

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, 10/04/2013

## 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 paliativas 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

<b>1</b>	<b>Global Media Transmission Protocol (GMTP)</b>	<b>1</b>
1.1	Visão Geral do GMTP . . . . .	3
1.1.1	Terminologias e Convenções . . . . .	5
1.1.2	Arquitetura . . . . .	8
1.1.3	GMTP Intra e GMTP Inter . . . . .	8
1.1.4	Principais funções . . . . .	10
1.1.5	Canais de Comunicação . . . . .	11
1.1.6	Tipos de Pacotes . . . . .	13
1.2	Definições, Relações e Restrições do GMTP . . . . .	15
1.3	Constituição da Rede de Favores $\eta$ . . . . .	19
1.3.1	Registro de participação de $r_d$ em $\eta$ . . . . .	20
1.3.2	Tabela de Recepção de Fluxos de Dados . . . . .	25
1.3.3	Formação de parcerias . . . . .	27
1.4	Distribuição do fluxo de dados $P$ através de $\eta$ . . . . .	34
1.4.1	Indexação de Conteúdo . . . . .	34
1.4.2	Estabelecimento de conexão e compartilhamento para obter $P$ . . .	38
1.4.3	Fase 1: primeira requisição a um fluxo de dados $P$ . . . . .	39
1.4.4	Fase 2: próximas requisições para obter $P$ . . . . .	43
1.4.5	Fase 3: busca por mais parceiros $r_q$ para obter $P$ . . . . .	44
1.4.6	Manutenção da conexão . . . . .	46
1.4.7	Compartilhamento de $P$ entre $s_a$ . . . . .	46
1.4.8	Envio e recebimento de $p_x \in P$ em $\eta$ . . . . .	47
1.5	Controle de Congestionamento em $\eta$ . . . . .	50

---

1.5.1	Controle de Congestionamento Unicast . . . . .	51
1.5.2	Controle de Congestionamento Multicast . . . . .	56
1.6	Autenticidade de $P$ . . . . .	59
1.6.1	Transmissão e assinatura de autenticidade de $p_x \in P$ . . . . .	61
1.6.2	Verificação de autenticidade de $p_x \in P$ . . . . .	62
1.6.3	Habilitar ou desabilitar o procedimento de segurança . . . . .	63
1.6.4	Obter a chave pública $K_{s_a}^+$ de $s_a$ . . . . .	64
1.7	Outras considerações sobre o GMTP . . . . .	65
1.7.1	Procedimentos para desconexão de nós $c_f$ , $l_w$ e $r_d$ . . . . .	65
1.7.2	Eleição de nós $l_w$ . . . . .	66
1.8	Sumário do Capítulo . . . . .	67

# Lista de Símbolos

3WHS - *Three Way Hand Shake*

BSD - *Berkley Software Distribution*

CCID - *Congestion Control IDentifier*

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* 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*

RTO - *Retransmission Timeout*

RTT - *Round Trip Time*

SCTP - *Stream Control Transmission Protocol*

Swift - *The Generic Multiparty Transport Protocol*

TCP - *Transport Control Protocol*

TTL - *Time-To-Live*

UDP - *User Datagram Protocol*

# Lista de Figuras

1.1	Analogia do Princípio da Cooperação de Brigadas utilizado no GMTP para distribuição de conteúdos multimídia ao vivo. . . . .	3
1.2	Cenário global de atuação do GMTP. . . . .	4
1.3	Rede de sobreposição construída pelo GMTP . . . . .	6
1.4	Tipos de Nós e modos de conexões do GMTP. . . . .	7
1.5	Arquitetura do Protocolo GMTP. . . . .	9
1.6	Tela da ferramenta de administração do OpenWRT [1] com suporte ao GMTP. Nessa tela, permitir que o administrador do roteador configure registros de participação em uma ou mais redes CDN. . . . .	10
1.7	Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP. . . . .	19
1.8	Exemplo de uma tabela de recepção de fluxo mantida por um nó $r_d$ . . . . .	26
1.9	Um nó $r_d$ precisa descobrir e selecionar seus parceiros $r_q$ , resultando na formação de um caminho $W_v$ entre um nó $s_a$ e um ou mais nós $c_f$ . . . . .	27
1.10	Cenário e passos para seleção de nós intra caminhos $W_v$ . . . . .	28
1.11	Cenário de falha do nó $r_6$ em um caminho $W_1$ , seguida de constituição de um novo caminho $W_3$ formado pelo procedimento de formação de parceria intra $W_v$ . . . . .	29
1.12	Cenário e passos para seleção de nós por interseção de caminhos $W_v$ . . . . .	31
1.13	Exemplo de rede para o estabelecimento de conexão do GMTP. . . . .	39
1.14	Passos do processo de estabelecimento de conexão do GMTP (Fase 1). . . . .	40
1.15	Tabela de recepção de fluxos de dados após a Fase 1. . . . .	43
1.16	Passos do processo de estabelecimento de conexão do GMTP (Fase 2). . . . .	43
1.17	Fase 3 de conexão do GMTP (Passo 1). . . . .	45
1.18	Fase 3 de conexão do GMTP (Passo 2). . . . .	45



1.19	Exemplo da estrutura do buffer de envio e recepção de um nó GMTP com tamanho de $17 p_x$ . . . . .	48
1.20	Exemplo do mapa de buffer de um nó GMTP com tamanho de $17 p_x$ . . . . .	49
1.21	Organização do algoritmo de controle de congestionamento no GMTP. . . . .	51
1.22	Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP, para $C = 2.4 Gbps$ . Os fluxos foram injetados na rede com base na distribuição pareto, com $E[L] = 25 pacotes$ e $shape = 1.2$ . Extraído de [2]. . . . .	56
1.23	Um nó $r_d$ mal-intencionados podem poluir o sistema com conteúdos que não foram gerados pelo nó $s_a$ . . . . .	60

# **Lista de Tabelas**

# Lista de Algoritmos

1	registerRelay( $s_a$ : PeerServer, $p_x = GMTP-Request$ ) . . . . .	22
2	onReceiveGMTPRegisterReply( $p_x = GMTP-Register-Reply$ ) . . . . .	24
3	matchPartnersByPathIntersection( $r_d$ : PeerRelay, $p_x = GMTP-Request$ ) . . .	33
4	Exemplo de requisição e resposta da lista de nomes dos fluxos de dados $P$ de um distribuidor de conteúdos multimídia. . . . .	35
5	Exemplo de uma mensagem SDP no pacote <i>GMTP-MediaDesc</i> . . . . .	37
6	respondToClients( $p_x$ : <i>GMTP-RequestNotify</i> ) . . . . .	42
7	digitalSignPacket( $p_x$ : <i>GMTP-Data</i> ) . . . . .	62
8	verifyPacketAuthenticity( $P'$ : <b>array of</b> <i>GMTP-Data</i> ) . . . . .	63

# Capítulo 1

## Global Media Transmission Protocol (GMTP)

O *Global Media Transmission Protocol* (GMTP) é um protocolo de rede 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 de dados multimídia ao vivo. Trata-se de um protocolo baseado em uma arquitetura híbrida P2P/CDN, onde os dados de um ou mais sistemas são transmitidos através de uma rede de pares P2P, onde os nós cooperam entre si a fim de obterem um conteúdo multimídia de interesse ao mesmo tempo que ocorrem interações entre os servidores de uma ou mais redes CDN, os quais atuam como super nós para a rede P2P, auxiliando-os no envio e recebimento dos fluxos de dados de eventos ao vivo. À medida que recebe um determinado fluxo de dados de um evento ao vivo, os nós cliente reproduzem tal conteúdo para o usuário final, 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 possuem nós clientes também interessados no mesmo conteúdo a fim de reproduzi-lo aos seus usuários finais.

As trocas de dados entre nós GMTP ocorrem por meio do envio e recebimento de pequenas partes do conteúdo de uma mídia, que são transmitidas por diferentes nós da rede, constituindo um fluxo de datagramas IP. Estes fluxos são transmitidos em modo *multicast* ou em múltiplos fluxos *unicast* compartilhados (multi-unicast), realizando-se controle de congestionamento sem garantia de entrega. A escolha do modo de transmissão utilizado para disseminar um determinado conteúdo ocorre automaticamente, ou seja, sem a influência da

aplicação que, simplesmente “sintoniza” sua conexão em um determinado canal definido pelo roteador, correspondente ao fluxo de dados de interesse do usuário final. 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.

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. Assim, promove-se a integração do GMTP em aplicações já existentes, consideradas futuras adoções, ao tempo que permite-se 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.

Nas próximas seções deste capítulo, detalha-se o funcionamento do GMTP, conforme a seguinte organização:

- Na Seção 1.1, apresenta-se uma visão geral do protocolo, como cenário de atuação, arquitetura, canais de comunicação e tipos de nós e pacotes.
- Na Seção 1.2, formaliza-se as definições e restrições do protocolo, que serão utilizadas nas seções subsequentes.
- Na Seção 1.3, descreve-se o processo de constituição da rede de favores, bem como aspectos de conexão multi-ponto através da introdução de um novo conceito de sockets P2P. Detalham-se os aspectos inerantes à constituição de uma rede P2P, tais como o registro de participação de um nó e o processo de seleção de nós parceiros.
- Na Seção 1.4, discute-se sobre os aspectos de transmissão e recepção de fluxos de dados, com os algoritmos utilizados para compartilhar um fluxos de dados e as estratégias de disponibilização e obtenção das partes de uma mídia.
- Na Seção 1.5, apresentam-se detalhes de funcionamento dos algoritmos de controle de congestionamento utilizados no GMTP.
- Na Seção 1.6, discute-se sobre os aspectos relacionados a validação de autenticidade de um fluxo de dados transmitido através do GMTP.

- E, por fim, na Seção 1.7, apresentam-se outros aspectos relacionados ao GMTP, tais como finalização de conexão, tolerância à desconexão e eleição de nós relatores.

## 1.1 Visão Geral do GMTP

O GMTP é composto por dois módulos chamados de *GMTP Intra* e *GMTP Inter*, que operam na camada de transporte e de rede, respectivamente. O *GMTP Intra* fornece serviços às aplicações de rede a fim de abstrair a complexidade na execução de tarefas comuns a qualquer sistema, tais como conexão multi-ponto, multiplexação/demultiplexação de segmentos IP e controle de congestionamento. O *GMTP Inter* é responsável por constituir uma rede de sobreposição P2P composta por roteadores, os quais funcionam como pontes de acesso aos servidores de uma rede CDN. Sendo assim, para viabilizar a disseminação de conteúdos multimídia, emprega-se o Princípio da Cooperação de Brigadas de Incêndio (*Fire Bucket Brigade Principle*), onde cada nó roteador de um caminho constituído entre o servidor que transmite a mídia e o cliente interessado em obtê-la, pode replicar o conteúdo sendo roteado para os clientes conectados diretamente a ele e assim sucessivamente, análogo a ilustração da Figura 1.1. No caso do GMTP, é como se cada roteador fosse responsável por apagar os focos de incêndio próximos a ele, ou melhor, atender a demanda dos clientes diretamente conectados a ele, ao passo que ajuda os outros a fazer o mesmo.



Figura 1.1: Analogia do Princípio da Cooperação de Brigadas utilizado no GMTP para distribuição de conteúdos multimídia ao vivo.

Na Figura 1.2, observa-se o cenário global 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. Na prática, os nós *Clientes GMTP* são aplicações de rede capazes de iniciar uma sessão GMTP, que transmitem, recebem e reproduzem

dados multimídia de um determinado evento. Os nós *Clientes GMTP* estão conectados a um nó *Repassador GMTP*, que é executado em um roteador de rede e, junto com outros nós *Repassadores GMTP*, efetivamente constituem a rede de sobreposição P2P. Os *Repassadores GMTP* também podem se conectar a um ou mais *Servidores GMTP*. Os *Servidores GMTP* são as fontes de conteúdos multimídia, obtidos através de três formas: i) diretamente a partir de uma unidade geradora de conteúdo (filmadora e/ou microfone); ii) a partir de um *Cliente GMTP*; e/ou iii) a partir de outro *Servidor GMTP* (troca de dados entre os servidores da CDN). Os *Servidores GMTP* recebem sinalizações de controle contendo requisições dos nós *Repassadores GMTP*, que ao receberem uma resposta correspondente a sua requisição, atendem a demanda de um ou mais nós *Clientes GMTP*.

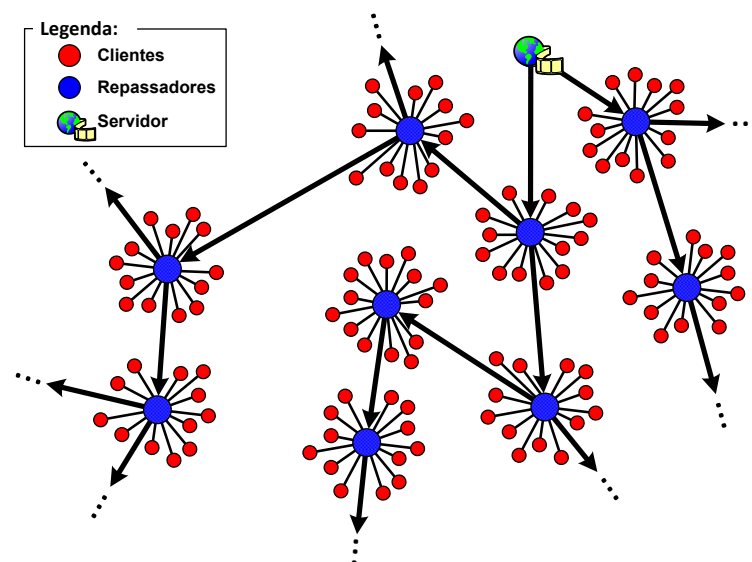


Figura 1.2: Cenário global de atuação do GMTP.

Quando um nó *Cliente GMTP* deseja reproduzir um determinado evento, este envia uma requisição destinada ao nó *Servidor GMTP* que está transmitindo o conteúdo de interesse, como atualmente acontece em qualquer conexão na Internet. A diferença é que um pedido de conexão é interceptado por algum nó *Repassador GMTP* durante o trajeto do pedido de conexão até o nó *Servidor GMTP*, que então determina os melhores parceiros para atendê-la. Em geral, isto ocorre já no roteador de borda do *Cliente GMTP*, que funciona como nó *Repassador GMTP* de origem. Caso o nó *Repassador GMTP* não encontre nenhum nó parceiro capaz repassar a mídia de interesse, este encaminha tal requisição ao nó *Servidor GMTP* que transmite a mídia correspondente. Em todo caso, sempre o nó *Repassador GMTP* de ori-

gem assumirá o controle da requisição do *Cliente GMTP*, habilitando-se como candidato a parceiro para outros nós *Repassadores GMTP*, quando motivados por requisições originadas pelos seus *Clientes GMTP*.

O posicionamento dos nós *Repassadores GMTP* e suas habilidades permitem a redução do número de fluxos de dados correspondente a um mesmo evento. Além disso, permite-se uma maior escalabilidade do número de nós *Clientes GMTP* interessado em receber um mesmo fluxo de dados. Por este mesmo motivo, o protocolo GMTP é flexível para permitir que um nó *Repassador GMTP* atue somente encaminhando conteúdos multimídias entre duas ou mais redes distintas, mesmo que este não esteja conectado a nenhum nó *Cliente GMTP* interessado por tal conteúdo. Desta forma, maximiza-se o uso de canais de transmissão ociosos, em particular das redes residenciais, as quais seus usuários muitas vezes estão ausentes e portanto sem fazer uso dos recursos disponíveis, não necessitando, inclusive, manter um determinado computador da sua rede interna ativo (ligado), como é obrigatório em todas as outras soluções similares e baseadas em arquitetura P2P.

