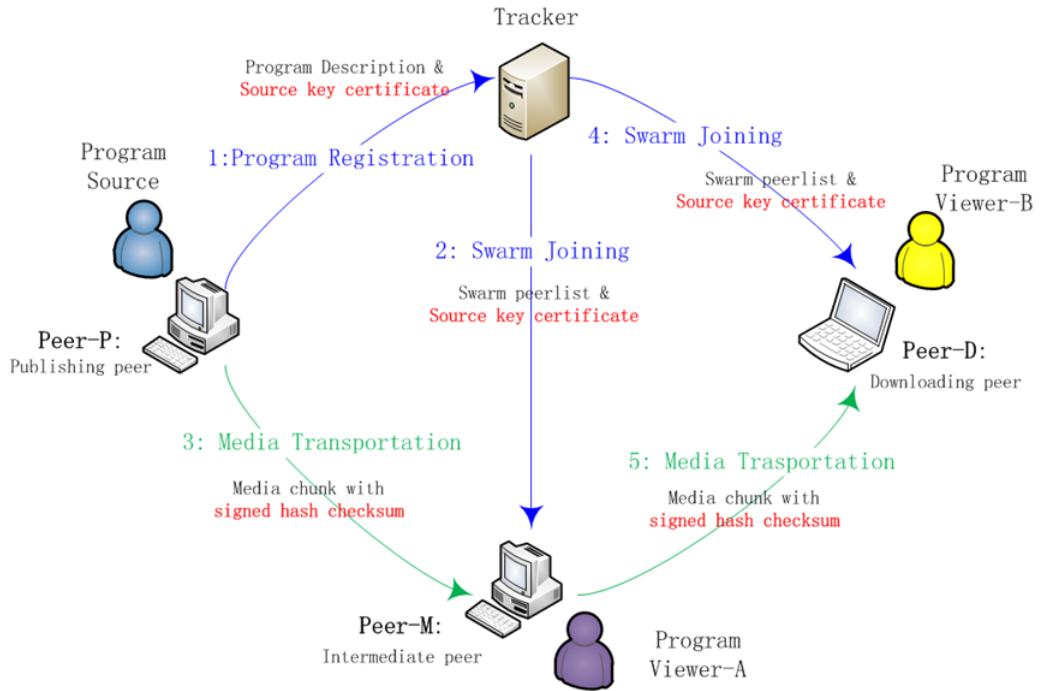


Figura 3.2: Arquitetura e funcionamento do protocolo PPSP/Swift.

porte de controle de congestionamento chamado de LEDBAT [158, 170], o mesmo adotado no BitTorrent.

Considerações sobre o trabalho

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

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

1. Ausência de suporte para extensão para recursos da aplicação, como por exemplo, descoberta, controle de congestionamento e tolerância à falhas. O GMTP é extensível nesse aspecto.
2. Não suporta compartilhamento de conexão com suporte a transmissão em modo multicast.
3. Menciona o uso futuro do algoritmo para controle de congestionamento TFRC, porém não possui suporte a controle de congestionamento em grupo. No GMTP isso é feito

utilizando o algoritmo MCC, apresentado nas Seções ?? e ??.

4. A transmissão de conteúdo com o transporte de *chunks* é interessante em aplicações para compartilhamento de arquivos, onde o tempo de resposta não é requisito 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 *chunck*. Esses procedimentos podem onerar o tempo em que um pacote de dados é entregue para a camada de aplicação, gerando-se um atraso no fluxo contínuo de dados para a camada de aplicação.
5. Uso do protocolo UDP, apesar de fornecer mecanismo para controle de congestionamento. Esta prática quebra a idéia da organização dos protocolos em camadas funcionais, onde uma camada fornece serviços para a camada superior e, obviamente, usufrui de serviços da camada inferior. Mecanismos para controle de congestionamento devem ser implementados na camada de transporte e não na camada de aplicação. Isso limita o uso dos recursos implementados no Swift apenas para aplicações que utilizam sua implementação, a *libswift*, disponível em forma de biblioteca de software. O GMTP é um protocolo de transporte e portanto independente de qualquer outro, além de implementar e suportar à adição de seus próprios algoritmos para controle de congestionamento.

3.3 Protocolos de Aplicação para Transmissão de Mídias

Nesta seção apresentam-se os trabalhos acadêmicos onde são protocolos para transmissão de conteúdos multimídia na camada de aplicação. Os trabalhos foram selecionados seguindo um critério de similaridade com as propostas do GMTP.

3.3.1 CoolStreaming

Preencher...

CoolStreaming [99]

3.3.2 Denacast

Preencher...

3.3.3 PeerCast

5935134

3.3.4 HTTP Live Streaming

Preencher...

3.3.5 PDTP – Peer Distributed Transfer Protocol

O protocolo *Peer Distributed Transfer Protocol* (PDTP) [197] surgiu em 2002 com a promessa de prover um método para transferência de arquivos e mídia em tempo real similar ao BitTorrent. O uso do protocolo foi perdendo força e no final de 2007 foi descontinuado. Sua implementação de referência era conhecida pelo nome de DistribuStream¹.

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

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

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

um *torrent* o conteúdo fica inacessível. Na Figura 3.3 ilustra-se a organização geral dos nós PDTP.

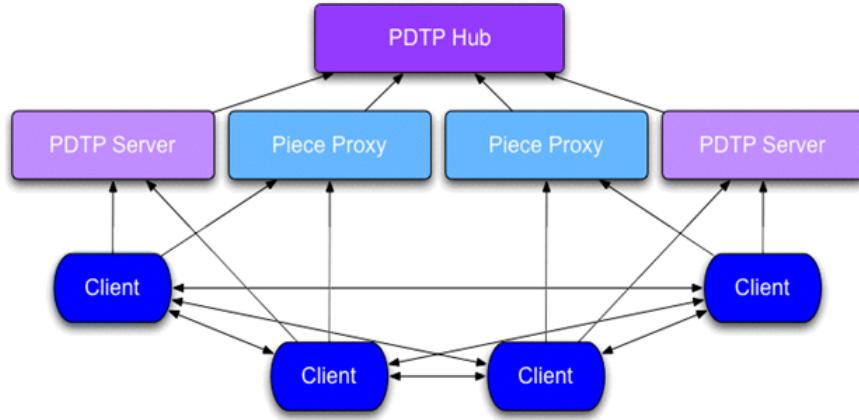


Figura 3.3: Organização dos Nós PDTP.

Considerações sobre o trabalho

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

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

3.3.6 CPM – *Cooperative Peer Assists and Multicast*

No *Cooperative Peer Assists and Multicast* (CPM) [108] propõe-se uma abordagem unificada para prover suporte eficiente de transmissão de vídeos sob demanda para ser utilizada por provedores de serviços. O CPM é um protocolo de aplicação que suporta transmissão em modo multicast, cache de dados nos nós clientes, compartilhamento de dados entre os clientes, onde o servidor utiliza modo de transmissão unicast.

Na Figura 3.4 ilustra-se a visão geral do funcionamento do CPM através de um diagrama de sequência. Primeiramente o cliente conecta o servidor para saber sobre a existência de al-

gum grupo multicast, e então passa a receber o conteúdo em modo multicast. Caso não exista um grupo multicast para o conteúdo de interesse, o cliente solicita, através de um servidor de diretórios a lista de nós que detém o conteúdo de interesse e então inicia a transferência. Caso não exista nenhum nó com o conteúdo requisitado, o cliente requisita o conteúdo diretamente para o servidor.

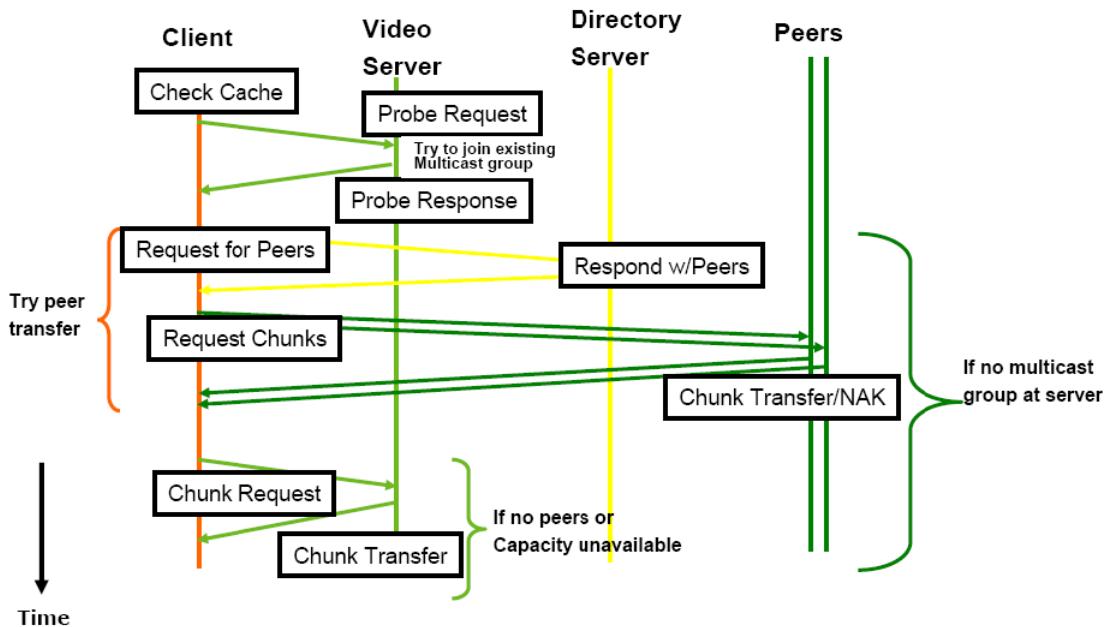


Figura 3.4: Diagrama de sequência do CPM (*Cooperative Peer Assists and Multicast*).

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

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

O protocolo de transferência assume que o vídeo deve estar completamente armazenado no servidor para permitir que os nós façam cache dos pedaços e redistribuí-los *a posteriori*. Quando um cliente envia um pedido de reprodução de vídeo, o servidor mapeia o conteúdo do vídeo requisitado, que é então formatado em sequências de pedaços e transmitidos para o cliente. Este procedimento é executado em paralelo com outros nós da rede. O modo de transmissão multicast é ativado pelo servidor e só ocorre quando múltiplos clientes têm interesse pelo mesmo conteúdo.

Considerações sobre o trabalho

A capacidade para transmitir o conteúdo em modo híbrido é um aspecto positivo para o CPM. A seguir enumeram-se os pontos fracos 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.
2. O mecanismo de transmissão de conteúdo quebra os segmentos em pedaços, o que torna o gerenciamento mais complexo devido ao espalhamento dos pedaços entre os nós participantes da transmissão. Isto pode gerar atrasos na reprodução do conteúdo no cliente devido a necessidade de localizar cada pedaço individualmente, embora o uso dessa abordagem pode aumentar a velocidade de download. No GMTP a existência da idéia de quebrar os segmentos em pedaços menores. Tal abordagem é deixada a cargo da aplicação.
3. O uso de servidor de diretórios para consultar a lista de nós que mantêm o conteúdo multimídia desejado não faz muito sentido para sistemas de transmissão de mídia ao vivo. No GMTP não utiliza-se este tipo de solução.
4. Como o CPM foi desenvolvido com foco em transmissão de vídeo sob demanda, os nós só podem começar a repassar o conteúdo se previamente o este já tenha reproduzido o vídeo no passado. No caso de sistemas de transmissão de vídeo em tempo real esta abordagem é completamente inútil. No GMTP qualquer nó é capaz de realizar o repasse de conteúdo, inclusive em modo multicast.

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

No *Hybrid Delivery System with Adaptive Content Management for IPTV Networks* (HySAC) [198] propõe-se uma nova arquitetura e um sistema adaptativo e híbrido que utiliza um esquema chamado de pre-população para distribuição de vídeos sob demanda em redes

IPTV. Os autores do HySAC criticam o protocolo CPM ao afirmarem que tal abordagem não utiliza os recursos de rede de forma otimizada, uma vez que o conteúdo de mídia não é armazenado de modo pré-planejado de acordo com a demanda dos nós clientes, o que eleva o consumo de recursos de rede e provê uma baixa qualidade na transmissão do conteúdo multimídia ao usuário final.

Diferente do CPM, o HySAC provê a arquitetura ilustrada na Figura 3.5. Para evitar que a grande quantidade de usuários concorrentes sobrecarregue os servidores de mídia, os autores do HySAC propõem um sistema adaptativo de transmissão de mídia que otimiza o processo de entrega de dados baseado na popularidade do conteúdo e nos recursos de rede disponíveis. O conteúdo é categorizado em diferentes classes e o modo de entrega do conteúdo é baseado na popularidade do mesmo. A popularidade de um conteúdo é computada baseando-se no interesse dos usuários e no número de requisições que chegam ao servidor. Os servidores HySAC categorizam os conteúdos e os servidores de indexação e descoberta utilizam Tabelas Dinâmicas de Hash (DHT) para encontrar os servidores que armazenam o conteúdo. Quando a localização de um conteúdo muda, os servidores de indexação são atualizados e quando um novo conteúdo é adicionado, os servidores cuidam da replicação do mesmo de acordo com a sua popularidade.

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

Considerações sobre o trabalho

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

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 me-

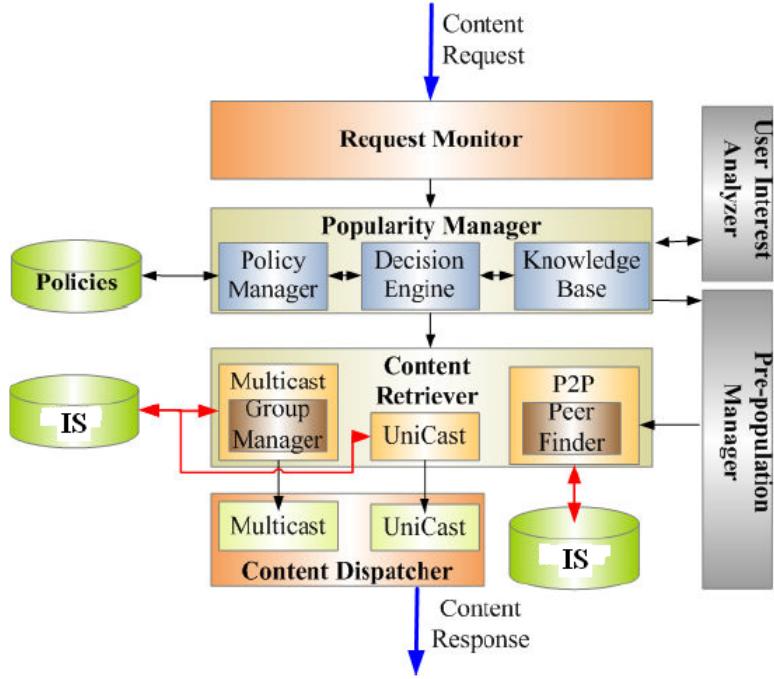


Figura 3.5: Arquitetura do HySAC (*Hybrid Delivery System with Adaptive Content Management for IPTV Networks*).

canismo de classificar o conteúdo requer um tempo de convergência, o que pode consumir recurso de rede desnecessariamente. No GMTP utiliza-se multicast sempre que possível, possibilitando que outros clientes recebam o conteúdo até mesmo sem precisar contactar o servidor, utilizando-se o modo de conexão rápida.

2. Da mesma forma que outras soluções, o HySAC utiliza o protocolo UDP para transmissão de dados da aplicação multimídia. Não foi encontrado nenhuma menção a respeito do uso de algoritmos para controle de congestionamento, ao contrário do GMTP.
3. Trata-se de um sistema de transmissão e não de um protocolo de rede propriamente dito. Isto significa que a proposta do HySAC servirá apenas para clientes que seguem sua especificação.

3.3.8 PULSE – *Peer-to-Peer Unstructured Live Streaming Experiment*

3.3.9 SmoothCache – *HTTP-Live Streaming Goes Peer-To-Peer*

<https://www.sics.se/roberto/files/Roverso-Smoothcache.pdf>

3.3.10 Pastry/SplitStream – *High-bandwidth content distribution*

Preencher...

3.3.11 LayeredCast – *Hybrid Peer-to-Peer Live Layered Video Streaming Protocol*

Preencher...

3.3.12 BitTorrent/LEDBAT – *Low Extra Delay Background Transport*

shalunov2011, 5560080

Preencher...

3.3.13 Outras propostas

Falar aqui que a quantidade de aplicação é tanta e estas podem ser clientes do GMTP.

Além das propostas de protocolos ou sistemas para distribuição de conteúdos multimídia, várias outras propostas foram encontradas durante o levantamento bibliográfico, com menor nível de similaridade com o trabalho aqui proposto. Por exemplo, vale ressaltar as tecnologias de distribuição de conteúdo conhecidas e utilizadas na indústria, tais como: GridMedia [97], Sopcast [98], AnySee [76], PPLive [100] e o ZIGZAG [102]. Estas tecnologias têm sido utilizadas com sucesso para a aplicações multimídia, contudo tais soluções foram concebidas para distribuição de conteúdo multimídia considerando um determinado propósito, com pouca flexibilidade no ponto de vista de extensibilidade e reuso dos mecanismos desenvolvidos.

***** Adicionar mais esses aplicativos a seguir nas referência acima. Pegar só os que tem artigos publicados *****

3.4 Redes Centradas no Conteúdo

3.4.1 CCNx – *Content Centric Network Protocol*

Preencher...

3.4.2 NDN – *Named Data Network*

Preencher...

3.5 Sumário Comparativo

Nas seções anteriores, presentou-se as considerações sobre cada trabalho com comparações entre o respectivo trabalho e o protocolo GMTP. Na Tabela 3.1 apresenta-se um sumário das comparações apresentadas anteriormente. Os critérios de comparação são apresentados a seguir.

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

Tabela 3.1: Tabela comparativa dos protocolos para transmissão de mídia em tempo real. **Legenda:** X = Suporta; $\frac{X}{2}$ = Suporta parcialmente, apenas para algoritmos de controle de congestionamento; S = Requer intervenção da aplicação; Ind. = Indefinido.

Protocolos	CT	CS	TMU	DND	CC	AFM	TD	ENA	CAS
UDP	X		S						X
TCP	X		S					$\frac{X}{2}$	X
DCCP	X							$\frac{X}{2}$	X
SCTP	X			X				$\frac{X}{2}$	X
PPETP				X	Ind.	X	X		S
PPSP/Swift				X	X		X		S
PDT							X		S
CPM			X	X		X	X		
HySAC			X	X		X	X		
Danacast	X	X	X	X	X	X	X	X	X
GMTP	X	X	X	X	X	X	X	X	X

3.6 Sumário do Capítulo

Neste capítulo, apresentou-se uma avaliação crítica acerca de um conjunto de sistemas e protocolos para distribuição de conteúdos multimídia. Discutiu-se sobre trabalhos com propostas semelhantes ao protocolo GMTP, entendendo-se que os principais trabalhos disponíveis no estado da arte não contempla os recursos e a forma de proposta do GMTP. De fato, apenas o Denacast contempla uma proposta próxima ao protocolo GMTP e, por este motivo, no Capítulo 5, discute-se em detalhes seu desempenho em comparação ao GMTP.

3.6.1 Preâmbulo ao GMTP

A definição do protocolo GMTP se baseou nos requisitos e estado da prática dos sistemas de distribuição de conteúdos (Capítulo 2), bem como no levantamento das proposta disponíveis no estado da arte, como as discutidas nas seções anteriores deste capítulo. Como complemento, três questionamentos foram primordiais motivadores para o projeto de tal protocolo,

são eles:

1. Quais as funcionalidades dos sistemas de distribuição de conteúdos ao vivo que poderiam ser implementadas na camada de aplicação por falta de um protocolo de transporte de dados ideal para esse tipo de aplicação?
2. Quais dessas funcionalidades podem ser implementadas na camada de transporte a fim de torná-las padronizadas para todas as aplicações existentes, evitando-se o retrabalho de desenvolvimento?
3. Como se pode, de forma eficiente, distribuir um mesmo fluxo de dados multimídia partindo de um servidor para múltiplos nós clientes conectados a diferentes redes, considerando o fato de que diferentes aplicações clientes possam ser utilizadas e com suporte a controle de congestionamento assistido pela rede?

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

Em termos arquiteturais, o GMTP está posicionado de tal forma que suas funções são disponibilizadas para diferentes processos de aplicação em execução em um sistema operacional, de modo que se torna independente de linguagem de programação, bibliotecas de funções e dos mais variados tipos de dispositivos de usuários finais (desktops, notebooks, tablets, smartphones, smartTVs, etc.), independente da mobilidade dos usuários e podendo ser embarcado no núcleo de qualquer sistema operacional moderno. Com isto, promove-se recursos

funcionais para a camada de aplicação cada vez mais estáveis e eficientes, propagando-se as boas práticas (algoritmos) que passam a ser utilizadas e testadas por um conjunto cada vez maior de desenvolvedores, aprimorando-as com o passar do tempo sem causar um impacto direto à camada de aplicação, tal como tem ocorrido com o protocolo TCP ao longo dos seus 32 anos de existência, desde da sua primeira versão em 1981.

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

1. constatou-se que os sistemas mais robustos de distribuição de conteúdos multimídia são baseados em arquiteturas híbridas P2P/CDN. Isto ocorre porque se obtém escalabilidade do número de usuários e redução de custos com infra-estrutura de rede por meio das redes P2P; e facilidade no gerenciamento e maior estabilidade de disponibilização dos serviços, por meio das CDNs [30–38];
2. observou-se que nos sistemas desse tipo não se viabiliza o acesso a um conteúdo ao vivo por parte de um usuário levando-se em consideração apenas o seu interesse em assistir a um evento ao vivo (*abordagem centrada no conteúdo*). Nas soluções existentes, a forma de acesso ao conteúdo de um evento ao vivo é dependente do local físico (servidor) onde tal conteúdo está sendo transmitido (*abordagem centrada no hospedeiro*), sendo crucial saber de qual nó fonte o evento está sendo recebido;
3. dado que na maioria dos sistemas de distribuição de conteúdos ao vivo se transmite fluxos de dados UDP, tais sistemas não executam mecanismos para controle de congestionamento. Desta forma, se isto for necessário em seus sistemas, os desenvolvedores são forçados a executar algoritmos de controle de congestionamento na camada de aplicação, sem qualquer padronização na forma como os fluxos de dados são controlados, com diferentes equipes de desenvolvimento implementando, das mais variadas formas, a mesma funcionalidade presente nesse tipo de aplicação, sem qualquer compartilhamento desse esforço [?, 199, 199, 199];
4. não há uma forma efetiva de centralizar e/ou disponibilizar as boas soluções e práticas (algoritmos) para as diferentes funcionalidades empregadas nesse tipo de sistema. De

fato, existem diversos *middlewares* de desenvolvimento de sistemas de distribuição de conteúdo que tentam suprir as limitações existentes dos protocolos da camada de transporte, tratando-se de soluções intermediárias e fragmentadas para o transporte de dados multimídia nas redes de computadores, salvas suas devidas contribuições científicas e reais [199, 199, 199, 199].

Capítulo 4

Global Media Transmission Protocol (GMTP)

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

As trocas de dados entre nós GMTP ocorrem por meio do envio e recebimento de partes de uma mídia (*chunks*), que são transmitidas por diferentes nós da rede, constituindo um fluxo de datagramas IP. Estes fluxos são transmitidos em modo *unicast* e compartilhados (*multi-unicast*) pelos roteadores, quando são entregues aos nós clientes em modo *multicast*, realizando-se controle de congestionamento sem garantia de entrega. A escolha do modo de transmissão utilizado para disseminar um fluxo de dados ocorre sem a influência da apli-

ção, que precisa simplesmente “sintoniza” sua conexão em um determinado canal *multicast* definido pelo roteador. Tal abstração para a camada de aplicação ocorre de modo que os processos em execução utilizam o GMTP através de uma API compatível com as especificações de socket BSD e POSIX, o que permite adaptações mais simples das atuais aplicações de rede de transmissão de mídias ao vivo.

Por conseguinte, o GMTP permite o estabelecimento de conexões entre diversas aplicações, executadas de forma distribuída em cada sistema final, tornando-as compatíveis entre si, uma vez que o protocolo desacopla a forma como os dados são transportados da forma como estes são exibidos ao usuário final, emulando os sistemas tradicionais de TV e rádio. Sendo assim, promove-se a integração do GMTP em aplicações já existentes, consideradas futuras adoções, ao tempo que permite a utilização dos novos recursos introduzidos no protocolo, evitando-se a complexidade de construção dos sistemas de transmissão de fluxos de dados de eventos ao vivo, especialmente aqueles baseados em arquitetura P2P/CDN.

Nas próximas seções deste capítulo, detalham-se os aspectos teóricos e computacionais empregados do GMTP de acordo com os blocos funcionais ilustrados na Figura 4.1, a fim de construir uma rede de sobreposição formada por roteadores, pela execução de quatro grandes passos:

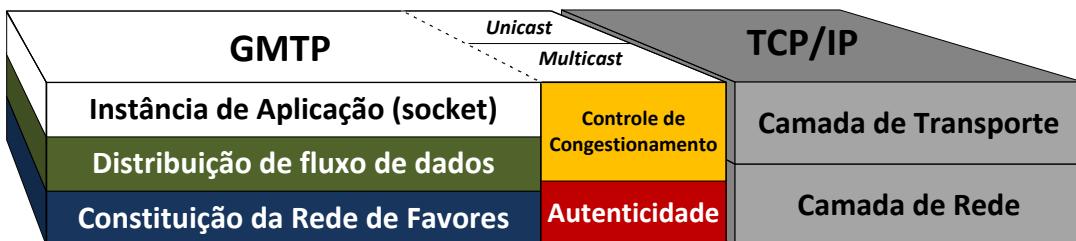


Figura 4.1: Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP.

