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

GMTP: Distribuição de Mídias Ao Vivo através de uma Rede de Favores Constituída entre Roteadores

Leandro Melo de Sales

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

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

Angelo Perkusich e Hyggo Almeida
(Orientadores)

Campina Grande, Paraíba, Brasil

Resumo

TBD

Abstract

TBD

Conteúdo

Capítulo 1

Global Media Transmission Protocol (GMTP)

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

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

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

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

Com base na ilustração da Figura 1.1, nas próximas seções deste capítulo, detalham-se os aspectos teóricos e computacionais empregados do GMTP, a fim de construir uma rede de sobreposição formada por roteadores, pela execução de quatro grandes passos:

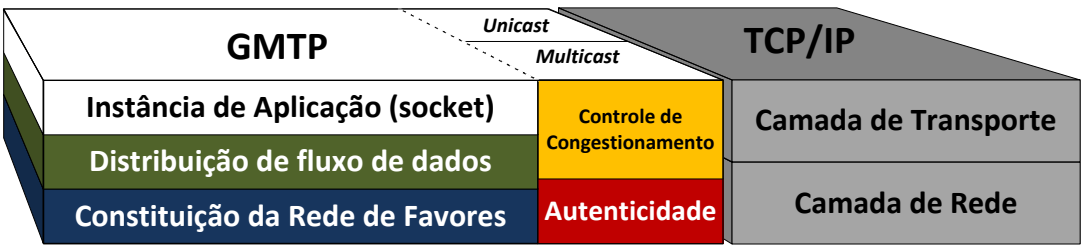

Figura 1.1: Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP.

1. *Constituição da rede de favores:* descobrir, definir, efetivar e desfazer parcerias entre os roteadores de acordo com o evento ao vivo a ser transmitido.

2. *Distribuição de fluxos de dados através de uma camada de socket:* conectar os nós

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

3. *Controle de congestionamento:* controlar a taxa de transmissão dos fluxos de dados distribuídos e utilizar as informações sobre a capacidade de transmissão de um canal para sugerir novas parcerias.

4. *Autenticidade do conteúdo:* verificar a autenticidade do fluxo de dados antes de repassá-los aos nós clientes, evitando-se ataques de poluição.

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

- 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, formalizam-se as definições e restrições do protocolo.

- Na Seção 1.3, descrevem-se o processo de constituição da rede de favores e os aspectos de conexão multi-ponto através da introdução de conceitos como sockets P2P, registro de participação de um nó e a seleção de nós parceiros.

- Na Seção 1.4, discutem-se os aspectos de transmissão e recepção de fluxos de dados, relacionando os algoritmos utilizados para compartilhar os fluxos de dados e as estratégias de disponibilização e obtenção das partes de uma mídia.

- Na Seção 1.5, apresentam-se detalhes de funcionamento dos algoritmos de controle de congestionamento utilizados no GMTP, bem como estes influenciam no processo de formação de parcerias.

- Na Seção 1.6, discutem-se os aspectos relacionados à autenticidade de um fluxo de dados.

- Na Seção 1.7, apresentam-se outros aspectos relacionados ao GMTP, tais como finalização de conexão, tolerância à desconexão e eleição de nós relatores para o funcionamento do algoritmo de controle de congestionamento em modo *multicast*.

- E, por fim, na Seção 1.8, apresenta-se o sumário deste capítulo, elencando brevemente os principais pontos discutidos.

1.1 Visão Geral

O protocolo GMTP é composto por dois módulos chamados de *GMTP-Intra* e *GMTP-Inter*, que operam na camada de transporte e de rede, respectivamente, definindo assim sua arquitetura, ilustrada na Figura 1.2.

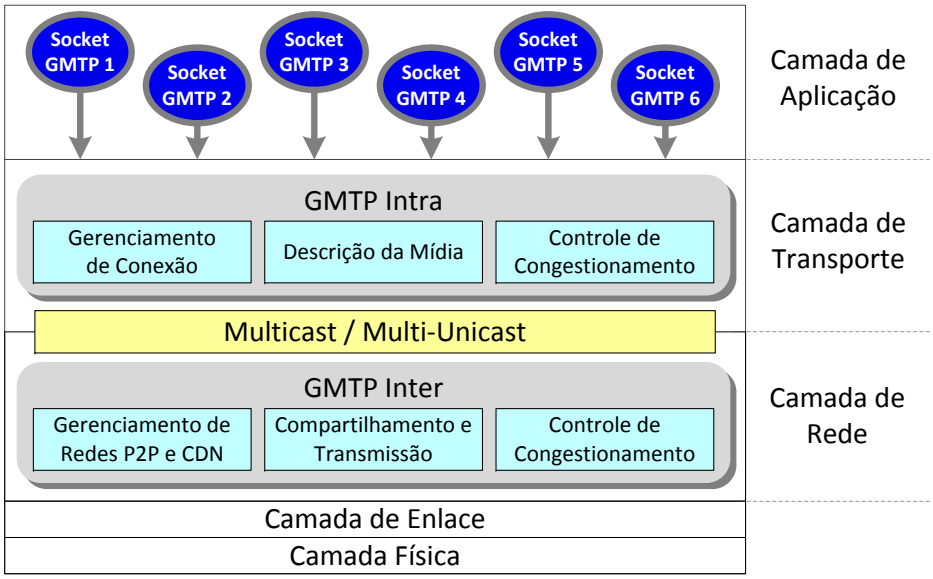

Figura 1.2: Arquitetura do Protocolo GMTP.

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

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

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

- **GMTP-Inter:** constituir uma rede de favores P2P composta por roteadores, os quais funcionam como pontes de acesso aos servidores de uma rede CDN. Trata-se do módulo em execução nos roteadores que cooperam entre si para constituir a rede de favores, aceitando conexões oriundas de um nó cliente, bem como instruções do nó servidor sobre formar parcerias com outros roteadores. No contexto de uma conexão, o GMTP-Inter mantém variáveis de estado relacionadas às funções de sua responsabilidade, tais como estabelecimento de conexão com nós servidores ou roteadores, seleção de nós roteadores parceiros, eleição de nós relatores, compartilhamento de fluxos multimídia e controle de congestionamento assistido pela rede. No GMTP-Inter, permite-se a configuração de parâmetros iniciais de configuração da rede de favores e da integração com servidores de uma ou mais CDN, como ilustra-se na Figura 1.3. Nesse caso, o usuário administrador de um nó repassador pode definir os seguintes parâmetros:

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

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

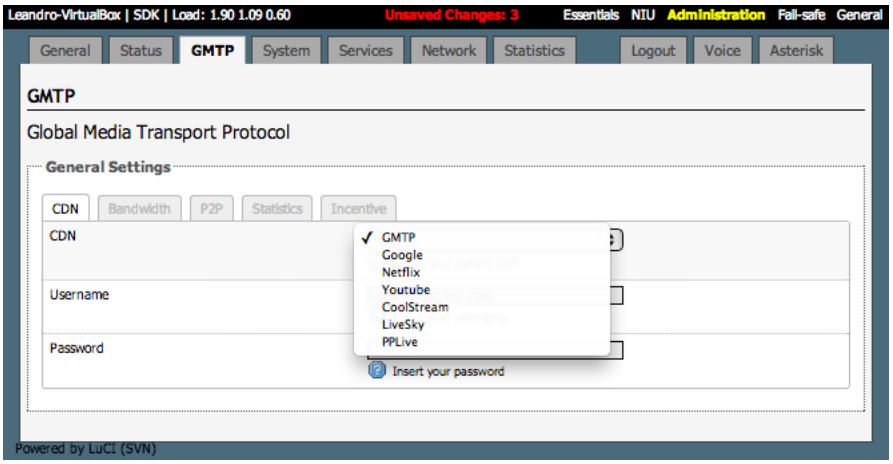

Figura 1.3: Tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP. Nessa tela, permite-se que o administrador do roteador configure parâmetros do módulo GMTP-Inter.