Pelo princípio da cooperação de brigadas empregado no GMTP, as requisições de conexão podem ser originados não apenas por nós *Clientes GMTP* para seu respectivo nó *Repassador GMTP*, mas também as requisições podem ocorrer entre nós *Repassadores GMTP* que, motivados pelos interesses dos seus nós *Clientes GMTP*, podem formar parcerias entre si. Isto significa que um nó *Repassador GMTP* pode agir como se fosse um nó *Servidor GMTP*, respondendo às requisições originadas por seus nós *Clientes GMTP* ou de outros nós *Repassadores GMTP*, como se a requisição estivesse alcançado o *Servidor GMTP* que oficialmente transmite o conteúdo, o que ocorre de forma transparente para a aplicação.

Na Figura 1.3, observam-se detalhes do cenário supracitado, introduzindo-se o conceito de 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-se que a rede entre em colapso de congestionamento.

### 1.1.1 Terminologias e Convenções

Nesta seção, apresentam-se algumas definições, terminologias e convenções utilizadas no restante deste documento, de acordo com a Figura 1.4.



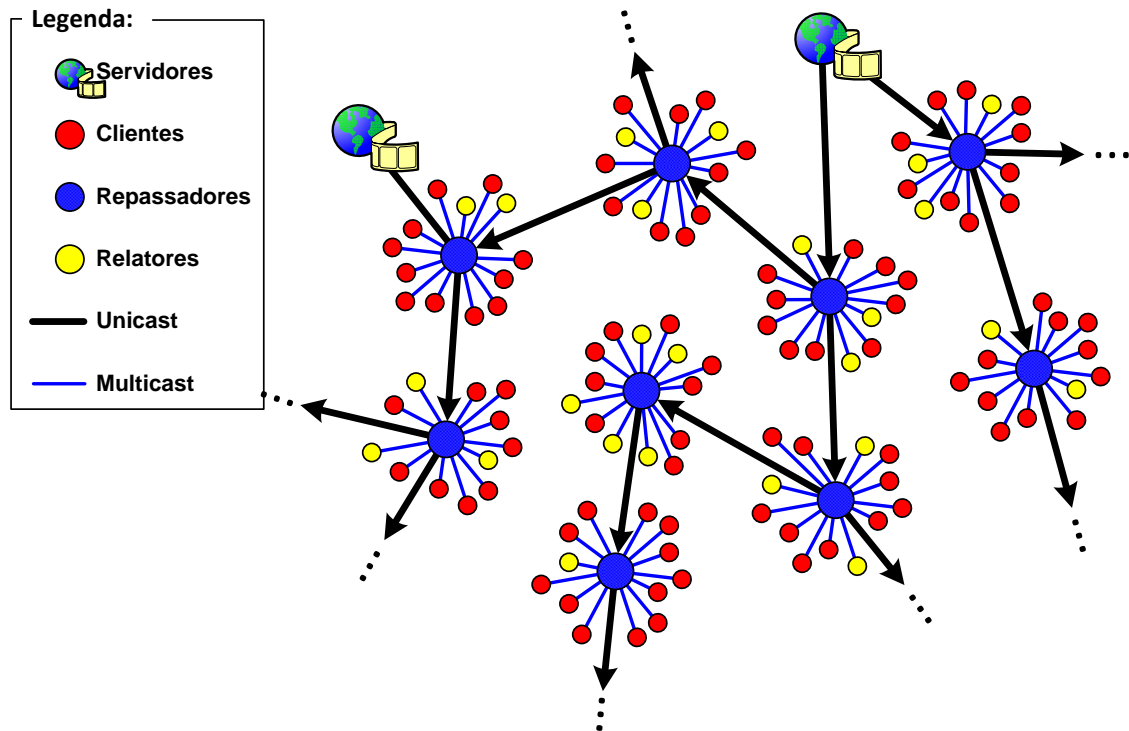


Figura 1.3: Rede de sobreposição construída dinamicamente pelo GMTP com a presença de nós repassadores e relatores.

#### Tipos de Nós:

- Nó GMTP ou Processador GMTP:** qualquer processador de rede que implementa o protocolo GMTP. É um sistema computacional que implementa parte ou todo do protocolo GMTP, sendo capaz de interpretar os cabeçalhos dos pacotes definidos pelo GMTP e realizar ações pré-definidas. Não há restrições de qual tipo de processador de rede pode implementar qual(is) parte(s) do GMTP.
- Cliente GMTP:** é um *nó GMTP* 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 para um processo em execução, contribuindo na execução do algoritmo de controle de congestionamento.
- Servidor GMTP:** é um *nó GMTP* capaz de capturar um evento ao vivo e gerar conteúdos digitais de áudio e vídeo ou ainda, receber de um *Cliente GMTP* tais conteúdos.

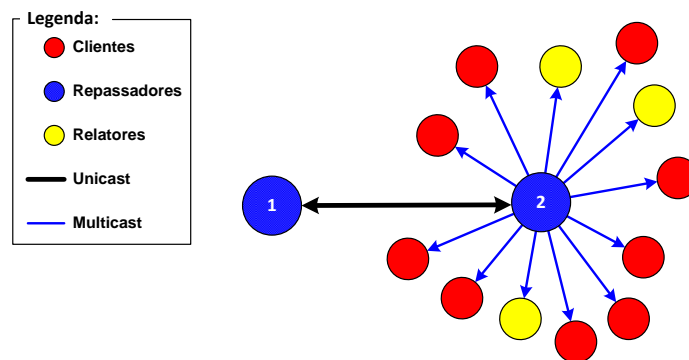


Figura 1.4: Tipos de Nós e modos de conexões do GMTP.

Em geral, um *Servidor GMTP* é um sistema final que participa de uma rede CDN.

- **Repassador GMTP:** é um *Nó GMTP* com habilidades de repassar os fluxos de dados originados de um ou mais *Servidores GMTP* ou de um outro nó *Repassador GMTP*.
- **Relator GMTP:** é um *Cliente GMTP* com habilidades de enviar relatórios periódicos ao repassador sobre o estado da transmissão.

#### Modos de Transmissão:

- **Unicast:** toda comunicação que ocorre entre dois nós *Repassadores GMTP*, com a interpretação do conceito definido por *unicast* em sua forma tradicional no contexto de redes de computadores.
- **Multi-unicast:** é um conjunto formado por dois ou mais canais de transmissão *unicast*.
- **Multicast:** toda comunicação que ocorre entre um nó *Repassador GMTP* e seus respectivos *Clientes GMTP*, com a interpretação do conceito definido por *multicast* em sua forma tradicional no contexto de redes de computadores.

O modo *multicast* sempre é utilizado para a transmissão dos datagramas correspondentes ao fluxo de dados multimídia, porém quando este modo não é suportado pela rede, executa-se o modo *multi-unicast* do protocolo. É mandatório que o modo *multicast* seja utilizado para transmissões entre um nó *Repassador GMTP* e seus *Clientes GMTP* diretos. O modo *unicast* é utilizado para que *Clientes GMTP* estabeleçam uma conexão com um *Servidor*

*GMTP* ou um *Repassador GMTP* e passe a distribuir o conteúdo de dados multimídia em sua rede local.

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

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.

As palavras “deve”, “não deve”, “requerido”, “pode”, “não pode”, “recomendado” e “opcional”, incluindo suas variações morfológicas, devem ser interpretadas como descrito na RFC 2119 [3], em inglês.

### 1.1.2 Arquitetura

Na Figura 1.5, ilustra-se a arquitetura geral do GMTP.

### 1.1.3 GMTP Intra e GMTP Inter

De acordo com a arquitetura apresentada na seção anterior, define-se:

- **GMTP Intra:** parte do protocolo GMTP executada na camada de transporte da pilha de protocolos TCP/IP, corresponde ao módulo interno de uma rede, composta por nós clientes e relatores, disponibilizado no sistema operacional e utilizado pela aplicação 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

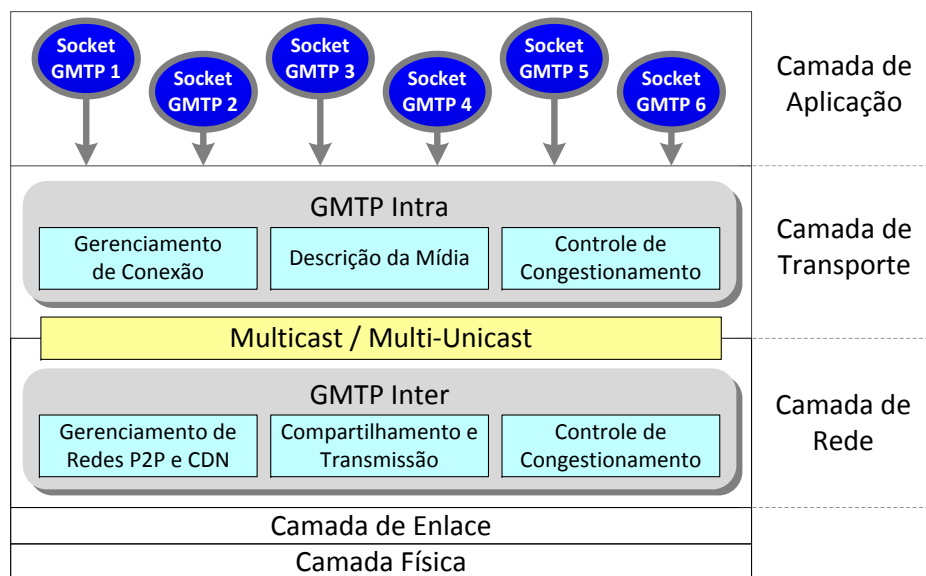


Figura 1.5: Arquitetura do Protocolo GMTP.

de estado que representam uma instância e executa algoritmos para gerenciamento de conexão (estabelecimento e desconexão) e eleição de nós parceiros, determinação do formato e preenchimento dos parâmetros que definem uma determinada mídia digital, permitindo que a aplicação defina os valores de tais parâmetros e ou obtenham acesso aos valores dos mesmos, controle de congestionamento e multiplexação e demultiplexação de datagramas IP. O GMTP não faz verificação de conteúdo por mecanismo de soma de verificação.

- GMTP Inter:** parte do protocolo GMTP executada na camada de rede da pilha de protocolos TCP/IP e corresponde ao módulo externo de uma rede, composta por vários nós repassadores que cooperam entre si. É executado por um roteador de rede e aceita conexões oriundas de um nó cliente ou de um nó repassador. 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 repassadores, seleção de nós repassadores parceiros, eleição de nós relatores, controle de congestionamento assistido pela rede e controle de compartilhamento de fluxos multimídia.

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 ilustrado na Figura 1.6. Nesse caso, o usuário administrador de um nó repassador pode definir os seguintes parâmetros:

- registro de participação em uma ou mais redes CDN;
- largura de banda (*download* e *upload*) que deseja 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;
- download automático ou não do certificado digital de um nó servidor; e
- realização de cache ou não dos certificados digitais obtidos.

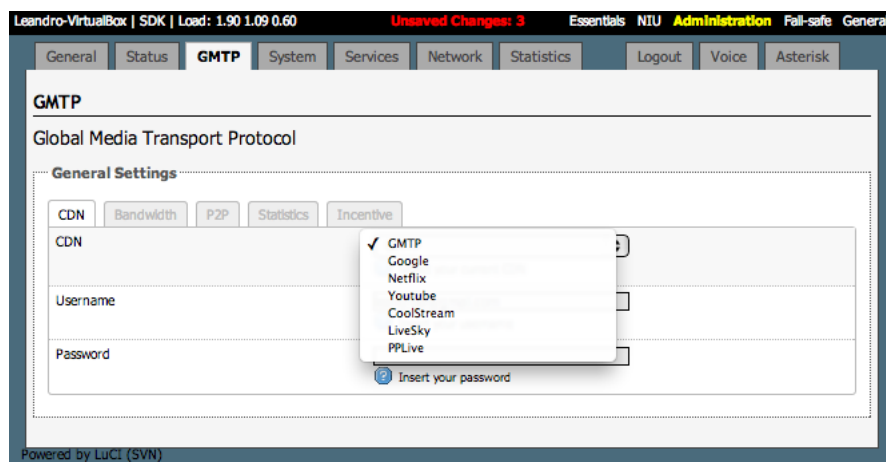


Figura 1.6: Tela da ferramenta de administração do OpenWRT [1] com suporte ao GMTP. Nessa tela, permitir que o administrador do roteador configure registros de participação em uma ou mais redes CDN.

### 1.1.4 Principais funções

- Registro de participação de um nó repassador em um nó servidor. Isto permite o suporte à pré-seleção de nós parceiros filtrados por métricas que influenciam na qualidade de experiência do usuário ao assistir um evento ao vivo, como atraso fim-a-fim.

- Acesso a uma transmissão 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.
- Descoberta de nós parceiros entre redes distintas e negociação de parcerias, com suporte a formação de parcerias baseadas em métricas que influenciam na qualidade de experiência do usuário.
- Envio e recebimento de fluxos de dados compartilhados entre nós da mesma rede através de multicast e uso de fluxos unicast entre redes distintas, porém sem a relação de uma conexão por cliente e assim evitando o fenômeno da tragédia dos bens comuns, discutido na Seção ??.
- Suporte a algoritmos de controle de congestionamento assistidos pela rede e de fluxos multicast. Troca de relatórios periódicos entre os nós repassadores sobre a transmissão.
- Eleição de nós relatores com suporte a tolerância a desconexões de nós, com notificação e reeleição de novos nós.
- Possibilidade de permitir que os nós clientes verifiquem a autenticidade das partes de uma mídia, por meio do uso de certificado digital determinado no nó servidor para impedir ataques de poluição.

### 1.1.5 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 fornece 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 [4], 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 *Internet Assigned Numbers Authority* – IANA<sup>1</sup> 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 1.4.2, 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

Além do canal de controle, define-se no protocolo GMTP um canal de repasse utilizado por um nó repassador para encaminhar datagramas vindos de um servidor ou de outro repassador para a rede local. Esse canal de repasse, na prática, é um socket multicast criado pelo

---

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

nó repassador para transmitir os datagramas para todos os seus clientes com interesse em reproduzir o mesmo fluxo de dados de um evento ao vivo.

O *socket de repasse multicast* deve ser criado quando um nó repassador passa a obter um determinado fluxo de dados correspondente a um determinado 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 para repassar os dados vindos do servidor ou de outro repassador para dentro de sua rede. A faixa de endereços IP multicast que o nó repassador deve utilizar para criar seu socket de repasse para um determinado fluxo de dados é a de escopo local 239.192.0.0/14, definida na RFC 2365 [4]. Como é uma faixa de endereços IP multicast de domínio local, não se faz necessário registrar o uso desses endereços. Isto significa que para todo fluxo de dados de um evento ao vivo, deve-se alocar um endereço IP e uma porta. No caso do esquema de endereçamento IPv4, com isto, será possível definir a transmissão de exatamente de 17.179.607.040 (dezesete 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.