1. *Constituição da rede de favores*: descobrir, definir, efetivar e desfazer parcerias entre os roteadores de acordo com o evento a ser transmitido.
2. *Distribuição de fluxos de dados através de uma camada de socket*: conectar os nós clientes interessados em receber um fluxo de dados de um evento, bem como transmitir tal fluxo de dados através da rede de sobreposição constituída no Passo 1.

3. *Controle de congestionamento:* controlar a taxa de transmissão dos fluxos de dados distribuídos no Passo 2.
4. *Autenticidade do conteúdo:* verificar a autenticidade do fluxo de dados antes de entregá-los aos nossos clientes.

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

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

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.

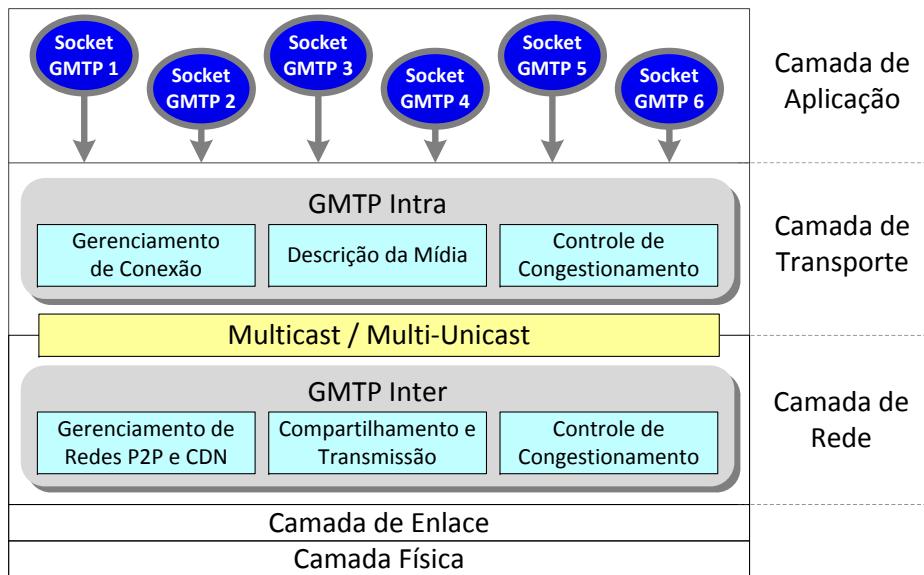


Figura 4.2: Arquitetura do Protocolo GMTP.

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 multi-ponto, multiplexação/demultiplexação de segmentos IP entre as camadas de transporte/rede/aplicação e controle de congestionamento. Este módulo compreende a instância do GMTP em execução no sistema operação do nó cliente, acessível através de uma API de socket GMTP. Um socket GMTP é a representação de uma instância do protocolo GMTP em execução, sendo responsável por gerenciar todas as atividades de comunicação da aplicação correspondente ao meio externo (outros processos GMTP). No contexto de uma conexão, o GMTP Intra mantém diversas variáveis de estado relacionadas à execução dos algoritmos para gerenciamento de conexão (estabelecimento e desconexão), controle de congestionamento multicast, multiplexação e demultiplexação dos datagramas, eleição de nós parceiros e determinação do formato e preenchimento dos parâmetros que definem uma mídia, permitindo-se que a aplicação defina os valores de tais parâmetros ou obtenham acesso aos seus valores.
- **GMTP Inter:** constituir uma rede de favores P2P composta por roteadores, os quais

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

- configurações sobre registro de participação em uma ou mais redes CDN. Este assunto foi detalhado na Seção 4.3.2;
- largura de banda máxima (*download* e *upload*) que o nó repassador está autorizado a compartilhar;
- o período (faixa de dias e horários) que o roteador funcionará como nó repassador;
- quantidade máxima de parcerias que podem ser realizadas;
- quantidade máxima de fluxos de dados que podem ser compartilhados;
- parâmetros avançados relacionados aos algoritmos de controle de congestionamento. Este assunto foi discutido na Seção 4.5; e
- configurações acerca dos certificados digitais, tais como *download* automático e realização de cache. Este assunto foi abordado na Seção 4.6.

Sendo assim, para viabilizar a disseminação de conteúdos multimídia, cada nó roteador no caminho entre o servidor que transmite a mídia e o cliente interessado em obtê-la, pode repassar os pacotes de dados para seus clientes locais também podem replicá-los para outros roteadores interessados em receber o fluxo de dados correspondente. No GMTP, permite-se que um roteador atende à demanda dos seus clientes locais, ao passo que ajuda os outros

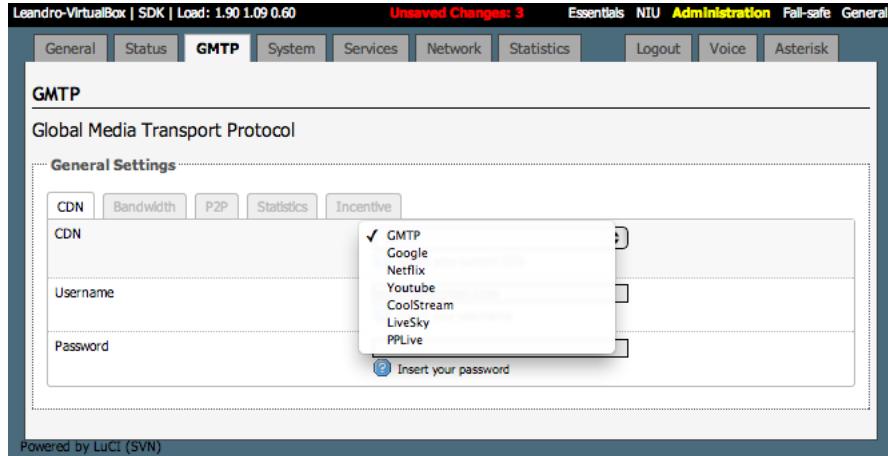


Figura 4.3: Tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP. Nessa tela, permitir que o administrador do roteador configure parâmetros do módulo GMTP Inter.

roteadores a fazerem o mesmo, evitando múltiplas conexões para obter um mesmo fluxo de dados no nó servidor.

Na Figura 4.4, observa-se o cenário geral de atuação do protocolo GMTP, onde ilustram-se os nós *Clientes GMTP* interessados em obter o conteúdo de um determinado evento ao vivo. Neste caso, observa-se também um *Servidor GMTP*, que está conectado a uma rede CDN e atua como fonte geradora de dados; ao passo que os nós *Clientes GMTP* conectam-se a um nó *Repassador GMTP* que, na prática, é um roteador de rede. Os *Repassadores GMTP* efetivamente constituem a rede de sobreposição P2P, conectando-se a um ou mais *Servidores GMTP*. Com base na Figura 4.4, definiu-se a Figura 4.5, onde se observa os seguintes tipos de nós GMTP:

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

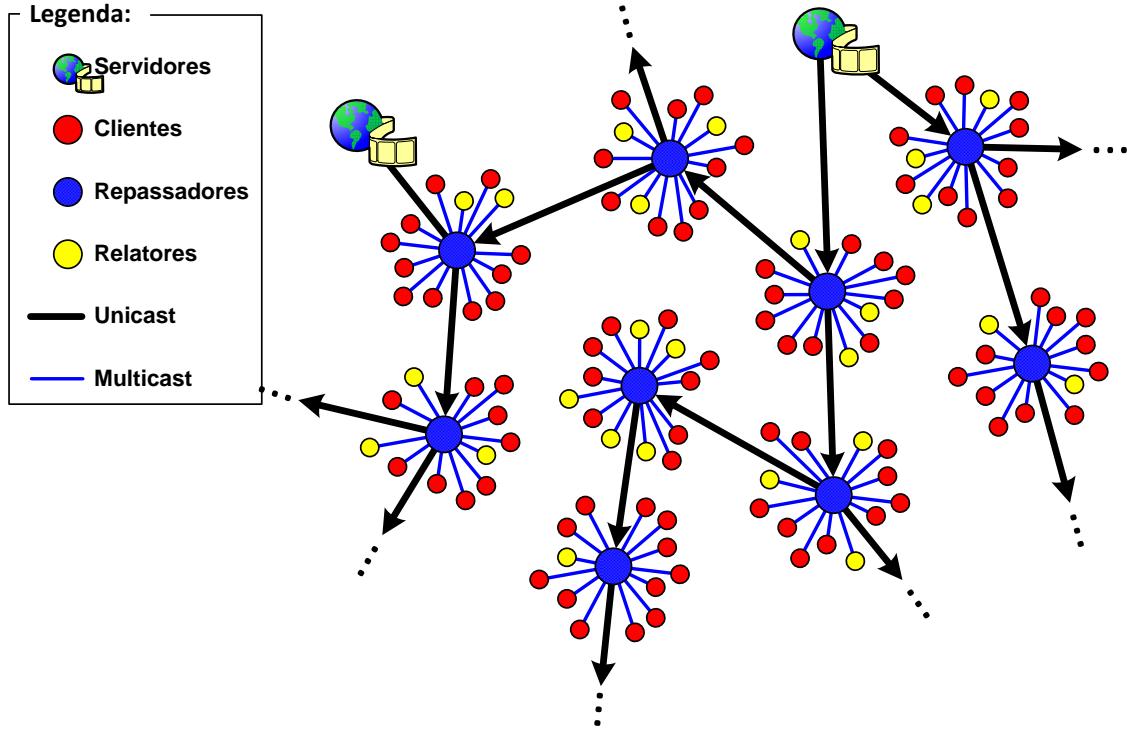


Figura 4.4: Rede de sobreposição construída dinamicamente pelo GMTP com a presença de nós repassadores e relatores.

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

- **Repassador GMTP**: participa efetivamente da rede de favores e tem a responsabilidade de repassar os fluxos de dados originados em um ou mais *Servidores GMTP* para outros nós *Repassador GMTP* até que o conteúdo de tal fluxo (pacotes de dados) alcancem os nós *Clientes GMTP*. Este nó sempre é um roteador de rede.
- **Relator GMTP**: é um *Cliente GMTP* com habilidades de enviar relatórios periódicos ao nó *Repassador GMTP* sobre o estado da transmissão.

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

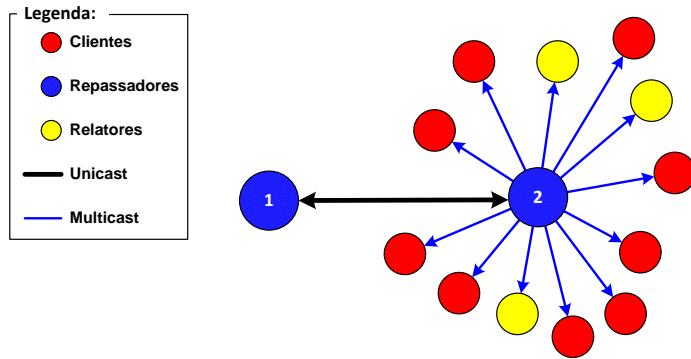


Figura 4.5: Tipos de Nós e modos de conexões do GMTP.

ítálico, bem como os termos *socket*, *unicast*, *multi-unicast* e *multicast*. Ademais, quando o termo *transmissão* ou *transmissão de um evento* for mencionado, denotar-se-á a transmissão de um fluxo de datagramas IP correspondente a um evento ao vivo, utilizando-se o protocolo GMTP.

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

O posicionamento dos nós repassadores e suas habilidades permitem a redução do número de fluxos de dados na rede correspondente a um mesmo evento, ao tempo que maximiza a quantidade de nós clientes interessados em receber o mesmo fluxo (escalabilidade). Por este mesmo motivo, o protocolo GMTP é flexível para permitir que um nó repassador atue somente encaminhando conteúdos multimídias entre duas ou mais redes distintas, mesmo que este não tenha demandas explícitas dos seus nós clientes por tal conteúdo. Desta forma,

maximiza-se o uso dos canais de transmissão ociosos, em particular das redes residenciais, principalmente quando seus usuários estão ausentes e portanto sem fazer uso dos recursos de redes disponíveis. Isto pode ocorrer sem a necessidade de manter um determinado computador ligado e conectado à rede, bastando apenas manter o roteador de rede ligado, diferentemente de todas as outras soluções existentes baseadas em arquitetura P2P ou P2P/CDN, que requer pelo menos um nó cliente ligado.

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

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 interface de rede de saída, 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 [200].

4.1.1 Resumo das principais funções

- Registro de participação de um nó repassador em um nó servidor. Isto permite que um nó repassador sinalize interesse em participar de uma rede CDN. Como resultado, pode-se pré-selecionar nós parceiros filtrados por métricas que influenciam na

qualidade de serviço oferecido aos sistemas finais. Este assunto será retomado na Seção 4.3.2.

- Acesso a uma transmissão de um evento ao vivo através de um processo de conexão em três-vias (*3WHS*), com a requisição de conexão transmitida ao servidor e podendo ser interceptada por um nó repassador em seu trajeto ao servidor, com suporte automático de detecção e uso dos modos de transmissão suportados pelo nó repassador (unicast e/ou multicast). Este assunto será retomado na Seção 4.4.2.
- Descoberta de nós parceiros entre redes distintas e negociação de parcerias, com suporte a formação de parcerias baseadas em métricas de rede, tal como a largura de banda fim-a-fim. Além disso, o GMTP é capaz de distribuir um fluxo de dados em múltiplas taxas de transmissão de acordo com a largura de banda dos nós repassadores. Para isto, o GMTP segmenta os canais de transmissão (rota entre um nó servidor e os nós repassadores) quando existem múltiplos nós repassadores em uma determinada rota e estes estão interessados em receber o mesmo fluxo de dados. Este assunto será retomado nas Seções 4.3 e 4.5.
- Envio e recebimento de fluxos de dados compartilhados entre nós da mesma rede através do uso do modo de transmissão multicast, sendo o modo de transmissão unicast restrito apenas para uso em transportar os fluxos de dados entre redes distintas, evitando-se a relação de uma conexão por cliente ao nó servidor. Este assunto será retomado na Seção 4.4.
- Uso de algoritmo de controle de congestionamento assistidos pela rede, em transmissões em modo unicast; e uso do *TCP Friendly Rate Control* (TFRC) adaptado às transmissões de fluxos de dados em modo multicast. Este assunto será retomado na Seção 4.5.
- Eleição de nós relatores com suporte a tolerância a desconexões de nós, com notificação e reeleição de novos nós. Este assunto será retomado na Seção 4.7.3.
- Verificação de autenticidade dos pacotes de dados que transportam parte de uma mídia, por meio do uso de assinaturas digitais disponibilizadas pelos nós servidores,

impedindo assim ataques de poluição de conteúdo. Este assunto será retomado na Seção 4.6.

4.2 Definições, Relações e Restrições

Nesta seção, descrevem-se as definições, relações e restrições do protocolo GMTP. Para isto, faz-se uso de fundamentos de álgebra booleana, lógica proposicional, teoria de conjuntos e teoria dos grafos [201–204].

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

- i. Seja P , o conjunto *parcialmente ordenado (poset)* dos pacotes de dados p_x transmitidos por um nó r_d , também chamado de fluxo de pacotes de dados ou apenas fluxo de dados, definido por $(P, \prec) = \{p_1, p_2, p_3, \dots, p_x\}$, tal que $x \in \mathbb{N}$. Trata-se de um *poset* porque o GMTP não garante entrega de p_x ;
 - ii. Seja C_i , uma função que denota os nós c_f relacionados a um nó r_d , de modo que nenhum nó $c_f \in C$ pode estar relacionado com dois ou mais nós r_d , definida por $C_i : r_d \rightarrow 2^C$, $\forall r_d, r_q \in R$, $C_i(r_d) \cap C_i(r_q) = \{\emptyset\}$, tal que $q \neq d$ e $q \in \mathbb{N}$;
 - iii. Seja L , o conjunto finito dos nós relatores, definido por $L = \{l_1, l_2, \dots, l_w\}$. Como todo nó c_f pode atuar como l_w , tem-se que $\exists L_\theta \in 2^{C_i(r_d)}$, tal que $l_w \in L_\theta$. Pelo item 6(b)ii, tem-se portanto que $L_\theta \subset L$ e $L_\theta \cup C_i(r_d) = C_i(r_d)$.
- (c) $W = \bigcup_{v=1}^j W_v$, onde $j \in \mathbb{N}$ e corresponde à quantidade de todos os possíveis caminhos W_v , tal que um caminho é definido por um conjunto *toset* (W_v, \prec) , que denota um dos possíveis caminhos por onde um fluxo de dados P pode ser transmitido, obrigatoriamente a partir de um nó servidor s_a até um nó r_1 , tal que:
- i. $(W_v, \prec) = \{w_m \mid s_a, r_1, r_2, r_3, \dots, r_d\}$, $\forall w_m, w_{m+1} \in W_v : w_m \prec w_{m+1}$ e $|W_v| \geq 2$;
 - ii. Um caminho W_v é dito *caminho semi-completo*, representado por W_v° , se e somente se $W_v \leftrightarrow \exists B_\theta$ (bijetora), tal que $B_\theta \in 2^B$ e $B_\theta \neq \{\emptyset\}$. Isto é, todos os roteadores $b_e \in B$ são sobrepostos por um nó $r_d \in W_v^\circ$;
 - iii. Um caminho W_v é dito *caminho completo*, representado por W_v^\bullet , se for W_v° e se $W_v \subset T$, tal que $T \subset Z$ é o conjunto dos nós r_d que transmitem os pacotes de dados $p_x \in P$ a seus nós $c_f \in C_i(r_d)$, definido por $T = \{t_u \mid \varphi(t_u, P) = 1\}$, sendo $u \in \mathbb{N}$ e φ uma função booleana que determina se um nó $t_u \in T$ transmite os pacotes $p_x \in P$ para $c_f \in C_i(t_u)$, ou seja:
 - A. $\varphi : (t_u, P) \rightarrow \{0, 1\}$, $\forall (t_u, P) \in \{T \times \{P\}\}$, onde 0 e 1 denotam, respectivamente, *falso* e *verdadeiro*.
- (d) Seja \sim , uma função reversa de um conjunto *toset*, tal que $\sim : (W_v, \prec) \rightarrow (W_v, \succ)$. Isto é, para um conjunto $(W_v, \prec) = \{w_m \mid s_a, r_1, r_2, \dots, r_d\}$, então $\sim(W_v)$ produzirá $(W_v, \succ) = \{w_m \mid r_d, r_{d-1}, r_{d-2}, \dots, r_1, s_a\}$;

- (e) Seja δ , 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 ζ 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 γ 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 γ calcula o custo apenas entre dois nós t_u, t_{u+1} , com base pela largura de banda disponível nos nós t_u . Porém, pode-se definir outras métricas, por exemplo, o número total de saltos no caminho W_v ou o RTT entre o nó s_a e um nó r_d ;
- (g) *Conjectura 1:* $\forall r_d \in R$ e $\forall c_f \in C$, r_d é mais estável que qualquer c_f com relação a sua disponibilidade e participação em uma rede de favores η . Em uma rede comutada por pacotes IP, um nó $b_e \in B$, ou seja, um nó r_d fica menos indisponível se comparado aos seus nós $C_i(r_d)$. Por exemplo, nas transmissões de dados na Internet, a participação de um roteador no processo de transmissão de um fluxo de dados P é fundamental, mesmo que seja apenas para rotear os respectivos pacotes. Apesar de óbvia, tal observação é importante porque para qualquer nó c_f receber os pacotes de dados $p_x \in P$, primeiramente os pacotes de dados p_x passam, obrigatoriamente, pelo roteador de c_f , ou seja, o seu roteador padrão. Sendo assim, quando um nó r_d se desconecta, todos seus nós $C_i(r_d)$ tornam-se capazes de receber P , mas a recíproca não é verdadeira – se um nó c_f se tornar indisponível, não necessariamente r_d também se torna indisponível. Com a aceitação dessa conjectura para a rede η , 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;

mente 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 pre-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, η representa formalmente a rede de sobreposição constituída pelo GTMP, 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 η

A constituição da rede de favores η 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_θ 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_θ , tal que $W_\theta = \min(\zeta(\forall W_v))$ e, sempre que possível, que W_θ seja um caminho completo W_θ^\bullet . Sempre buscar um caminho completo é importante porque como todos os nós de tal caminho são roteadores sobrepostos por r_d e utilizados para transmitir P , pode-se distribuir P para mais nós c_f sem que sejam necessárias múltiplas conexões em s_a , evitando a tragédia dos bens comuns, discutida no Capítulo 1. Além disso, quanto mais

nós r_d estiverem disponíveis na rede, menor será o impacto causado pelas desconexões nos sistemas finais que recebem o fluxo de dados P .

4.3.1 Tipos de Pacotes

Antes de prosseguir com os passos do GMTP para constituir uma rede de favores η , é importante entender os diferentes tipos de pacotes utilizados no GMTP. Isto porque, toda comunicação entre dois ou mais nós GMTP ocorre através da troca de pacotes IP, os quais carregam sinalizações de controle e/ou dados da aplicação. No mundo real (Internet), será necessário registrar na *Internet Assigned Numbers Authority* – IANA¹ o uso de um código para o campo *Protocolo* do cabeçalho de um datagrama IP. Com a padronização do protocolo GMTP e a publicação da sua RFC, provavelmente será utilizado o código 100, como já está definido no documento *Protocol Numbers*² da IANA.

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

0. *GMTP-Request*: o nó cliente envia requisição para obter um fluxo de dados multimídia com base no nome do fluxo de interesse;
1. *GMTP-RequestNotify*: o nó repassador notifica um cliente que um fluxo de dados está prestes a ser transmitido ou já está sendo transmitido em um determinado canal de repasse multicast. O campo de dados desse tipo de pacote contém a descrição da mídia a ser reproduzida;
2. *GMTP-Response*: o nó repassador confirma o estabelecimento de uma parceria com outro nó repassador, dado um determinado fluxo de dados;

¹IANA: <http://www.iana.org/>

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

3. *GMTP-Register*: o nó repassador registra participação no servidor para funcionar como distribuidor de um fluxo de dados;
4. *GMTP-Register-Reply*: o nó servidor responde sobre o pedido de registro de participação enviado por um nó repassador;
5. *GMTP-RelayQuery*: o nó repassador pode solicitar ao servidor uma lista de possíveis nós repassadores parceiros;
6. *GMTP-Data*: qualquer nó utiliza esse tipo de pacote para transmitir dados da aplicação;
7. *GMTP-Ack*: qualquer nó utiliza esse tipo de pacote para confirmar a recepção de um determinado pacote, seja pacotes previamente contendo dados ou não;
8. *GMTP-DataAck*: combinação dos pacotes GMTP-Data e GMTP-Ack (*PiggyBack*);
9. *GMTP-MediaDesc*: o nó servidor transmite esse pacote para descrever informações sobre a mídia sendo transmitida em uma determinado fluxo de dados (conexão);
10. *GMTP-DataPull-Request*: o nó repassador envia um pedido para obter o mapa de buffer atual de um outro repassador parceiros;
11. *GMTP-DataPull-Response*: resposta ao pedido para obtenção de um mapa de buffer;
12. *GMTP-Elect-Request*: o nó repassador envia para um cliente o pedido para tal cliente atuar como nó relator;
13. *GMTP-Elect-Response*: o nó cliente envia para o repassador uma confirmação de que pode atuar como relator;
14. *GMTP-Close*: os nós servidor, repassador ou cliente solicitam o término de uma conexão;
15. *GMTP-Reset*: determina, incondicionalmente, a finalização de uma conexão;
16. *Reservado*: a partir deste identificador ao 31, tratam-se de valores reservados para uso futuro e ignorado pelos nós que o processa.

4.3.2 Registro de participação de r_d em η

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

Para realizar um registro de participação, um nó r_d envia uma mensagem para um nó s_a utilizando o pacote *GMTP-Register*, o que permite a descoberta de um caminho W_v . Isto porque todos os nós repassadores existentes no caminho entre r_d e s_a devem adicionar seu identificador no pacote *GMTP-Register* antes de roteá-lo para o próximo salto da rota em direção ao nó s_a . Para definir um identificador de um nó r_d , gera-se um código *hash* da soma dos endereços MAC (*Media Access Control*) de todas as interfaces de rede do roteador. Quando o pacote *GMTP-Register* alcançar o destino s_a , o nó s_a conhecerá o caminho W_v composto por todos os nós r_d até s_a e o armazenará como sendo um dos possíveis caminhos para distribuir um fluxo de dados P . Como resposta ao nó r_d , o nó s_a deve enviar um pacote do tipo *GMTP-Register-Reply*, que confirma o registro de participação. O caminho W_v pode ser utilizado futuramente no processo de formação de parcerias, a ser discutido na Seção 4.3.4.