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

Na Figura 1.4, observa-se o cenário geral de atuação do protocolo GMTP, onde ilustram-se os nós *Clientes GMTP* interessados em obter o conteúdo de um determinado evento ao vivo. Neste caso, observa-se também um *Servidor GMTP*, que está conectado a uma rede CDN e atua como fonte geradora de dados; ao passo que os nós *Clientes GMTP* se conectam a um nó *Repassador GMTP* que é um roteador de rede. Com base na Figura 1.4, definiu-se a Figura 1.5, onde se observa os seguintes tipos de nós GMTP:

- **Cliente GMTP:** é capaz de reproduzir e gerar conteúdos multimídia ao vivo. Em geral, um *Cliente GMTP* é um sistema final que executa um processo em nível de sistema operacional, representando uma aplicação manipulada pelo usuário final. A maioria dos *Clientes GMTP* funciona apenas de forma passiva, recebendo o fluxo de dados de um conteúdo multimídia e entregando-o para um processo de aplicação em execução, sendo um sub-conjunto destes, contribuidores efetivos no processo de execução do algoritmo de controle de congestionamento em transmissões *multicast*.

- **Servidor GMTP:** é um sistema final que participa de uma rede CDN e obtém a mídia a ser transmitida através de três formas: i) diretamente a partir de uma unidade gera-

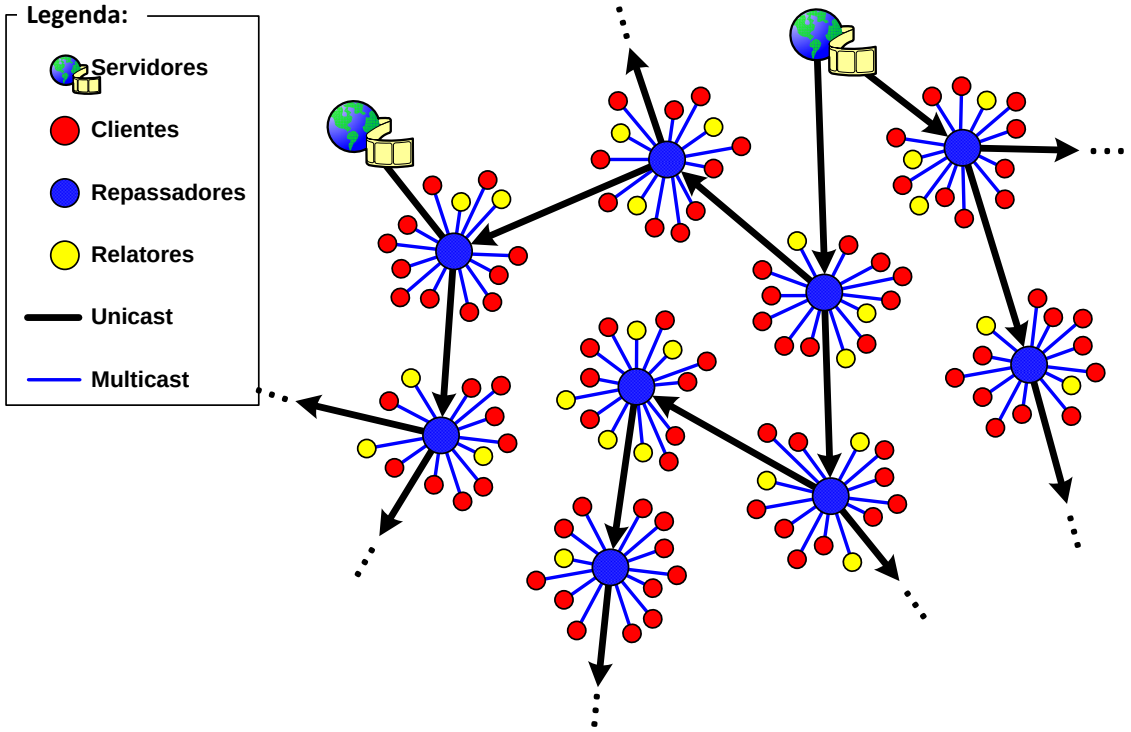

Figura 1.4: Rede de sobreposição construída dinamicamente pelo GMTP com a presença de nós repassadores e relatores.

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

- **Repassador GMTP:** roteadores que participam efetivamente da rede de favores, com a responsabilidade de repassar os fluxos de dados originados em um ou mais *Servidores GMTP* para outros nós *Repassadores GMTP* até que os pacotes de dados alcancem os nós *Clientes GMTP*.

- **Relator GMTP:** é um *Cliente GMTP* com habilidades de enviar relatórios periódicos ao nó *Repassador GMTP* sobre o estado da transmissão.

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

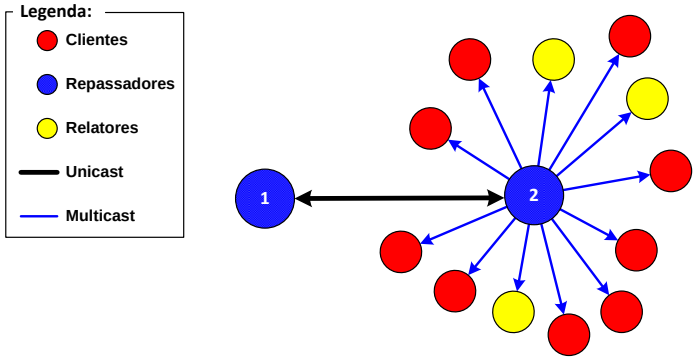

Figura 1.5: Tipos de Nós e modos de conexões do GMTP.

como os termos *socket*, *cache*, *unicast*, *multi-unicast* e *multicast*. Quando o termo *transmissão* ou *transmissão de um evento ao vivo* for mencionado, denotar-se-á a transmissão de um fluxo de datagramas IP correspondente a um evento ao vivo com o uso do protocolo GMTP. Embora alguns autores considerem os termos "repasse" e "roteamento" como conceitos distintos, neste trabalho ambos os termos são considerados sinônimos e devem ser interpretados como a capacidade que um nó GMTP tem de receber dados em uma interface de rede de entrada e encaminhar estes dados através de uma ou mais interface de rede de saída, ao mesmo tempo, permitindo-se que uma mesma interface de rede seja utilizada como entrada e saída ao mesmo tempo. Ademais, nas seções subsequentes, as palavras "deve", "não deve", "requerido", "pode", "não pode", "recomendado" e "opcional", incluindo suas variações morfológicas, devem ser interpretadas como descrito na RFC 2119 [1].