### 1.1.6 Tipos de Pacotes

Toda comunicação entre dois ou mais nós GMTP ocorre através da troca de datagramas IP, que carregam sinalizações de controle e/ou dados da aplicação. Para isso, faz-se necessário registrar o uso de um código para o campo *Protocolo* do cabeçalho de um datagrama IP à IANA. Com a padronização do protocolo GMTP e a publicação da sua RFC, provavelmente será utilizado o código 100, como já está definido no documento *Protocol Numbers*<sup>2</sup> da IANA.

No cabeçalho dos pacotes GMTP, existe um campo denominado *tipo do pacote* com tamanho de 4 *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.

---

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



0. *GMTP-Request*: Cliente envia requisição para obter um fluxo de dados multimídia dado um nome do fluxo de interesse;
1. *GMTP-RequestNotify*: 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. Campo de dados contém a descrição da mídia a ser reproduzida;
2. *GMTP-Response*: Repassador confirma o estabelecimento de uma parceria com outro nó repassador, dado um determinado fluxo de dados;
3. *GMTP-Register*: Repassador registra participação no servidor para funcionar como distribuidor de um ou mais fluxos de dados;
4. *GMTP-Register-Reply*: Servidor responde ao repassador sobre seu pedido de registro de participação;
5. *GMTP-RelayQuery*: Repassador pode solicitar ao servidor uma lista de possíveis nós repassadores parceiros;
6. *GMTP-Data*: Qualquer nó utiliza para transmitir dados da aplicação;
7. *GMTP-Ack*: Qualquer nó utiliza 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*: Servidor transmite esse pacote para descrever informações sobre a mídia sendo transmitida em uma determinada transmissão;
10. *GMTP-DataPull-Request*: 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*: Repassador envia para um cliente o pedido para tal cliente atuar como nó relator;
13. *GMTP-Elect-Response*: Cliente envia para o repassador uma confirmação de que pode atuar como relator;

14. *Reservado*: Reservado para uso futuro e ignorado pelos nós que o processa;
15. *Reservado*: Reservado para uso futuro e ignorado pelos nós que o processa;
16. *GMTP-Close*: Servidor, repassador ou cliente solicita o término de uma conexão;
17. *GMTP-Reset*: Determina, incondicionalmente, a finalização de uma conexão.

Nas próximas seções, descreve-se todas as possíveis ações do GMTP e os tipos de pacotes envolvidos na comunicação entre os seus nós. No Apêndice ??, apresenta-se detalhes acerca do uso dos tipos de pacotes do GMTP, sendo seu teor bastante técnico e portanto dedicado aos leitores interessados em sua implementação.

## 1.2 Definições, Relações e Restrições do GMTP

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 [5–8].

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  $(\mathbb{P}, \prec) = \{p_1, p_2, p_3, \dots, p_h\}$ , onde  $h \in \mathbb{N}$ . Note que o símbolo  $\prec$  é utilizado para representar precedência entre dois elementos diferentes.

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  $s_a$  ou repassados 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, definida por  $C_i : r_d \rightarrow 2^C, \forall r_d, r_{d+1} \in R, C_i(r_d) \cap C_i(r_{d+1}) = \{\emptyset\}$ ;

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, que determina que dois nós  $c_f$  não podem estar relacionados a mais de um nó  $r_d$ , 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  $W_v \neq \{\emptyset\}$  e  $|W_v| > 1$ ;

ii. Um caminho  $W_v$  é dito *caminho semi-completo*, representado por  $W_v^\circ$ , se e somente se  $W_v \leftrightarrow \exists B_\theta$  (bijetora), tal que  $B_\theta \in 2^B$  e  $B_\theta \neq \{\emptyset\}$ . Isto é, todos os roteadores  $b_e \in B$  são sobrepostos por um nó  $r_d \in W_v^\circ$ ;

iii. Um caminho  $W_v$  é dito *caminho completo*, representado por  $W_v^\bullet$ , se for  $W_v^\circ$  e se  $W_v \subset T$ , tal que  $T \subset Z$  é o conjunto dos nós que transmite os pacotes

de dados  $p_x \in P$ , definido por  $T = \{t_u \mid \varphi(t_u, P) = 1\}$ , sendo  $u \in \mathbb{N}$  e  $\varphi$  uma função booleana que determina se um nó  $t_u \in T$  transmite os pacotes  $p_x \in P$  para  $c_f \in C_i(t_u)$ , 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  $\delta$ , uma função que define um sub-caminho de  $W_v$ , representado por  $W_v^\triangleleft$ , a partir de um nó  $t_u \in W_v$  até um nó  $t_1 \in W_v$ , tal que  $\delta : (t_u, W_v) \rightarrow (W_v^\triangleleft, \prec)$ . Ou seja, para um caminho qualquer  $(W_v, \prec) = \{t_{u+1}, t_{u+1}, t_u, t_{u-1}, t_{u-2}, \dots, t_2, t_1\}$ ,  $\delta(t_u, W_v) = W_v^\triangleleft = \{t_u, t_{u-1}, t_{u-2}, \dots, t_2, t_1\}$ . Neste caso, como  $\delta$  faz um corte no conjunto  $W_v$ , pode-se obter *caminho semi-completo* ou *completo*, representados por  $W_v^{\triangleleft\circ}$  e  $W_v^{\triangleleft\bullet}$ , respectivamente;
- (f) Seja  $\zeta$  uma função que calcula o custo total para transmitir um pacote  $p_x \in P$ , através de um caminho  $W_v$ , definida por  $\zeta : \sum_{v=1}^{|W_v|} \gamma(w_m, w_{m+1})$ , tal que  $\gamma$  é uma função que determina o custo para transmitir o pacote  $p_x$  entre dois nós distintos  $\forall w_m, w_{m+1} \in W_v$ . No GMTP, o custo é calculado pelo RTT entre dois nós distintos, mas podendo-se utilizar outras métricas, como número total de saltos no caminho  $W_v$ ;
- (g) *Conjectura 1*:  $\forall r_d \in R$  e  $\forall c_f \in C$ ,  $r_d$  é mais estável que qualquer  $c_f$  com relação a sua disponibilidade e participação em uma rede de favores  $\eta$ . Em uma rede comutada por pacotes IP, um nó  $b_e \in B$ , portanto para o GMTP 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  $p_x$  devem, obrigatoriamente, passar 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 base na aceitação dessa conjectura, especificamente para a rede  $\eta$ , pretende-se permitir que outros nós  $c_f$  possam continuar recebendo  $P$ , mesmo ocorrendo a desconexão de um nó  $c_f$  que esteja recebendo  $P$  durante a recepção de um fluxo de dados  $P$ . No GMTP, adota-se tal estratégia quando um nó  $r_d$  passa a manter estado sobre tal transmissão e não mais por qualquer nó  $c_f$ , antes prática comumente adotada em soluções tradicionais de distribuição de conteúdos multimídia baseado em uma arquitetura P2P ou em qualquer protocolo de transporte e rede disponível no estado da arte;

- (h) *Conjectura 2*: as tabelas de roteamento dos nós  $w_m \in W_v$  não mudam frequentemente e são independentes umas das outras. Em redes comutadas por pacotes IP, as rotas entre quaisquer nós  $c_{f_1}$  e  $c_{f_2} \in C$  não se alteram com um nível de 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, é possível antecipar a formação de parcerias entre os nós em  $Z$  antes da efetiva transmissão de um fluxo de dados  $P$ . Essa estratégia é adotada no GMTP.

Desta forma,  $\eta$  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.

Nas próximas seções, detalham-se os aspectos teóricos e computacionais empregados do GMTP a fim de construir  $\eta$ , em três partes distintas de acordo com os blocos funcionais ilustrados na Figura 1.7:

1. *Constituição da rede de favores  $\eta$* : descobrir, definir, efetivar e desfazer parcerias entre os nós  $r_d \in R$  de acordo com o evento  $\mathcal{E}$  a ser transmitido (Seção 1.3);
2. *Distribuição do fluxo de dados  $P$  em  $\eta$ , através das Instâncias de Aplicações (sockets)*:

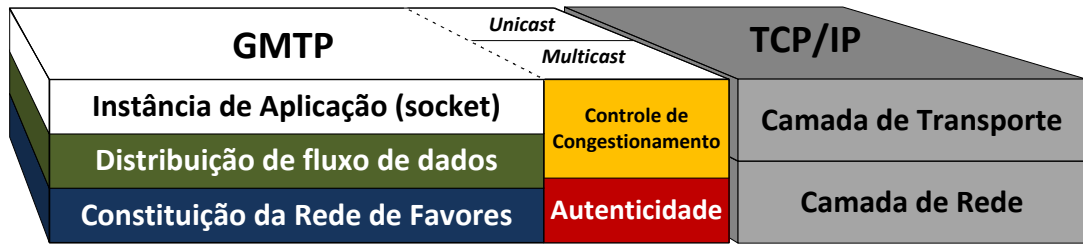


Figura 1.7: Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP.

- conectar os nós  $c_f \in C$ ,  $r_d \in R$  e  $s_a \in S$ , bem como transmitir os pacotes de dados  $p_x \in P$  através da rede de sobreposição constituída na Fase 1 (Seção 1.4);
3. *Controle de congestionamento em  $\eta$* : controlar a taxa de transmissão dos fluxos de dados  $P$  transmitidos em  $\eta$  na Fase 1 (Seção 1.5); e
  4. *Segurança em  $\eta$* : verificar a autenticidade do conteúdo de  $P$  antes de entregar aos nós  $c_f \in P$  (Seção 1.6).

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

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

O GMTP tenta determinar um caminho sub-ótimo  $W_\theta$  através do qual os pacotes de dados  $p_x \in P$  sejam entregues o mais rápido possível ao nó  $c_f$  interessado em obter  $P$ . Para isto, deve-se determinar  $W_\theta$ , tal que  $W_\theta = \min(\zeta(\forall W_v))$  e, sempre que possível, que  $W_\theta$  seja um caminho completo  $W_\theta^\bullet$ . Sempre buscar um caminho completo é importante porque como todos os nós de tal caminho são nós repassadores sobrepostos em cada roteador da rota de rede utilizada para transmitir  $P$ , consequentemente pode-se distribuir  $P$  para mais nós  $c_f$  sem que sejam necessárias múltiplas conexões em  $s_a$ , evitando cenários da tragédia dos bens comuns, discutidos no Capítulo ???. Além disso, por possuir mais nós  $r_d$ , um utilizar um caminho completo torna os sistemas que transmitem um fluxo de dados  $P$  menos sensíveis

às desconexões.

### 1.3.1 Registro de participação de $r_d$ em $\eta$

O procedimento de registro de participação de um nó  $r_d$  em uma rede  $\eta$  é o primeiro passo, e um dos mais importante. O registro de participação permite que um nó  $r_d$  se registre a um nó  $s_a$  para sinalizar interesse em funcionar como um nó repassador de um ou mais fluxos 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.

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 código no pacote *GMTP-Register*, no momento de roteá-lo para o próximo salto da rota em direção ao nó  $s_a$ . Na prática, o identificador de um nó repassador pode ser o endereço IP, porém o método mais apropriado seria gerar um código-nome baseado na combinação dos endereços MAC (*Media Access Control*) das interfaces de rede do roteador. Quando o pacote *GMTP-Register* alcançar o destino  $s_a$ , o nó  $s_a$  conhecerá a lista ordenada dos nós  $r_d$  até  $s_a$  e a 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 1.3.3. Pelas conjecturas 6g e 6h, apresentadas na Seção 1.2, o nó  $s_a$  pode utilizar os caminhos conhecidos  $W_v \in W$  para coordenar o processo de formação de futuras parcerias entre os nós  $r_d \in R$ .

Sendo assim, um registro de participação ocorre quando um nó deseja participar da rede de sobreposição, não necessariamente quando se deseja obter um fluxo de dados  $P$ . Mesmo assim, um registro de participação pode ocorrer no mesmo instante que um nó  $r_d$  desejar receber um fluxo de dados  $P$ . 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  $s_a$ , inicialmente sem interesse por um evento  $\mathcal{E}$  qualquer, será possível mapeá-lo antecipadamente e seleccionar 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  $p_x$  mesmo quando  $C_i(r_d) = \{\emptyset\}$ , ou seja, quando o nó  $r_d$  não tenha nó 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 dinâmicos, que poderiam ser acionados quando fosse conveniente, por exemplo, se ele for um nó comum entre dois caminhos, necessitando apenas de enviar apenas um fluxo de dados  $P$ .

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 o fluxo de dados  $P$ , o registro de participação já terá ocorrido e o fluxo já estará sendo recebido pelo nó  $r_d$  em questão, vindo do nó  $s_a$ , mas não necessariamente direto de  $s_a$ , mas sim podendo ser repassado por outros nó  $r_d$ . Como consequência, reduzindo-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 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.

Analogamente, a motivação do registro de participação empregado no GMTP é transformar cada roteador de uma rede IP como se fosse uma antena de recepção de uma transmissora de TV. Tal antena é capaz de receber um ou mais sinais de canais de TV diferentes, nesse caso, os fluxos de dados  $P$ , e distribuí-los para os aparelhos de TV da cidade, ou melhor, distribuí-los para os sistemas finais da respectiva rede, ou seja, seus nós  $c_f \in C$ . Este assunto será retomado na Seção 1.4, onde discute-se as funções de conectividade IP empregadas no GMTP.

No Algoritmo 1, resume-se o procedimento de registro de participação explicado anteriormente. Um nó  $r_d$  utiliza tal algoritmo para realizar o 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, pois lembre-se, o nó  $r_d$  se comporta, por analogia, como uma antena de TV. Porém, da mesma forma que é possível uma antena de TV obter apenas o sinal correspondente a um terminal canal (frequência de TV), também é possível que um nó  $r_d$  especifique exatamente qual fluxo de dados  $P$  que deseja receber. Neste caso, 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.



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

---

```

/* 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  $p_x \neq NULL$  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    $channel \leftarrow \text{isFlowBeingReceived}(P)$ ; /* See Section 1.3.2 */
   /* Add  $c_f$  in the list of receivers waiting  $P$ . */
5    $\text{addClientWaitingFlow}(c_f, P)$ ;
6   if  $channel \neq NULL$  then
   /* Let  $c_f$  know that  $P$  is already registered in this
       $r_d$  and is available from a multicast channel. */
7    $\text{respondToClients}(\text{GMTPRequestReply}(channel))$ ;
8   return 0;
9 else /* Flow  $P$  not registered yet. */
   /* Send request to  $s_a$  and wait registration reply.
      When  $GMTP\text{-}Register\text{-}Reply$  is received, executes
       $\text{onReceiveGMTPRegisterReply}$  (Algorithm 2). */
10  if not  $\text{isWaitingRegisterReply}(P)$  then
11     $\text{isWaitingRegisterReply}(P, \text{true})$ ;
12     $\text{sendPktRdt}(\text{GMTPRegister}(s_a, P))$ ;
13    return 0;
14  end
   /* Ask  $C_i(r_d)$  to wait registration reply for  $P$ . */
15   $\text{respondToClients}(\text{GMTPRequestReply}(P))$ ;
16  return 0;
17 end
18 end
19 if not  $\text{isWaitingRegisterReply}(s_a)$  then
20   return  $\text{sendPktRdt}(\text{GMTPRegister}(s_a))$ ;
21 end
22 return 0;

```

---

---

No Algoritmo 2, resume-se os passos descritos anteriormente, executados após o nó  $r_d$  receber o pacote do tipo *GMTP-Register-Reply*, correspondente ao registro de participação.