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 em questão, vindo diretamente

do nó s_a ou repassado por outros nós r_d . Como consequência, reduz-se o tempo de início de reprodução do referido fluxo de dados P para aqueles nós c_f que solicitarem o mesmo fluxo de dados P após o primeiro nó repassador pedir, bastando apenas que os próximos nós c_f “sintonizem” sua interface de comunicação (socket de rede) no canal apropriado e informado por r_d , pois a transmissão ocorre em modo multicast. Por analogia, o registro de participação faz com que o roteador de uma rede funcione como se fosse uma antena de recepção de uma transmissora de TV, podendo-se receber um ou mais sinais de canais de TV diferentes. Em seguida, estes sinais são repassados para os clientes conectados diretamente à antena, ou melhor, ao roteador.

No Algoritmo 1, executado por um r_d , resume-se os passos para o envio do pedido de registro de participação em um nó s_a . Note que o nó r_d não é obrigado a informar qual fluxo de dados P está interessado em obter. Se o fluxo de dados P for especificado, o nó s_a executará um procedimento para determinar se aceita ou não o pedido de registro de participação para transmitir P a r_d . Em caso de aceite, a transmissão do fluxo de dados P de s_a para r_d ocorrerá em modo unicast, caso contrário o nó s_a instruirá um outro nó r_q a transmitir o referido fluxo de dados ao nó r_d solicitante. Já no Algoritmo 2, resume-se os passos após um nó r_d receber uma resposta do tipo *GMTP-Register-Reply* transmitida pelo nó s_a , referente ao pedido de registro de participação transmitido anteriormente por r_d .

Note que, no GMTP, 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 η 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* [205].

Além disso, um nó r_d pode sinalizar explicitamente sua desconexão a s_a quando não desejar mais participar da rede de favores η ou receber 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 no processo de formação de parcerias e enviar para este um pacote do tipo *GMTP-Reset*.

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    $\text{channel} \leftarrow \text{isFlowBeingReceived}(P)$ ; /* See Section 4.3.3 */
      /* 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  $\text{GMTP-Register-Reply}$  is received, executes
        $\text{onReceiveGMTPRegisterReply}$  (Algorithm 2). */
12    if not  $\text{isWaitingRegisterReply}(P)$  then
13       $\text{isWaitingRegisterReply}(P, \text{true})$ ;
14       $\text{sendPktRdt}(\text{GMTPRegister}(s_a, P))$ ;
15    end
16    /* Ask  $C_i(r_d)$  to wait registration reply for  $P$ . */
17     $\text{respondToClients}(\text{GMTPRequestReply}(P))$ ;
18  return 0;
19 end
20 if not  $\text{isWaitingRegisterReply}(s_a)$  then
21   return  $\text{sendPktRdt}(\text{GMTPRegister}(s_a))$ ;
22 end
23 return 0;

```

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

```

/* 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 isWaitingRegisterReply( $P, \text{false}$ );
2 if  $p_x = OK$  then                                /*  $s_a$  confirmed registration */
3    $s_a \leftarrow \text{getPacketFieldValue}(p_x, \text{'server'})$ ;    /* Gets  $s_a$  in  $p_x$  */
4    $P \leftarrow \text{getPacketFieldValue}(p_x, \text{'flow'})$ ;        /* Gets  $P$  in  $p_x$  */
5   if  $P \neq \text{NULL}$  then          /* Reply to  $C_i(r_d)$ , waiting for  $P$  */
6     if  $s_a$  enabled security layer then          /* Section 4.6.4 */
7       | getAndStoreServerPublicKey( $s_a$ );
8     end
9      $channel \leftarrow \text{createMulticastChannel}(s_a, P)$ ;
10    updateFlowReceptionTable( $channel$ ); /* Section 4.3.3 */
11    /* Let  $c_f \in C_i(r_d)$  know the multicast channel to
12       receive  $P$  (Section 4.4.2). */           */
13    respondToClients(GMTPRequestReply( $channel$ ));
14    /* Start to relay  $P$  to clients (Section 4.4.6). */    */
15    startRelay( $channel$ );
16  end
17  /* It was just a reply of a registration of
18     participation. Update flow reception table. */
19  updateFlowReceptionTable( $s_a$ );             /* Section 4.3.3 */
20
21 else
22   /*  $s_a$  refused to accept the registration of
23     participation. This  $r_d$  must notify the clients
24     waiting for receiving  $P$ . */
25    $errorCode \leftarrow \text{getPacketFieldValue}(p_x, \text{'error'})$ ;
26   respondToClients(GMTPRequestReply( $errorCode, P$ ));
27
28 end

```

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

Manutenção do registro de participação:

Após o registro de participação, o nó r_d deve enviar periodicamente sinalizações de controle (*polling*) sobre sua participação na rede de favores η . Este procedimento 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 [206], discute-se sobre outros aspectos de expiração no tempo que podem ser adaptadas para o GMTP.

4.3.3 Tabela de Recepção de Fluxos de Dados

Antes de seguir com a explicação sobre o processo de estabelecimento de conexão do GMTP, é importante entender que cada nó r_d mantém uma tabela chamada *Tabela de Recepção de Fluxos de Dados*, como ilustra-se na Figura 4.6. O nó r_d utiliza tal tabela para registrar todos

os fluxos de dados que estão sendo repassados para seus nós $c_f \in C_i(r_d)$, mantendo-se as seguintes informações:

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

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

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

Conceitualmente, quando um nó r_d adiciona um registro na tabela de recepção de fluxos de dados, define-se $\varphi(r_d, P) = 1$, ou seja, $r_d \in T$. Um nó r_d consulta a tabela de recepção de fluxos de dados quando recebe um pedido de conexão (*GMTP-Register*) para obter um fluxo de dados P , tal como apresentou-se na Linha 6 do Algoritmo 1, Seção 4.3.2. 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 10 do Algoritmo 2, Seção 4.3.2. Mais adiante, na Seção 4.4.2, discute-se em mais detalhes as ações de consulta e atualização da tabela de recepção de fluxos de dados.

4.3.4 Formação de parcerias

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

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

Para entender detalhes desse processo, considere a Figura 4.7. No Passo 1, ilustra-se um cenário de rede $\eta = G(Z, W)$, onde $Z = \{s_1, r_{1..19}\}$, $W = \{\emptyset\}$ e $\mathcal{T} = \{\{\emptyset\}, \{\emptyset\}, \{\emptyset\}\}$, ou seja, sem qualquer fluxo de dados P sendo transmitido, tampouco nenhuma parceria efetivada e suprimindo-se os nós $c_f \in C_i(r_{1..19})$. Já no Passo 2, ilustra-se a mesma rede η , 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 4.3.2) e, a partir de sua confirmação, processada pelo nó s_1 e enviada ao nó r_9 , este começa a receber os pacotes $p_x \in P$. Com este procedimento, o nó s_1 passa a conhecer o caminho W_1 , que pode ser

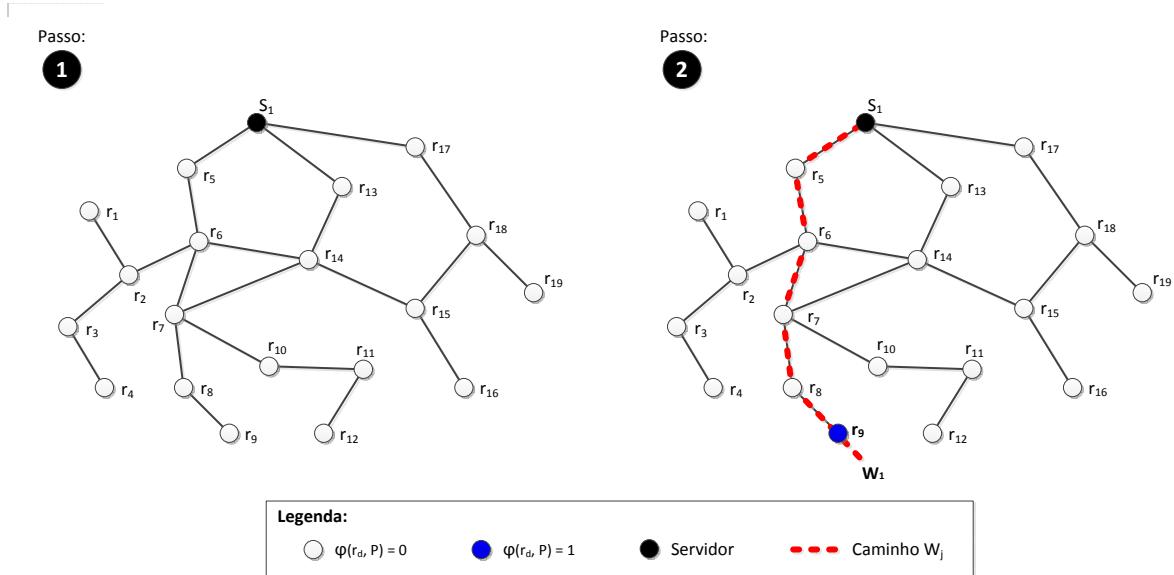
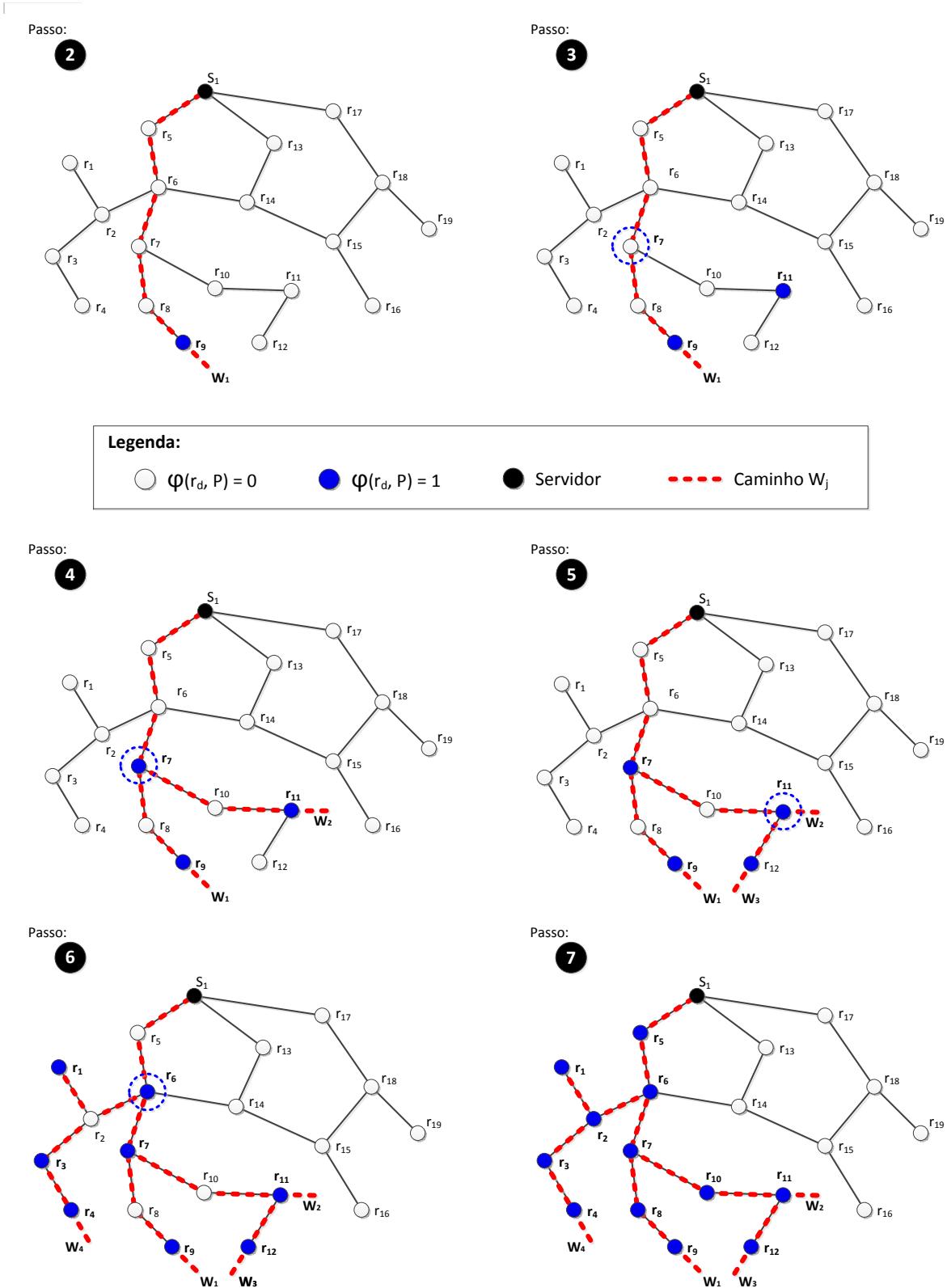


Figura 4.7: Cenário e passos para seleção de nós (exemplo 1).

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

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

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

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

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

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

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

Note que, o nó r_d que interceptar um pedido de conexão para um fluxo de dados P , deve transmitir para o nó s_a uma notificação sobre a(s) parceria(s) formada(s) por intersecção. No caso do exemplo anterior, os nós r_6 , r_7 e r_{11} devem realizar tal notificação enviando um pacote do tipo *GMTP-Register*, como explicado na Seção 4.3.2. Para isso, deve-se ativar o bit *intercepted* do pacote *GMTP-Register*. Esta ação é importante devido aos aspectos gerenciais de uma transmissão, onde uma aplicação poderá contabilizar os nós r_d que estão recebendo P , mesmo que indiretamente, por meio da interceptação de registros de participação. Na prática, não se faz necessário que o nó r_d envie tal notificação no instante da interceptação de um pedido de registro de participação. Em vez disso, pode-se acumular diversos registros de participação durante um determinado intervalo de tempo e, em seguida, transmiti-los para o nó s_a . Como se trata de um aspecto a nível de implementação, tal decisão está fora do escopo dessa discussão. No caso da implementação do GMTP realizada em simulador e utilizada neste trabalho, definiu-se que para todo registro de participação interceptado, gera-se e transmite-se uma notificação ao nó s_a .

No Algoritmo 3, resume-se os passos descritos anteriormente na perspectiva do nó s_a , a fim de determinar a formação de parcerias por intersecção. Executa-se tal algoritmo quando o nó s_a recebe um pedido de registro de participação enviado por um nó r_d para obter um fluxo de dados P . Através dessa estratégia de formação de parceria, permite-se repasses de pacotes de dados levando-se em consideração o nome do fluxo de dados de interesse e não o nó que o produz e transmite. Em todo caso, o destino da requisição é sempre o nó servidor, garantindo-se que se nenhum nó repassador interceptar o pedido de registro de participação, com certeza tal pedido alcançará o nó servidor e o estabelecimento de conexão ocorrerá normalmente. Esta decisão é fundamental para manter a compatibilidade com as aplicações de rede existentes na Internet.

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

```

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

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

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

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

```

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

4.4 Transmissão de $p_x \in P$ através de η

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, seguindo o princípio das redes centradas no conteúdo. Na prática, cada fluxo de dados P

corresponde a uma mídia gerada a partir de um evento real \mathcal{E} , por exemplo, a transmissão de um jogo de futebol, corrida de fórmula 1 etc.

No GMTP, define-se um nome de um fluxo de dados P por um código de *hash* no formato UUID (*Universally Unique IDentifier*) de 128 bits [207]. Na sua forma canônica, representa-se P por uma sequência de 32 dígitos hexadecimal, exibidos em cinco grupos separados por hífen, na forma de {8}-{4}-{4}-{4}-{12}. Por exemplo, $P = 641f931f-d3ac-50e3-b625-537574541f1f$. O nome de um fluxo de dados P sempre será informado no campo *nome do fluxo de dados (data flow name)*, disponível no cabeçalho de transporte dos pacotes *GMTP-Register*, *GMTP-Request*, *GMTP-Data* e *GMTP-Ack*.

Na prática, para gerar o nome para um fluxo de dados P , utiliza-se uma função de *hash* do tipo SHA1. Sendo assim, para determinar o nome de um fluxo de dados P , disponibilizado por um servidor s_a , utiliza-se $MD5(IP_{s_a} + : + PORTA_{s_a})$. 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 $MD5("200.17.113.98:21200") = f8ea01fd-4d71-5d95-89ec-35646e11d7fe$. 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 , os nós r_d mantêm a tabela de recepção de fluxo de dados que estão repassando para seus clientes $c_f \in C_i(r_d)$ e, sendo assim, podem compartilhá-la para outros nós repassadores. Dessa forma, o GMTP não requer alteração na camada de aplicação para informar o fluxo de dados de interesse – a aplicação continua informando endereço IP e número da porta, mantendo-se a compatibilidade com as aplicações existentes.

De posse de um identificador de um fluxo de dados P , um nó GMTP poderá solicitar os pacotes de dados $p_x \in P$. 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 [208]. Para o GMTP, criou-se um novo tipo de registro de DNS chamado de SID (*Streaming IDentifier*).

No Quadro 4, ilustra-se um exemplo de uma requisição DNS, utilizando a ferramenta *dig*, um comando de terminal para Linux. Nesse exemplo, apresenta-se a lista dos nomes dos fluxos de dados transmitidos pelo domínio administrativo *globo.com*. Por ser uma consulta

simples de DNS, qualquer sistema final conectado à Internet pode realizar tal procedimento, enaltecendo-se a facilitar de adaptar aplicações multimídia existentes para utilizar o GMTP. Ao indexar o conteúdo através de um serviço de DNS, permite-se desacoplar a forma de indexar um determinado conteúdo e a forma de obtê-lo, que passa a ser de responsabilidade da infra-estrutura de rede e não de uma ou mais aplicações isoladamente. 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 que as desenvolveu. Isto somente é possível porque o GMTP também oferecer 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

Uma outra característica do GMTP é permitir a descrição da mídia a ser transmitida e com isso promover a compatibilidade entre diferentes aplicações e reduzir o tráfego de rede para

um mesmo fluxo de dados P . Para isto, incorporou-se ao GMTP o protocolo o SDP (*Session Description Protocol*), definido na RFC 2327 [94], permitindo-se que as aplicações consigam obter mais detalhes sobre a mídia transmitida, flexibilizando-se o acesso a um determinado conteúdo, descrevendo para a aplicação o formato do conteúdo, que então o decodificará e o reproduzirá ao usuário através da aplicação final. Com esta decisão, torna-se mais fácil implementar novas aplicações multimídia, ao passo que também fica mais fácil adaptar aplicações existentes para fazer uso do GMTP, uma vez que, em sua grande maioria, utiliza-se o protocolo SDP. Do ponto de vista de engenharia de software, isto evitará a repetição de esforço com implementações já consolidadas e que, com o passar dos anos, provou-se funcionar a contento, como foi o caso do SDP. Consequentemente, caso seja necessário a atualização do referido padrão, tal atualização será realizada internamente no GMTP e todas as aplicações automaticamente já poderão usufruir dos novos recursos disponibilizados. Na prática, isto significa uma atualização a nível de sistema operacional.

No ponto de vista de uma aplicação em execução no nó s_a , esta precisa apenas determinar as informações da mídia e as fornece ao GMTP através de passagem de parâmetro via socket GMTP. 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 [209] */
14 a=framerate:23.0

```

No exemplo apresentado no Quadro 5, utiliza-se a primeira versão do protocolo SDP e descreve-se a transmissão de dois fluxos de dados P (Linhas 10 e 11), sendo um de áudio e outro de vídeo. A distribuição dos fluxos de dados P ocorre com a geração dos pacotes de dados $p_x \in P$ em três nós s_a (Linha 2), dos quais dois são acessíveis através de endereços IPv4 e um através de um endereço IPv6. Os fluxos de áudio e vídeo são repassados por um nó r_d , acessível por um endereço IPv4 (Linha 6), através das portas 49170 e 51372, respectivamente (Linhas 9 e 10). As informações de endereço IP e porta do nó r_d são utilizadas para que os nós $c_f \in C_i(r_d)$ possam sintonizar seus sockets de conexão e iniciar a

reprodução da mídia, através do modo de transmissão multicast (Linha 10). Em seguida, na Linha 8, observa-se uma URL do certificado digital a ser utilizado pelo nó r_d para verificar a autenticidade do conteúdo de pacote de dados $p_x \in P$ – na Seção 4.6, discute-se este assunto em mais detalhes. Por fim, entre as Linhas 11 e 17 especificam-se outros parâmetros para descrever a mídia, tais como o nível de qualidade da mídia, que varia entre 1 e 10, as taxas de bits para cada fluxo de dados, sendo para o áudio variando entre 16000 *Bytes* e 20000 *Bytes* e, para o vídeo, variando entre 156250 *Bytes* e 625000 *Bytes*. É importante salientar que os nós r_d utilizam as informações de taxa de bits para determinar o tamanho do buffer necessário para permitir a transmissão da mídia, o que ocorre ao adicionar uma nova entrada na tabela de recepção de fluxos de dados.

4.4.2 Estabelecimento de conexão entre c_f e s_a para obter P

No GMTP, divide-se o processo de estabelecimento de conexão em três fases. A Fase 1 acontece quando, por exemplo, um nó qualquer $c_1 \in C_i(r_d)$ deseja obter P transmitido por um nó s_1 e não existe nenhum outro nó $c_f \in C_i(r_d)$ em sua rede local recebendo P . Já a Fase 2 acontece quando um outro nó $c_2 \in C_i(r_d)$ precisa obter o mesmo fluxo de dados P , solicitado previamente pelo nó c_1 . E, por fim, a Fase 3 acontece quando o nó r_d começa a buscar novos nós parceiros r_q a fim de obter P . Na Figura 4.9, ilustram-se um nó s_a , que gera um fluxo de dados P e 12 nós r_d , que constituem uma rede de diferentes domínios administrativos, sendo o nó r_1 o repassador de um desses domínios, composto por 6 nós $c_f \in C_i(r_d)$ (Rede Local).

A regra geral é que um nó r_d deve consultar a tabela de recepção de fluxo de dados todas as vezes que receber um pacote do tipo *GMTP-Request* ou do tipo *GMTP-Register*, transmitido por um nó $c_f \in C_i(r_d)$. Com base no estado da referida tabela, que define a fase de conexão para um determinado fluxo de dados P solicitado, o nó r_d realiza uma determinada ação de registro de participação e repasse.

4.4.3 Fase 1: primeira requisição a um fluxo de dados P

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

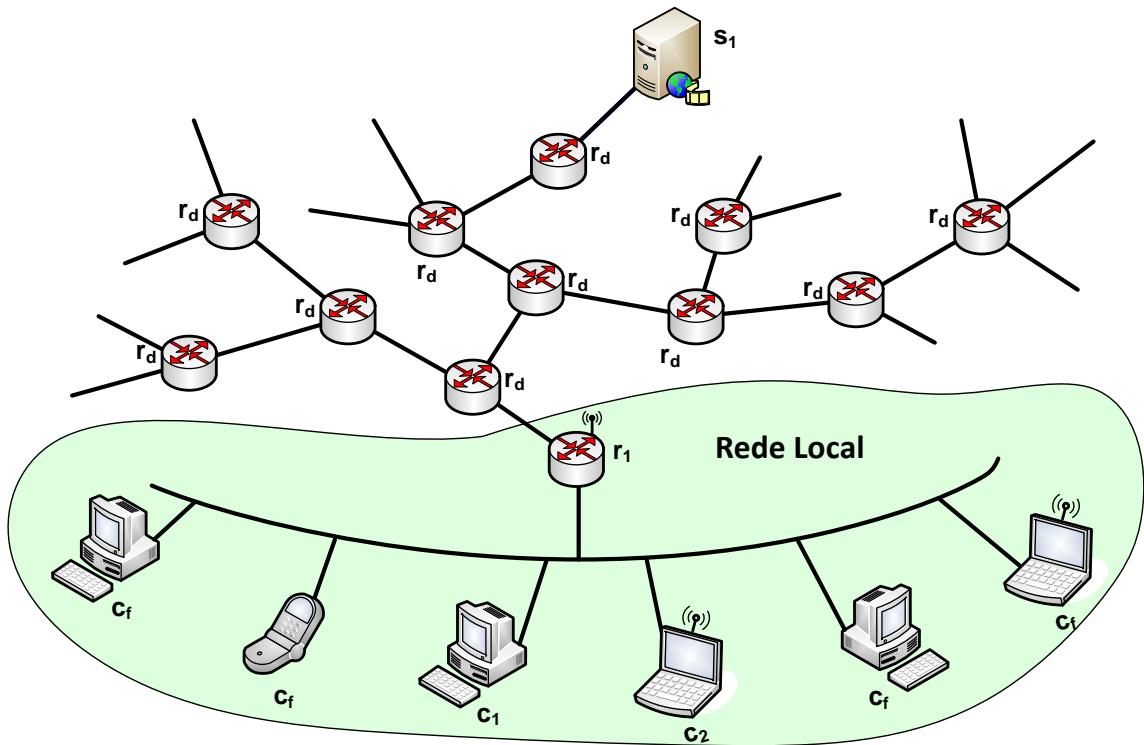


Figura 4.9: Exemplo de rede para o estabelecimento de conexão do GMTP.