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

motivados por requisições originadas pelos seus respectivos nós clientes.

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

As requisições de conexão podem ser originados não apenas por nós clientes para seu respectivo nó repassador, mas também estas podem ocorrer entre nós repassadores que, motivados pelos interesses dos seus nós clientes, formam parcerias entre si. Isto significa que um nó repassador pode agir como se fosse um nó servidor, respondendo às requisições originadas por seus nós clientes GMTP ou por outros nós repassadores, como se a requisição estivesse alcançado o nó servidor que originalmente transmite o conteúdo. Essa estratégia gera uma diminuição significativa do número de requisições de conexão aos nós servidores, evitando-se portando a tragédia dos bens comuns, como discutiu-se na Seção **??**. Além disso, na Figura 1.4, observa-se um grupo especial de nós chamados de nós *Relatores GMTP*. Estes nós são responsáveis por enviar relatórios periódicos sobre o estado da transmissão ao seu nó *Repassador GMTP*, que os utiliza para regular a taxa de transmissão de um ou mais fluxos de dados, impedindo que a rede entre em congestionamento.

1.1.1 Resumo das principais funcionalidades

- Registro de participação de um nó repassador em um nó servidor. Isto permite que um nó repassador sinalize interesse em participar de uma rede CDN. Assim, pode-se pré-selecionar nós parceiros filtrados por métricas que influenciam na qualidade de serviço oferecido aos sistemas finais. Este assunto será retomado na Seção 1.3.1.

- Acesso a uma transmissão de um evento ao vivo através de um processo de conexão em três-vias (*3WHS*), com a requisição de conexão transmitida ao servidor e podendo ser interceptada por um nó repassador em seu trajeto ao servidor, com suporte automático de detecção e uso dos modos de transmissão suportados pelo nó repassador (unicast e/ou multicast). Este assunto será retomado na Seção 1.4.2.

- Descoberta de nós parceiros entre redes distintas e negociação de parcerias, com suporte a formação de parcerias baseadas em métricas de rede, tal como a largura de banda fim-a-fim. Além disso, o GMTP é capaz de distribuir um fluxo de dados em múltiplas taxas de transmissão de acordo com a largura de banda dos nós repassadores. Para isto, o GMTP segmenta os canais de transmissão (rota entre um nó servidor e os nós repassadores) quando existem múltiplos nós repassadores em uma determinada rota e estes estão interessados em receber o mesmo fluxo de dados. Este assunto será retomado nas Seções 1.3 e 1.5.

- Envio e recebimento de fluxos de dados compartilhados entre nós da mesma rede através do uso do modo de transmissão multicast, sendo o modo de transmissão unicast restrito apenas para transportar os fluxos de dados entre redes distintas, evitando-se a relação de uma conexão por cliente ao nó servidor. Este assunto será retomado na Seção 1.4.

- Uso de algoritmo de controle de congestionamento assistidos pela rede, em transmissões em modo unicast (GMTP-UCC); e uso do *TCP Friendly Rate Control* (TFRC) adaptado às transmissões de fluxos de dados em modo multicast (GMTP-MCC). Este assunto será retomado na Seção 1.5.

- Verificação de autenticidade dos dados multimídia, por meio do uso de assinaturas digitais disponibilizadas pelos nós servidores, impedindo assim ataques de poluição de conteúdo. Este assunto será retomado na Seção 1.6.

- Eleição de nós relatores com suporte a tolerância a desconexões de nós, com notificação e reeleição de novos nós. Este e outros assuntos relacionados serão retomados na Seção 1.7.

1.1.2 Tipos de pacotes

Toda comunicação entre dois ou mais nós GMTP ocorre através da troca de datagramas IP, os quais carregam sinalizações de controle e/ou dados da aplicação. No mundo real (Internet), será necessário registrar na *Internet Assigned Numbers Authority* – IANA[1] o uso de um código para o campo *Protocolo* do cabeçalho IP. Com a padronização do protocolo GMTP e a publicação da sua RFC, provavelmente será utilizado o código 100, como já está definido no documento *Protocol Numbers*[2] da IANA.

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

0. *GMTP-Request:* o nó cliente envia requisição para obter um fluxo de dados multimídia com base no nome do fluxo de interesse;

1. *GMTP-RequestNotify:* o nó repassador notifica um cliente que um fluxo de dados está prestes a ser transmitido ou já está sendo transmitido em um determinado canal de repasse multicast. O campo de dados desse tipo de pacote contém a descrição da mídia a ser reproduzida;

2. *GMTP-Response:* o nó repassador confirma o estabelecimento de uma parceria com outro nó repassador, dado um determinado fluxo de dados;

3. *GMTP-Register:* o nó repassador registra participação no servidor para funcionar como distribuidor de um fluxo de dados;

4. *GMTP-Register-Reply:* o nó servidor responde sobre o pedido de registro de participação enviado por um nó repassador;

[1] IANA: http://www.iana.org/

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

5. *GMTP-Route-Notify:* pacote que contém um caminho (rota) entre o nó repassador e um nó servidor. Em geral, o nó repassador envia esse tipo de pacote ao nó servidor;

6. *GMTP-RelayQuery:* o nó repassador pode solicitar ao servidor uma lista de possíveis nós repassadores parceiros;

7. *GMTP-RelayQuery-Reply:* o nó servidor envia uma resposta ao nó repassador com uma lista de candidatos a parceiros;

8. *GMTP-Data:* qualquer nó utiliza esse tipo de pacote para transmitir dados da aplicação;

9. *GMTP-Ack:* qualquer nó utiliza esse tipo de pacote para confirmar a recepção de um determinado pacote, seja pacotes previamente contendo dados ou não;

10. *GMTP-DataAck:* combinação dos pacotes GMTP-Data e GMTP-Ack (*PiggyBack*);

11. *GMTP-MediaDesc:* o nó servidor transmite esse pacote para descrever informações sobre a mídia sendo transmitida em uma determinado fluxo de dados (conexão);

12. *GMTP-DataPull-Request:* o nó repassador envia um pedido para obter o mapa de buffer atual de um outro repassador parceiros;

13. *GMTP-DataPull-Response:* resposta ao pedido para obtenção de um mapa de buffer;

14. *GMTP-Elect-Request:* o nó repassador envia para um cliente o pedido para tal cliente atuar como nó relator;

15. *GMTP-Elect-Response:* o nó cliente envia para o repassador uma confirmação de que pode atuar como relator;

16. *GMTP-Close:* os nós servidor, repassador ou cliente solicitam o término de uma conexão;

17. *GMTP-Reset:* determina, incondicionalmente, a finalização de uma conexão;

18. *Reservado:* a partir deste identificador ao 31, tratam-se de valores reservados para uso futuro e ignorado pelos nós que o processa.

1.2 Definições, Relações e Restrições

Para melhor organizar as discussões sobre o funcionamento do GMTP, nesta seção, descrevem-se suas definições, relações e restrições. Para isto, faz-se uso de fundamentos de álgebra booleana, lógica proposicional, teoria de conjuntos e teoria dos grafos [2–5].