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

---

```

/* The node  $r_d$  executes this algorithm when receives a
   packet of type GMTP-Register-Reply, as response for a
   registration of participation sent to a  $s_a$  node. */
1 isWaitingRegisterReply( $P, false$ );
2 if  $p_x = OK$  then                                     /*  $s_a$  confirmed registration */
3    $s_a \leftarrow \text{getPacketFieldValue}(p_x, 'server');$     /* Gets  $s_a$  in  $p_x$  */
4    $P \leftarrow \text{getPacketFieldValue}(p_x, 'flow');$         /* Gets  $P$  in  $p_x$  */
5   if  $P \neq NULL$  then                                  /* Reply to  $C_i(r_d)$ , waiting for  $P$  */
6     if  $s_a$  enabled security layer then                 /* Section 1.6.4 */
7       getAndStoreServerPublicKey( $s_a$ );
8     end
9      $channel \leftarrow \text{createMulticastChannel}(s_a, P);$ 
10    updateFlowReceptionTable( $channel$ ); /* Section 1.3.2 */
    /* Let  $c_f \in C_i(r_d)$  know the multicast channel to
       receive  $P$  (Section 1.4.2). */
11    respondToClients(GMTPRequestReply( $channel$ ));
    /* Start to relay  $P$  to clients (Section 1.4.8). */
12    startRelay( $channel$ );
13  end
    /* It was just a reply of a registration of
       participation. Update flow reception table. */
14  updateFlowReceptionTable( $s_a$ ); /* Section 1.3.2 */
15 else
    /*  $s_a$  refused to accept the registration of
       participation. This  $r_d$  must notify the clients
       waiting for receiving  $P$ . */
16   $errorCode \leftarrow \text{getPacketFieldValue}(p_x, 'error');$ 
17  respondToClients(GMTPRequestReply( $errorCode, P$ ));
18 end

```

---

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  $\eta$  através de uma função tradicionalmente conhecida 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* [9].

Além disso, um nó  $r_d$  pode sinalizar explicitamente sua desconexão a  $s_a$ , quando não desejar mais participar da rede de favores  $\eta$  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 explícita através do envio de *GMTP-Close*, o nó  $s_a$  deve desconsiderar  $r_d$  no processo de formação de parcerias e enviar para o nó  $r_d$  um pacote do tipo *GMTP-Reset*.

Para finalizar esse assunto, 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. Consequentemente, quanto mais caminhos forem conhecidos, mais parcerias poderão ser formadas entre os nós  $r_d$ . E, portanto, 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  $r_d$  (antena na cidade!), 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, por exemplo, pode configurar seu roteador para registra-lo em múltiplas redes de distribuição, como ilustra-se na Figura 1.6. 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.

### 1.3.2 Tabela de Recepção de Fluxos de Dados

Antes de seguir com a explicação sobre o processo de estabelecimento de conexão do GMTP, é importante entender que cada nó  $r_d$  mantém uma tabela chamada *Tabela de Recepção de*

*Fluxos de Dados*, como ilustra-se na Figura 1.8. O nó  $r_d$  utiliza tal tabela para registrar todos os fluxos de dados que estão sendo repassados para seus nós  $c_f \in C_i(r_d)$ , 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 1.8: Exemplo de uma tabela de recepção de fluxo mantida por um nó  $r_d$ .

- **Nome do Fluxo de Dados  $P$ :** é uma sequência de 128 *bits* que determina o nome de um fluxo de dados, como descrito na Seção 1.4.1;
- **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  $c_f \in C_i(r_d)$ .

Um nó  $r_d$  consulta a tabela de recepção de fluxos de dados quando recebe um pedido de conexão para obter um fluxo de dados  $P$ , tal como apresentou-se na Linha 6 do Algoritmo 1, Seção 1.3.1. Além disso, um nó  $r_d$  atualiza a tabela de recepção de fluxos de dados após receber uma confirmação do registro de participação, tal como apresentou-se na Linha 10 do Algoritmo 2, Seção 1.3.1. Mais adiante, na Seção 1.4.2, discute-se em mais detalhes as ações de consulta e atualização da tabela de recepção de fluxos de dados.

### 1.3.3 Formação de parcerias

A formação de parcerias resulta na constituição de um ou mais caminhos  $W_v \in W$ , que interligará um nó  $s_a$  a um ou mais nós  $c_f \in C_i(r_d)$ . Tal processo ocorre entre dois ou mais nós  $r_d$  e consiste na seleção de nós parceiros  $r_q$  para auxiliar  $r_d$  a obter os pacotes  $p_x \in P$ , análogo a ilustração da Figura 1.9.

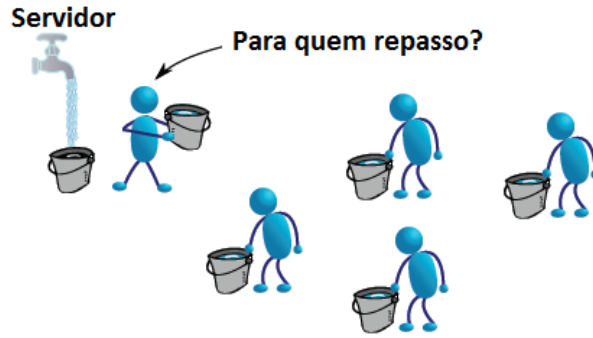


Figura 1.9: Um nó  $r_d$  precisa descobrir e selecionar seus parceiros  $r_q$ , resultando na formação de um caminho  $W_v$  entre um nó  $s_a$  e um ou mais nós  $c_f$ .

No GMTP, a seleção de nós parceiros  $r_q$  ocorre antes e durante a transmissão de um fluxo de dados  $P$ , através dos procedimentos de formação de parcerias apresentados a seguir, executados concorrentemente.

1. Formação de parcerias intra  $W_v$ ;
2. Formação de parcerias por intersecção de  $W_v$ .

#### 1. Formação de parcerias intra $W_v$

No procedimento de formação de parceria intra  $W_v$ , os nós  $w_m \in W_v$  são automaticamente considerados parceiros entre si e por isto qualquer nó  $w_m$  pode repassar um fluxo de dados  $P$  para o nó  $r_d$  em questão. A formação da parceria ocorre de forma transparente durante o pedido de conexão de um nó  $c_f$  ao nó  $s_a$  para obter  $P$ . Dessa forma, qualquer nó  $w_m \in W_v$  pode agir como se fosse um nó  $s_a$ , sendo este o procedimento mais simples e direto que um nó  $r_d$  pode obter um fluxo de dados  $P$  através de nós parceiros  $r_q$ .

Na Figura 1.10, ilustram-se os passos executados por este procedimento de formação de parceria. No Passo 1, ilustra-se um cenário de rede  $\eta$  com 19 nós  $r_d$  e dois nós  $s_a$ , sem

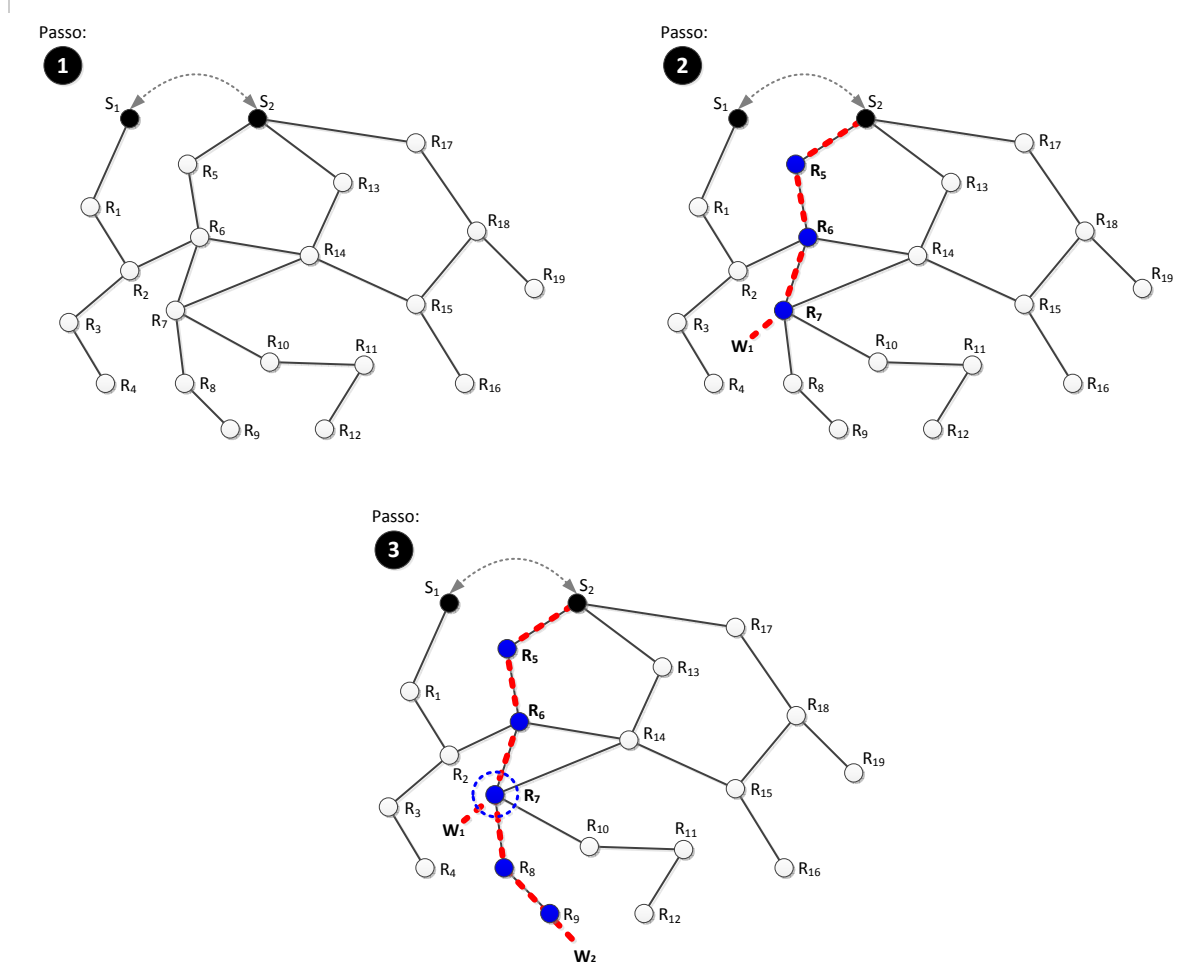


Figura 1.10: Cenário e passos para seleção de nós intra caminhos  $W_v$ .

qualquer fluxo de dados  $P$  transmitido, tampouco nenhuma parceria efetivada. No Passo 2, ilustra-se a transmissão de um fluxo de dados  $P$  que ocorre entre os nós  $s_2$  e  $r_7$ , caracterizando o caminho  $W_1$ , ilustrado pela linha tracejada na cor vermelha. Este cenário será utilizado como base para todos os exemplos apresentados a seguir.

Com base no exemplo ilustrado, para formar o caminho  $W_1$ , o nó  $r_7$  transmite o pedido de registro de participação ao nó  $s_2$ , como explicado na Seção 1.3.1. Assim, quando houver um nó qualquer  $r_d$  interessado por um fluxo de dados  $P$ , motivado por algum nó  $c_f \in C_i(r_d)$ , qualquer nó  $w_m$  pode interceptar um pedido de conexão transmitido para  $s_a$ , desde que  $w_m, r_d \in W_v$  e  $\varphi(w_m, P) = 1$ . Ao interceptar o pedido de conexão enviado por  $r_d$ , o nó  $w_m$  enviar uma resposta a  $r_d$ , como se fosse o servidor  $s_a$ , uma vez que o nó  $w_m$  já está repassando o fluxo de dados  $P$  para seus nós  $c_f \in C_i(w_m)$ . Para este caso, até antes do momento em que  $w_m$  interceptara o pedido de conexão enviado por  $r_d$ , o fluxo de dados  $P$  estava sendo

repassado apenas para os nós  $c_f \in C_i(w_m)$ . Porém, com o emprego do procedimento de formação de parcerias intra  $W_v$ , o mesmo fluxo de dados  $P$  é “estendido” até  $r_d$ , e tal nó poderá “estender” o fluxo de dados  $P$  para mais outros nós  $r_d$  que utilizem o caminho  $W_v$  para alcançar  $s_a$ . Este caso pode ser observado no Passo 3 da Figura 1.10. Uma vez que  $\varphi(r_7, P) = 1$  este intercepta o pedido de conexão transmitido pelo nó  $r_9$ , formando o caminho  $W_2^{\triangleleft} = \{r_9, r_8\} \cup W_1$ .

Note que quando um nó  $r_d$  interceptar um pedido de conexão para um fluxo de dados  $P$ , tal como explicado anteriormente, o nó  $r_d$  deve também transmitir para o nó  $s_a$  uma notificação que informa sobre tal pedido utilizando o pacote do tipo *GMTP-Register*, como explicado na Seção 1.3.1.

Esse procedimento de formação de parceria é muito interessante porque pode reduzir o tempo de estabelecimento de conexão e reduzir a quantidade de requisições para o nó  $s_a$  que transmite um fluxo de dados  $P$ . Isto ocorre porque se permite que  $r_d$  descubra mais rapidamente outros candidatos a parceiros  $r_q$ , tal que  $\varphi(r_q, P) = 1$ , efetivando-se parcerias quando ocorrer  $C_i(r_d) \neq \{\emptyset\}$ .

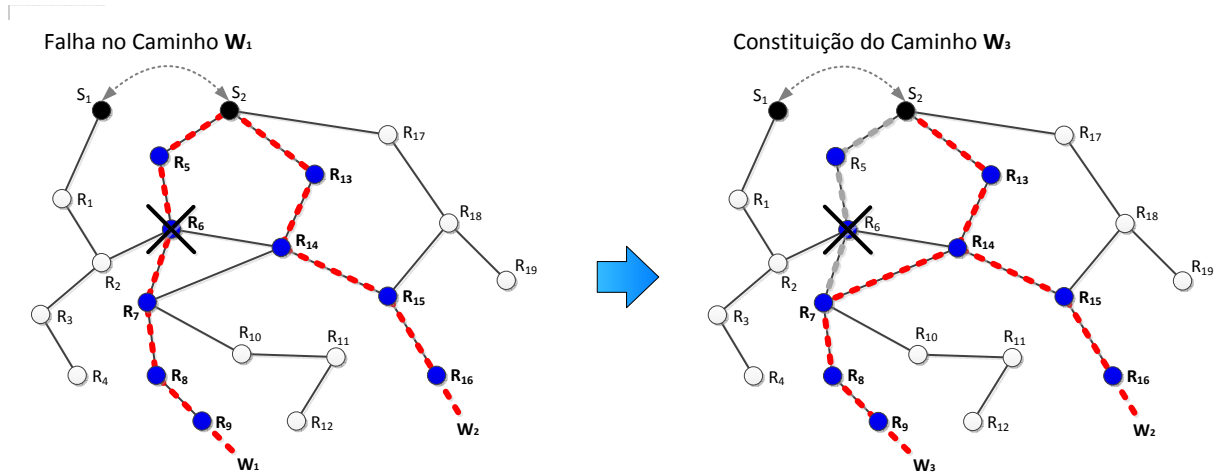


Figura 1.11: Cenário de falha do nó  $r_6$  em um caminho  $W_1$ , seguida de constituição de um novo caminho  $W_3$  formado pelo procedimento de formação de parceria intra  $W_v$ .