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

Para obter o fluxo de dados P , o nó c_1 inicia o canal de controle GMTP (detalhado na Seção 4.7.1) e transmite um pacote do tipo *GMTP-Request* (Figura 4.10, Passo 1). Para construir o pacote do tipo *GMTP-Request*, qualquer nó c_f deve especificar o valor para o endereço IP de destino como sendo o endereço do nó s_a que transmite P , com o valor para o campo do cabeçalho de rede $TTL=1$. Além dos valores para o IP de destino e para o TTL , o nó c_f também deve informar o nome do fluxo de dados P que o usuário deseja reproduzir, presente no cabeçalho de transporte do pacote do tipo *GMTP-Request*. O valor de $TTL=1$ é intencional, pois faz com que o nó r_d intercepte o referido pacote de requisição, evitando-se extrapolar o domínio administrativo de sua rede local.

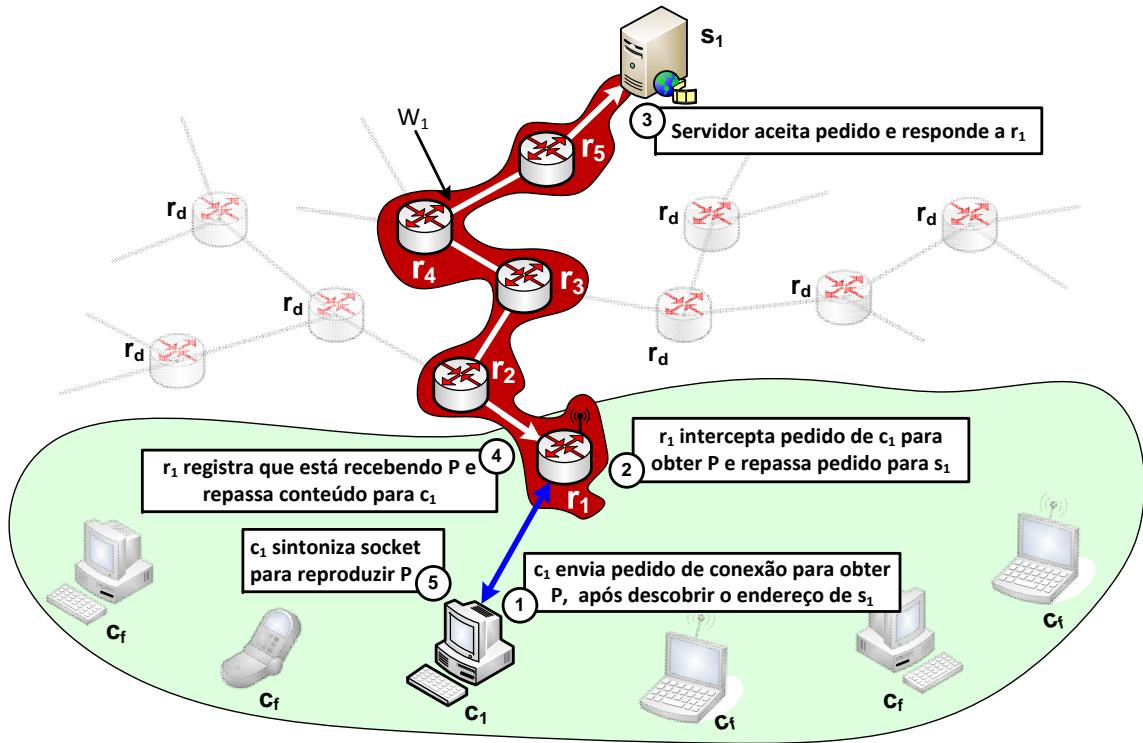


Figura 4.10: Passos do processo de estabelecimento de conexão do GMTP (Fase 1).

Quando o pacote *GMTP-Request* alcançar o nó r_1 (Passo 2 da Figura 4.10), este consulta a tabela de recepção de fluxos de dados e constata que não há qualquer registro para o fluxo de dados P . Nesse instante, o nó r_d inicia um processo de registro de participação para obter o fluxo de dados P . Isto significa que a execução do procedimento *registerRelay*(s_a , p_x) (Seção 4.3.2), onde p_x é o pacote do tipo *GMTP-Request*, fará o nó r_1 transmitir um pacote do tipo *GMTP-Register* em direção ao nó s_1 . À medida que os nós r_d repassam o pacote *GMTP-Register* até alcançar o nó s_1 , constitui-se o caminho $W_1 = \{r_1, r_2, r_3, r_4, r_5, s_1\}$ (Passo 3 da Figura 4.10 e destacado na cor vermelha), conforme discutiu-se na Seção 4.3.4.

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

No Algoritmo 6, resume-se os passos descritos anteriormente para iniciar a transmissão dos pacotes de dados $p_x \in P$ aos nós $c_f \in C_i(r_d)$, após r_d receber o pacote do tipo *GMTP-RequestReply*. Note que, o nó r_d invoca tal procedimento nas Linhas 7 e 14 do Algoritmo 1 e nas Linhas 11 e 17 do Algoritmo 2 (Seção 4.3.2). Como resultado da Fase 1, gera-se uma nova entrada na tabela de recepção de fluxos de dados do nó r_d , tal como ilustra-se na Figura 4.11. Com base no exemplo citado, a tabela de recepção antes vazia, agora contém uma entrada que informa a ocorrência de recepção do fluxo de dados $P = 72c44591-7d82-427c-825f-722f015787c1$, originado no nó s_a , cujo endereço é $177.135.177.241$, com porta de recepção 49170 . Além disso, define-se o canal multicast no endereço $239.192.68.79$ e porta 1900 , através do qual os nós $c_f \in C_i(r_d)$ podem receber os pacotes de dados $p_x \in P$.

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 11 and 17 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 $errorCode \leftarrow \text{getPacketFieldValue}(p_x, \text{'errorCode'});$

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

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

7 $\text{sendPkt}(p_x);$

8 **return** 0;

9 **end**

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

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

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

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

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

19 $\text{sendPkt}(p_x);$

20 **end**

21 **return** 0;

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

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

4.4.4 Fase 2: próximas requisições para obter P

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

4.4.5 Fase 3: busca por mais parceiros r_q para obter P

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

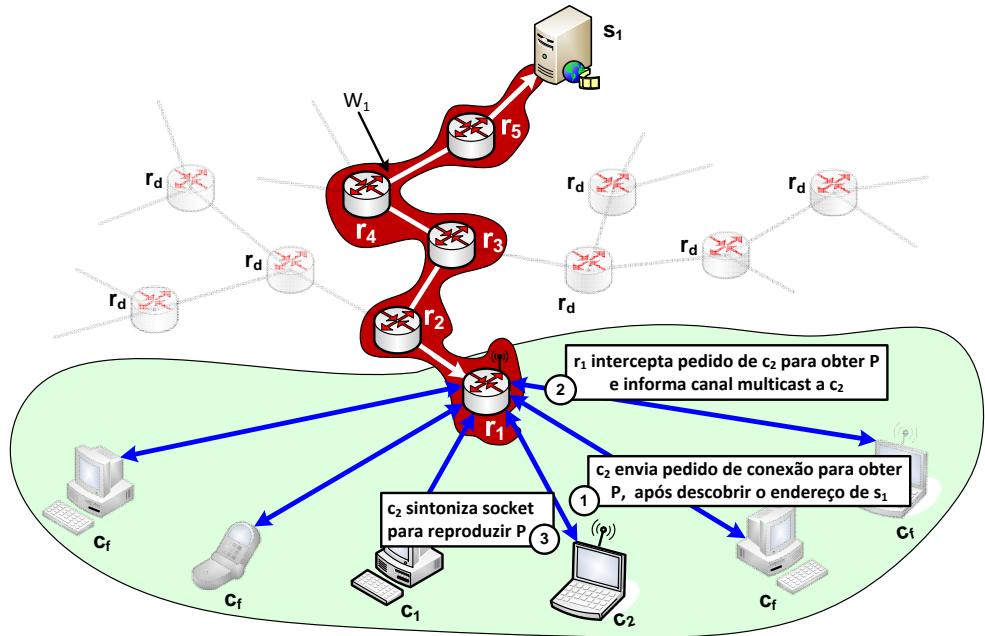


Figura 4.12: Passos do processo de estabelecimento de conexão do GMTP (Fase 2).

1. Maior capacidade de transmissão do caminho W_v . Define-se este critério com base na taxa de transmissão disponível nos nós $w_m \in W_v$ com menor capacidade de transmissão 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.

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 o nó r_d não estabelece parcerias com todos eles. Em vez disso, executa-se repetidamente as seguintes ações:

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

A periodicidade de requisições do tipo *GMTP-RelayQuery* e a quantidade máxima de parcerias efetivas são parâmetros controláveis pelo administrador do nó r_d . Na implementação do GMTP utilizada neste trabalho, definiu-se o tempo de 5 minutos para a periodicidade de requisições do tipo *GMTP-RelayQuery* e 5 para a quantidade de parcerias efetivas. Os valores destes parâmetros são padrões em aplicações tradicionais de distribuição de conteúdos multimídia ao vivo baseados em redes P2P.

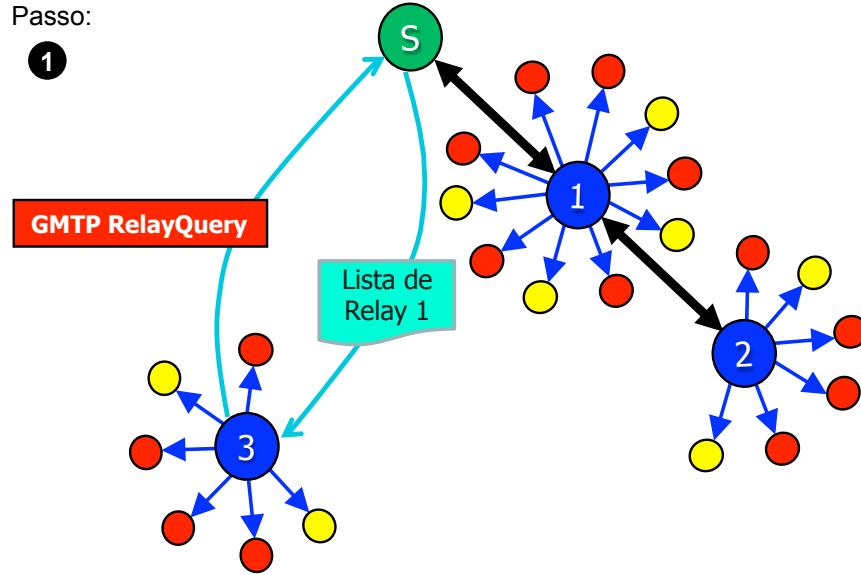


Figura 4.13: Fase 3 de conexão do GMTP (Passo 1).

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

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

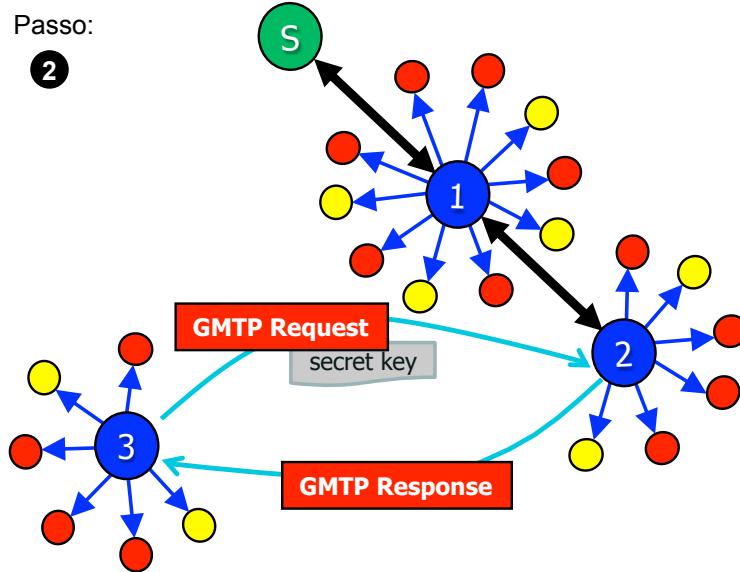


Figura 4.14: Fase 3 de conexão do GMTP (Passo 2).

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

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

do GMTP supre uma antiga deficiência das soluções tradicionais de transmissão multicast, pois as aplicações tinham que se adaptar às configurações estáticas dos canais multicast definidos pelo administrador de rede, e até os próprios administradores de rede tinham que fazer tal configuração de forma manual, obrigatoriamente em todos os nós roteadores de um determinado caminho. Isto é impraticável devido à independência dos diferentes domínios administrativos.

Até o presente momento, não se conhece nenhuma solução que permita configuração dinâmica de canais multicast aliado a formação de uma rede de sobreposição constituída entre roteadores, com benefícios diretos para a aplicação e para a rede, fazendo-se uso dos recur-

sos computacionais e de rede de forma mais apropriada, como será discutido no Capítulo 5 (Resultados).

4.4.6 Envio e recebimento de $p_x \in P$ em η

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 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 o GMTP executa os procedimentos de transmissão e recepção desses pacotes de dados.

Buffer de Envio e Recepção:

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

O buffer de envio e recepção do GMTP tem seu tamanho definido no processo de estabelecimento de conexão, de acordo com o tipo da mídia sendo transmitido e o nó r_d pode determinar um limite máximo. Para isto, o administrador de um nó r_d pode definir esse limite máximo do buffer para qualquer fluxo de dados P , evitando-se ataques de negação de serviço (*Denied of Service*). Isto porque, no GMTP, uma aplicação passa a poder definir o tamanho do buffer que cada nó w_m deverá alocar para repassar os pacote de dados p_x .

de um fluxo de dados P . Após definir o tamanho do buffer para um fluxo de dados P , tal tamanho permanece fixo durante todo o ciclo de vida de uma conexão GMTP. Essa decisão é importante porque permite um nó s_a alocar previamente o recurso necessário para o transporte de um determinado fluxo de dados P . O tamanho do buffer é especificado pelo nó s_a e propagado para os demais nós em um caminho W no cabeçalho do pacote do tipo *GMTP-Register-Reply* ou *GMTP-MediaDesc*.

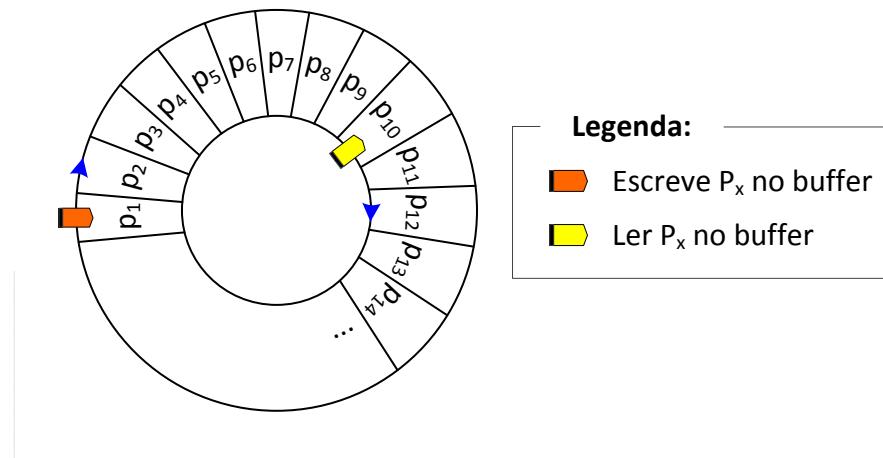


Figura 4.16: Exemplo da estrutura do buffer de envio e recepção de um nó GMTP com dois ponteiros, um para escrever e outro para ler pacotes p_x .

Mapa de Buffer:

O mapa de buffer do GMTP descreve o estado atual do buffer de envio e recepção de um nó GMTP. Como ilustrado na Figura 4.17, trata-se de uma estrutura de dados que determina se um pacote p_x está ou não presente no buffer de um respectivo nó GMTP. O conteúdo de cada posição é o número de sequência do pacote, que determina a ordem que um pacote foi gerado e transmitido pelo nó s_a , tal como ocorre no protocolo TCP.

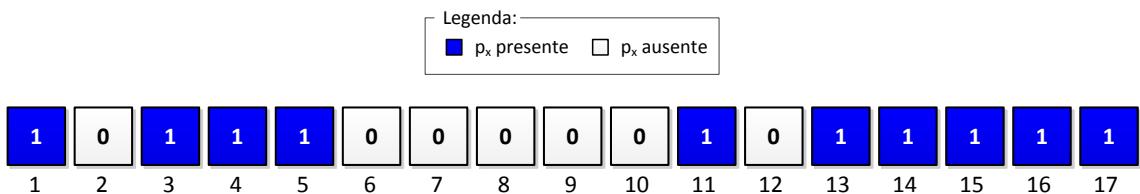


Figura 4.17: Exemplo do mapa de buffer de um nó GMTP com tamanho de 17 p_x .

O mapa de buffer é utilizado por um nó GMTP para sinalizar seu atual estado com relação a um determinado fluxo de dados P . Um nó GMTP pode enviar o mapa de buffer completo, como ilustrado na Figura 4.17, ou o mapa de buffer apenas dos p_x presentes ou ausentes. Na prática, um nó r_d envia para um nó parceiro r_q o mapa de buffer dos p_x presentes quando deseja indicar a sua atual disponibilidade; ao passo que envia o mapa de buffer dos p_x ausentes quando desejar obtê-los. 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 automaticamente o uso do método *pull*, em vez do método *push*, que é o modo padrão do GMTP. Salienta-se que deve-se evita o método pull devido à transitoriedade dos pacotes de dados p_x (transmissão de dados ao vivo) e um nó r_d deve apenas realiza tal procedimento após completar a Fase 3 do processo de estabelecimento de conexão. Isto porque um nó r_d pode nunca receber a resposta para uma requisição do tipo pull.

Quando um nó r_d percebe a falta de um ou mais pacotes p_x , este pode solicitar a um ou mais nós r_d os pacotes p_x ausentes e então obtê-los usando o método *pull*. Para isso, um nó r_d enviar aos seus nós parceiros r_q o mapa de buffer dos pacotes p_x ausentes e aguarda as respostas sobre tal disponibilidade. Essa sinalização ocorre através do uso do pacote do tipo *GMTP-DataPull-Request*, que é preenchido com o mapa de buffer dos pacotes ausentes e transmitido aos respectivos nós parceiros. Ao receber esse tipo de requisição, um nó parceiro avalia seu conteúdo e responde com o pacote do tipo *GMTP-DataPull-Response*, o qual contém o mapa de buffer dos pacotes disponíveis, seguido dos pacotes p_x do tipo *GMTP-Data*. Note que os pacotes do tipo *GMTP-DataPullRequest* e *GMTP-DataPull-Response* são transmitidos com garantia de entrega

Na prática, o mapa de buffer utilizado para sinalizar a presença ou ausência de p_x é representado por faixas de acordo com o índice do buffer. Por exemplo, para representar o mapa de buffer dos pacotes ausentes ilustrados na Figura 4.17, o nó GMTP preenche o pacote do tipo *GMTP-DataPull-Request* com a sequencia 2;6-10;12. Ao receber esta sequência, o nó parceiro r_q responde com o pacote do tipo *GMTP-DataPull-Response*, que contém o mapa de buffer de quais pacotes serão enviados e começa a transmití-los.

Descarte de pacotes:

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

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

4.5 Controle de Congestionamento em η

Na GMTP, executa-se um algoritmo para controle de congestionamento híbrido, cujo comportamento dependerá se o nó que o executa está transmitindo em modo unicast ou em multicast. Como ilustra-se na Figura 4.18, na prática, trata-se de dois algoritmos para controle de congestionamento, um que atua em transmissões unicast, chamado de *GMTP Unicast Congestion Control* (GMTP-UCC) e outro que atua em transmissões multicast, chamado de *GMTP Multicast Congestion Control* (GMTP-MCC). Nesta seção, discute-se o funcionamento de cada um desses algoritmos.

Em modo de transmissão unicast, utilizado na comunicação entre os nós r_d , define-se a taxa de transmissão de um nó GMTP através de uma versão modificada do protocolo RCP (Rate Control Protocol) [55]. Já em modo de transmissão multicast, executa-se um algoritmo

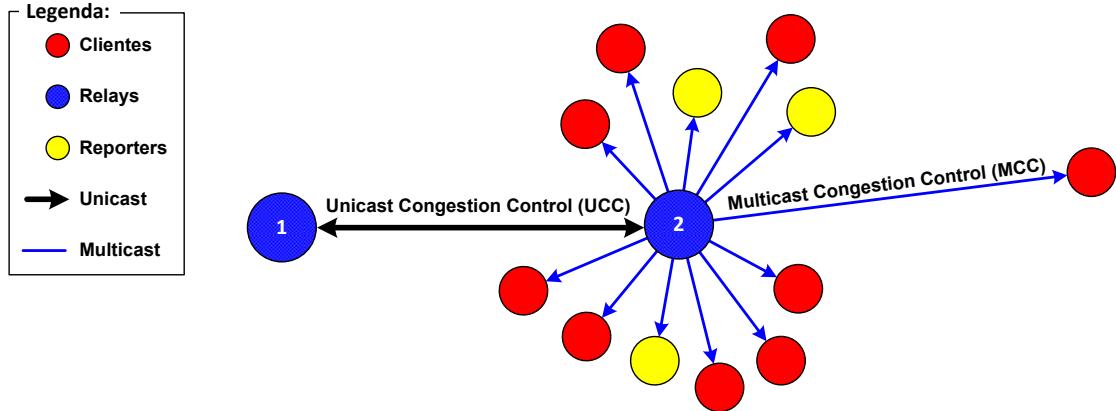


Figura 4.18: Organização do algoritmo de controle de congestionamento no GMTP.

baseado em relatórios (*feedbacks*) transmitidos pelos nós clientes l_w , eleitos em cada rede e controlados por um nó r_d , tal que $l_w \in C_i(r_d)$.

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 [1]. Nesse contexto, entende-se que se um roteador pudesse obter a informação exata sobre o número de fluxos de entrada em um instante t , a taxa de transmissão ideal para cada fluxo de dados seria $R_{ps}(t) = \frac{C}{N(t)}$, onde C corresponde à capacidade do link e $N(t)$ ao número de fluxos no instante t .

Partindo desse ponto, Nandita et. al [55] argumenta que para um roteador funcionar de forma equânime, este deve oferecer a mesma taxa de transmissão para todos os fluxos transmitidos através dele, mantendo-se o número de pacotes na fila de roteamento perto de zero, a fim de evitar que apenas os fluxos que tem pacotes na fila de repasse compartilhem a largura de banda disponível. Com base nisso, Nandita et. al [55] determinou a Equação 4.1, onde $R(t)$ 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 $\alpha(C - y(t)) - \beta \frac{q(t)}{h_0}$ (mudança agregada) e a divide por $N(t)$. Porém, como é impossível determinar o valor exato de $N(t)$, estima-se $\hat{N}(t) = \frac{C}{R(t-T)}$ e para atualizar $R(t)$ com mais frequência do que no tempo de um RTT, escala-se a mudança

agregada por $\frac{T}{h_0}$, resultando na Equação 4.2, onde:

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

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

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

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

- $H = \min(RTT_{user}, h_0)$, sendo RTT_{user} um tempo definido pelo usuário (administrador do roteador), caso seja necessário atualizar $R(t)$ mais rápido do que o tempo de h_0 ;
- $R(t - T)$, é a última taxa de transmissão medida;
- $y(t)$, é a taxa de tráfego de entrada medida no intervalo entre a última atualização da taxa de transmissão e h_0 ;
- $q(t)$, é o tamanho instantâneo da fila de repasse, em bytes. Note que no GMTP esse valor é obtido pela soma de todos os pacotes p_x presentes em todos os buffers de recepção e envio para cada fluxo de dados P . 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 $c_f \in C_i(r_d)$;
- α e β , são parâmetros pré-definidos que determinam a estabilidade e o desempenho;
- C , é a capacidade do link.