1. Seja o conjunto finito dos nós repassadores, definido por $R = \{r_1, r_2, r_3, \ldots, r_d\}$, tal que $d \in \mathbb{N}$.

2. Seja o conjunto finito dos roteadores de uma rede de computadores, definido por $B = \{b_1, b_2, b_3, \ldots, b_e\}$, tal que $e \in \mathbb{N}$. Existe uma relação $R \to B$ que determina a sobreposição dos nós repassadores $r_d \in R$ sob os roteadores em B .

3. Seja o conjunto finito dos nós servidores, definido por $S = \{s_1, s_2, s_3, \ldots, s_a\}$, tal que $a \in \mathbb{N}$.

4. Seja o conjunto finito dos nós clientes, definido por $C = \{c_1, c_2, c_3, \ldots, c_f\}$, 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 $(\mathbb{P}, \prec) = \{p_1, p_2, p_3, \ldots, p_h\}$, tal que $h \in \mathbb{N}$. Note que se utiliza o símbolo $\prec$ para representar precedência entre dois elementos.

6. Seja um grafo determinado pelo conjunto de vértices Z, que podem estar interligados entre si por um conjunto de diferentes arestas, chamadas de caminhos W, por onde se transmite o fluxo de dados P, definido por $\eta = G(Z, W)$, tal que:

   (a) $Z = S \cup R$;

   (b) Sejam as relações e restrições estabelecidas entre os diferentes tipos de nós de uma transmissão de um evento $\mathcal{E}$, definida por $\mathcal{T} = \{Z, P, C_i\}$, tal que:

      i. Seja P, o conjunto *parcialmente ordenado (poset)* dos pacotes de dados p_x transmitidos por um nó r_d, também chamado de fluxo de pacotes de dados ou apenas fluxo de dados, definido por $(P, \prec) = \{p_1, p_2, p_3, \ldots, p_x\}$, tal que $x \in \mathbb{N}$. Trata-se de um *poset* porque o GMTP não garante entrega de p_x;

ii. Seja C_i, uma função que denota os nós c_f relacionados a um nó r_d, de modo que nenhum nó $c_f \in C$ pode estar relacionado com dois ou mais nós r_d, definida por $C_i : r_d \rightarrow 2^C$, $\forall r_d$,$r_q \in R$, $C_i(r_d) \cap C_i(r_q) = \{\emptyset\}$, tal que $q \neq d$ e $q \in \mathbb{N}$;

iii. Seja L, o conjunto finito dos nós relatores, definido por $L = \{l_1, l_2, \ldots, l_w\}$. Como todo nó c_f pode atuar como l_w, tem-se que $\exists L_\theta \in 2^{C_i(r_d)}$, tal que $l_w \in L_\theta$. Pelo item 6(b)ii, tem-se portanto que $L_\theta \subset L$ e $L_\theta \cup C_i(r_d) = C_i(r_d)$.

(c) $W = \bigcup_{v=1}^{j} W_v$, onde $j \in \mathbb{N}$ e corresponde à quantidade de todos os possíveis caminhos *toset* $(W_v, \prec)$, que denota um dos possíveis caminhos por onde um fluxo de dados P pode ser transmitido, obrigatoriamente a partir de um nó servidor s_a até um nó r_1, tal que:

i. $(W_v, \prec) = \{w_m \mid s_a, r_1, r_2, r_3, \ldots, r_d\}$, $\forall w_m, w_{m+1} \in W_v : w_m \prec w_{m+1}$ e $|W_v| \geq 2$;

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

iii. Um caminho W_v é dito *caminho completo*, representado por $W_v^\bullet$, se for W_v° e se $W_v \subset T$, tal que $T \subset Z$ é o conjunto dos nós r_d que transmitem os pacotes de dados $p_x \in P$ a seus nós $c_f \in C_i(r_d)$, definido por $T = \{t_u \mid \varphi(t_u, P) = 1\}$, $u \in \mathbb{N}$, tal que:

A. φ uma função booleana que determina se um nó $t_u \in T$ transmite os pacotes $p_x \in P$ para $c_f \in C_i(t_u)$, definida por $\varphi : (t_u, P) \rightarrow \{0, 1\}$, $\forall (t_u, P) \in \{T \times \{P\}\}$, onde 0 e 1 denotam, respectivamente, *falso* e *verdadeiro*.

(d) Seja $\sim$, reversa de um conjunto *toset*, tal que $\sim : (W_v, \prec) \rightarrow (W_v, \succ)$. Isto é, para um conjunto $(W_v, \prec) = \{w_m \mid s_a, r_1, r_2, \ldots, r_d\}$, então $\sim(W_v)$ produzirá $(W_v, \succ) = \{w_m \mid r_d, r_{d-1}, r_{d-2}, \ldots, r_1, s_a\}$;

(e) Seja δ, uma função que define um sub-caminho de W_v, representado por $W_v^\triangleleft$, a partir de um nó $t_u \in W_v$ até um nó $t_1 \in W_v$, tal que $\delta : (t_u, W_v) \rightarrow (W_v^\triangleleft, \prec)$. Ou seja, para um caminho qualquer $(W_v, \prec) = \{t_{u+2}, t_{u+1}, t_u, t_{u-1}, t_{u-2}, \ldots, t_2, t_1\}$,

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

(f) Seja ζ uma função que calcula o custo total para transmitir um pacote $p_x \in P$ através de um caminho W_v, definida por $\zeta : \sum_{v=1}^{|W_v|} \gamma(w_m, w_{m+1})$, tal que γ é uma função que determina o custo para transmitir o pacote p_x entre dois nós distintos $\forall w_m,w_{m+1} \in W_v$. No GMTP, a função γ calcula o custo apenas entre dois nós t_u,t_{u+1}, com base pela largura de banda disponível nos nós t_u. Porém, pode-se definir outras métricas, por exemplo, o número total de saltos no caminho W_v ou o RTT entre o nó s_a e um nó r_d;

(g) *Conjectura 1*: $\forall r_d \in R$ e $\forall c_f \in C$, r_d é mais estável que qualquer c_f com relação a sua disponibilidade e participação em uma rede de favores η. Em uma rede comutada por pacotes IP, um nó $b_e \in B$, ou seja, um nó r_d fica mais disponível se comparado aos seus nós $C_i(r_d)$. Por exemplo, nas transmissões de dados na Internet, a participação de um roteador no processo de transmissão de um fluxo de dados P é fundamental, mesmo que seja apenas para rotear os respectivos pacotes. Apesar de óbvia, tal observação é importante porque para qualquer nó c_f receber os pacotes de dados $p_x \in P$, primeiramente os pacotes de dados p_x passam, obrigatoriamente, pelo roteador de c_f, ou seja, o seu roteador padrão. Sendo assim, quando um nó r_d se desconecta, todos seus nós $C_i(r_d)$ tornam-se incapazes de receber P, mas a recíproca não é verdadeira – se um nó c_f se tornar indisponível, não necessariamente r_d também se torna indisponível. Com a aceitação dessa conjectura para a rede η, permite-se que outros nós c_f possam continuar recebendo P, mesmo ocorrendo a desconexão de um nó c_f que também esteja recebendo P. No GMTP, adota-se tal estratégia quando um nó r_d passa a manter estado sobre a transmissão de P e não mais os nós c_f, antes prática comumente adotada em soluções tradicionais de distribuição de conteúdos multimídia baseado em uma arquitetura P2P ou em qualquer protocolo disponível no estado da arte;