Além disso, uma ação específica é realizada se um nó  $w_m \in W_v$  circunstancialmente falhar. Para tratar estes casos, definiu-se que o nó  $r_d$  pode formar parcerias com os próprios nós  $r_d \in W_v^{\triangleleft}$ , tal que  $W_v^{\triangleleft} = \sim(\delta(w_{m+1}, W_v))$ , através de uma outra rota de rede que também alcance o nó  $s_a$  – isto pode acontecer devido à execução de algoritmos de roteamento



dinâmico *Intra-AS* (*Autonomous Systems*), por exemplo, o OSPF, e *Inter-AS*, por exemplo, o BGP [10].

Para entender o comportamento do GMTP em caso de falha de um nó  $r_d$ , observe a Figura 1.11. Se o nó  $r_6$  falhar e o nó  $r_{14}$  também estiver repassando  $P$ , uma nova parceria é formada transparentemente entre o nó  $r_7$  e o nó  $r_{14}$ . Isto porque o nó  $r_{14}$  interceptará os pacotes de controle de *keep-alive* que o nó  $r_7$  está transmitindo para o nó  $s_a$ . Mas, pode acontecer o caso de que  $\varphi(r_{14}, P) = 0$  e  $\varphi(r_{13}, P) = 0$  para o fluxo de dados  $P$ , portanto o pedido de conexão enviado pelo nó  $r_9$  alcançará o nó  $s_2$  como antes. Isto resultará na constituição de um novo caminho  $W_3 = W_1^{\triangleleft} \cup W_2^{\triangleleft}$ , tal que  $W_1^{\triangleleft} = \sim(\delta(r_6, W_1))$  e  $W_2^{\triangleleft} = \delta(\sim(W_2), r_{15})$ . Isto fará com que todo o caminho  $W_3$  repasse o fluxo de dados  $P$ . Como consequência, aumenta-se a possibilidade de parcerias futuras com nós  $r_d$  cujo pedido de conexão para obter  $P$  seja roteado pelo caminho  $W_3$ . Para este caso, criou-se o procedimento de formação de parceria por intersecção de caminhos  $W_v$ , detalhado mais adiante.

Na Seção 1.7.1, apresenta-se uma discussão geral sobre o comportamento do GMTP em outros casos de desconexões. O procedimento de formação de parceria intra  $W_v$ , apresentado nesta seção, está intimamente relacionado com o processo de estabelecimento de conexão do GMTP, detalhado mais adiante na Seção 1.4.2.

## 2. Formação de parcerias por intersecção de $W_v$

O procedimento de formação de parcerias por intersecção consiste na identificação de um nó  $w_m$  comum a dois ou mais caminhos  $W_v$ . Esse caso foi parcialmente introduzido no final da seção anterior, quando apresentou-se um cenário de falha de um nó  $r_d$ . Porém, o procedimento de formação de parceria introduzido nesta seção permite que o referido nó  $w_m$  comece replicar o fluxo de dados  $P$  mesmo quando  $C_i(w_m) = 0$ . Para este caso, o nó  $s_a$  transmite apenas um único fluxo de dados  $P$  para duas ou mais sub-caminhos que contém o nó  $w_m$ . Para entender melhor este caso, acompanhe a explicação a seguir com base na ilustração da Figura 1.12.

Seja o caminho  $W_1$ , ilustrado no Passo 1 da Figura 1.12. A formação de parceria por intersecção ocorre quando um nó  $r_d$  transmite um pedido de registro de participação ou um pedido de conexão para obter um fluxo de dados  $P$ , através de um caminho  $W_v$  que já esteja repassando  $P$ . Assim, após o registro de participação de  $r_9$  e a consequente constituição de

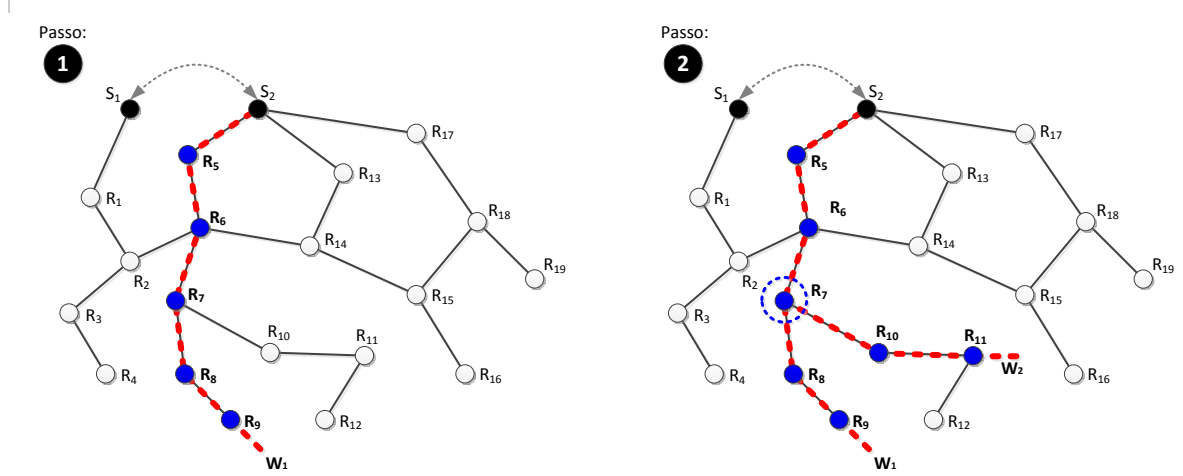


Figura 1.12: Cenário e passos para seleção de nós por intersecção de caminhos  $W_v$ .

$W_1$  (Passo 1), se qualquer um dos nós  $r_{10..12}$ , suponha  $r_{11}$ , enviar um registro de participação ou um pedido de conexão para obter  $P$  (Passo 2 da Figura 1.12),  $r_7$  se tornará candidato a parceiro de  $r_{11}$  (se ocorrer apenas o registro de participação), ou automaticamente começará a repassar o fluxo de dados  $P$  para  $r_{11}$ , caso  $r_{11}$  envie um pedido de conexão para obter  $P$ , em vez do simples registro de participação. Com o registro de participação de  $r_9$  e  $r_{11}$ , o nó  $s_a$  descobrirá, por intersecção, um sub-caminho de nós  $r_d \in (W_1 \cap W_2)$  (Passo 2).

No Algoritmo 3, resume-se os passos descritos anteriormente que o nó  $s_a$  executa a fim de realizar a formação de parcerias por intersecção. No GMTP, executa-se tal algoritmo quando o nó  $s_a$  recebe um pedido de conexão originado por um nó  $r_d$  para obter um fluxo de dados  $P$ . O nó  $s_a$  compara os caminhos conhecidos  $W_v \in W$  com o caminho  $W_{r_d}$  contido no pedido de conexão transmitido por  $r_d$ . Se  $s_a$  determinar um nó  $w_m$  em comum entre algum  $W_v$  e  $W_{r_d}$ ,  $s_a$  instrui o nó  $w_m$  a interceptar os próximos pedidos de conexão para obter  $P$ . Caso contrário,  $s_a$  aceita o novo pedido de conexão e todo o caminho  $W_{r_d}$  passa a ser alimentado com o fluxo de dados  $P$ , permitindo-se futuras parcerias intra ou por intersecção. Note que após este procedimento, o nó  $s_a$  passa a conhecer o caminho  $W_{r_d}$ .

Para o cenário ilustrado na Figura 1.12, o algoritmo identifica um sub-caminho  $W_1^\triangleleft = W_2^\triangleleft = \sim(\delta(r_7, W_1)) \cup \sim(\delta(r_7, W_2)) = \{r_7, r_6, r_5, s_2\}$ . Ao identificar tal intersecção, o nó  $s_2$  deve enviar apenas um fluxo de dados  $P$  para  $r_7$ , que o repassará para os nós  $r_d$  que enviar um pedido de conexão para obter  $P$ , ou seja, qualquer requisição originada a partir de qualquer  $r_d \in \delta(r_7, W_v)$ ,  $\forall W_v$ . No caso específico do pedido de conexão transmitido

pelo nó  $r_{11}$ , na resposta a este pedido transmitida pelo nó  $s_2$  em direção ao nó  $r_{11}$ ,  $s_2$  deve sinalizar que os pacotes de dados  $p_x \in P$  também devem ser repassados para  $r_{11}$ . Para isso,  $r_7$  deve manter um mapeamento entre o nome do fluxo de dados  $P$  e a lista dos nós  $r_d$  que tal fluxo deve ser repassado. Um aspecto técnico muito importante que deve ser ressaltado é que apenas o nó  $r_d$  que precisar repassar  $p_x \in P$  para seus nós  $c_f \in C_i(r_d)$  deve manter um mapeamento entre o fluxo de dados  $P$ , o endereço e porta em que tal fluxo deve ser entregue. 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 em modo multicast, faz-se necessária apenas uma entrada no mapeamento do nó  $r_d$  por fluxo de dados  $P$ , portanto utilizando-se pouco recurso de memória de um roteador.

O algoritmo explicado anteriormente permite repasse de pacotes de dados  $p_x$  levando em consideração o fluxo de dados de interesse  $P$ , portanto trata-se de uma abordagem centrada no conteúdo (*content centric networks*), e não centrada no sistema final (*host centric networks*). Sendo assim, essa proposta quebra um paradigma que tradicionalmente os pacotes de dados de uma aplicação são roteados com base no sistema final de destino. No caso do GMTP, diferentemente de qualquer outros protocolo existente, para o cenário ilustrado no exemplo supracitado, quando qualquer nó  $c_f \in \sum_{d=8}^{12} C_i(r_d)$  solicitar o fluxo de dados  $P$ , apesar de tal pedido ser enviado em direção ao nó  $s_a$ , o nó  $r_7$  analisará o pedido pelo nome do fluxo de dados  $P$  e decidirá por encaminhar os pacotes de dados  $p_x \in P$  para o respectivo nó  $r_d$  solicitante, naquele caso o nó  $r_{11}$ . Isto só é possível porque o nó  $s_a$  instrui o nó  $r_7$ , que é o primeiro nó comum entre os caminhos  $W_1$  e  $W_2$ , a se tornar transmissor de  $P$ , ou seja,  $\varphi(r_7, P) = 1$ . Como resultado, o nó  $r_7$  aciona um filtro para interceptar todo pacote de requisição para obter o fluxo de dados  $P$ . A partir desse instante, como o nó  $r_7$  interceptará todos os pedidos de conexão com interesse em obter  $P$ ,  $r_7$  responderá ao nó  $r_d$  solicitante como se fosse o nó  $s_2$ . Salienta-se também que há originalidade nesse algoritmo a partir do momento em que se permite que um nó  $s_a$  se comunique com um nó  $r_d$  a fim de delegar a responsabilidade em distribuir um determinado fluxo de dados  $P$ .

**Algoritmo 3:** matchPartnersByPathIntersection( $r_d$ : PeerRelay,  $p_x = GMTP\text{-}Request$ )

---

```

/*  $s_a$  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 name for  $P$ . */
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  $W_{r_d} \leftarrow$  getPacketFieldValue( $p_x$ , 'path'); /* Extracts  $W_{r_d}$  in  $p_x$  */
4  $W_P \leftarrow$  getKnownPathsOfFlow( $P, W$ );          /*  $W_P \subset W$  */
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  $P$ .  $s_a$  activates
       the flag 'relay' of the GMTP-Response packet. */
15     $W[\text{length}(W)] \leftarrow W_{r_d}$ ;
16    return GMTPResponse( $w_m$ , 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  $W_{r_d}$ . */
19  $W[\text{length}(W)] \leftarrow W_{r_d}$ ;
20 return GMTPResponse( $r_d$ , relay=0);

```

---

## 1.4 Distribuição do fluxo de dados $P$ através de $\eta$

No GMTP, a transmissão de dados é feita utilizando uma estratégia híbrida *pull/push* para obtenção do fluxo de dados  $P$ . O método *push* é adotado como padrão, onde os nós  $s_a$  iniciam a transmissão de  $p_x \in P$  para os demais nós  $w_m \in W_v$ , onde  $w_1 = s_a$ . Já o método *pull* é utilizado 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*. Os nós  $r_d$  e  $c_f$  mantêm seus próprios mapas de *buffer*, sendo que um nó  $r_d$  sempre terá um mapa de *buffer* mais atualizado do que os mapas de *buffer* dos nós  $c_f \in C_i(r_d)$ .

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  $P$ .

### 1.4.1 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 UUID (*Universally Unique Identifier*) de 128 bits [11]. Na sua forma canônica,  $P$  é representado por uma sequência de 32 dígitos hexadecimal, exibidos em cinco grupos separados por hífen, na forma de {8}-{4}-{4}-{4}-{12}. Por exemplo,  $P = 641f931f-d3ac-50e3-b625-537574541f1f$ .

Na prática, a aplicação GMTP em execução no nó  $s_a$ , que é responsável por gerar os pacotes de dados  $p_x \in P$ , define o identificador para um fluxo de dados  $P$  e o divulga 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 uma lista contendo os nomes dos fluxos 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, o que caracteriza um modelo distribuído de indexação de conteúdo.

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 [12]. 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 em grandes proporções das aplicações multimídia sem se preocupar como localizar um determinado conteúdo, extrapolando-se as barreiras administrativas de cada sistema de geração de conteúdos multimídia, bastante para isso apenas todos passarem a adotar o GMTP (quando desejado).

---

**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
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 $P$

Como uma das principais motivação do GMTP é promover a compatibilidade entre diferentes aplicações que o utiliza, afim de reduzir o tráfego de rede para um mesmo fluxo, faz-se necessária uma forma flexível de permitir que qualquer aplicação consiga reproduzir o fluxo de dados  $P$ . Para isto, incorporou-se um mecanismo para permitir que as aplicações consigam obter mais detalhes sobre a mídia transmitida, flexibilizando-se o acesso a um determinado conteúdo, instruindo a aplicação como o conteúdo deve ser decodificado para ser reproduzido e consumido (escutado ou visualizado) pelo usuário final. Nesse contexto, adotou-se o SDP (*Session Description Protocol*), definido na RFC 2327 [13], que permite um sistema final servidor descrever, de forma padronizada, o conteúdo mídia, tornando possível que os sistemas finais clientes possam interpretá-lo corretamente a fim de reproduzi-lo para o usuário final.

No caso do GMTP, o nó  $s_a$  gera uma descrição SDP e envia para os nós  $c_f \in C$  que, com base na descrição da mídia recebida, a interpreta e a reproduzir para o usuário 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 o 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, a aplicação em execução no nó  $s_a$  determina as informações da mídia as fornecem ao GMTP, através de passagem de parâmetro via socket GMTP. Em seguida, o GMTP gera um pacote do tipo *GMTP-MediaDesc*, que encapsula o conteúdo de descrição SDP e fica pronto para transmiti-lo para o nó  $r_d$  interessado em obter o fluxo de dados  $P$ , além de poder disseminá-lo para seus nós parceiros  $r_q$ . No Quadro 5, apresenta-se um exemplo de uma mensagem SDP, que deve ser disponibilizada no campo de dados do pacote do tipo *GMTP-MediaDesc*. Em seguida, apresenta-se o significado de cada atributo de uma mensagem SDP.