No GMTP-UCC, o algoritmo para controle de congestionamento, adaptado do RCP, funciona da seguinte forma (acompanhe os passos de acordo com a Figura 4.19):

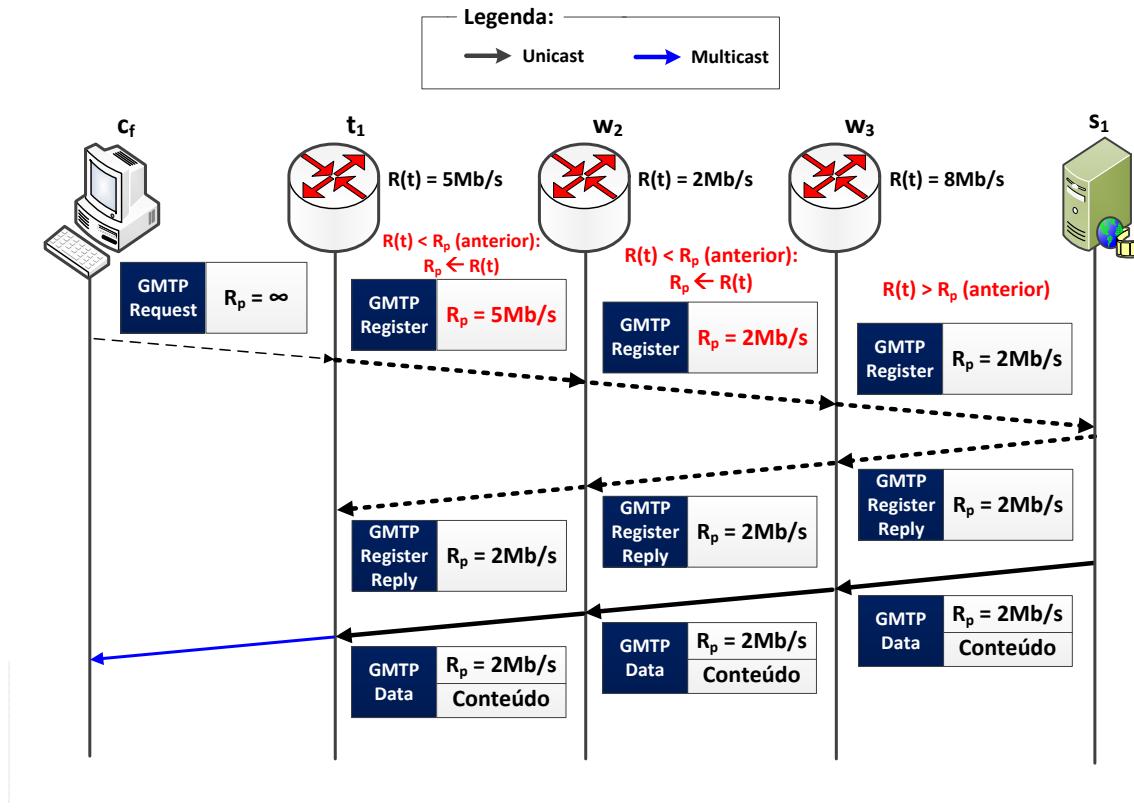


Figura 4.19: Cada r_d mantém uma única taxa de transmissão $R(t)$ que é atribuída em todos os pacotes de dados p_x no cabeçalho de qualquer pacote GMTP. A medida que o pacote passa através dos roteadores em um caminho W_v , se a taxa atual $R(t)$ no roteador for menor do que R_p informado no pacote sendo processado, o roteador o sobrescreve com sua taxa atual $R(t)$. Quando o pacote alcançar o nó s_a , este utiliza R_p para transmitir o fluxo de dados, pois trata-se da taxa de transmissão do roteador mais congestionado no caminho W_v .

1º Seja um caminho W_v , todo nó w_m mantém uma única taxa de transmissão $R(t)$, que é oferecida para todos os fluxos de dados passando por w_m em um certo instante t . Cada nó w_m atualiza $R(t)$ aproximadamente a cada RTT.

2º Todo pacote GMTP carrega duas informações de controle no campo de cabeçalho:

- *taxa de transmissão proposta (R_p)*: corresponde à taxa de transmissão do roteador com menor capacidade de transmissão em um instante 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 t_u e t_{u+1} que processam o respectivo pacote p_x de um fluxo de dados P , a fim de repassar aos seus nós $c_f \in (C_i(t_u) \cup$

$C_i(t_{u+1})$). Apesar de uma mudança suscinta, este aspecto diferencia o GMTP-UCC do RCP, a ser discutido em mais detalhes a seguir.

- 3° No início da transmissão de um fluxo de dados P , o nó w_m , motivado por um ou mais nós $c_f \in C_i(w_m)$, transmite um pacote p_x com o valor de $R_p = \infty$ em direção a s_a .
- 4° Todo nó $w_m \in W_v$, ao receber um pacote p_x , verifica se $R(t) < R_p$ e, em caso afirmativo, então $R_p \leftarrow R(t)$, caso contrário nenhuma modificação é realizada nesse campo. Nesse ínterim, se $\varphi(w_m, P) = 1$, w_m também executa as seguintes ações:
 - (a) repassa p_x para seus nós c_f em modo multicast, como discutiu-se na Seção 4.4.6;
 - (b) utilizar RTT_s , presente no pacote p_x , para atualizar a média móvel do RTT no intervalo de h_0 (Equação 4.3) e, em seguida, atualiza $R(t)$;
 - (c) cria um pacote *GMTP-Ack* contendo R_p e o envia em direção a s_a . Este procedimento permitirá que o nó s_a saiba em quais pontos o caminho W_v está sendo segmentado. Este assunto será discutido em mais detalhes a seguir.
- 5° O nó t_u deve usar R_p como a nova taxa de transmissão para enviar os próximos pacotes de dados p_x para seu nó parceiro t_{u+1} . Assim, R_p é a menor taxa de transmissão oferecida ao longo do sub-caminho $W'_v \subset W_v$, tal que $W'_v = \{t_u, \dots, t_{u+1}\}$. Note que esta regra automaticamente se aplica também ao nó s_a , já que este é o último salto de um caminho W_v .
- 6° Todo nó r_d atualiza sua taxa de transmissão local $R(t)$ de acordo com a Equação 4.2, no intervalo de tempo correspondente a H .

Sendo assim, no caso do GMTP-UCC, a ideia básica é a seguinte: para quaisquer dois nós $t_1, t_2 \in W_v$, a taxa de transmissão a ser utilizada por t_1 e t_2 será definida pela menor taxa de transmissão oferecida pelos nós $w_m \in W_v$ posicionados entre t_1 e t_2 . Isto significa que o GMTP-UCC segmenta um caminho W_v em vários sub-caminhos W'_v . Com isto, se existir largura de banda disponível entre t_1 e t_2 , ou seja, $C - y(t) > 0$, então o GMTP-UCC compartilhará igualmente o canal entre todos os fluxos, inclusive para o fluxo entre t_1 e t_2 . Caso contrário, ou seja, se $C - y(t) < 0$, considera-se o canal saturado e o GMTP-UCC reduzirá a taxa de transmissão igualmente para todos os fluxos, inclusive para o fluxo entre

t_1 e 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.

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 em R_p . 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, observe o cenário ilustrado na Figura 4.20, 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 $c_f \in (C_i(t_1) \cup C_i(t_3))$ recebendo 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 Mb/s tanto para os nós $c_f \in C_i(t_1)$ quanto para os nós $c_f \in C_i(t_3)$. Ora, 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 $R(t)$ é igual a 4 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 $R(t)$ corresponde a 8 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} . Considerando o mesmo exemplo ilustrado na Figura 4.20, mas adotando essa estratégia de segmentar o caminho W_v , tal cenário corresponde ao ilustrado na Figura 4.21. Note que no sub-caminho $W_1^\triangleleft = \{t_3, w_2, t_1\}$ a taxa de transmissão de t_3 em direção a t_1 será de 1 Mb/s , ao passo que no sub-caminho $W_2^\triangleleft = \{s_1, w_4, t_3\}$ será de 4 Mb/s . Sendo assim, os nós $c_f \in C_i(t_1)$ receberão o fluxo de dados P em uma taxa de 1 Mb/s , ao passo que os nós $c_f \in C_i(t_3)$ receberam a uma taxa de 4 Mb/s .

Ordenação dos melhores caminhos com base em $R(t)$:

Na Seção 4.4.5, discutiu-se a Fase 3 de conexão do GMTP, que permite um nó s_a sugerir possíveis nós r_q como candidatos a parceiros de um nó solicitante r_d . O primeiro critério para sugerir nós parceiros r_q é priorizar aqueles que fazem parte de um caminho W_v com

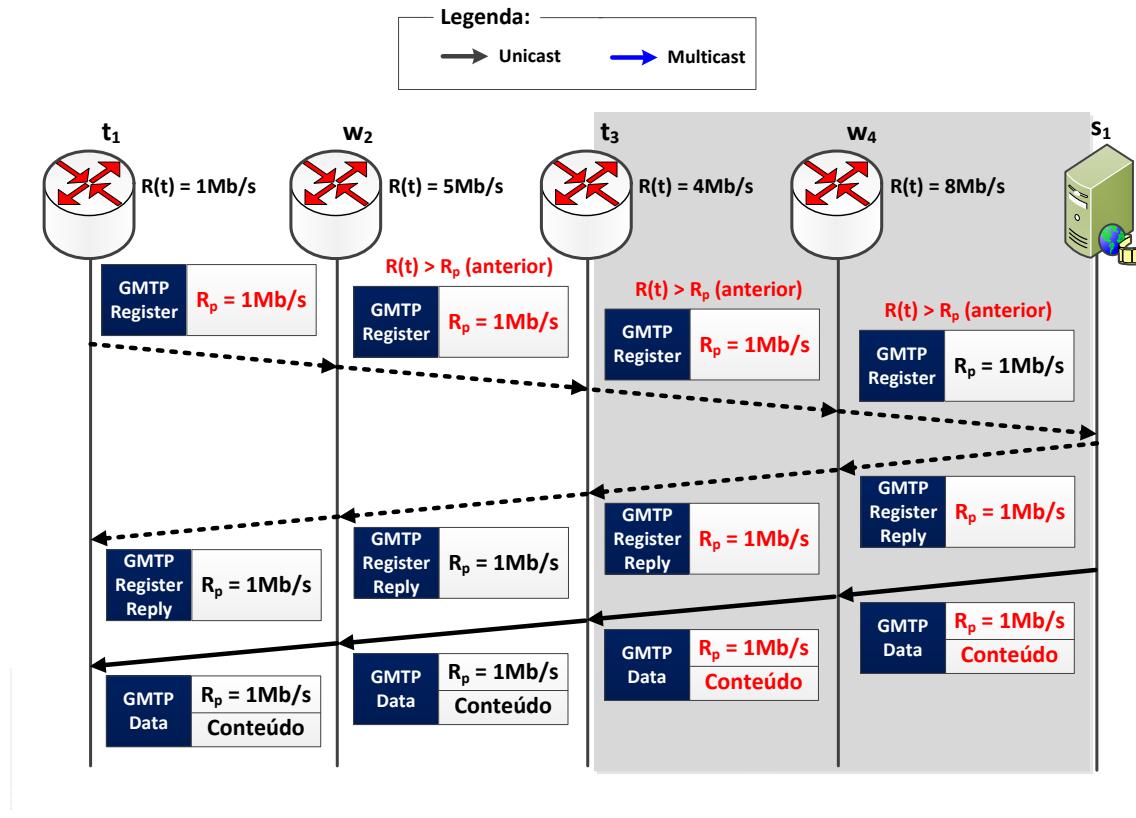


Figura 4.20: O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão, porém isto pode limitar alguns nós clientes a receberem os pacotes de dados em uma taxa maior.

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 de congestionamento, obtidos pelo Passo 4c do algoritmo GMTP-UCC. Sendo assim, dependendo da posição de um nó solicitante r_d em um caminho W_v , o nó s_a determina os parceiros r_q e os sugere ao nó r_d através da intersecção de caminhos conhecidos, utilizando como critério de ordenação as capacidades de transmissão dos caminhos conhecidos W .

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

A motivação para o RCP é identificar um algoritmo para controle de congestionamento simples e prático para emular um PS 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

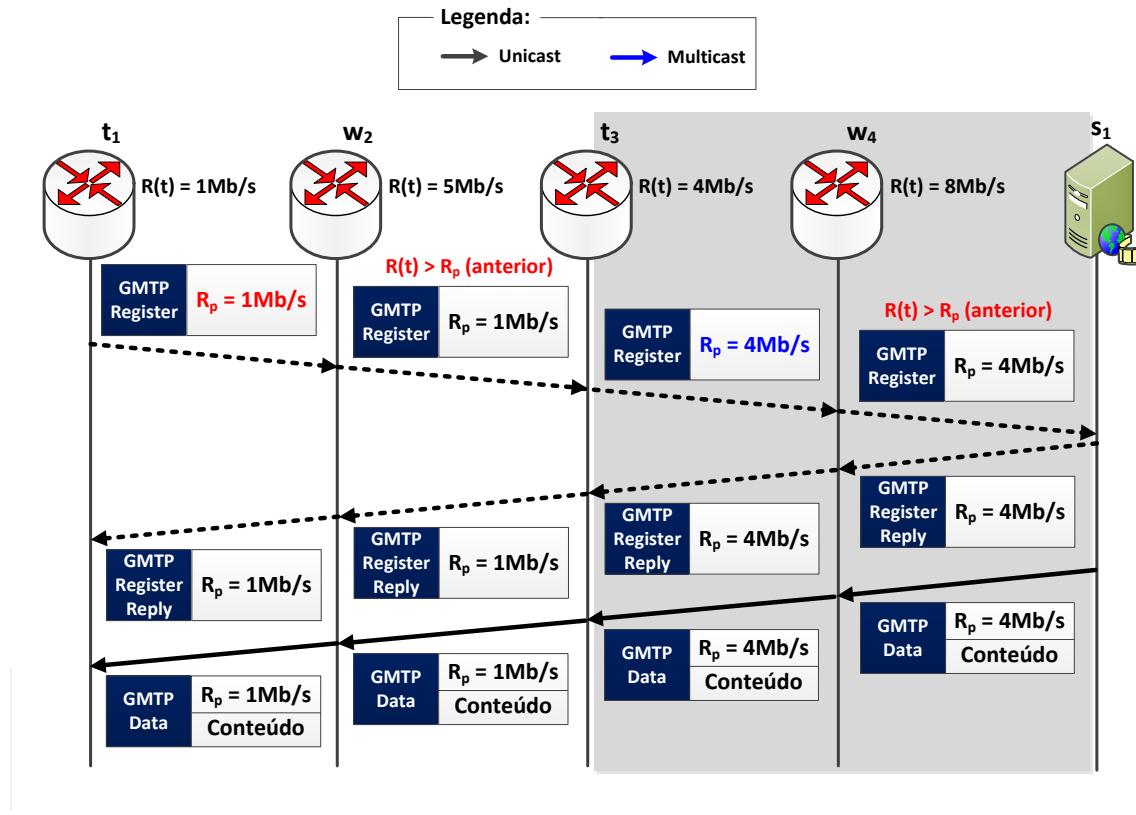


Figura 4.21: O GMTP-UCC segmenta o caminho e dessa forma não limita a taxa de transmissão de um fluxo de dados para certos nós capazes de receber em uma taxa de transmissão maior.

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 [1]. 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 $R_{rcp}(t)$ e $R_{xcp}(t)$ é bastante diferente, alocando-se tais taxas para cada fluxo de dados a fim de emular a taxa de transmissão do PS, definida por $R_{ps}(t)$. 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 [1].

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 equi-

líbrio onde todos os transmissores transmitirão pacotes de dados a uma taxa de transmissão $R_{xcp}(t)$, ao passo que o RCP calcula uma única taxa de transmissão que deve ser utilizada por todos os nós transmissores em um certo instante t . Apesar dessa diferença suscinta, deve-se entender minuciosamente o que isto significa.

No caso do XCP, o protocolo aumenta ou diminui a janela de congestionamento de um fluxo de dados de acordo com o tamanho atual da sua janela de congestionamento. Isto significa que o XCP reduz gradativamente os tamanhos da janela de congestionamento dos fluxos com $R_{xcp}(t)$ maior do que o $R_{ps}(t)$ estimado, aumentando-se gradativamente o tamanho das janelas de congestionamento dos fluxos com $R_{xcp}(t)$ menor do que $R_{ps}(t)$ estimado. Porém, 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. Por exemplo, nos gráficos da Figura 4.22, compara-se o TCP e o XCP com um PS ideal com base em uma rede simulada, com taxa de entrada de pacotes de dados de um fluxo 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 oferecida igual a 0.9. No gráfico esquerdo, ilustra-se o tempo médio de duração (quanto tempo o respectivo protocolo gasta para completar o fluxo) em função do tamanho do fluxo. No gráfico direito, ilustra-se o número de fluxos ativos em função do tempo. Os valores de PS foram calculados a partir de expressões analíticas [210] e mostram que os fluxos poderiam ser finalizados uma ordem de magnitude mais rápida do que o TCP.

Com base nos gráficos da Figura 4.22, observa-se que os fluxos TCP demoram para finalizar porque consome-se múltiplos RTTs na fase de partida lenta para encontrar uma taxa de transmissão equânime, além do mais, muitas vezes o fluxo acaba antes que tal taxa seja encontrada. Em seguida, quando o fluxo TCP entra no modo de prevenção de congestionamento, o TCP adapta-se lentamente devido ao método de aumento aditivo, o que aumenta o tempo de finalização do fluxo. Além disso, o TCP deliberadamente preenche o buffer dos roteadores saturados de modo a ajustar a taxa de transmissão com base nos descartes de pacotes, mas buffers adicionais resulta em aumento no tempo (atraso) para entregar um pacote de dados, impactando no tempo total de duração de um fluxo. Já o XCP funciona melhor em

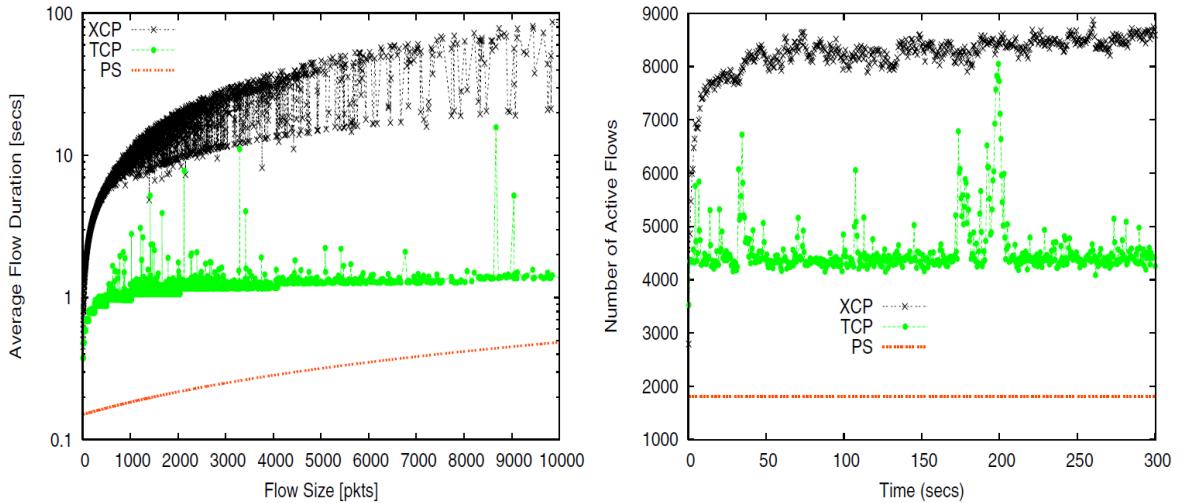


Figura 4.22: No gráfico esquerdo, ilustra-se o tempo médio de duração (quanto tempo leva para completar) de um fluxo versus o tamanho do fluxo utilizando o TCP e o XCP. No gráfico direito, ilustra-se o número de fluxos ativos versus o tempo. Ambos os gráficos são resultados de simulações com chegada 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 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 [1].

redes com altos produtos largura de banda–atraso. Os roteadores disponibilizam para as fontes transmissoras relatórios sobre as mudanças da janela de congestionamento, enviados em múltiplos RTTs, que funcionam a contento quando todos os fluxos são de longa duração. Por isso, em um ambiente dinâmico, o XCP pode aumentar a duração de cada fluxo em relação ao PS ideal, resultando em mais fluxos de dados em trânsito na rede em qualquer instante, principalmente os fluxos de curta duração.

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

Como observa-se no gráfico da Figura 4.23, a estratégia do RCP de compartilhar uma

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

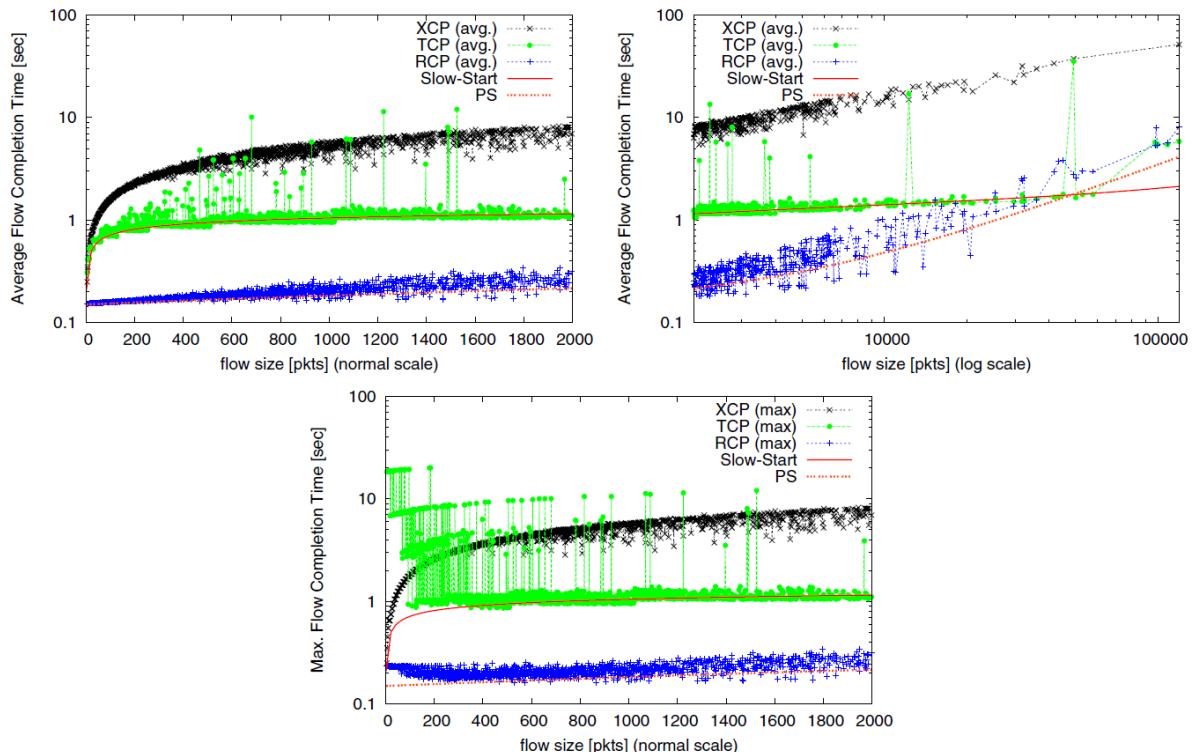


Figura 4.23: Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP como resultados de simulações com taxa de entrada de dados em Poisson e tamanhos do fluxo em distribuição Pareto com média de 25 pacotes (1000 bytes pacote), shape igual a 1.2, capacidade do link igual a 2.4 Gbps e RTT igual a 100 ms , com carga ofertada igual a 0.9. Os valores de PS foram calculados a partir de expressões analíticas. Extraído de [1].

O XCP é computacionalmente mais complexo do que o RCP, uma vez que o XCP define diferentes valores de *feedback* para cada fluxo, envolvendo operações matemáticas (multiplicação e soma) para cada pacote, o que torna o XCP mais lento que o RCP. Pela estratégia de mudança no tamanho da janela de congestionamento, o XCP pode levar múltiplos RTTs

para a maioria dos fluxos alcançarem a taxa de transmissão equânime entre eles, mas que mudam com o passar do tempo à medida que novos fluxos são injetados na rede e outros são finalizados, devido à natureza dinâmica das redes. No caso do RCP, essa complexidade é menor e há uma redução significativa na convergência entre a taxa de transmissão praticada $R_{rcp}(t)$ e a taxa estimada do PS ($R_{ps}(t)$). Isto porque mantém-se uma única taxa de transmissão para todos os fluxos, não envolvendo qualquer computação adicional por pacote p_x que passa por r_d . Além disso, para determinar $R_{rcp}(t)$, utiliza-se apenas o tamanho da fila e a taxa agregada de entrada, sem necessitar manter estado por fluxo de dados e operações matemáticas por pacote de dados.

Desta forma, os aspectos que determinam o funcionamento do RCP são fundamentais 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, permite-se que fluxos de dados GMTP/RCP e TCP/RCP coexistam na Internet de forma equânime, aliado às funções do GMTP de distribuição de conteúdo assistida pela rede, evitando-se sobrecarga nos nós s_a .

4.5.2 Controle de Congestionamento Multicast

Da mesma forma que no GMTP-UCC, o objetivo principal do GMTP-MCC é determinar uma taxa de transmissão equânime entre os fluxos de dados transmitidos pelo GMTP e por outros protocolos, como o TCP, porém em modo de transmissão multicast. No caso GMTP-MCC, trata-se de um algoritmo responsável pelo controle de congestionamento em uma rede local constituída por $\eta_{local} = r_d \cup C_i(r_d)$. Na prática, os nós da rede η_{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 [211]). O TFRC é um mecanismo para controle de congestionamento de fluxos unicast que tenta prevê a taxa de transmissão de um fluxo TCP e utilizá-la em protocolos diferentes do TCP [151]. 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. 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 [212, 213]. 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 [214], $R(s, p)$ é a taxa de transmissão medida em bytes/segundo definida em função de s , que é o tamanho do pacote medido em bytes e p , que corresponde a taxa de perda de pacotes observado pelo nó receptor; RTT é o tempo de ida-volta entre o nó transmissor e o receptor, medido em segundos.