(h) *Conjectura 2*: as tabelas de roteamento dos nós $w_m \in W_v$ não mudam frequentemente e são independentes umas das outras. Em redes comutadas por pacotes IP, as rotas entre quaisquer nós c_{f_1} e $c_{f_2} \in C$ não se alteram com uma frequência que desestabilize a comunicação entre estes. Mesmo se estas mudanças ocorrerem

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

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

1.3 Constituição da Rede de Favores η

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

1.3.1 Registro de participação de r_d em η

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

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

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

s_a, ao passo que s_a poderá utilizar W_v para instruir os nós r_d a realizarem parcerias a fim de distribuir um fluxo de dados P, como se discute na Seção 1.3.3.

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

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

É importante salientar que toda transferência de pacotes de controle entre nós r_d ocorre com garantia de entrega, representando-se tais ações pelas funções com nomes contendo o sufixo *Rdt* (*Reliable data transfer*). Uma outra decisão importante tomada no GMTP é que um nó r_d deve periodicamente sinalizar sua participação na rede de favores η através de um método conhecido por *keep-alive*, comumente utilizado em outros protocolos de rede consolidados, como o TCP. Nesse aspecto, o GMTP segue a RFC 1122, *Requirements for Internet Hosts - Communication Layers* [6].

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

s_a, quando não desejar mais participar da rede de favores η ou continuar recebendo um fluxo de dados P. Para isto, deve-se enviar um pacote do tipo *GMTP-Close*. Em qualquer um dos casos de desconexão, por expiração do tempo (devido ao procedimento de *keep-alive*) ou explicitamente através do envio do pacote do tipo *GMTP-Close*, o nó s_a deve desconsiderar r_d em futuras formações de parcerias, finalizando o procedimento de desconexão com o envio do pacote do tipo *GMTP-Reset*. Na Seção 1.7, discute-se em mais detalhes sobre o processo de desconexão de um nó r_d, pois a suspensão de transmissão dos pacotes de dados P não ocorre instantaneamente, dependerá se o nó r_d em fase de desconexão está ou não repassando P para algum nó parceiro r_q.

Algoritmo 1: registerRelay(s_a: PeerServer, p_x = *GMTP-Request*)

```
/* The node r_d executes this algorithm to send a register
   of participation to a given node s_a. If p_x is given,
   node c_f wants to receive the flow P, so notify s_a.   */
if p_x ≠ NULL then
    P ← getPacketFieldValue(p_x, 'flow');  /* Extracts P in p_x */
    c_f ← getPacketFieldValue(p_x, 'client'); /* Extracts c_f in p_x */
    channel ← isFlowBeingReceived(P);   /* Ver Seção 1.3.2 */
    /* Add c_f in the list of receivers waiting P.        */
    addClientWaitingFlow(c_f, P);
    if channel ≠ NULL then
        /* Let c_f know that P is already registered in this
           r_d and is available from a multicast channel.  */
        respondToClients(GMTPRequestReply(channel));
        return 0;
    else                          /* Flow P not registered yet. */
        /* Send request to s_a and wait registration reply.
           When GMTP-Register-Reply is received, executes
           onReceiveGMTPRegisterReply (Algorithm 2).       */
        if not isWaitingRegisterReply(P) then
            isWaitingRegisterReply(P, true);
            sendPktRdt(GMTPRegister(s_a, P));
        end
        /* Ask C_i(r_d) to wait registration reply for P.  */
        respondToClients(GMTPRequestReply(P));
        return 0;
    end
end
if not isWaitingRegisterReply(s_a) then
    return sendPktRdt(GMTPRegister(s_a));
end
return 0;
```

Algoritmo 2: onReceiveGMTPRegisterReply(p_x = *GMTP-Register-Reply*)

```
   /* The node r_d executes this algorithm when receives a
      packet of type GMTP-Register-Reply, as response for a
      registration of participation sent to a s_a node.      */
1  setWaitingRegisterReply(P, false);
2  if p_x = OK then                          /* s_a confirmed registration */
3      W_v ← getPacketFieldValue(p_x, 'way');     /* Gets W_v in p_x */
4      s_a ← getPacketFieldValue(p_x, 'server');  /* Gets s_a in p_x */
5      P ← getPacketFieldValue(p_x, 'flow');      /* Gets P in p_x */
6      if P ≠ NULL then          /* Reply to C_i(r_d), waiting for P */
7          if s_a enabled security layer then          /* Section 1.6.4 */
8              getAndStoreServerPublicKey(s_a);
9          end
10         channel ← createMulticastChannel(s_a, P);
11         updateFlowReceptionTable(channel);  /* Section 1.3.2 */
           /* Let c_f ∈ C_i(r_d) know the multicast channel to
              receive P (Section 1.4.2 for more details).     */
12         respondToClients(GMTPRequestReply(channel));
           /* Start to relay P to clients (Section 1.4.6).    */
13         startRelay(channel);
14     end
       /* It was just a reply of a registration of
          participation.  Update flow reception table.        */
15     updateFlowReceptionTable(s_a);          /* Section 1.3.2 */
16     sendWayBackToServer(W_v);
17 else
       /* s_a refused to accept the registration of
          participation.  This r_d must notify the clients
          waiting for receiving P.                            */
18     errorCode ← getPacketFieldValue(p_x, 'error');
19     respondToClients(GMTPRequestReply(errorCode, P));
20 end
```

Por fim, salienta-se que o registro de participação do GMTP permite que quanto mais nós r_d se registrarem em nós s_a, mais caminhos W_v sejam conhecidos. Quanto mais caminhos forem conhecidos, mais parcerias poderão ser formadas entre os nós r_d. Quanto mais parcerias forem formadas, maior será o número de nós c_f capazes de receber um fluxo de dados P originado em s_a, disponibilizado indiretamente através dos seus respectivos nós r_d, sem nenhuma influência da camada de aplicação. No mundo real (Internet), os nós r_d podem passar a constituir dinamicamente a rede de distribuição de conteúdos de uma empresa. Por exemplo, um usuário de uma conexão residencial xDSL pode configurar seu roteador para registra-lo em múltiplas redes de distribuição, como ilustrou-se na Figura 1.3. Nesses casos, as redes de distribuição podem fazer uso do roteador desse usuário em momentos ociosos de recepção e transmissão de dados através da Internet. Como consequência, relações comerciais podem ser construídas entre o usuário e os provedores de rede, mas essa discussão está fora do escopo deste trabalho.