**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 1.4.1 */
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 1.6 */
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 [14] */
14 a=framerate:23.0

```

---

- $v$ , a versão do SDP;
- $o$ , a lista de nós  $s_a$  que a distribui;
- $s$ , o nome da mídia, como discutido na Seção 1.4.1;
- $i$ , o título da mídia;
- $u$ , a URI que descreve detalhes sobre a mídia;
- $c$ , as informações de conexão, como o tipo da rede, 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 1.6;
- $m$ , 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.

No exemplo apresentado no Quadro 5, utiliza-se a primeira versão do protocolo SDP e descreve-se a transmissão de dois fluxos  $P$  (Linhas 10 e 11), sendo um deles 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  $p_x \in P$  – este assunto será discutido com mais detalhes na Seção 1.6. 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-se entre 16000 Bytes e 20000 Bytes e, para o vídeo, variando-se 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 – este assunto será retomado na Seção 1.4.8.

### 1.4.2 Estabelecimento de conexão e compartilhamento 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 1.13, 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).

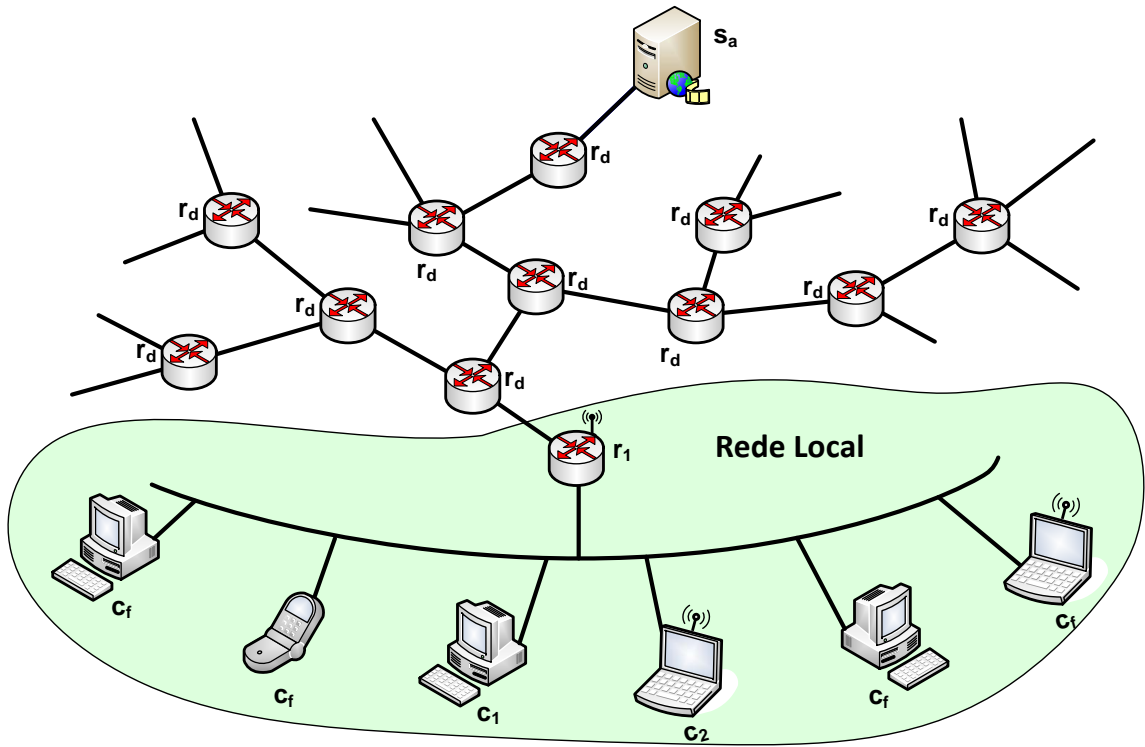


Figura 1.13: Exemplo de rede para o estabelecimento de conexão do GMTP.

A regra geral é que um nó  $r_d$  deve consultar a tabela de recepção de fluxo de dados (Seção 1.3.2) todas as vezes que receber um pacote do tipo *GMTP-Request*, 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.

### 1.4.3 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 1.14, onde ilustra-se um exemplo de conexão na Fase 1, considere:

- $P$ , um fluxo de dados cujo nome é definido por:  
 $72c44591-7d82-427c-825f-722f015787c1$ ;
- $s_1$ , 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



Na Fase 1 do processo de estabelecimento de conexão, considera-se que não existe nenhum nó  $c_f \in C_i(r_1)$  recebendo o fluxo de dados  $P$ . Sendo assim, quando o pacote *GMTP-Request* alcançar o nó  $r_1$  (Passo 2 da Figura 1.14), o nó  $r_1$  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  $P$ , como descreveu-se na Seção 1.3.1. Isto significa que a execução do procedimento *registerRelay*( $s_a, p_x$ ), 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 1.14 e destacado na cor vermelha), conforme discutiu-se na Seção 1.3.3.

Em seguida, ao receber o pacote do tipo *GMTP-Register-Reply*, como resposta ao registro de participação, o nó  $r_1$  cria um canal multicast e envia um pacote do tipo *GMTP-RequestNotify* para um ou mais clientes  $c_f \in C_i(r_1)$  (Passo 4 da Figura 1.14). Esta notificação sinalizará os nós clientes  $c_f$  qual canal multicast seus respectivos sockets devem ser “sintonizado”. No caso do exemplo supracitado, o nó  $c_1$ , após sintonizar o socket no canal multicast informado pelo nó  $r_1$ , começa a receber os pacotes de dados  $p_x$  do tipo *GMTP-Data* ou *GMTP-DataAck* (Passo 5 da Figura 1.14). Na Seção 1.4, discute-se em detalhes sobre os aspectos de transmissão de pacotes do tipo *GMTP-Data* ou *GMTP-DataAck*.

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 15 do Algoritmo 1 e nas Linhas 11 e 17 do Algoritmo 2 (Seção 1.3.1).

Como resultado da Fase 1, gera-se uma nova entrada na tabela de recepção de fluxos de dados do nó  $r_d$ , tal como ilustra-se na Figura 1.15. 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  $1900$ , por onde 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 15 of Algorithm 1 and in
   Lines 11 and 17 of the Algorithm 2. */
1  $destAddress \leftarrow getCtrlChannel()$ ; /* 238.255.255.250:1900 */
2  $setPacketFieldValue(p_x, 'destinationAddress', destAddress)$ ;
3  $P \leftarrow getPacketFieldValue(p_x, 'flow')$ ; /* Extracts  $P$  in  $p_x$  */
4  $errorCode \leftarrow getPacketFieldValue(p_x, 'errorCode')$ ;
5 if  $errorCode \neq NULL$  then
6    $removeClientsWaitingForFlow(P)$ ; /* See Algorithm 1 */
7    $sendPkt(p_x)$ ;
8   return 0;
9 end
10  $channel \leftarrow getPacketFieldValue(p_x, 'channel')$ ;
11 if  $channel \neq NULL$  then
12   /* Node  $r_d$  is already receiving  $P$  and clients  $C_i(r_d)$ 
13     must know the media description. */
14    $mediaDescription \leftarrow getMediaDescription(P)$ ;
15    $setPacketFieldValue(p_x, 'data', mediaDescription)$ ;
16   /* In Algorithm 1, Line 5,  $c_f$  nodes are added in a list
17     of clients waiting for flow  $P$ . Now,  $r_d$  notifies
18     them, wait confirmation (ACKs) from them and start
19     relaying  $p_x \in P$  to them through given channel. */
20    $sendPkt(p_x)$ ;
21    $C_i(r_d) \leftarrow getClientsWaitingForFlow(P)$ ;
22    $waitAck(C_i(r_d), P)$ ;
23 else /* Let  $C_i(r_d)$  know  $r_d$  is waiting for registration. */
24    $setPacketFieldValue(p_x, 'waitingRegistration', true)$ ;
25    $sendPkt(p_x)$ ;
26 end
27 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 1.15: Tabela de recepção de fluxos de dados após a Fase 1.

#### 1.4.4 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 1.16. 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-se o pacote do tipo *GMTP-Request-Reply*. Este procedimento ocorre no registro de participação, especificamente no trecho de código definidos entre as Linhas 2-8 do Algoritmo 1. Em seguida, o pacote do tipo *GMTP-Request-Reply* é transmitido para o nó  $c_2$ , como descreve-se no trecho de código entre as Linhas 10-16 do Algoritmo 6. Tal procedimento se repete para cada novo nó  $c_f \in C_i(r_1)$  interessado em obter  $P$ .

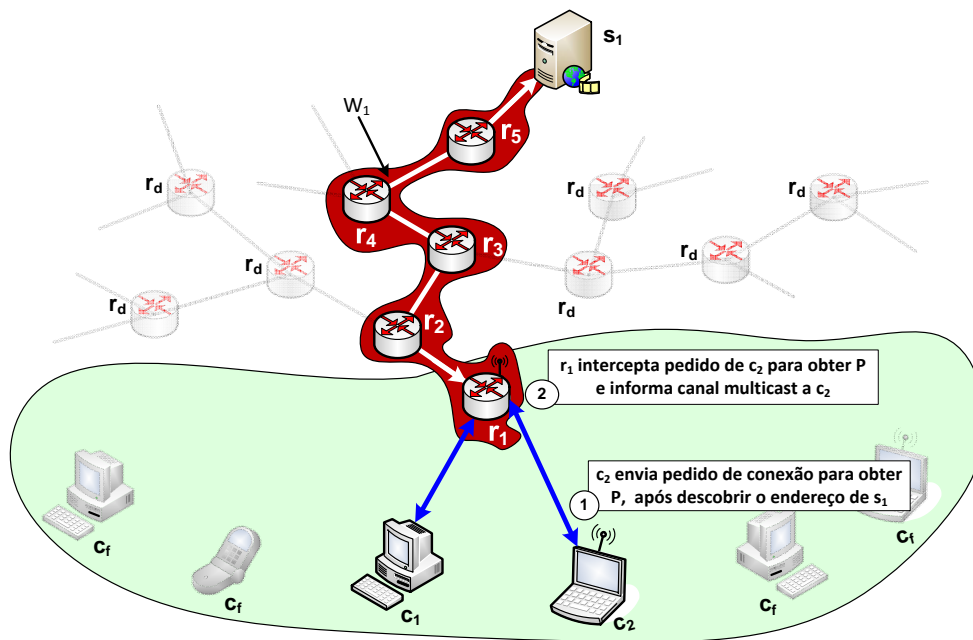


Figura 1.16: Passos do processo de estabelecimento de conexão do GMTP (Fase 2).

### 1.4.5 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  $p_x \in P$  e caminhos  $W$  alternativos em caso de falha e/ou desconexões de algum nó parceiro  $r_q$ . Ao considerar os aspectos discutido na Seção 1.3.3, nota-se que na Fase 1 e 2 utiliza-se os modos de formação de parcerias intra  $W$  e por intersecção, porém ainda resta fazer uso do modo de formação de parceria por combinação de  $W$  (Figura ??). Na fase 3 de conexão, o GMTP explora tal recurso.

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 ilustrado na Figura 1.17. Note que a lista de nós parceiros enviada pelo nó  $s_a$  é construída usando o algoritmo ?? e, portanto, os nós  $s_a$  funcionam como um indexador (*tracker*) de nós parceiros  $r_q$ , executando uma pré-seleção de nós parceiros para  $r_3$ . Esta pré-seleção ajuda o nó  $r_3$  a selecionar os melhores parceiros disponíveis, de acordo com os critérios definidos em ??.

Diante do exposto, faz-se necessário registrar três procedimentos importantes realizados pelo GMTP na Fase 3:

1. um nó  $r_d$  pode enviar periodicamente requisições do tipo *GMTP-RelayQuery* para o servidor a fim de descobrir melhores parceiros e aumentar seu leque de opções. Apesar disso, a quantidade de possíveis parceiros de um nó  $r_d$  não significa, necessariamente, que tal nó mantém a mesma quantidade de parcerias efetivas para obter um fluxo de dados  $P$ . Os parâmetros de periodicidade de requisições do tipo *GMTP-RelayQuery* e a quantidade máxima de parcerias efetivas pode ser alteradas pelo administrador de  $r_d$  e tem valores padrões de 10 minutos e 5 nós, respectivamente;
2. como ilustra-se na Figura 1.18, apenas na Fase 3, permite-se requisições do tipo *GMTP-Request* partindo de um nó  $r_{d_3}$  em direção a outro nó  $r_{d_2}$ , que irá enviar um resposta do tipo *GMTP-Response* se  $r_{d_1}$  enviar uma chave secreta aceita por  $r_{d_2}$  e encaminhada para  $r_{d_1}$  pelo nó  $s_a$ , que a obteve de  $r_{d_2}$  no processo de registro de participação discutido na Seção 1.3.1. Note que nesse caso, mesmo se nó  $r_{d_2}$  não estiver recebendo o fluxo de dados  $P$  de interesse de  $r_{d_1}$ , o nó  $r_{d_2}$  deve estabelecer uma conexão (Fase 1) para obtê-lo e então repassar  $P$  para  $r_{d_1}$ ;

3. como se considera uma arquitetura híbrida P2P/CDN, o nó  $s_a$  pode facilmente realizar um mecanismo de balanceamento de carga, incluindo na lista, como se fosse um nó  $r_d$ , um outro nó  $s_a$ , levando-se em consideração, inclusive, todos os critérios estabelecidos na Seção ??.

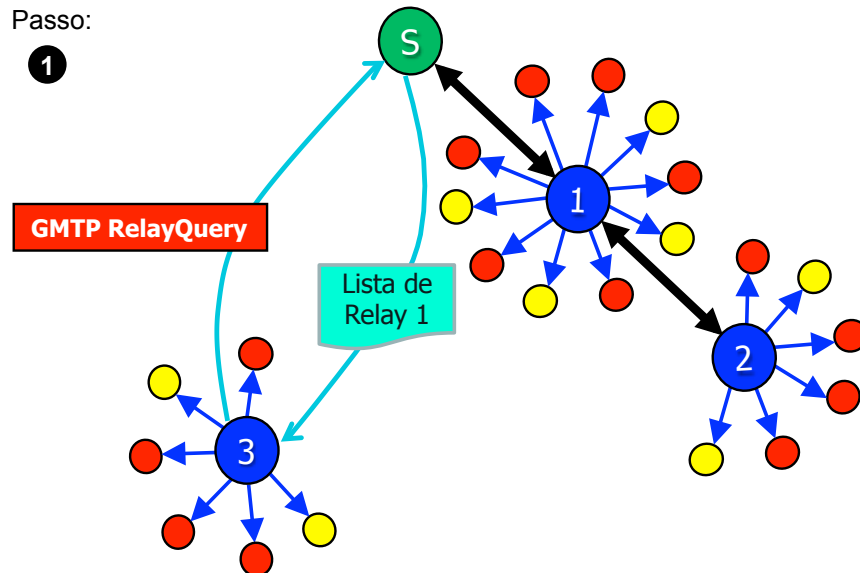


Figura 1.17: Fase 3 de conexão do GMTP (Passo 1).

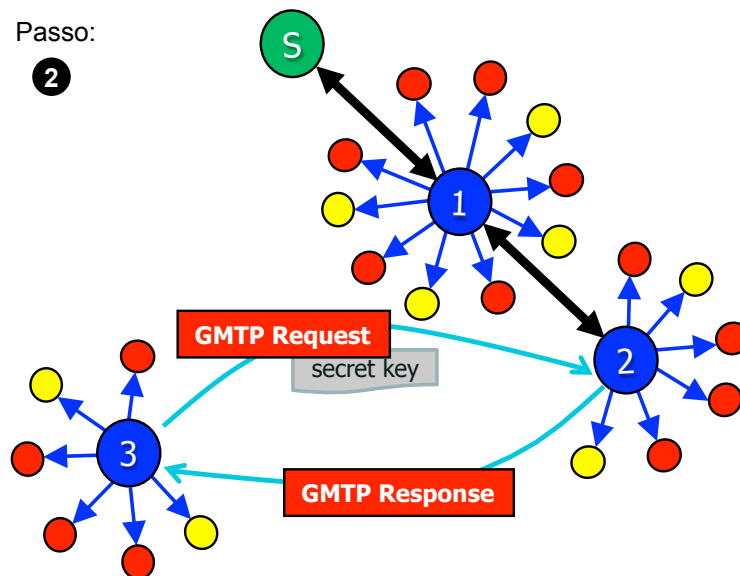


Figura 1.18: Fase 3 de conexão do GMTP (Passo 2).



### 1.4.6 Manutenção da conexão

Note que, por serem pacotes de controle, estes pacotes são transmitidos com garantia de entrega, similar ao mecanismo adotado pelo TCP.

Após o registro de participação, o nó  $r_d$  deve enviar periodicamente sinalizações de controle (*polling*) sobre sua participação na rede de favores  $\eta$ . Este procedimento deve ser feito usando o pacote do tipo *GMTP-Ack* em um tempo  $t = \max(300, t_{user}) - RTT$ , onde  $t$  e  $RTT$  são definidos em segundos, e  $t_{user}$  é um parâmetro de tempo definido pelo usuário. 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 no máximo 3 vezes, quando  $r_d$  deve considerar a conexão finalizada por tempo de expiração (*timeout*) e enviar um pacote do tipo *GMTP-Reset*. Na RFC 5482 [15], discute-se sobre outros aspectos de expiração no tempo que podem ser adaptadas para o GMTP.

### 1.4.7 Compartilhamento de $P$ entre $s_a$

Além do processo transparente para se obter um fluxo de dados  $P$  empregado no GMTP, como os nós  $s_a$  constituem uma rede CDN, estes podem negociar entre si o envio e a recepção de um fluxo de dados  $P$  de acordo com as requisições submetidas aos nós  $r_d$ . Desta forma, se um nó  $r_d$  enviar uma requisição para obter  $P$  de um evento  $\mathcal{E}$  a um nó  $s_a$  e este não esteja recebendo tal fluxo,  $s_a$  poderá solicitá-lo a outros nós  $s_a$  da CDN que participa. A partir desse ponto, o nó  $s_a$  passará a servir o nó  $r_d$  normalmente. Como no GMTP se faz uso indireto dessa função das redes CDNs, a qual já está consolidada, resolveu-se suprimir maiores detalhes a respeito deste assunto. Para maiores informações sobre a função de distribuição de conteúdos ao vivo entre os servidores de uma rede CDN, o leitor pode consultar as referências [16–18].

Desta forma, o processo de conexão do GMTP é fundamental para a efetiva distribuição de mídias ao vivo, pois permite-se que as aplicações compartilhem fluxos de dados entre si, mesmo que tais aplicações 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 um a outra. Como resultado, reduz-se para 1 o número de transmissões para um mesmo fluxo de dados  $P$  destinados a uma mesma rede ou para um subconjuntos de redes adjacentes. Além dessa diferença substancial, a forma de conexão do GMTP supre uma antiga deficiência das soluções tradicionais de transmissão multicast, as quais os nós clientes, camada de aplicação, 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, que obrigatoriamente tem que ser realizada em todos os nós roteadores de um determinado caminho. Até o presente momento, não se conhece nenhuma solução que permita configuração dinâmica de canais multicast da forma que foi explicada nesta seção, com benefícios diretos para a aplicação e para a rede, fazendo-se uso dos recursos computacionais e de rede de forma mais apropriada, como será discutido no Capítulo ??.

#### 1.4.8 Envio e recebimento de $p_x \in P$ em $\eta$

Após o estabelecimento de conexão, os nós  $r_d$  trocam dados entre si em modo unicast a fim de obter  $P$ , constituído de pacotes  $p_x$  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. Nesta seção, detalha-se como o GMTP executa as funções para transmissão que, em ambos os casos, realiza-se controle de congestionamento, sendo tal função descrita na Seção ??.

Após o processo de estabelecimento de conexão, o GMTP entra no estado de transmissão de dados. Se 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*. Nesta seção, discute-se o funcionamento do mecanismo de transmissão e recepção de dados no GMTP.

Para o transporte de dados da aplicação, um  $s_a$  ou um  $r_d$  deve criar pacotes do tipo *GMTP-Data* ou o *GMTP-DataAck* e enviá-los aos nós  $c_f$  através do socket correspondente à conexão estabelecida. Embora o protocolo GMTP transmite dados sem garantia de entrega, em alguns casos, dados de controle podem ser transmitidos de forma confiável. Nestes casos, durante a transmissão de dados, um nó GMTP utiliza-se do pacote do tipo *GMTP-Data* para enviar dados, ao passo que utiliza-se pacote *GMTP-Ack* para confirmar a recepção de

pacotes, ou ainda, utiliza-se *GMTP-DataAck* para enviar pacotes de dados e ao mesmo tempo confirmar a recepção de pacotes de dados vindos da direção oposta (*piggyback*).

### 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 no formato de uma estrutura de dados do tipo array, onde cada posição é utilizada para armazenar um pacote  $p_x$  (Figura 1.19). 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) nós interessados, seja em modo unicast e/ou em modo multicast. Isto porque é possível que um nó  $r_{d_1}$  repasse  $p_x$  para um outro nó  $r_{d_2}$  (unicast) ao mesmo tempo que  $r_{d_1}$  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, sendo determinado um valor mínimo e um valor máximo, sendo estes permanecendo fixos durante todo o ciclo de vida de uma conexão GMTP. Essa decisão é importante porque permite um nó  $r_d$  alocar previamente o recurso necessário para um determinado fluxo de dados  $P$ . O tamanho do buffer é especificado pelo nó  $s_a$  e sempre é propagado para os demais nós no cabeçalho do pacote do tipo *GMTP-MediaDesc*, como discutido a seguir. Este aspecto é muito importante, pois a aplicação que deve tomar tal decisão, de acordo com o tipo e formato da mídia a ser transmitida. Para o GMTP, é importante apenas ter conhecimento sobre o tamanho do buffer para executar ações de descarte de  $p_x$ .

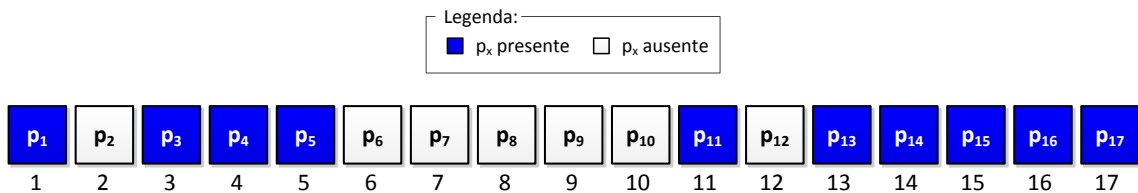


Figura 1.19: Exemplo da estrutura do buffer de envio e recepção de um nó GMTP com tamanho de 17  $p_x$ .

**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 1.22, trata-se de uma estrutura de dados binária que determina se um pacote  $p_x$  está ou não presente no buffer de um respectivo nó GMTP.

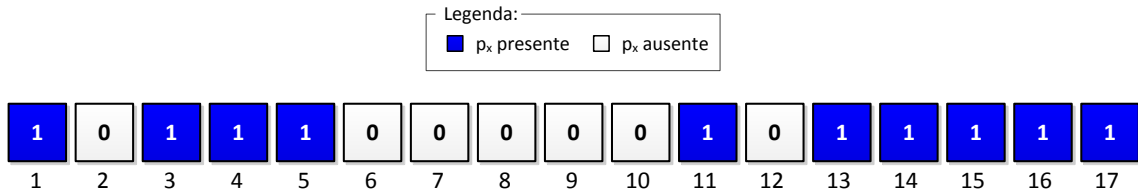