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

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

- 1° O nó r_d executa um algoritmo de eleição de nós relatores $l_w \in C_i(r_d)$. Na Seção 4.7.3, descreve-se o procedimento para eleger os nós l_w .
- 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 [211, 214];
- 4° O RTT é calculado entre o nó l_w e o nó r_d , com o temporizador controlado pelos nós l_w e não pelo nó r_d . Isto evita que o nó r_d tenha que manter estado de temporizador para cada fluxo de dados P transmitido para os nós $c_f \in C_i(r_d)$. Para determinar o valor do parâmetro RTT e calcular a taxa de transmissão através da Equação 4.4, o GMTP-MCC utiliza a Equação 4.3, que também é utilizada no GMTP-UCC, porém com $\theta = 0.25$, valor igual ao utilizado no TCP e no DCCP;
- 5° A taxa de transmissão a ser utilizada pelo nó r_d é a média aritmética de todas as taxas enviadas pelos nós l_w ;
- 6° Repete-se todos os passos a partir do passo 2 a cada intervalo igual ao RTT ou quando um intervalo de perda p é determinado.

Teoricamente, o GMTP-MCC seria um protocolo *TCP-Friendly* se $R(s, p)$ fosse o valor máximo entre as taxas de transmissão relatadas pelos nós l_w . Porém, optou-se por utilizar a média 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 ms , que é o mesmo adotado no TCP. Quando um nó c_f envia um pedido de conexão utilizando o pacote do tipo *GMTP-Request*, o mesmo deve realizar a sua primeira medição do valor de RTT, iniciando-se o marcador de tempo para o cálculo do RTT quando enviar o primeiro *GMTP-Request* e parando-o quando receber o pacote do tipo *GMTP-Response*. Em seguida, deve-se acionar o mecanismo de cálculo da taxa de transmissão através da Equação 4.4, caso o respectivo nó c_f seja eleito um nó relator.

4.6 Autenticidade de P

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

conteúdo multimídia errados, consequentemente não consumindo recursos computacionais desnecessários.

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

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

1. apenas os nós w_m , tal que $\varphi(w_m, P) = 1$ devem realizar o procedimento de verificação de autenticidade do fluxo de dados P ; e
2. os nós w_m , definidos pela Regra 1, não devem verificar todos os pacotes $p_x \in P$, mas apenas uma quantidade $pc(t)$ de pacotes de dados $p_x \in P$, em um instante t . Nesse caso, define-se $pc(t)$, apresentada na Equação 4.5, em função de:
 - $bs(t, P)$, o número de pacotes $p_x \in P$ presentes no buffer de repasse de w_m em um instante t ;
 - $\frac{1}{|W_v^\triangleleft|-1}$, a probabilidade de um nó $r_d \in W_v^\triangleleft$ ter alterado o conteúdo de um ou mais p_x presente(s) no buffer de repasse de w_m , onde $W_v^\triangleleft = \sim(\delta(w_{m+1}, W_v))$ e W_v é o caminho através do qual se transmite os pacotes de dados $p_x \in P$;

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

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

4.6.1 Transmissão e assinatura de autenticidade de $p_x \in P$

Quando o nó s_a gerar cada pacote de dados $p_x \in P$, este deve gerar uma assinatura digital dos dados da aplicação a serem transportados. Em seguida, o nó s_a deve incluir a assinatura digital gerada no cabeçalho do pacote de dados p_x , no campo assinatura (*signature*). Para assinar digitalmente o conteúdo da aplicação, utiliza-se o método de criptografia assimétrica RSA, onde $K_{s_a}^-$ e $K_{s_a}^+$ representam a chave privada e a chave pública de s_a , respectivamente. No Trecho de Código 7, apresenta-se o procedimento de assinatura de um pacote $p_x \in P$ adotado no GMTP, utilizando-se a mesma técnica apresentada na Seção 2.3.

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 proce-

dimento de verificação de autenticidade de um ou mais pacotes de dados $p_x \in P$.

Dada a quantidade $pc(t)$ de pacotes que w_m deve verificar suas respectivas autenticidades, o nó w_m escolhe aleatoriamente (distribuição uniforme) os pacotes p_x disponíveis no buffer de recepção, gerando um conjunto $P' \subset P$. Uma vez definido P' , w_m executa o procedimento de verificação de autenticidade que funciona a seguinte forma. Para cada pacote $p_x \in P'$, extrai-se a assinatura do pacote p_x , gerada pelo nó s_a , como explicado na Seção 4.6.1. Em seguida, extrai-se o campo de dados para que se possa verificar sua autenticidade. Para isto, gera-se o valor de *hash* do campo de dados e compara-se com o valor de *hash* gerado pelo nó s_a no momento da transmissão do pacote p_x . Note que o valor de *hash* gerado pelo nó s_a é obtido através de processo de decriptar a assinatura do pacote de dados p_x utilizando a chave pública do nó s_a . Assim, se o valor de *hash* gerado com base no conteúdo transportado no pacote p_x for igual ao valor de *hash* disponível na assinatura do pacote, conclui-se que o pacote p_x não foi alterado por nenhum nó $w_m \in W_v^\triangleleft = \sim(\delta(w_{m+1}, W_v))$. Se o pacote de dados p_x não foi alterado, marca-o como aprovado para ser repassado, caso contrário, marca-o como desaprovado e deve ser descartado. No Trecho de Código 8, apresenta-se o procedimento de verificação de autenticidade de um pacote $p_x \in P$.

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

```

/*  $w_m$  executes this Algorithm to check if the content of
   a subset of packets  $P' \subset P$  was modified. It marks
   each  $p_x \in P'$  to be relayed or discarded.  $w_m$  uses the
    $s_a$  public key to decrypt the  $p_x$  signature and compares
   it to the hash value of the  $p_x$  content. It marks  $p_x$  to
   be relayed if  $p_x$  content was not modified, otherwise it
   marks  $p_x$  to be discarded, because  $p_x$  was modified by a
   node in  $W_v^\triangleleft = \sim(\delta(w_{m+1}, W_v))$ . */
1 verifiedPackets  $\leftarrow$  array of boolean;
2 foreach  $p_x \in P$  do
3    $signature \leftarrow getPacketFieldValue(p_x, 'signature');$ 
4    $data \leftarrow getPacketFieldValue(p_x, 'data');$ 
5    $verifiedPackets[x] \leftarrow (\text{hash}(data) = \text{decrypt}(K_{s_a}^+, signature));$ 
6 end
7 return  $verifiedPackets$ ;

```

4.6.3 Habilitar / desabilitar validação dos pacotes $p_x \in P$

A função de verificação de autenticidade de um fluxo de dados P do GMTP é opcional e desabilitada por padrão. Isto porque um sistema de transmissão, em execução na camada de aplicação, pode ou não desejar tal função. Por isso, considera-se que apenas o nó s_a tem o controle de habilitar tal funcionalidade, e este procedimento requer sinalizar os nós w_m para que estes executem o procedimento de verificação de autenticidade descrito na Seção 4.6.2. Para isto, o nó s_a ativa a opção assinado (*signed*), disponível no pacote de dados *GMTP-Register-Reply*, sinalizando que todos os pacotes de dados $p_x \in P$ conterá a assinatura da porção de dados sendo transportados naquele pacote de dados, podendo ser verificado pelos nós $w_m \in W_v$, desde que $\varphi(w_m, P) = 1$.

Note que quando um nó $c_f \in C_i(r_d)$ solicitar um fluxo de dados P , em resposta a tal pedido, o nó r_d retornará um pacote do tipo *GMTP-Request-Notify*. No cabeçalho desse pacote, o nó r_d deve também ativar a opção assinado (*signed*) para que o nó c_f seja notificado

e entenda que seu nó r_d realizará a verificação de autenticidade do fluxo de dados P da forma descrita anteriormente na Seção 4.6.2. Este procedimento permitirá que a aplicação em execução no nó c_f possa informar ao usuário final que tal funcionalidade está habilitada, por exemplo.

Além disso, como parâmetros de configuração, o usuário administrador do nó r_d pode habilitar ou desabilitar a opção de verificação de autenticidade dos fluxos de dados P , mesmo que o nó s_a possibilite tal verificação, como descrito anteriormente. Por fornecer essa função de verificação da porção de dados transportado em um pacote, no GMTP não realiza-se checagem de erro por soma de verificação (checksum), tradicionalmente utilizado 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 4.3.2.

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

4.7 Outras considerações

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

4.7.1 Canais de Comunicação

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

Canal de Controle

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

A decisão do uso do endereço IP multicast 238.255.255.250 foi baseada na RFCs 2365 [215], que define o escopo administrativo do uso dos endereços multicast entre 239.0.0.0 e 239.255.255.255. O endereço 238.255.255.250 é definido no escopo de uso global e sua alocação deve ser confirmada pela IANA antes do uso massivo do GMTP na Internet.

Canal de Transmissão Unicast

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

Do ponto de vista de roteamento, todo nó repassador deve avaliar os datagramas GMTP e realizar as ações apropriadas, definidas nas próximas seções deste capítulo. Por exemplo, no processo de estabelecimento de conexão, a ser detalhado na Seção 4.4.2, ao processar um pacote GMTP transmitido por um nó cliente, o nó repassador deve verificar se o pacote

é do tipo *GMTP-Request* e, em caso positivo, deve-se retornar um pacote do tipo *GMTP-Response* ao nó cliente, se o fluxo de dados de interesse do nó cliente especificado no pacote *GMTP-Request* já estiver sendo recebido por tal nó repassador.

Canal de Repasse Multicast

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

O *socket de repasse multicast* deve ser criado quando um nó repassador começa a receber um determinado fluxo de dados correspondente a um evento de interesse de pelo menos um dos seus clientes. Na prática, quando isto acontece, o repassador deve criar um socket multicast em um endereço IP e número de porta escolhida aleatoriamente na faixa de endereços IP de escopo local 239.192.0.0/14, definida na RFC 2365 [215]. 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 interim, 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 interim, o nó l_w em processo de desconexão deve continuar enviando pacotes do tipo *GMTP-Ack* para o nó r_d .

Desconexão de um nó r_d

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

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

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

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

4.8 Sumário do Capítulo

Neste capítulo, apresentou-se os fundamentos do *Global Media Transmission Protocol* (GMTP), um protocolo de transporte e rede baseado em uma arquitetura híbrida P2P/CDN para distribuição de fluxos de dados multimídia ao vivo. Tal arquitetura é caracterizadas por um conjunto de nós servidores que obtém o conteúdo multimídia da fonte geradora e o transmite para muitos nós receptores ($1 \rightarrow n$). O GMTP foi proposto para operar principalmente na Internet, permitindo a transmissão de pacotes de dados com suporte a controle de congestionamento sem garantia de entrega, tudo ocorrendo de forma transparente para a aplicação. O GMTP opera na camada de transporte e rede da pilha de protocolos GMTP, realizando transmissão em modo multicast ou de múltiplos fluxos unicast compartilhados entre os nós participantes da transmissão. Neste segundo caso, tal ação ocorre através de uma rede de favores constituída dinamicamente entre os roteadores da rede, evitando a relação de uma

conexão por cliente ao nó servidor.

Ao contrário de todos os outros protocolos de transporte e das soluções de aplicação para redes P2P, o foco de definição do GMTP foi reduzir responsabilidade dos nós clientes e aumentar a responsabilidade dos roteadores de rede no processo para distribuição de um determinado conteúdo multimídia. Este foco teve como principal motivação a proposta das Redes Centradas no Conteúdo (CCN), onde o roteador passa a ter um papel com maior participação no processo de entrega de um conteúdo para os nós interessados. Com vistas nos aspectos da CCN, o GMTP oferece um mecanismo de conexão separado em duas fases, quando se decide a forma como um determinado nó cliente obterá o conteúdo de interesse, contando com o suporte dos roteadores nesse processo. Nesse interim, uma grande peculiaridade do GMTP é a função que os nós roteadores passam a ter de realizar parcerias entre si a fim de obter um determinado conteúdo multimídia de interesse, identificado por um nome, como especificado pela teoria das redes centradas no conteúdo.

Diversas estratégias adotadas no GMTP e apresentadas neste capítulo discutidas são diferenciais que permitem a disseminação mais rapidamente de um determinado fluxo de dados originado em um nó servidor. Incorporou-se um mecanismo de *registro de participação* que, após um nó repassador se registrar em um nó servidor, permite-se que os servidores determinem quais são os candidatos a parceiros de um nó repassador, o que ocorre periodicamente. A vantagem é que, *a priori*, permite-se que os nós repassadores avaliem seus parceiros sem necessariamente um nó estar recebendo um fluxo de dados de um determinado evento. Com isto, um nó repassador pode repassar um fluxo de dados para um outro nó repassador sem que o primeiro tenha interesse no referido fluxo, mas devido ao seu posicionamento na rede e sua capacidade computacional e de vazão, pode melhorar o processo de disseminação de um determinado fluxo de dados. Além disso, como se trata de uma rede de favores e os dados são trocados de forma distribuída, ou seja, nem sempre com a participação de um nó servidor, pode-se empregar um mecanismo para validação dos dados transmitidos pelo servidor, evitando-se ataques de poluição, por exemplo.

No GMTP, os responsáveis por formar as parcerias P2P são os nós repassadores e não mais os nós clientes, como em soluções tradicionais de distribuição de conteúdo P2P. Como consequência, melhora-se o desempenho das transmissões de conteúdos multimídia ao vivo, pois o GMTP não é influenciado por fatores que impactam negativamente no funcionamento

da rede P2P, tais como a capacidade de processamento, armazenamento (memória), mobilidade e dinâmica de conexão/desconexão (*churn*) dos nós clientes. Esses dois últimos fatores são mais críticos se comparados aos demais, principalmente com a popularização dos dispositivos móveis e usar esse tipo de cliente para compartilhar seus recursos em uma rede P2P não é apropriado.

Um aspecto importante do GMTP são seus dois algoritmos para controle de congestionamento de fluxos de dados sem garantia de entrega, o GMTP-UCC e o GMTP-MCC. No primeiro, a ser aplicado na transmissão de fluxos de dados unicast entre os nós roteadores, emprega-se uma solução para controle de congestionamento assistido pela rede, onde oferta-se para cada fluxo de dados uma taxa de transmissão igual para todos os fluxos passando por todos os roteadores de um caminho. Nesse caso, a taxa de transmissão é determinada de acordo com a capacidade de transmissão do menor roteador em uma determinada rota. Já no segundo algoritmo, a ser aplicado em fluxos de dados multicast, utiliza-se um algoritmo de controle de congestionamento baseado na equação TFRC (*TCP Friend Rate Control*), fazendo-se uso de nós especiais chamados de relatores para determinar a próxima taxa de transmissão que o roteador deverá utilizar para distribuir o conteúdo multimídia para os nós clientes diretamente conectados a ele.

Por fim, discutiu-se sobre outras funcionalidades do protocolo GMTP, tais como seu mecanismo para finalização de conexão dos tipos de nós do GMTP, eleição de nós relatores e considerações sobre segurança. No próximo capítulo, apresentam-se os resultados e discussões acerca do uso do protocolo GMTP para a distribuição de conteúdos multimídia ao vivo.

Capítulo 5

Análise de Desempenho do GMTP

LINKAR TUDO APRESENTADO COMO RESULTADO À UM DESES 3 GRUPOS:

Olhar os gráficos desse artigo

ver ref 6386696

- Mostrar resultados que não justificam baixar chunks de múltiplos nós, mostrando que a contagem de duplicações foi alta demais, o que resulta em descartes no roteador e portanto, consumo desnecessário dos canais de transmissão. A melhor parceria estará sempre no caminho até o servidor, caso contrário, pega do servidor, o que alimenta todo o caminho e os demais pedidos de conexão se penduram no caminho. Se não houver um caminho alimentado, uma nova conexão no servidor e tudo se repete.

- Avaliar a quantidade de parcerias (fazer vários subgrafos mostrando a qualidade do vídeo, pela quantidade de nós, variando-se o número de parcerias usando *GMTP-RelayQuery*)

- MPEG-2, 1MB, 8s de vídeo

- Resultados * Comentar gráficos * Lembrar de falar sobre redes móveis * Sobre churn (roteador pode reduzir churn) * Quantidade de fluxos em transito

- Colocar uma seção de preâmbulo, explicando cenário e parâmetros

chunk loss ratio, chunk delivery ratio , continuity index , chunk delivery delay , end to end delay

contagem de interceptações (isso significa uma conexão de deixou de ser estabelecida no servidor, ou seja, menos um fluxo de dados repetido na rede)

denacast: in denacast every node select chunk from neighbors that have them. and it consider balancing policy, it select chunk from neighbor that have more upload bandwidth

thanks for your answer. this works for neighbor selection but , what about chunk selection? i mean is there any priority based policy to download some particular chunks sooner than the others? for example (rarest first policy or something else) ? no , in denacast this policy (rarest)has not benn consider

Current version of denacast is a mesh topology P2P video streaming. But what is your idea about tree topology. Is it a distributed tree or local tree? For creation of distributed tree topology you must change the overlay structure of denacast. But for local or push-pull based you can do it in the application layer. Regards,

=====

A Survey of Peer-to-Peer Simulators and Simulation Technology

- Fazer gráficos * Gráfico que mostra o consumo de memória de um repass, dependendo da qualidade da mídia. Será que dar pra fazer o de consumo de processamento? * Avaliar o uso de memória para cache de playback, colocar uma tabela os principais formatos e o tempo gasto * Usar baseline * Usar Coolstreaming / Denacast * Startup time * Distorção * Restart time * Quantidade de nós * Lembrar do churn e da heterogeneidade * Além destas, duas outras métricas são discutidas em uma perspectiva mais geral: a escalabilidade da quantidade de nós em uma transmissão e o impacto do *churn* nos sistemas de transmissão de conteúdos ao vivo.

=====

e de um proeminente protocolo de distribuição de conteúdos multimídia ao vivo, o Denacast [199], uma versão estendida do CoolStreaming [199, 199] com suporte a uma rede híbrida P2P/CDN, detalhado no Capítulo 3

Devido a complexidade para realizar experimentos reais dos cenários de aplicação, constituiu-se os resultados com base em um ambiente de simulação que permite determinar as configurações e avaliar os limites e os impactos dos recursos propostos com relação às métricas de estudo estabelecidas.

=====

avaliar a capacidade do protocolo em lidar com o *churn* dos nós e o impacto disso sobre a qualidade de experiência Além disso, avaliar o impacto do comportamento de participação dos nós sob outros nós na rede, como por exemplo, o impacto sob a qualidade do conteúdo multimídia recebido por um nó quando há uma falha em um nó de repasse, ou ainda, qual é o

máximo atraso aceitável sem que o usuário perceba interrupções na reprodução do conteúdo multimídia.

=====

Neste capítulo, apresentam-se os resultados preliminares do uso do protocolo GMTP para a transmissão de fluxos de dados multimídia em cenários com muitos nós receptores. Os resultados estão organizados de acordo com três métricas avaliadas, a escalabilidade quanto ao número de nós clientes interessados por um mesmo fluxo de mídia, a taxa de transmissão e o atraso. Essas métricas foram escolhidas segundo as sugestões de avaliação de protocolos apresentadas no documento *Metrics for the Evaluation of Congestion Control Mechanisms* (RFC 5166) [216].

5.1 Análise da Escalabilidade do Número de Clientes

Em se tratando da redução da quantidade de conexões simultâneas ao servidor por parte dos clientes, este tem sido um aspecto fundamental do GMTP. Isto porque ao utilizar-se do GMTP, constata-se uma redução significativa na quantidade de fluxos de dados no servidor se comparado ao protocolo DCCP.

Observando-se o gráfico na Figura 5.1, nota-se que no nível 9 da topologia de rede utilizada, fazem-se necessárias 10230 conexões simultâneas dos clientes ao servidor. Seguindo-se esta constatação, observa-se que ao utilizar o protocolo GMTP são realizadas, no máximo 1023 conexões (pior caso). O pior caso acontece se cada nó localizado em cada rede local distinta não encontrar um nó relay, sendo necessário um cliente por rede local estabelecer uma conexão direta com o servidor. Este pior caso é muito difícil de acontecer, pois o mecanismo de busca por nós relays empregado no GMTP cuidará de encontrá-los, evitando-se conexões desnecessárias no servidor, exceto se o nó relay estiver 5 saltos de distância do cliente interessado pelo fluxo de dados.

Nas simulações de rede executadas, avaliou-se a evolução do protocolo GMTP quanto à disponibilidade de nós relays e o uso destes por parte dos clientes. No caso da topologia de rede utilizada nas simulações, constatou-se apenas duas conexões ao nó servidor e em cada nó relay (melhor caso). Isto porque a quantidade de níveis da árvore da topologia de rede foi pequena (apenas 10 níveis), então os clientes localizados em níveis mais extremos

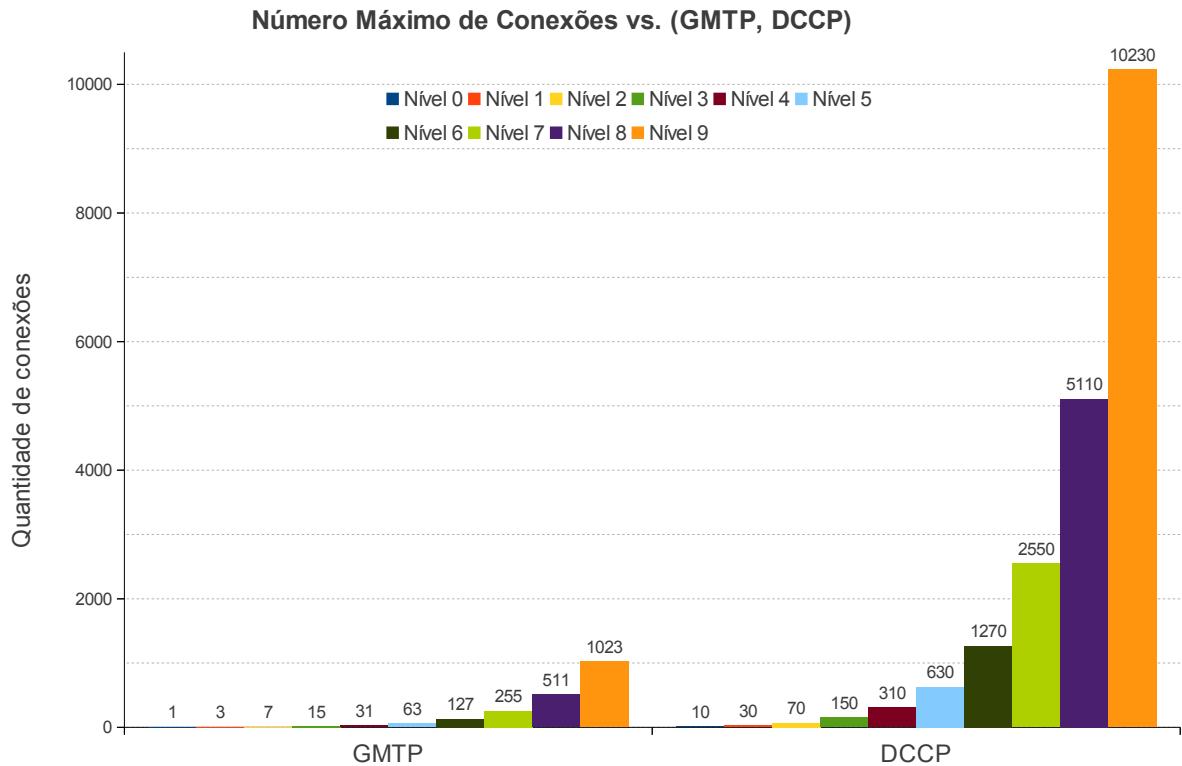


Figura 5.1: Gráfico da quantidade de conexões simultâneas ao servidor com o uso dos protocolos GMTP e o DCCP.

da topologia da rede não alcançou o limiar de atraso necessário para ativar o mecanismo empregado no GMTP de conexão direta ao servidor, sem o uso de nós relays. Este aspecto do GMTP foi discutido no final da Seção ???. Neste contexto, faz-se necessário mais estudos de simulações a fim de atingir o limiar de atraso configurado pela aplicação e então avaliar o comportamento do GMTP no tocante à quantidade de conexões simultâneas que serão necessárias ao nó servidor caso aconteça esta situação. Mais adiante na Seção 5.3 serão discutidos outros resultados relacionados ao atraso com o uso do GMTP.