Manutenção do registro de participação:

O procedimento de manutenção de um registro de participação deve ser feito usando o pacote do tipo *GMTP-Ack*, em um tempo $t = max(300, t_{user})$, onde t_{user} é definido em segundos e corresponde a um tempo definido pelo administrador do nó r_d, caso deseje um tempo menor que $300\,s$ para mantém o registro de participação ativo. Quando s_a receber um pacote do tipo *GMTP-Ack* do nó r_d, este deve enviar um pacote do mesmo tipo. Caso r_d não receba *GMTP-Ack* no período de $4 \times RTT$, deve-se repetir tal procedimento por 3 vezes e somente após essas tentativas, o nó r_d deve considerar a conexão finalizada por tempo de expiração (*timeout*) e enviar um pacote do tipo *GMTP-Reset*. Na RFC 5482 [7], discute-se sobre outros aspectos de expiração no tempo que podem ser adaptados para o GMTP.

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 1.6. O nó r_d utiliza tal tabela para registrar todos os fluxos de dados que estão sendo repassados para seus nós $c_f \in C_i(r_d)$ e os respectivos

canais multicast utilizados, mantendo-se as seguintes informações:

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

Figura 1.6: Exemplo de uma tabela de recepção de fluxo mantida por um nó r_d.

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

Conceitualmente, quando um nó r_d adiciona um registro na tabela de recepção de fluxos de dados, define-se $\varphi(r_d, P) = 1$, ou seja, $r_d \in T$. Um nó r_d consulta a tabela de recepção de fluxos de dados quando recebe um pedido de conexão (*GMTP-Register*) para obter um fluxo de dados P, tal como apresentou-se na Linha 6 do Algoritmo 1, Seção 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 11 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

Dado que as parcerias ocorrem entre os nós $r_d \in R$ e não entre os nós $c_f \in C$, a formação de parcerias consiste em determinar intersecções de caminhos W_v, considerando o nó *pivot* s_a e diversos nós r_d interessados em obter P, a pedido de seus nós $c_f \in C_i(r_d)$. Este processo pode ocorrer antes e durante a transmissão de um fluxo de dados P gerado por um nó s_a, de forma transparente para a aplicação em execução em c_f, durante seu pedido de conexão transmitido em direção ao nó s_a. Como consequência, constitui-se um ou mais caminhos $W_v \in W$, os quais interconectam um nó s_a e os nós $c_f \in C_i(w_m)$, tal que $\exists W_v \mid w_m \in W_v$. Como regra geral para formação de parcerias, definem-se três critérios:

1. o melhor nó s_a para servir um nó r_d é aquele que está especificado em seu pedido de registro de participação, pois deve-se respeitar as regras de balanceamento de carga definida pela CDN. Em geral, o servidor DNS define tais regras com base no endereço IP do nó solicitante;

2. se $\varphi(w_m, P) = 1$, então w_m pode agir como se fosse um nó s_a;

3. se o nó $w_m \in W_v$, tal que W_v é parte ou todo do caminho entre r_d e s_a; e se w_m se enquadra no Item 2, então o melhor nó s_a para servir r_d será o mesmo que serve o nó w_m.

Para entender detalhes desse processo, considere a Figura 1.7. No Passo 1, ilustra-se um cenário de rede $\eta = G(Z, W)$, onde $Z = \{s_1, r_{1..19}\}$, $W = \{\emptyset\}$ e $\tau = \{\{\emptyset\}, \{\emptyset\}, \{\emptyset\}\}$, ou seja, sem qualquer fluxo de dados P sendo transmitido, tampouco nenhuma parceria efetivada e suprimindo-se os nós $c_f \in C_i(r_{1..19})$. Já no Passo 2, ilustra-se a mesma rede η, porém com $\tau = \{\{s_1, r_{5..9}\}, P, C_i(r_9)\}$, constituindo-se o caminho $W_1 = \{s_1,...,r_9\}$ (linha tracejada e vermelha) e, portanto, $W = \{W_1\}$ com $\varphi(r_9, P) = 1$. Nesse exemplo do Passo 2, o nó r_9 recebe o fluxo de dados P em modo unicast e repassa P para todos os nós $c_f \in C_i(r_9)$ em modo multicast. Para constituir o caminho W_1, o nó r_9 deve transmitir o pedido de registro de participação ao nó s_1 (como discutiu-se na Seção 1.3.1) e, a partir de sua confirmação, processada pelo nó s_1 e enviada ao nó r_9, este começa a receber os pacotes $p_x \in P$. Com este procedimento, o nó s_1 passa a conhecer o caminho W_1, que pode ser

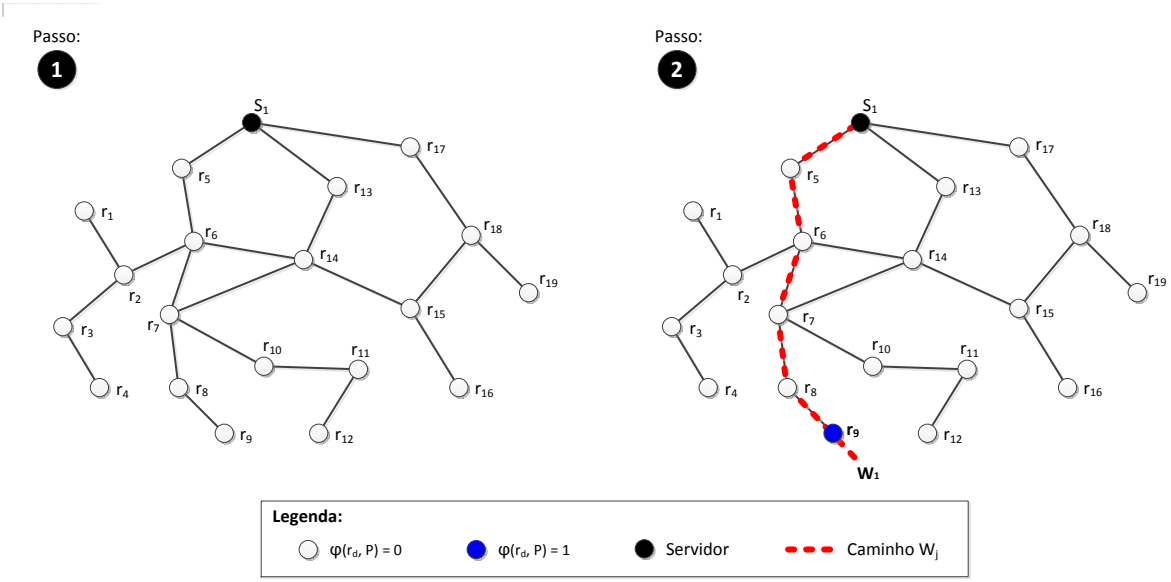

Figura 1.7: Cenário e passos para seleção de nós (exemplo 1).