Figura 1.20: Exemplo do mapa de buffer de um nó GMTP com tamanho de 17  $p_x$ .

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 GTMP pode enviar o mapa de buffer completo, como ilustrado na Figura 1.22, 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;

As trocas do mapa de buffer entre os nós GMTP ocorrem sob demanda, utilizando o método *pull*, uma vez que o método *push* é utilizado por padrão. Neste caso, 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, ou seja, caso sejam perdidos, o GMTP garante sua retransmissão. Para

isto, o GMTP utiliza o mecanismo básico de envio e confirmação utilizando o pacote do tipo *GMTP-DataAck* ou *GMTP-DataAck*. No caso de falha na execução de uma requisição utilizando o método *pull*, o nó GMTP pode reavaliar a necessidade de retransmitir o pedido, pois é possível que os  $p_x$  ausentes já tenham sido expirados e requisitá-los novamente não fará mais sentido.

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 1.22, o nó GMTP preenche o pacote do tipo *GMTP-DatPull-Request* com a sequência 2;6-10;12. Ao receber esta sequência, o nó parceiro  $r_q$  responde com o pacote do tipo *GMTP-DatPull-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:** 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 descarta seletivo de pacotes, primeiro os que tenham menos impacto na qualidade da mídia, por exemplo, pacotes de dados contendo quadros B (codificação mpeg). Isto não impede que o vídeo seja reproduzido, porém com perda de qualidade, ao passo que permite-se a transmissão de conteúdo com os recursos disponíveis;
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  $p_x$ .

## 1.5 Controle de Congestionamento em $\eta$

No GMTP, disponibiliza-se um arcabouço para adição de novos algoritmos de controle de congestionamento de forma modularizada. Desta forma, permite-se a adição e remoção de novos algoritmos de controle de congestionamento. Atualmente, o GMTP oferece dois

algoritmos, um voltado para transmissões em modo unicast e outro voltado para transmissões em modo multicast.

Na prática, definiu-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. 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 um algoritmo de janela deslizante baseado em uma equação cúbica, com suporte aos protocolos RCP [1] e ConEx [1]. Já em modo de transmissão multicast, executa-se um algoritmo baseado em relatórios transmitidos pelos nós  $l_w$ , eleitos em cada rede controlado por um nó  $r_d$ , tal que  $l_w \in C_i(r_d)$ . Como ilustrado na Figura 1.21, para a parte do algoritmo que funciona em modo unicast, dar-se o nome de *GMTP Unicast Congestion Control* (GMTP-UCC), ao passo que para a parte do algoritmo que funciona em modo multicast, dar-se o nome de *GMTP Multicast Congestion Control* (GMTP-MCC).

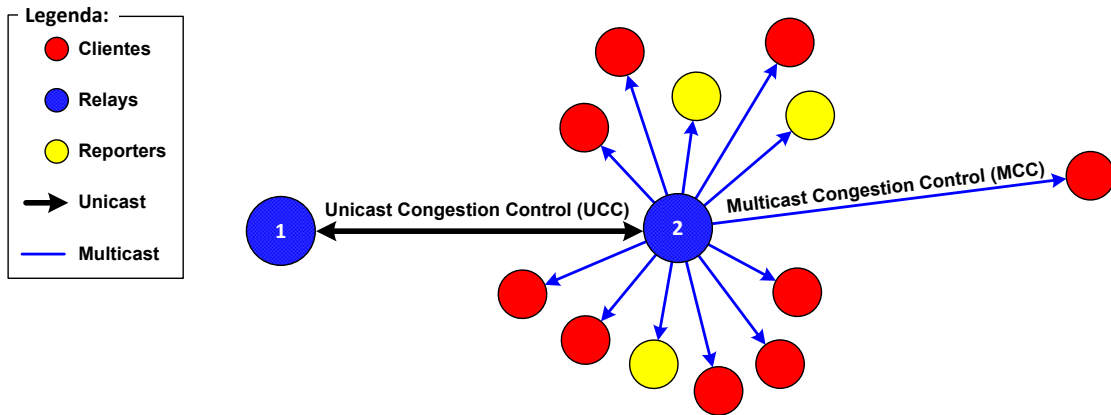


Figura 1.21: Organização do algoritmo de controle de congestionamento no GMTP.

### 1.5.1 Controle de Congestionamento Unicast

O GMTP-UCC funciona de forma similar ao protocolo RCP (Rate Control Protocol) [19], 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 Computador Compartilhado, ou *Processor Sharing* (PS), por exemplo, um roteador [2]. Nesse caso, 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  $t$ .

Dessa forma, Nandita et. al [19] argumenta que para um roteador funcionar de forma equânime, tal roteador 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 [19] determinou a Equação 1.1, onde  $R(t)$  é a taxa de transmissão que deve ser oferecida para cada fluxo de dados que passa pelo roteador. Pela Equação 1.1, estima-se a largura de banda disponível em um determinado canal, representado pela porção  $\alpha(C - y(t)) - \beta \frac{q(t)}{d_0}$  (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  $\frac{T}{d_0}$ , resultando na Equação 1.2, onde:

$$R(t) = R(t - d_0) + \frac{\alpha(C - y(t)) - \beta \frac{q(t)}{d_0}}{\hat{N}(t)} \quad (1.1)$$

$$R(t) = R(t - T) \left[ 1 + \frac{\frac{T}{d_0} \left( \alpha(C - y(t)) - \beta \frac{q(t)}{d_0} \right)}{C} \right] \quad (1.2)$$

- $d_0$ , é a média móvel dos valores de  $RTT_s$ , calculada através da Equação 1.3, onde  $\theta$  é o ganho e corresponde a 0.02. Note que quanto maior o valor de  $\theta$ , mais rápida será a convergência de  $d_0$  ao valor de  $RTT_s$ .

$$d_0 = \theta \times RTT_s + (1 - \theta) \times d_0 \quad (1.3)$$

- $T = \min(RTT_{user}, d_0)$ , sendo  $RTT_{user}$  um tempo definido pelo usuário caso seja necessário atualizar  $R(t)$  mais rápido do que o tempo de  $d_0$ ;
- $R(t - T)$ , 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  $d_0$ ;
- $q(t)$ , é o tamanho instantâneo da fila de repasse, em bytes;

- $\alpha$  e  $\beta$ , 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:

1° Todo nó  $r_d$  mantém uma única taxa de transmissão  $R(t)$ , que é oferecida para todos os fluxos de dados passando por  $r_d$  em um certo instante  $t$ . Cada nó  $r_d$  atualizada  $R(t)$  aproximadamente a cada RTT.

2° Todo pacote dos tipos *GMTP-Ack*, *GMTP-Data* ou *GMTP-DataAck* carrega duas informações de controle (campo no cabeçalho):

- *taxa de transmissão proposta* ( $R_p$ ): corresponde à taxa de transmissão necessária para transmitir um fluxo de dados  $P$ , em geral, calculada pelo nó  $s_a$ ;
- *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  $c_f \in (C_i(t_u) \cup C_i(t_{u+1}))$ .

3° No início de uma transmissão de um fluxo de dados  $P$ , o nó  $s_a$  transmite um pacote  $p_x$  com o valor de  $R_p$  correspondente à taxa de transmissão desejada para transmitir o referido fluxo  $P$ , com o valor para  $RTT_s = \infty$ . A taxa de transmissão desejada  $R_p$  deve ser calculada pela aplicação, de acordo com a taxa de bits da mídia a ser transmitida e repassada à instância do GMTP no nó  $s_a$ .

4° Todo nó  $w_m = r_d$  que receber um pacote  $p_x$ , se  $R(t) < R_p$ , então  $R_p \leftarrow R(t)$ , caso contrário nenhuma modificação é realizada nesse campo. Nesse ínterim, se existir pelo menos um nó  $c_f \in C_i(w_m)$  interessado em obter os pacotes  $p_x \in P$  (Seção 1.4.2),  $w_m$  executa as seguintes ações:

- repassa  $p_x$  para seus nós  $c_f$  em modo multicast (Seção 1.4.8);
- cria um pacote *GMTP-Ack* contendo  $R_p$  e o envia de volta para seu nó parceiro  $r_q = w_{m-1}$ . O pacote *GMTP-Ack* também carrega um campo de  $RTT_s$ . Quando



$w_m$  receber um pacote *GMTP-Ack*, deve-se utilizar  $RTT_s$  para atualizar a média móvel do RTT no intervalo de  $d_0$ .

5° O nó  $w_m$  deve usar  $R_p$  como a nova taxa de transmissão para enviar os próximos pacotes de dados  $p_x$  para seu nó parceiro  $r_q = w_{m+1}$ . Assim,  $R_p$  é a menor taxa de transmissão oferecida ao longo do caminho  $W_v$  no intervalo de  $RTT_s$ .

6° Todo nó  $r_d$  atualiza periodicamente sua taxa de transmissão local  $R(t)$  de acordo com a Equação 1.2.

Sendo assim, no caso do GMTP, 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  $t_2$ . Com isto, se existir largura de banda disponível entre  $t_1$  e  $t_2$ , ou seja,  $C - y(t) > 0$ , então o GMTP 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 reduzirá a taxa de transmissão igualmente para todos os fluxos, inclusive para o fluxo entre  $t_1$  e  $t_2$ . Especificamente, a largura de banda necessária para repassar todos os pacotes  $p_x \in P$  que estão na fila de roteamento no intervalo de um RTT corresponde à  $\frac{q(t)}{d_0}$  [2].

### **Escolha do algoritmo RCP em detrimento ao XCP:**

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 as taxas de transmissão  $R_{rcp}(t)$  e  $R_{xcp}(t)$ , respectivamente, para emular a taxa de transmissão do PS, definida por  $R_{ps}(t)$ , é bastante diferente. Dito isto, o grande dilema foi decidir qual dos dois protocolos seria mais adequado para o GMTP-UCC. Para tomar tal decisão, estudou-se a diferença entre tais protocolos e decidiu-se utilizar o RCP, de acordo com o que se discute a seguir.

Especificamente, a diferença está no tipo de informação enviada para um nó transmissor de um fluxo de dados para atualizar o valor de  $R_{rcp}(t)$  ou de  $R_{xcp}(t)$ . O XCP continuamente tenta convergir a taxa de transmissão para um ponto de equilíbrio onde todos os transmissores transmitirão pacotes de dados a uma taxa de transmissão  $R_{xcp}(t)$ , ao passo que o RCP calcula

uma única taxa de transmissão que deve ser utilizada por todos os nós transmissores em um certo instante  $t$ . Apesar dessa diferença suscinta, deve-se entender minuciosamente o que isto significa.

No caso do XCP, o protocolo aumenta ou diminui a janela de congestionamento de um fluxo de dados de acordo com o tamanho atual da sua janela de congestionamento. Isto ocorre porque 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 equânimes para todos os fluxos de dados.

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  $d_0$  que, no caso do GMTP, tal valor não é definido entre o nó  $s_a$  e o nó  $c_f$ , como na versão original do RCP. Em vez disso, o tempo  $d_0$  é definido entre dois nós  $r_d$  consecutivos em um caminho  $W_v$ , tal que  $r_d \in T$ .

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 de convergência entre a taxa de transmissão praticada  $R_{rcp}(t)$  e a  $R_{ps}(t)$ . 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  $R_{rcp}(t)$ , necessita-se apenas o tamanho da fila e da 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 para a distribuição de conteúdos multimídia ao vivo. Isto porque, ao tempo que 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  $s_a$ , como discutiu-se anteriormente.

Ao perceber essa diferenças, nandita et al. PLATAR O GRÁFICO

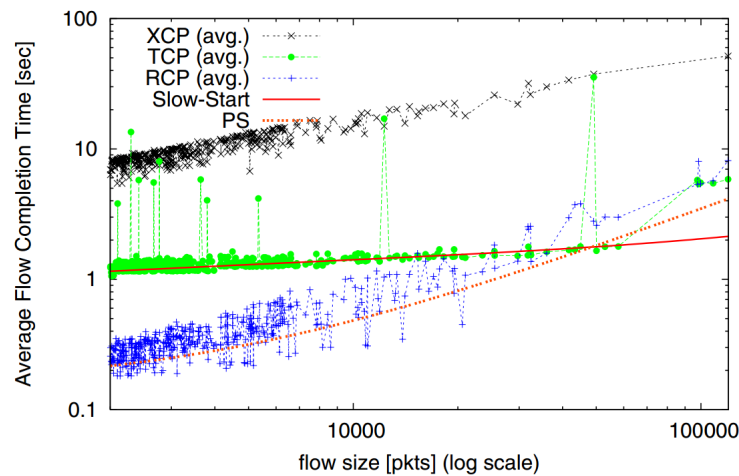


Figura 1.22: Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP, para  $C = 2.4 \text{ Gbps}$ . Os fluxos foram injetados na rede com base na distribuição pareto, com  $E[L] = 25 \text{ pacotes}$  e  $\text{shape} = 1.2$ . Extraído de [2].

## 1.5.2 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  $\eta_{sub} = r_d \cup C_i(r_d)$ . Na prática, os nós da rede  $\eta_{sub}$  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 da taxa de transmissão para um fluxo de dados  $P$  seja tão próximo ao valor da taxa de transmissão que o fluxo TCP utilizaria caso fosse transmitido na rede, portanto 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 [20]). 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 [21]. 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*). Algoritmos desse tipo são adotados em diversos protocolos, como é o caso dos CCIDs 3 e 4 do DCCP [22, 23]. Em linhas gerais, o algoritmo TFRC funciona da seguinte forma:

- 1° o receptor mede a taxa de perda de pacotes e envia essa informação para o transmissor;
- 2° o transmissor usa esse relatório para medir o RTT até o receptor;
- 3° o transmissor utiliza a Equação 1.4 para determinar qual será a sua próxima taxa de transmissão em função do relatório de perdas e o RTT obtidos anteriormente;
- 4° o transmissor então ajusta sua taxa de transmissão para o valor calculado no passo anterior.

$$R(p) = \frac{s}{RTT \times \left( \sqrt{\frac{2 \times p}{3}} + \left( 12 \times \sqrt{\frac{3 \times p}{8}} \right) \times p \times (1 + 32 \times p^2) \right)} \quad (1.4)$$

Na Equação 1.4 [24],  $T$  é a taxa de transmissão medida em bytes/segundo definida em função de  $s$ , que é o tamanho do pacote medido em bytes;  $RTT$ , é o RTT entre o nó transmissor e o receptor, medido em segundos e  $p$ , a taxa de perda de pacotes observado pelo nó receptor.

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 GMTP 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 1.7.2, descreve-se o procedimento para eleger os nós  $l_w$ .
- 2° Os nós  $l_w$  calculam a taxa de transmissão utilizando a Equação 1.4, ao invés 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 [20, 24], 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;
- 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 1.4, o GMTP-MCC utiliza a Equação 1.3, com  $\theta = 0.25$ , padrão do TCP;

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

Um aspecto importante na medição do RTT está relacionado com o início de uma conexão GMTP, pois não se sabe o valor para inicial para RTT até o final do processo de estabelecimento de uma conexão. Nesse caso, deve-se utilizar um valor consideravelmente alto para evitar taxas de transmissões maiores do que a rede tem capacidade de suportar. No GMTP, utiliza-se o valor inicial de RTT igual a  $150\text{ ms}$ . 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 1.4, caso o respectivo nó  $c_f$  seja eleito um nó relator.

## 1.6 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 (Figura 1.23).



Figura 1.23: Um nó  $r_d$  mal-intencionados podem poluir o sistema com conteúdos que não foram gerados pelo nó  $s_a$ .

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  $p_x$ .

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, como discutido nas Seções 1.3 e 1.4.

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  $\varphi(w_m, P) = 1$ , ou seja, se  $w_m = t_u \in T$ , 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 1.5, 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  $p_x \in P$ ;

$$pc(t) = \left\lfloor bs(t) \times \left(1 - \frac{1}{|W_v^\triangleleft|-1}\right) \right\rfloor \quad (1.5)$$

Sendo assim, quanto mais distante um nó  $w_m$  estiver do nó  $s_a$ , que gera o fluxo de dados  $P$ , mais pacotes  $p_x \in P$  deve-se verificar sua autenticidade. 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 manipular os referidos pacotes de dados para que seja possível verificar sua autenticidade. Este procedimento é explicado a seguir.

### 1.6.1 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  $p_x \in P$ .



**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 node  $r_d$  to
   verify the packet  $p_x$  authenticity executing the
   Algorithm 8.
*/
1 data  $\leftarrow$  getPacketFieldValue( $p_x$ , 'data');
2 hashValue  $\leftarrow$  hash(data);
3 signature  $\leftarrow$  encrypt( $K_{s_a}^-$ , hashValue);
4 setPacketFieldValue( $p_x$ , 'signature', signature);
5 return  $p_x$ ;