Dando continuidade às discussões acerca da escalabilidade do GMTP no quesito de número de clientes interessados pelo mesmo fluxo de dados, no gráfico da Figura 5.2, observa-se um comparativo do comportamento do uso do GMTP e a taxa de transmissão média obtida por cada nó cliente à medida que aumentou-se o número de clientes (níveis 0-9). Constata-se claramente que quando se compara o comportamento do GMTP e do DCCP, com o uso do GMTP, melhora-se sobremaneira a taxa efetiva de recepção de dados por parte dos nós clientes, ao passo que a quantidade de perda de pacotes de dados por parte do DCCP aumenta exponencialmente. Observa-se que mesmo aumentando a quantidade de nós clientes, por

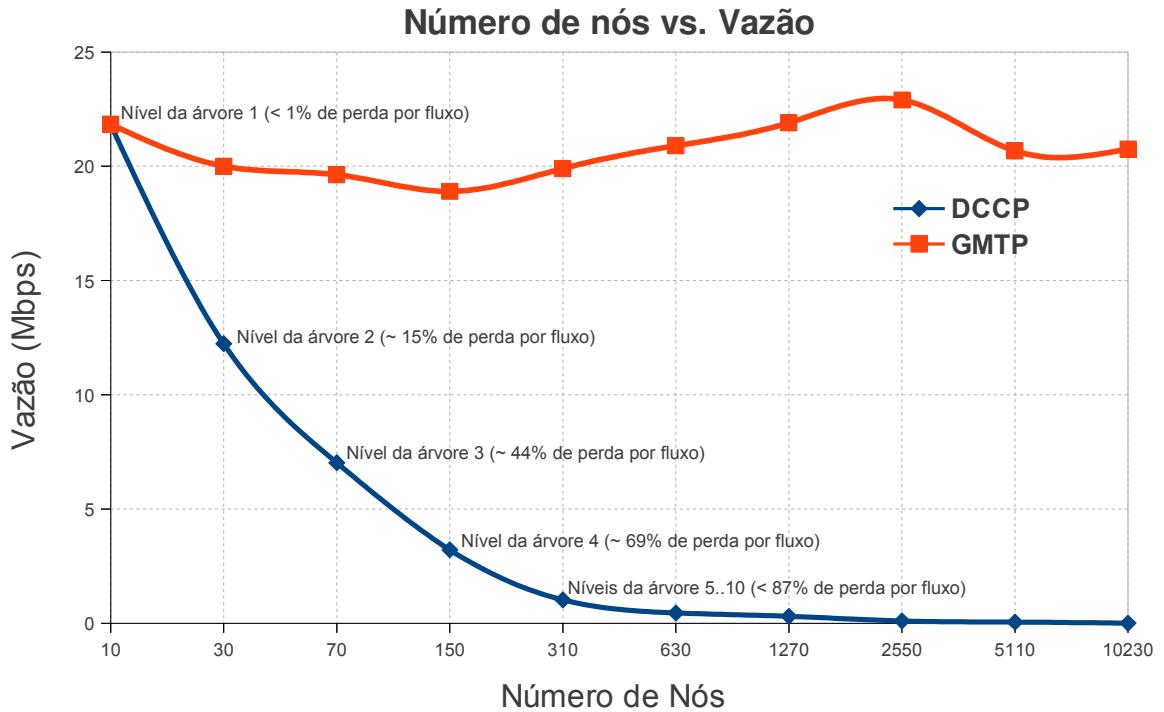


Figura 5.2: Gráfico da taxa média de recepção e perda de pacotes pela quantidade de nós clientes entre o nível 1 (10 clientes) e o nível 9 (10230 clientes), utilizando-se os protocolos GMTP e DCCP.

exemplo, entre 150 e 2550, a taxa de transmissão média continuou sendo satisfatória, sendo crescente à medida que aumentou-se a quantidade de nós clientes. Isso é justificado pelo fato de que quanto mais nós GMTP na rede, aumentam-se as chances de mais nós clientes se tornaram relays, o que consequentemente aumentam as chances de existirem mais clientes recebendo fluxos de dados em modo multicast e não somente em modo unicast.

5.2 Análise da Taxa de Transmissão

O estudo da métrica *taxa de transmissão* ocorreu através da análise do comportamento dos protocolos GMTP, DCCP e TCP para a transmissão de fluxos de dados multimídia no cenário apresentado no Capítulo ??.

No gráfico apresentado na Figura 5.3, observa-se o comportamento dos protocolos estudados quando utilizados em uma transmissão de vídeo, segundo as considerações estabelecidas na Seção ?? . No eixo das abscissas representa-se o tempo de simulação em segundos, ao

passo que no eixo das ordenadas representa-se a taxa de transmissão obtida a cada segundo. Note que os valores plotados no gráfico refletem a taxa de recepção para um dos ensaios efetuados do Tratamento 1. A quantidade de repetições realizadas para cada protocolo, assim como um sumário dos valores de todas as métricas estudadas são apresentadas na Seção 5.4.

De acordo com o gráfico, observa-se um desempenho superior do protocolo GMTP com relação à taxa de recepção obtida pelos nós clientes, comparando-o com o TCP e com o DCCP. Percebe-se que o TCP e o DCCP tiveram comportamentos semelhantes, com uma leve vantagem para o protocolo DCCP. Note que o fluxo de dados transmitido pelo GMTP se mantém de forma estável, entre $18\ Mbps$ e $22\ Mbps$. A principal explicação para o desempenho satisfatório do GMTP com relação a taxa de transmissão é porque tal protocolo faz uso do compartilhamento de conexões para obtenção do conteúdo multimídia, obtendo-se o conteúdo através de nós relays, que repassa para seus nós próximos os pacotes de dados em modo multicast, sempre que possível.

Além disso, o GMTP faz uso de um algoritmo para controle de congestionamento híbrido, tratando-se de formas distintas a forma como os recursos de rede são consumidos *intra* e *inter* redes. Consequentemente, utilizando-se de forma mais eficiente os canais dos *backbones* com alta disponibilidade de largura de banda (os canais de $1\ Gbps$ da topologia de rede utilizada), ao passo que minimiza o congestionamento nas redes locais, reduzindo-se significativamente a quantidade de conexões simultâneas no servidor ao utilizar o modo de transmissão multicast.

5.3 Análise do Atraso Fim-a-Fim

Após uma avaliação do comportamento do GMTP com relação a métrica taxa de transmissão e escalabilidade do protocolo, apresenta-se agora discussões acerca da métrica *atraso*. No gráfico ilustrado na Figura 5.4, apresenta-se a evolução do atraso com relação aos protocolos GMTP e o DCCP, considerando-se a topologia de rede estudada. No eixo das abscissas, representa-se a distância entre um grupo de clientes (os níveis da árvore da topologia de rede utilizada) até o servidor localizado na raiz da árvore, ao passo que no eixo das ordenadas representa-se a média do atraso de recepção observado em cada nó GMTP.

Note que, à medida que a distância em saltos aumenta entre um nó cliente (níveis mais



Figura 5.3: Gráfico da taxa de recepção para 10230 clientes (nível 9), considerando-se o uso dos protocolos GMTP, DCCP e TCP em uma transmissões de vídeo.

altos da topologia de rede) e o servidor localizado na raíz, o atraso de recepção aumenta exponencialmente para cada cliente DCCP. Não observa-se este fato ao utilizar o protocolo GMTP, onde o aumento do atraso de recepção é linear e suave. De fato, isto ocorre por conta do uso da combinação do modo de transmissão unicast e multicast empregado no GMTP. Ao utilizar o protocolo DCCP, muitas conexões são abertas ao servidor, o que resulta em múltiplos fluxos com conteúdos iguais que são transmitidos na rede, fazendo-se mal uso do canal de transmissão e elevando-se o nível de congestionamento da rede. A consequência disso é uma inundação de pacotes de dados no roteador que, quando estes não os destartam, demoram demasiadamente para processá-los, o que aumenta o atraso fim-a-fim.

5.4 Compilação dos Resultados

Na Tabela 5.1, apresenta-se um sumário das métricas taxa de transmissão, quantidade de dados transmitidos e perdidos, atraso e o número de repetições dos tratamentos para os protocolos GMTP, DCCP e TCP. Considera-se importante salientar a quantidade de repetições que foram realizadas durante a execução dos tratamentos. Os valores apresentados na coluna

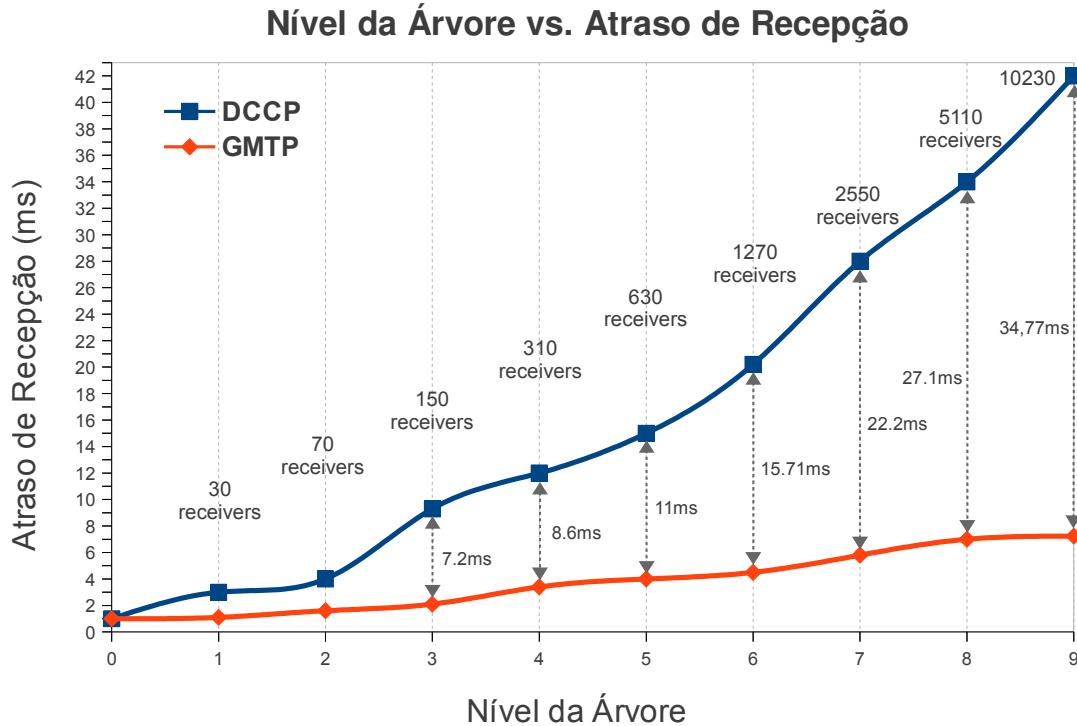


Figura 5.4: Gráfico do atraso médio de recepção de pacotes de dados pela quantidade de nós receptores, entre o nível 1 (10 clientes) e o nível 10 (10230 clientes) da topologia de rede estudada, considerando-se o uso dos protocolos GMTP e DCCP.

n da Tabela 5.1 expressa a quantidade de vezes que um determinado tratamento foi repetido, utilizando-se todo o suporte estatístico discutido na Seção ??.

De acordo com os valores apresentados na Tabela 5.1, constata-se que para as métricas avaliadas, o GMTP promove melhorias significativas se comparado ao uso de protocolos tradicionais, tais como o DCCP e o TCP em cenários de aplicações estudados neste trabalho. Observou-se melhorias da taxa de transmissão média obtida por um fluxo GMTP de quase 211 vezes mais, comparando-se ao DCCP e de quase 320 vezes mais comparando-se ao TCP. Além disso, obteve-se melhorias bastante expressivas com o uso do GMTP com relação às métricas de quantidade de dados efetivamente transmitidos e o atraso.

Considerando-se a taxa de transmissão média obtida por cada fluxo de dados GMTP, conseguiu-se transmitir, em média, 29103,8 MBytes, com perda apenas de 724 MBytes. Comparando-se o desempenho do GMTP aos protocolos DCCP e TCP, o resultado é bastante satisfatório. Com o uso do DCCP, transmitiu-se efetivamente apenas 87,62 MBytes, de 141,62 MBytes originados do servidor. Já com o uso do TCP, transmitiu-se apenas

28,15 MBytes, de 141,62 MBytes originados do servidor. Este resultado obtido com o uso do TCP é muito ruim porque tal protocolo implementa retransmissão de dados e a perda de dados causa a redução da taxa de transmissão e o aumento do atraso fim-a-fim devido ao acúmulo de dados que precisa ser retransmitido.

Já com relação ao atraso, observa-se um tempo de atraso muito menor do GMTP se comparado aos protocolos DCCP e TCP. Enquanto que a média de atraso entre pacotes (*jitter*) ao utilizar o GMTP foi de 7,23 ms, o atraso médio observado pelos nós DCCP e TCP foram 88,34 ms e 288,64 ms, respectivamente.

Tabela 5.1: Sumário da taxa de transmissão, quantidade de dados transmitidos e perdidos e o atraso para os protocolos GMTP, DCCP e TCP.

Protocolos	Taxa de Trans. (Mbps)	Transmitido / Perdido (MBytes)	Atraso (ms)	Repetições (n)
GMTP	20,43 (18,96 – 21,9)	29103,8 (29707,6 – 29948)	7,23 (6,39 – 8,07)	32
DCCP	0,097 (0 – 0,12)	87,62 (77,48 – 97,76)	88,34 (86,5 – 90,18)	29
TCP	0,064 (0 – 0,084)	28,15 (18,84 – 37,46)	288,64 (287,39 – 289,89)	37

5.5 Sumário do Capítulo

Neste capítulo, apresentou-se os resultados preliminares do uso do protocolo GMTP em um cenário de transmissão de vídeo com um único nó transmissão e milhares de nós receptores.

As métricas estudadas foram a taxa de transmissão, o atraso e a escalabilidade quanto ao número de nós clientes interessados por um mesmo fluxo de mídia. Os resultados foram apresentados através de gráficos e as discussões foram feitas a fim de prover ao leitor e desenvolvedores de aplicações razões que justificam o uso do GMTP em suas aplicações.

Durante todo o processo de discussão acerca dos resultados obtidos, comparou-se o desempenho do GMTP em relação aos protocolos DCCP e o TCP, onde constatou-se melhorias significativas para as métricas estudadas do uso do protocolo GMTP em transmissões de con-

teúdos multimídia para 10230 nós receptores.

De acordo com os resultados apresentados ao longe deste capítulo e discussão apresentada na Seção 5.4, constatou-se que o desempenho do GMTP foi显著mente superior em todas as métricas estudadas se comparado aos outros protocolos analisados.

Capítulo 6

Considerações Finais

Futuro: - if the video is encoded with multiple bit rate (MBR) streams [8] or scalable stream.

Citar o MDC ?? Multiple Description ???? - adaptação de fluxo por descarte de pacotes que carregam quadros B - ConEx - Analisar melhor o per-flow state para calcular o RTT - Estudar uma nova equação para o GMTP-MCC (Baseada em Cubic)? - Avaliar com o ConEx??? - Terminar a implementação no kernel do Linux - RFC - Estudar a possibilidade de determinar padrões de parcerias (troca de tráfego, disponibilidade por dia/hora, usar parceiros ociosos etc.) - Distribuição de registros de participação entre os nós da rede CDN - Um ataque pode gerar pacotes errados de um fluxo P e parar a recepção. Tem que pensar em um mecanismo de blacklist e reputação. Limitações: - Repetição de UUID - Muitas msgs no canal de controle - Dizer que os estudos realizados na formação de redes P2P podem servir para pesquisas/trabalhos futuros - Futuro: nem sempre colocar os ids de todos os roteadores no pacote register, pode-se pensar em algo otimizado (apesar de que um roteador pode fazer parte de caminhos diferentes, portanto, pode estar sempre em posições diferentes, que isso a necessidade de sempre colocar sua posição no pacote register. Isso pode ser melhorado?). - Futuro: Trabalho futuro vanets - Futuro: aplicar o ALTO?

Os sistemas para transmissão de mídias ao vivo vêm se tornando uma importante solução para disseminação da informação e entretenimento pela Internet. Esses sistemas fazem parte de uma categoria de aplicações chamada de IPTV e têm atraído a atenção dos usuários porque permitem o livre acesso a conteúdos antes possíveis apenas através dos sistemas tradicionais de TV, que exigem altos custos de manutenção principalmente do usuário final.

Para permitir o funcionamento desses sistemas, utilizam-se basicamente três métodos

de transmissão. No primeiro utiliza-se o modelo tradicional cliente-servidor, hoje em dia escasso devido às limitações de suportar quantidades significativas de nós receptores em uma transmissão. No segundo método, utiliza-se redes de distribuição de conteúdo, que tem altos custos operacionais e por conseguinte apenas grandes empresas as utilizam. E o terceiro método e mais promissor, o uso de sistemas cooperativos P2P ou P2-IPTV.

Nos sistemas P2-IPTV, os nós interessados por uma transmissão cooperam no esforço para disseminar o conteúdo multimídia entre si através de uma rede de sobreposição. Tal método tem chamado atenção dos pesquisadores e provedores de serviços, permitindo-se a transmissão de mídias ao vivo em larga escala sem sobrecarregar as fontes geradoras e seus canais de transmissão. Isto porque otimiza-se o uso de recursos computacionais já que não é necessário alocar banda de rede (primeiro método) e evita-se ter que distribuir e aumentar o número de nós replicadores (segundo método) proporcionalmente ao número de nós interessados por uma determinada transmissão.

Apesar da existência de sistemas que exploram eficientemente o cooperativismo entre os nós participantes de uma transmissão, tais sistemas continuam à mercê de protocolos de transportes tradicionais (TCP, UDP, DCCP e SCTP), que não foram projetados para transmitir dados multimídia em larga escala.

6.1 Conclusões

6.1.1 Benefícios e Aplicabilidade

Nesta seção, apresentam-se os benefícios e justificativas para diversas tomadas de decisões realizadas no processo de especificação do GMTP.

O uso do GMTP nas aplicações de distribuição de mídia ao vivo fomenta benefícios em três vertentes: para a rede, para o desenvolvedor da aplicação e para os usuários interessados em reproduzir uma mídia ao vivo.

Considerando-se as discussões realizadas nas Seção 1.2 e os requisitos dos sistemas de transmissão de mídia em tempo real, o GMTP atende-os na medida em que:

- FALAR DA TORRE DA OPERADORA DE CELULAR E MOBILIDADE

- possibilita que os sistemas de transmissão de mídia ao vivo obtenham um melhor de-

sempenho quanto a escalabilidade do número de usuários e possivelmente na qualidade da experiência do usuário (*Quality of Experience* – QoE). Nesse aspecto, o GMTP é capaz de identificar a melhor forma que o conteúdo será transportado pela rede, permitindo-se o uso do modo de transmissão multicast sempre que possível ou de múltiplos fluxos unicast caso contrário, mas não proporcional ao número de clientes interessados;

- permite que as aplicações façam uso de soluções estáveis e largamente testadas, uma vez adicionadas ao protocolo em questão. Neste caso, diferentes equipes de desenvolvimento podem adicionar e testar suas propostas em um protocolo de rede que, uma vez consideradas estáveis podem ser compartilhadas entre os diferentes sistemas. Até mesmo soluções já implementadas nos sistemas existentes podem ser portadas para o GMTP;
- padroniza a forma como os fluxos de dados multimídia gerados pelos sistemas considerados são transportados na Internet, incluindo aspectos de controle de congestionamento e compartilhamento desses fluxos de dados entre os nós participantes de uma transmissão.
- indiretamente diminui os fluxos de dados na rede sem qualquer controle de congestionamento, pois aplicações para distribuição de mídia ao vivo atualmente fazem uso do protocolo UDP ou variantes;
- flexibilidade no acesso a rede por parte dos nós participantes, pois eles podem entrar e sair da rede a qualquer momento, realizando parcerias com um subconjunto de nós participantes a fim de receber o conteúdo multimídia interessado. Neste caso específico, a estratégia de distribuição empregada no referido protocolo naturalmente reduz o impacto do churn, uma vez que os nós clientes passam a ter uma participação menos importante no processo de distribuição de uma mídia;
- elimina ou pelo menos inibe a presença de nós *free-riders* na rede, uma vez que todo nó GMTP é obrigado a compartilhar a mídia a partir do momento que começa recebê-la;
- disponibiliza uma solução unificada que permite o uso da API padrão de *sockets* BSD, o que facilita e acelera a adaptação de aplicações existentes para usar o GMTP. Além

disso, permite-se que as aplicações continuem utilizando outros padrões de redes definidos pela IETF, tais como o RTP;

- unifica as aplicações clientes no ponto de vista do processo de conexão e obtenção dos dados, uma vez que todo esse processo ocorre na camada de transporte sem qualquer influência da aplicação. Isto significa que se existir um nó executando um cliente do sistema PPLive e um outro cliente do sistema SopCast, portanto desenvolvido por equipes diferentes, ambos ainda sim serão compatíveis e capazes de cooperar entre si na obtenção do fluxo de mídia de interesse comum.

De uma forma geral, dentre os diversos sistemas P2P para transmissão de mídia ao vivo que podem se beneficiar com o uso do GMTP, destacam-se os baseados em uma arquitetura em malha e sem organização rígida dos nós participantes do sistema. Isto também se aplica a todas as variantes dessa arquitetura como, por exemplo, híbrido por encaminhamento automático e pedido explícito (*Push-Pull*) e híbrido árvore-malha (vide Capítulo 2). Na perspectiva das aplicações de rede para distribuição de conteúdos multimídia mais conhecidas, a grande maioria pode se beneficiar com o uso do GMTP , especificamente as seguintes aplicações: Coolstreaming ??, LiveSky, Sopcast [98], o PPLive [100] e o GridMedia [180,187].

6.2 Perguntas Frequentes

, principalmente no tocante aos frequentes questionamentos realizados durante as palestras sobre o GMTP ministradas pelo autor

- Por que um protocolo de rede, invés de um middleware? 1 - porque visualizei o GMTP como a forma primitiva de acesso, podendo ser acomplhado em aplicações simples como sistemas mais avançados, que podem fazer uso de middlewares já existentes 2 - a necessidade de executar algoritmos na camada de rede obriga uma interface de comunicação disponível na camada de transporte

- Como seria a integração com o ConEx? Por que não foi utilizado?

- Pode ocorrer a inundação de um roteador próximo do servidor? Não, pois quanto mais conexões forem enviadas para s, mais roteadores distantes de s interceptará o pedido de conexão

- Por que utilizar roteadores?
- Qual a diferença entre NAT e o processo de conexão do GMTP? Em conexões NAT, os fluxos não são compartilhados (o roteador não conhece o conteúdo, mas sim o host). Em NAT ocorre apenas tradução de nomes, com o GMTP é muito mais do que isso (controle de congestionamento e o compartilhamento do fluxo), além da parte que interceptação de requisições para obter P entre roteadores. Falar do canal multicast, o NAT não faz isso

Estou propondo o que posso chamar de um NAT distribuído pelo nome do fluxo, (ou seja, porém com conhecimento do conteúdo)? Distribuído porque todos os repassadores interceptam, pedem e repassam.

Não tem nada em todos os roteadores

- Relacione os conceitos ICN, CCN, NDN ao protocolo GMTP? (pode ser que seja melhor isso aparecer nos trabalhos relacionados)

GMTP não precisa alterar a app, ela continua pedindo via ip + porta

GMTP é específico para streaming, CCN é mais para propósito geral (pensar se falar isso)

Como no GMTP a informação de endereçamento ainda existe, roteadores que não suportam GMTP não terão problema de rotear os pacotes. Isso não é verdade para o CCNx.

As parcerias no GMTP é feita por intersecções de caminho, no CCNx não é assim (ver como é! ou se faz).

O GMTP não precisa que todos os roteadores rodem o gmtp-inter.

Olher esse artigo: CCN Naming Scheme for Scalable Routing and Incremental Deployment stamped

Md. F. Bari, S.R. Chowdhury, R. Ahmed, and B. Mathieu, A Survey of Naming and Routing in Information-Centric Networks, IEEE Communications Magazine, pp. 44-53, Dec. 2012. [2] V. Jacobson, D.K. Smetters, J.D. Thornton, M. Plass, N. Briggs, and R. Braynard, Networking Named Content, ACM CoNext, pp. 1-12, Dec. 2009. [3] Yuan, Haowei, and Patrick Crowley, Experimental Evaluation of Content Distribution with NDN and HTTP. Proceedings of IEEE INFOCOM 2013 Mini-Conference, Apr. 2013.

Artigos no desktop

- Qual o impacto do desempenho em um roteador? Falar do Mikrotik, 36 portas, cada uma com um processador dedicado de 1.5Ghz e 16GB de memória.

- O mal funcionamento (intencional) de um nó repassador, pode quebrar toda a lógica do GMTP para formação de parceria? Sim, da mesma forma que o mal funcionamento do roteador pode quebrar o funcionamento do TCP, UDP, HTTP etc, basta para isso, tal roteador passar a desencapsular os cabeçalhos de tais pacotes e alterar seus cabeçalhos.
- Qual o processo ideal para o deploy do GMTP na Internet?
- Por que gerar um id (Streaming Identifier) se IP e Porta já faz isso?
- Qual é a complexidade para adaptar os protocolos existentes de transporte usarem o GMTP?
 - Quais são as similares e diferenças entre o GMTP e o LibSwift?
 - O GMTP faz computação Per-Flow, se sim, quanto?
 - Por que não usar diretamente o traceroute? e o tcptraceroute?
 - Por que não manter um único RTT s? R: porque todos os acks teriam que ser repassados para o nó originador do fluxo de dados, o que geraria uma explosão de acks e além disso, qual valor deveria ser considerado, já que um fluxo transmitido pelo servidor pode ser recebido indiretamente por muitos outros nós? Além disso, isto limitaria a taxa de transmissão $R(t)$ entre um sub-caminho
- O GMTP Inter pode ser executado em um cliente?
 - Em um mesmo domínio administrativo, contendo múltiplos roteadores, sendo um de borda, é possível que ter apenas um nó repassador instalado em um roteador e os demais roteadores funcionarem como cliente/repassador?
 - Por que um cliente não pode ter mais de um repassador? R: o tráfego sempre passará pelo roteador padrão, onde tem o gmtp inter rodando
 - ... E considerando um rede com múltiplos roteadores? R: limitação do GMTP? tem que escolher um roteador como rota padrão
 - Por que usou a equação do reno? R: melhor seria usar cubic-like, interações com o pesquisador, contratado pela samsung, trabalhos futuros
 - Qual a vant de um r repassar?
 - tem que ler header de transporte? (pode colocar uma flag no campo opt do ip)
 - Por que tipos de pacotes, invez de flags
 - Por que não explorar o Anycast? Só serve mais para protocolos não orientados à conexão.