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

Na Figura 1.8, considera-se a formação de parceria por intersecção do fluxo de dados P, a partir do Passo 2 da Figura 1.7. Este procedimento ocorre quando um outro nó r_d envia um pedido de registro de participação em direção ao nó s_1, a fim de obter o fluxo de dados P, motivado por algum nó $c_f \in C_i(r_d)$. Nesse caso, se um nó r_d transmitir um pedido de registro de participação através de um sub-caminho $W_v^{\triangleleft}$ tal que $\exists W_v \in W$, o nó s_a determina a intersecção de ambos e instrui o nó comum w_m a repassar o fluxo de dados P também para r_d, sem a necessidade de enviar um segundo fluxo de dados na mesma direção de $W_v^{\triangleleft}$. Sendo assim, a resposta de s_1 não resulta em uma nova transmissão do fluxo de dados P, mas sim em uma mensagem de controle para o nó w_m, após identificá-lo como o nó comum a dois ou mais caminhos W_v. Isto implicará que o referido nó w_m replique o fluxo de dados P, mesmo quando $|C_i(w_m)| = 0$, mas de modo conveniente para evitar múltiplas transmissões do fluxo de dados P, originadas no nó s_a. A fim de compreender o funcionamento desse procedimento, acompanhe a explicação a seguir, com base na ilustração da Figura 1.8 e no caminho W_1.

Se qualquer um dos nós $r_{7,8,10,11,12}$, suponha r_{11}, enviar um registro de participação em direção à s_1 para obter um fluxo de dados P (Passo 3 da Figura 1.8), o nó s_1 descobrirá o

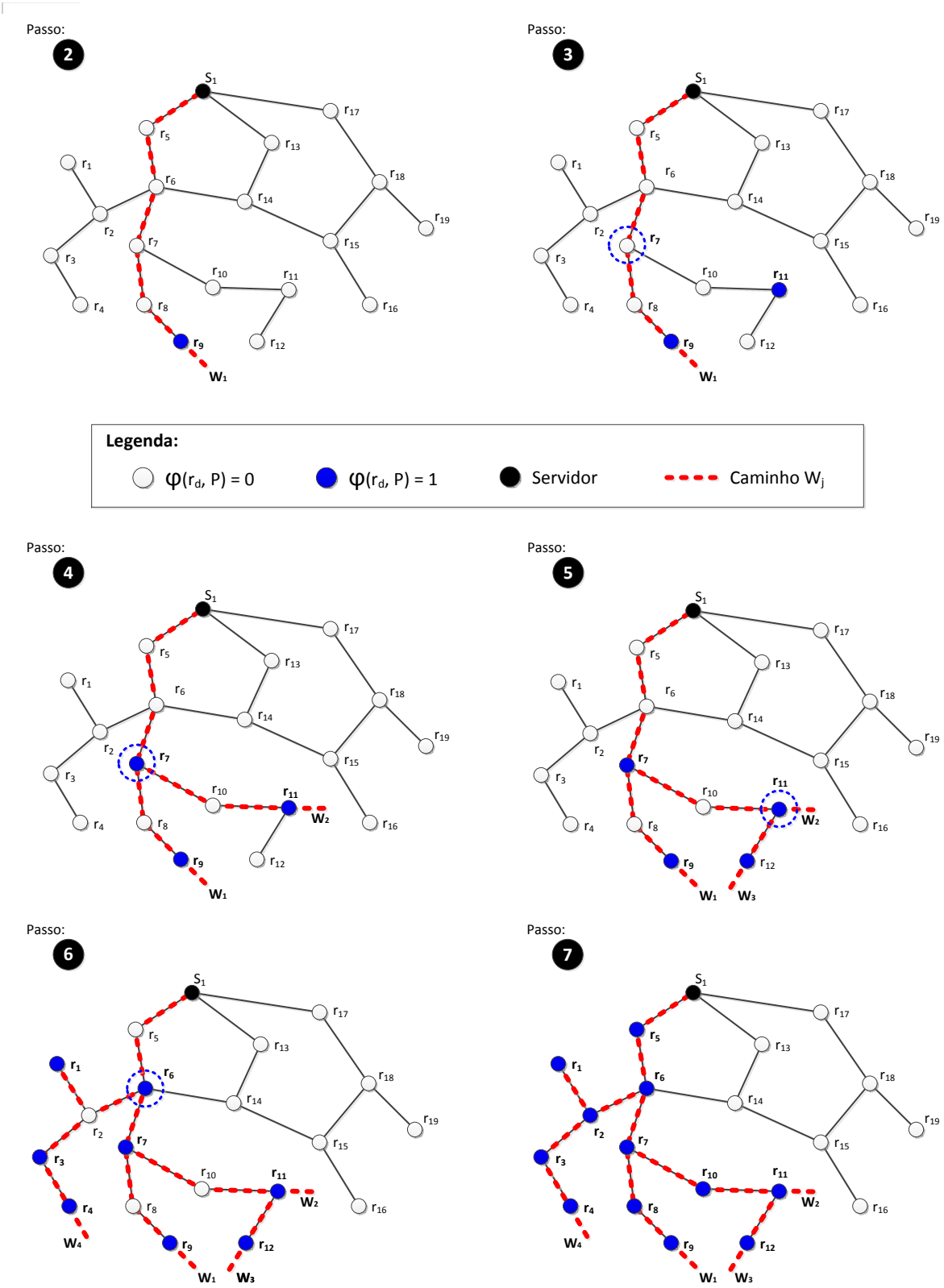

Figura 1.8: Cenário para seleção de nós por interseção de caminhos W_v.

caminho $W_2 = \{r_5, r_6, r_7, r_{10}, r_{11}\}$ (Passo 4). Em seguida, pela intersecção $(W_1 \cap W_2)$, o nó s_1 determinará que o nó r_7 é o nó comum e portanto instruirá que r_7 repasse o fluxo de dados P também para o nó solicitante r_{11}. A instrução de s_1 para r_7 deve determinar $\varphi(r_7, P) = 1$. Em termos práticos, isto obriga o nó r_7 a adicionar uma nova entrada na tabela de recepção de fluxos de dados referente a P, mesmo se $|C_i(r_7)| = 0$ para P. É óbvio que, se posteriormente $|C_i(r_7)| > 0$ para P, será necessário apenas r_7 criar um canal multicast para a transmissão local de P, evitando-se um novo registro de participação em s_1. Na Seção 1.4.2, discute-se em mais detalhes este aspecto do GMTP, explicando-se os procedimentos de pedido de conexão de um nó c_f.

Ao estender a discussão sobre o cenário ilustrado na Figura 1.8, percebe-se que se o nó r_{10} necessitar obter o mesmo fluxo de dados P, seu pedido de registro de participação será interceptado pelo nó r_7 e parte do procedimento supracitado se repete. Uma situação similar ocorre se o nó r_{12} ou qualquer nó $r_d \in W_4$ também desejar obter o fluxo de dados P, tal que $W_4 = \{r_1, r_2, r_3, r_4\}$ (Passo 5 e 6). Para o caso do nó r_{12}, o nó r_{11} interceptará o pedido de registro de participação de r_{12}, ao passo que se for qualquer nó $r_d \in W_4$, o nó r_6 realizará tal interceptação, pois o nó s_1 determinará $\varphi(r_6, P) = 1$, depois do primeiro pedido de registro de participação originado por qualquer nó $r_d \in W_4$. A única diferença nesses últimos casos é que, como $\varphi(r_7, P) = 1$ e $\varphi(r_{11}, P) = 1$, o nó r_7 tem autonomia para responder ao nó r_{10} e ao nó r_{11} como se fosse o nó s_1, sem repassar tal pedido em direção ao nó s_1.