```

---

**1.6.2 Verificação de autenticidade de  $p_x \in P$** 

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

Dada a quantidade  $pc(t)$  de pacotes que  $w_m$  deve verificar suas respectivas autenticidades, o nó  $w_m$  escolhe aleatoriamente (distribuição uniforme) os pacotes  $p_x$  disponíveis no buffer de recepção, gerando um conjunto  $P' \subset P$ . Uma vez definido  $P'$ ,  $w_m$  executa o procedimento de verificação de autenticidade que funciona a seguinte forma. Para cada pacote  $p_x \in P'$ , extrai-se a assinatura do pacote  $p_x$ , gerada pelo nó  $s_a$ , como explicado na Seção 1.6.1. 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]  $\leftarrow$  (hash(data) = decrypt( $K_{s_a}^+$ , signature));
6 end
7 return verifiedPackets;

```

---

### 1.6.3 Habilitar ou desabilitar o procedimento de segurança

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 1.6.2. Para isto, o nó  $s_a$  ativa o da 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 do conteúdo de dados sendo transportados e que poderá 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 1.6.2. 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.

#### 1.6.4 Obter a chave pública $K_{s_a}^+$ de $s_a$

Um nó  $r_d$  obtem 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 1.4.1. 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 X, Seção 1.3.1.

Após obter o referido certificado digital do nó  $s_a$ , o nó  $r_d$  pode realizar *cache* de tal conteúdo para que os próximos nós  $c_f$  e evitar 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 salvá-lo, por meio de *upload*, por exemplo, manualmente 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  $s_a$ .

## 1.7 Outras considerações sobre o GMTP

Nesta seção, apresentam-se brevemente outras funcionalidades do GMTP, tais como o procedimento de desconexão, adaptação de fluxo, eleição de nós relatores.

### 1.7.1 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 1.5.2, 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. São candidatos a nó  $l_w$  os nós  $c_f$  já recebendo o fluxo de dados  $P$ , e 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  $P$ . 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 no GMTP 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  $r_d$  continuará repassando o fluxo de dados  $P$  para seus parceiros.

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  $P$ . 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  $r_d$ .

### 1.7.2 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 unicast 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  $r_d$ .

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 por ele. Note que este modo de transmissão deve ser implementado com garantia de entrega, ou seja, com a confirmação de recepção de pacotes e retransmissão caso este tipo de pacote seja perdido. Assim, um nó  $r_d$  poderá ter controle sobre a quantidade de nós  $l_w$  e receber relatórios apenas dos nós  $l_w \in L$ .

Uma outra situação que se faz necessária a eleição de nós  $l_w$  é no procedimento de desconexão, como explicado na Seção 1.7.1. 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.

## 1.8 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 caracterizada 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.

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.

# Bibliografia

- [1] TBE. Tbe, 3 2008.
- [2] Nandita Dukkupati, 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.
- [3] S. Bradner. Key words for use in rfcs to indicate requirement levels, 3 1997. <http://www.ietf.org/rfc/rfc2119.txt>. Último acesso: 11 de Dezembro de 2013.
- [4] D. Meyer. Administratively scoped ip multicast, 7 1998. <http://www.ietf.org/rfc/rfc2365.txt>. Último acesso: 11 de Dezembro de 2013.
- [5] Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory (Discrete Mathematics and Its Applications)*. CRC Press, 2 2003. ISBN: 978-1584880905.
- [6] R Séroul. *Programming for Mathematicians*. Springer-Verlag, 2 2000. ISBN: 978-3540664222.
- [7] 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.
- [8] K. J. Devlin. *Fundamentals of Contemporary Set Theory*. Springer, 9 1979. ISBN: 978-0387904412.
- [9] R. Braden. Requirements for Internet Hosts - Communication Layers, 10 1989. Último acesso: 11 de Dezembro de 2013.



- 
- [10] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Addison Wesley, trad. 3 ed. edition, 2006.
- [11] 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: 11 de Dezembro de 2013.
- [12] 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: 11 de Dezembro de 2013.
- [13] M. Handley and V. Jacobson. Sdp: Session description protocol, 4 1998. <http://www.ietf.org/rfc/rfc2327.txt>. Último acesso: 11 de Dezembro de 2013.
- [14] H. Alvestrand. Tags for the identification of languages, 2 1995. <http://www.ietf.org/rfc/rfc1766.txt>. Último acesso: 11 de Dezembro de 2013.
- [15] L. Eggert and F. Gont. TCP User Timeout Option, 3 2009. <http://www.ietf.org/rfc/rfc5482.txt>. Último acesso: 11 de Dezembro de 2013.
- [16] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, August 2010.
- [17] A. Vakali and G. Pallis. Content delivery networks: status and trends. *Internet Computing, IEEE*, 7(6):68–74, nov.-dec. 2003.
- [18] Mukaddim Pathan, Rajkumar Buyya, and Athena Vakali. Content delivery networks: State of the art, insights, and imperatives. In Rajkumar Buyya, Mukaddim Pathan, and Athena Vakali, editors, *Content Delivery Networks*, volume 9 of *Lecture Notes Electrical Engineering*, pages 3–32. Springer Berlin Heidelberg, 2008.
- [19] Nandita Dukkupati. *Rate control protocol (rcp): congestion control to make flows complete quickly*. PhD thesis, Stanford, CA, USA, 2008. AAI3292347.

- 
- [20] 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: 11 de Dezembro de 2013.
- [21] 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.
- [22] 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: 11 de Dezembro de 2013.
- [23] 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.
- [24] 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.