6.3 Trabalhos Futuros

Como discutido no Capítulo 1, o problema central abordado nesse trabalho é concernente à carência de soluções para dar suporte ao transporte eficiente de dados em sistemas de transmissão de mídias ao vivo baseados em arquiteturas P2P. As soluções existentes, detalhadas e comparadas ao GMTP no Capítulo 3, são paleativas porque são disseminadas em forma de sistemas ou protocolos de aplicação que, embora possam ser executadas com sucesso na Internet, é impossível evitar a pulverização e fragmentação das mesmas, aumentando-se a complexidade de implantação de uma solução unificada em larga escala na Internet.

Neste contexto, considerou-se primordial a definição de uma solução padronizada e de implantação em escala global, que estabeleça uma camada de software para extensão e compartilhamento dos principais componentes que caracterizam um sistemas de transmissão de mídias em tempo real. Em busca de alcançar esse objetivo, apresentou-se o *Global Media Transmission Protocol* (GMTP). Este protocolo é baseado em uma arquitetura P2P para distribuição de fluxos de dados multimídia com um nó transmissor e milhares de nós receptores espalhados na Internet.

Inicialmente, como detalhado nos Capítulos 4 e ??, apresentou-se um cenário geral de funcionamento do GMTP, considerando-se uma topologia de rede híbrida *mesh*-árvore. Ainda, apresentou-se uma visão geral do GMTP com destaque para a sua arquitetura, discutindo sua flexibilidade em compartilhar as boas práticas (algoritmos) entre diferentes aplicações. Além disso, deve-se enfatizar a abstração proporcionada pelo GMTP no que se refere a complexidade para construir uma infra-estrutura de comunicação entre os nós participantes de uma transmissão de mídias em redes P2P. Para tanto, permite-se a adição de novos mecanismos para descoberta e seleção de nós parceiros, tolerância à desconexão de nós e adaptação e controle de congestionamento.

De acordo com os resultados apresentados no Capítulo 5, seguindo as metodologias discutidas no Capítulo ??, constatou-se que o protocolo GMTP apresenta um desempenho satisfatório no que diz respeito ao uso eficiente dos recursos de rede. Neste contexto, discutiu-se sobre o desempenho do GMTP quanto às métricas de taxa de transmissão, atraso e escalabilidade do número de nós receptores, concluindo-se que o GMTP é capaz de compartilhar o canal de transmissão com outros fluxos de dados (TCP), ao passo que ainda mantém taxas

de transmissão em níveis aceitáveis para uma transmissão multimídia.

Considerando-se os objetivos deste trabalho e os resultados obtidos até o momento, tem-se o cenário descrito a seguir. Os principais elementos que compõem o protocolo GMTP estão definidos e implementados em um simulador de rede. A versão do GMTP implementada no simulador de rede permitiu uma avaliação acerca do desempenho do protocolo. Entretanto tal versão não contempla a proposta do arcabouço para a extensão do GMTP, que só será implementado na próxima versão a ser desenvolvida no núcleo do sistema operacional Linux.

Os trabalhos futuros estão organizados de acordo com as seguintes atividades:

1. **melhorias gerais na especificação do GMTP:** faz-se necessário definir um diagrama de estados e transições de tal protocolo, assim como na estrutura do cabeçalho dos pacotes definidos;
2. **especificar e implementar o arcabouço de extensão:** definir os pontos de extensão do protocolo, bem como a forma como novos algoritmos são adicionados, carregados e descarregados do protocolo. Isto inclui a definição das mensagens que devem ser trocadas entre os nós para notificação de mudanças ou pedido de execução de algum algoritmo específico;
3. **desenvolvimento de um protótipo executável:** implementar o protocolo GMTP no sistema operacional Linux de acordo com as especificações discutidas ao longo deste documento (capítulos 4 e ??)
4. **aplicação modelo:** implementar uma aplicação que utilize os recursos disponíveis no GMTP. Após avaliar a viabilidade, pretende-se também alterar uma implementação de algum sistema existente para utilizar o GMTP;
5. **experimentos para validação dos protótipos desenvolvidos:** executar experimentos para analisar o funcionamento do GMTP e da aplicação desenvolvidos;
6. **redação de *Internet-Drafts*:** redigir duas propostas de RFC sobre o GMTP e submetê-las para a IETF. A primeira proposta contemplará discussões acerca dos principais problemas encontrados em sistemas de transmissão multimídia em tempo real, bem como

as motivações para a disponibilização de um protocolo de transporte com suporte aos requisitos de tais sistemas. Já a segunda proposta de RFC contemplará a especificação do GMTP e seus detalhes de implementação, incluindo diagramas de sequência e pseudo-algoritmos. O título da primeira proposta será *Problem Statement for Global Media Transmission Protocol (GMTP)*, ao passo que o da segunda proposta será *Global Media Transmission Protocol (GMTP) - Protocol Specification*;

7. **redação da tese:** finalizar a escrita do documento adicionando os avanços alcançados após a defesa de qualificação. No documento deverão constar discussões sobre a definição e formalização do arcabouço de extensão do GMTP, bem como análises relacionadas à validação dos protótipos desenvolvidos, diagramas de sequência e pseudo-algoritmos para facilitar o entendimento dos procedimentos realizados no GMTP.

Na Tabela 6.1 apresenta-se o cronograma das atividades a serem desenvolvidas no período de 1 ano e 2 meses.

Tabela 6.1: Cronograma de Atividades.

Atividades	2012												2013	
	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez	Jan	Fev
Melhorias gerais do GMTP	X	X	X	X										
Definição do arcabouço de extensão				X	X									
Desenvolvimento de protótipo executável					X	X	X	X	X	X				
Desenvolvimento de aplicação modelo								X	X					
Redação de <i>Internet Draft</i>							X	X	X	X	X			
Experimentos p/ validar protótipos												X	X	
Redação da tese												X	X	X

Bibliografia

- [1] 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.
- [2] Yonghong Tian, J. Srivastava, Tiejun Huang, and N. Contractor. Social multimedia computing. *Computer*, 43(8):27–36, Aug.
- [3] N. Leavitt. Network-Usage Changes Push Internet Traffic to the Edge. *Computer*, 43(10):13 –15, 10 2010.
- [4] Cisco Systems. The zettabyte era. Technical report, Cisco Systems, 5 2012. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.pdf.
- [5] San Murugesan. Understanding web 2.0. *IT Professional*, 9(4):34–41, 2009.
- [6] Kwei-Jay Lin. Building web 2.0. *Computer*, 40(5):101–102, 5 2007.
- [7] 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.
- [8] 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: 1 de Fevereiro de 2014.

- [9] 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: 1 de Fevereiro de 2014.
- [10] Eric Franchi. Live streaming continues momentum with march madness, 3 2009. <http://www.mediapost.com/publications/article/101750/#axzz200qSKoNN>. Último acesso: 1 de Fevereiro de 2014.
- [11] Marshall Kirkpatrick. The numbers are in, live video online is blowing up, 3 2008. http://readwrite.com/2008/06/05/live_video_big. Último acesso: 1 de Fevereiro de 2014.
- [12] Liz Gannes. The obama inauguration live stream stats, 1 2009. <http://gigaom.com/2009/01/20/the-obama-inauguration-live-stream-stats/>. Último acesso: 1 de Fevereiro de 2014.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.

- [18] 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.
- [19] Jameson Berkow. Netflix Proclaimed “King” of North American Internet Use, 5 2011. http://business.financialpost.com/2011/05/17/netflix-named-king-of-north-american-internet/?__lsa=60e0-e253. Último acesso: 1 de Fevereiro de 2014.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] Jin Li. On peer-to-peer (p2p) content delivery. *Peer-to-Peer Networking and Applications*, 1(1):45–63, 2008.
- [24] 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, June.
- [25] 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.
- [26] Darshan Purandare and R. Guha. An Alliance Based Peering Scheme for P2P Live Media Streaming. *Multimedia, IEEE Transactions on*, 9(8):1633–1644, 2007.

- [27] A. Vakali and G. Pallis. Content delivery networks: status and trends. *Internet Computing, IEEE*, 7(6):68–74, nov.-dec. 2003.
- [28] 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.
- [29] 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.
- [30] Dongyan Xu, Sunil Suresh Kulkarni, Catherine Rosenberg, and Heung keung Chai. A cdn-p2p hybrid architecture for cost-effective streaming media distribution. *Computer Networks*, 44:353–382, 2004.
- [31] Dongyan Xu, SunilSuresh Kulkarni, Catherine Rosenberg, and Heung-Keung Chai. Analysis of a cdn-p2p hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11:383–399, 2006.
- [32] Melika Meskovic, Himzo Bajric, and Mladen Kos. Content delivery architectures for live video streaming: Hybrid cdn-p2p as the best option. In *The Fifth International Conference on Communication Theory, Reliability, and Quality of Service*, pages 26–32, 2012.
- [33] R. Buyya, A.-M.K. Pathan, J. Broberg, and Z. Tari. A case for peering of content delivery networks. *Distributed Systems Online, IEEE*, 7(10):3–3, Oct.
- [34] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Understanding hybrid cdn-p2p: why limelight needs its own red swoosh. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '08*, pages 75–80, New York, NY, USA, 2008. ACM.
- [35] Kideok Cho, Hakyung Jung, Munyoung Lee, Diko Ko, T.T. Kwon, and Yanghee Choi. How can an ISP merge with a CDN? *Communications Magazine, IEEE*, 49(10):156–162, Oct.

- [36] S. M Y Seyyedi and B. Akbari. Hybrid cdn-p2p architectures for live video streaming: Comparative study of connected and unconnected meshes. In *Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on*, pages 175–180, Feb.
- [37] Z. Chen, H. Yin, C. Lin, Y. Chen, and M. Feng. Towards a universal friendly peer-to-peer media streaming: metrics, analysis and explorations. *Communications, IET*, 3(12):1919–1933, December.
- [38] Xuening Liu, Hao Yin, and Chuang Lin. A novel and high-quality measurement study of commercial cdn-p2p live streaming. In *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, volume 3, pages 325–329, Jan.
- [39] Yee-Ting Li, D. Leith, and R.N. Shorten. Experimental evaluation of tcp protocols for high-speed networks. *Networking, IEEE/ACM Transactions on*, 15(5):1109–1122, 10 2007.
- [40] Douglas E. Comer. *Interligação em Redes com TCP/IP: Princípios, Protocolos e Arquitetura*. ELSEVIER, 2 edition, 9 2004.
- [41] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM.
- [42] Kunwoo Park, Dukhyun Chang, Junghoon Kim, Wonjun Yoon, and T. Kwon. An analysis of user dynamics in p2p live streaming services. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6, May.
- [43] 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.
- [44] Chi-Chung Cheung, Man-Ching Yuen, and A.C.H. Yip. Dynamic dns for load balancing. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 962–965, May.
- [45] J. Pan, S. Paul, and R. Jain. A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26–36, July.

- [46] C. Severance. Van jacobson: Content-centric networking. *Computer*, 46(1):11–13, Jan.
- [47] Philip Eardley, Michalis Kanakakis, Alexandros Kostopoulos, Tapio Levä, Ken Richardson, and Henna Warma. The Future Internet. chapter Deployment and adoption of future internet protocols, pages 133–144. Springer-Verlag, Berlin, Heidelberg, 2011.
- [48] 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, December.
- [49] 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.
- [50] 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.
- [51] Dina Katabi. *Decoupling congestion control and bandwidth allocation policy with application to high bandwidth-delay product networks*. PhD thesis, Cambridge, MA, USA, 2003.
- [52] N. Dukkipati, G. Gibb, N. McKeown, and Jiang Zhu. Building a rcp (rate control protocol) test network. In *High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on*, pages 91–98, Aug.
- [53] 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.
- [54] 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.

- [55] Nandita Dukkipati. *Rate control protocol (rcp): congestion control to make flows complete quickly*. PhD thesis, Stanford, CA, USA, 2008. AAI3292347.
- [56] 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.
- [57] 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.
- [58] 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.
- [59] Yong Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One more bit is enough. *Networking, IEEE/ACM Transactions on*, 16(6):1281–1294, Dec.
- [60] Huixiang Zhang, Guanzhong Dai, Lei Yao, and Hairui Zhou. Fast convergence of variable-structure congestion control protocol with explicit precise feedback. In FrancoP. 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.
- [61] 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.
- [62] 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.

- [63] 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.
- [64] 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: 1 de Fevereiro de 2014.
- [65] Marcelo Bagnulo and Nandita Dukkipati. Congestion exposure group, 6 2010. <https://datatracker.ietf.org/doc/charter-ietf-conex/>. Último acesso: 1 de Fevereiro de 2014.
- [66] K.C. Almeroth. The evolution of Multicast: From the MBone to Interdomain Multicast to Internet2 Deployment. *Network, IEEE*, 14(1):10–20, Jan 2000.
- [67] 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.
- [68] 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.
- [69] R Rodrigues and P Druschel. Peer-to-peer streaming systems. *Communications of the ACM*, 53(10):3–39, 2010.
- [70] 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.
- [71] Sachin Agarwal, Jatinder Pal Singh, Aditya Mavlankar, Pierpaolo Bacchicet, 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.
- [72] 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.
- [73] 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.
- [74] 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.
- [75] 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.
- [76] 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.
- [77] 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.
- [78] 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.
- [79] 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 *Pro-*

- ceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming*, P2PMMS'05, pages 21–28, Hilton, Singapore, 2005. ACM. ACM ID: 1099388.
- [80] 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.
 - [81] 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.
 - [82] 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.
 - [83] 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.
 - [84] 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.
 - [85] Fabio Pianese. A survey of p2p data-driven live streaming systems. *Streaming Media Architectures, Techniques, and Applications: Recent Advances*, 1:295–310, 10 2011.
 - [86] Chuan Wu, Baochun Li, and Shuqiao Zhao. Diagnosing network-wide p2p live streaming inefficiencies. In *INFOCOM 2009, IEEE*, pages 2731–2735, April.
 - [87] 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.

- [88] 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.
- [89] 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.
- [90] 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, December.
- [91] Jeff Seibert, David Zage, Sonia Fahmy, and Cristina Nita-Rotaru. Experimental comparison of peer-to-peer streaming overlays: An application perspective. In *33rd IEEE Conference on Local Computer Networks (LCN)*, pages 20–27, Washington, DC, USA, October 2008. IEEE Computer Society.
- [92] Yang Guo, Chao Liang, and Yong Liu. A survey on peer-to-peer video streaming systems. *PeertoPeer Networking and Applications*, 1(1):18–28, 2008.
- [93] E. Setton, P. Baccichet, and B. Girod. Peer-to-peer live multicast: A video perspective. *Proceedings of the IEEE*, 96(1):25–38, Jan.
- [94] M. Handley and V. Jacobson. Sdp: Session description protocol, 4 1998. <http://www.ietf.org/rfc/rfc2327.txt>. Último acesso: 1 de Fevereiro de 2014.
- [95] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, 7 2003. <http://www.ietf.org/rfc/rfc3550.txt>. Último acesso: 1 de Fevereiro de 2014.
- [96] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp), 4 1998. <http://www.ietf.org/rfc/rfc2326.txt>. Último acesso: 1 de Fevereiro de 2014.
- [97] 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.

- [98] 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.
- [99] 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.
- [100] 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.
- [101] F. Pianese, J. Keller, and E.W. Biersack. Pulse, a flexible p2p live streaming system. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6, April 2006.
- [102] 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.
- [103] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Addison Wesley, trad. 3 ed. edition, 2006.
- [104] Mouna Allani, Benoit Garbinato, and Fernando Pedone. *Middleware for Network Eccentric and Mobile Applications*, volume 3, chapter Application Layer Multicast, pages 191–218. Springer Berlin Heidelberg, 9 2009.
- [105] Yao Liu, L. Guo, Fei Li, and Songqing Chen. A case study of traffic locality in internet p2p live streaming systems. In *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, pages 423–432, June.
- [106] 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.

- [107] 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.
- [108] 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.
- [109] 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.
- [110] 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.
- [111] 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.
- [112] 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.
- [113] 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.
- [114] Constantinos Vassilakis and Ioannis Stavrakakis. Minimizing node churn in peer-to-peer streaming. *Comput. Commun.*, 33(14):1598–1614, September 2010.
- [115] Y. Sakata, K. Takayama, R. Endo, and H. Shigeno. A chunk scheduling based on chunk diffusion ratio on p2p live streaming. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, pages 74–81, Sept.

- [116] J. Seedorf, S. Kiesel, and M. Stiemerling. Traffic localization for p2p-applications: The alto approach. In *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, pages 171–177, Sept.
- [117] 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.
- [118] 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.
- [119] 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.
- [120] 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.
- [121] 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.
- [122] 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.
- [123] 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.

- [124] 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.
- [125] 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.
- [126] 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.
- [127] S. Sanghavi and B. Hajek. A new mechanism for the free-rider problem. *Automatic Control, IEEE Transactions on*, 53(5):1176–1183, June.
- [128] 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>.
- [129] 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.
- [130] 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.
- [131] 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.
- [132] C. K. Yeo, B. S. Lee, and M. H. Er. A framework for multicast video streaming over ip networks. *J. Netw. Comput. Appl.*, 26(3):273–289, August 2003.
- [133] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable application layer multicast. *SIGCOMM Comput. Commun. Rev.*, 32(4):205–217, August 2002.

- [134] 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.
- [135] Yi Cui, Baochun Li, and K. Nahrstedt. ostream: asynchronous streaming multicast in application-layer overlay networks. *Selected Areas in Communications, IEEE Journal on*, 22(1):91–106, Jan 2004.
- [136] 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.
- [137] Minseok Kwon and Sonia Fahmy. Path-aware overlay multicast. *Comput. Netw.*, 47(1):23–45, January 2005.
- [138] Feng Liu, J. Huang, Xicheng Lu, and Yuxing Peng. An efficient distributed algorithm for constructing delay- and degree-bounded application-level multicast tree. In *Parallel Architectures,Algorithms and Networks, 2005. ISPAN 2005. Proceedings. 8th International Symposium on*, pages 6 pp.–, Dec 2005.
- [139] 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.
- [140] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP), 3 2006. <http://www.ietf.org/rfc/rfc4340.txt>. Último acesso: 1 de Fevereiro de 2014.
- [141] L.M. de Sales, R. de Amorim Silva, H.O. Almeida, and A. Perkusich. Multi(uni)cast dccp for live content distribution with p2p support. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 3253–3258, April.

- [142] 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.
- [143] 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.
- [144] Garrett Hardin. The Tragedy of the Commons. *Science*, 162(3859):1243 – 1248, 3 1968.
- [145] Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 17(1):1–14, June 1989.
- [146] Leandro Melo de Sales, Hyggo Oliveira, Angelo Perkusich, and Rafael A. Silva. Distribuição de conteúdo multimídia em tempo real com transporte de fluxos controlados e não confiáveis entre pares. In *Proceedings of Simpósio Brasileiro de Redes de Computadores 2011. VII Workshop de Redes Dinâmicas e Sistemas P2P*, volume 1, pages 131–146, 5 2011. http://sbrc2011.facom.ufms.br/files/workshops/wp2p/ST04_1.pdf.
- [147] J. R Iyengar, P. D Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent End-to-End paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, October 2006.
- [148] C. -M Huang and M. -S Lin. Multimedia streaming using partially reliable concurrent multipath transfer for multihomed networks. *IET Communications*, 5(5):587–597, March 2011.
- [149] 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.

- [150] 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: 1 de Fevereiro de 2014.
- [151] 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.
- [152] 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.
- [153] Sally Floyd. Highspeed tcp for large congestion windows, 12 2003. <http://www.ietf.org/rfc/rfc3649.txt>. Último acesso: 1 de Fevereiro de 2014.
- [154] 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.
- [155] Cheng Peng Fu and Soung C. Liew. TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks. volume 21, pages 216–228, 2 2003.
- [156] 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.
- [157] 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.
- [158] 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.
- [159] 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.

- [160] 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.
- [161] 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.
- [162] 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.
- [163] 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.
- [164] R. Stewart, J. Stone, D. Otis, K. Morneau, H. Schwarzbaier, 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: 1 de Fevereiro de 2014.
- [165] 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.
- [166] 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.
- [167] 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.
- [168] 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.

- [169] Sangtae Ha, Long Le, Injong Rhee, and Lisong Xu. Impact of Background Traffic on Performance of High-speed TCP Variant Protocols. volume 51, pages 1748–1762, New York, NY, USA, 2007. Elsevier North-Holland, Inc.
- [170] 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: 1 de Fevereiro de 2014.
- [171] 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.
- [172] L.M. de Sales, H.O. Almeida, A. Perkusich, and K. Gorgonio. About encouraging residential users to share upload bandwidth with cdn/p2p live streaming systems. In *International Conference on Consumers Electronics (ICCE), 2013 IEEE*, pages 673–674, Jan.
- [173] 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.
- [174] Larry L. Peterson and Bruce S. Davie. *Computer Networks, A System Approach*. Morgan Kaufmann, 5 ed. edition, 3 2011.
- [175] Luiz Fernando Gomes Soares. *Redes de Computadores: das LANs, MANs e WANs às Redes ATM*. Campus, 2 edition, 1995.
- [176] 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.
- [177] 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.
- [178] 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.
- [179] 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.
- [180] 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.
- [181] Yi Cui, Yanchuan Cao, Liang Dai, and Yuan Xue. Optimizing P2P streaming throughput under peer churning. *Multimedia Systems*, 15(2):83–99, November 2008.
- [182] 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.
- [183] 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.
- [184] 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.
- [185] 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.

- [186] 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.
- [187] 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.
- [188] 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.
- [189] 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.
- [190] Leslie Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11):770–772, 1981.
- [191] 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.
- [192] Presidencia da República Federativa do Brasil. Medida provisória número 2.200-2, Agosto 2001.
- [193] Hong Liu and P. Mouchtaris. Voice over IP Signaling: H.323 and Beyond. In *Communications Magazine, IEEE*, volume 28, pages 142 – 148, 10 2000.
- [194] J. Rosenberg, H. Schulzrinne, and G. Camarillo. SIP: Session Initiation Protocol, 6 2002. <http://www.ietf.org/rfc/rfc3261.txt>. Último acesso: 1 de Fevereiro de 2014.

- [195] A. Jungmaier. *SCTP for Beginners*. Computer Network Tecnology Group, 1 edition, 9 2003.
- [196] Riccardo Bernardini, Roberto Rinaldo, and Roberto Fabbro. Peer-to-Peer Epi-Transport protocol, 7 2011. <http://tools.ietf.org/html/draft-bernardini-pptp-02>. Último acesso: 1 de Fevereiro de 2014.
- [197] A. Arcieri. Peer distributed transfer protocol, 7 2004. <ftp://ftp.good.net/dl/bd/convention.cdroms/defcon12/Arcieri/draft-arcieri-peer-distributed-transfer-protocol.txt>. Último acesso: 1 de Fevereiro de 2014.
- [198] C. Kulatunga, G. Kandavanam, A.I. Rana, S. Balasubramaniam, and D. Botvich. Hy-sac: A hybrid delivery system with adaptive content management for iptv networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5, june 2011.
- [199] TBE. Tbe, 3 2008.
- [200] S. Bradner. Key words for use in rfc's to indicate requirement levels, 3 1997. <http://www.ietf.org/rfc/rfc2119.txt>. Último acesso: 1 de Fevereiro de 2014.
- [201] Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory (Discrete Mathematics and Its Applications)*. CRC Press, 2 2003. ISBN: 978-1584880905.
- [202] R Séroul. *Programming for Mathematicians*. Springer-Verlag, 2 2000. ISBN: 978-3540664222.
- [203] 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.
- [204] K. J. Devlin. *Fundamentals of Contemporary Set Theory*. Springer, 9 1979. ISBN: 978-0387904412.
- [205] R. Braden. Requirements for Internet Hosts - Communication Layers, 10 1989. Último acesso: 1 de Fevereiro de 2014.

- [206] L. Eggert and F. Gont. TCP User Timeout Option, 3 2009. <http://www.ietf.org/rfc/rfc5482.txt>. Último acesso: 1 de Fevereiro de 2014.
- [207] 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: 1 de Fevereiro de 2014.
- [208] 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: 1 de Fevereiro de 2014.
- [209] H. Alvestrand. Tags for the identification of languages, 2 1995. <http://www.ietf.org/rfc/rfc1766.txt>. Último acesso: 1 de Fevereiro de 2014.
- [210] W. Wolff. *Stochastic Modeling and the Theory of Queues*. PrenticeHall, 1989.
- [211] 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: 1 de Fevereiro de 2014.
- [212] 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: 1 de Fevereiro de 2014.
- [213] 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.
- [214] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, New York, NY, USA, 1998. ACM Press.
- [215] D. Meyer. Administratively scoped ip multicast, 7 1998. <http://www.ietf.org/rfc/rfc2365.txt>. Último acesso: 1 de Fevereiro de 2014.

- [216] Sally Floyd. Metrics for the evaluation of congestion control mechanisms, 3 2008.
<http://www.ietf.org/rfc/rfc5166.txt>. Último acesso: 1 de Fevereiro de
2014.