Para generalizar essa discussão sobre o processo de formação de parcerias do GMTP, caso existam outros nós r_q interessados em obter um fluxo de dados P e estão interligados direto ou indiretamente a r_d, tal que $\varphi(r_d, P) = 1$, o nó r_d sempre interceptará o pedido de registro de participação dos nós r_q e atuará como se fosse o nó s_1. No caso do exemplo que se discute, independente da ordem em que as requisições de registro de participação sejam enviadas por $w_m \in (W_1 \cup W_2 \cup W_3 \cup W_4)$, será necessário transmitir apenas um fluxo de dados P para "alimentar" os quatro caminhos referidos. Isto significa que todos os nós $c_f \in C_i(W_1 \cup W_2 \cup W_3 \cup W_4)$ receberão um único fluxo de dados, com repasse dos pacotes $p_x \in P$ realizado em modo multicast em cada sub-rede de cada nó w_m (Passo 7). Como a transmissão será em modo multicast, torna-se indiferente a quantidade de nós c_f desses caminhos, mas faz-se necessário um mecanismo para controle de congestionamento em modo multicast, a ser discutido na Seção 1.5.

Note que, o nó r_d que interceptar um pedido de conexão para um fluxo de dados P, deve transmitir para o nó s_a uma notificação sobre a(s) parceria(s) formada(s) por intersecção. No caso do exemplo anterior, os nós r_6, r_7 e r_{11} devem realizar tal notificação enviando um pacote do tipo *GMTP-Register*, como explicado na Seção 1.3.1. Para isso, deve-se ativar o bit *intercepted* do pacote *GMTP-Register*. Esta ação é importante devido aos aspectos gerenciais de uma transmissão, onde uma aplicação poderá contabilizar os nós r_d que estão recebendo P, mesmo que indiretamente, por meio da interceptação de registros de participação. A propósito, em ICN isso não é possível.

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

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

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

```
/* s_a executes this algorithm to finds the first node w_m
   common to a known path W_v and the path W_{r_d}. W_v is
   already used for transporting P to node in δ(w_m, W_v),
   and W_{r_d} contains all nodes between r_d (requester) and
   s_a. The packet p_x carries W_{r_d} and the P flow name. */
1  done ← false;            /* It becomes true when w_m is found */
2  P ← getPacketFieldValue(p_x, 'flow');     /* Extracts P in p_x */
3  W_{r_d} ← ~(getPacketFieldValue(p_x, 'path'));
4  W_P ← getKnownPathsOfFlow(P);                  /* W_P ⊂ W */
5  foreach W_v ∈ W_P do
6      foreach w_m ∈ W_v do
7          if w_m ∈ W_{r_d} then
               /* The node w_m is common in W_v and in W_{r_d}. */
8              done ← true;
9              break;
10         end
11     end
12     if done then
           /* s_a stores W_{r_d} as a known path and replies to r_d,
              asking w_m to act as a relay for P. s_a actives
              flag 'relay' of the GMTP-RegisterReply. */
13         W_P[length(W_P)] ← W_{r_d};
14         return GMTPRegisterReply(w_m, relay=1);
15     end
16 end
   /* s_a must register W_{r_d} as a known path and reply to r_d by
      accepting its connection request, since no node w_m is
      intersecting W_{r_d}. In this case, s_a starts the
      transmission of p_x ∈ P to r_d. */
17 W[length(W)] ← W_{r_d};
18 return GMTPRegisterReply(r_d, relay=0);
```

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

Por fim, um outro aspecto no processo de formação de parcerias do GMTP é o uso de informações sobre a capacidade de transmissão de um caminho W_v para determinar as parcerias entre os nós r_d, bem como a qualidade do fluxo de dados a ser transmitido pelo nó servidor. Diferentemente se comparado a soluções como as empregadas com o uso de HTTP (por exemplo, DASH), em vez de um servidor GMTP gerar múltiplos fluxos de dados, codificados em diferentes taxas de bits, gera-se o fluxo de dados com a taxa de bits correspondente a capacidade máxima de transmissão do caminho $W_v$em um determinado instante. Isto ocorre porque o GMTP expõe, ao nó s_a, a informação de controle de congestionamento percebida pelos nós $w_m \in W_v$. Sendo assim, o nó s_a codifica a mídia em uma qualidade possível de ser transportada através de $W_v$utilizando um codificador do tipo VBR (*Variable Bit Rate*), como o H.264 [8]. Na Seção 1.5.1, discute-se essa função em mais detalhes.

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

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

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

1.4.1 Indexação de conteúdo

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

Define-se um nome de um fluxo de dados P por um código de *hash* no formato UUID (*Universally Unique IDentifier*) de 128 bits [9]. Na sua forma canônica, representa-se P por uma sequência de 32 dígitos hexadecimal, exibidos em cinco grupos separados por hífen, na forma de {8}-{4}-{4}-{4}-{12}. Por exemplo, P = 641f931f-d3ac-50e3-b625-537574541f1f. O nome de um fluxo de dados P sempre será informado no campo *nome do fluxo de dados* (*data flow name*), disponível no cabeçalho de transporte dos pacotes *GMTP-Register*, *GMTP-Request*, *GMTP-Data* e *GMTP-Ack*.

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

nomes dos fluxos de dados P, de posse de um identificador de um fluxo de dados P, um nó GMTP poderá solicitar os pacotes de dados $p_x \in P$. Além disso, os nós r_d mantêm a tabela de recepção de fluxo de dados que estão repassando para seus clientes $c_f \in C_i(r_d)$ e, sendo assim, podem compartilhá-la para outros nós repassadores. Atualmente, não se explora o compartilhamento da tabela de repasse, mas pode ser feito para formar parcerias entre dois nós r_d sem precisar consultar o nó s_a, por exemplo. Com esse esquema de nomes baseado no endereço IP e porta, o GMTP não requer alterações na camada de aplicação para informar o fluxo de dados de interesse – a aplicação continua informando endereço IP e número da porta no momento do estabelecimento de uma conexão, mantendo-se a compatibilidade com as aplicações existentes e, portanto, futuras adoção do GMTP na Internet.

No caso do uso do DNS, o nó s_a divulga os identificadores de todos os eventos sendo transmitido por meio de um mecanismo de atualização dinâmica de registro de DNS, como especificado na RFC 2136 [10]. Para o GMTP, criou-se um novo tipo de registro de DNS chamado de SID (*Streaming IDentifier*).

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

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

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

Descrição de um fluxo de dados P:

O GMTP permite nativamente a descrição da mídia a ser transmitida e com isso promover a compatibilidade entre diferentes aplicações. Para isto, incorporou-se ao GMTP o protocolo o SDP (*Session Description Protocol*), definido na RFC 2327 [11]. Com o SDP, permite-se que as aplicações obtenham mais detalhes sobre a mídia, flexibilizando-se o acesso a um determinado conteúdo. Como consequência, as aplicações se preocupam apenas com a decodificação e a reprodução do conteúdo ao usuário, independente de qual sistema remoto que o gerou.

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

A título comparativo, considerando os protocolos baseado em HTTP e os sistemas de distribuição multimídia, descritos no Capítulo **??**, cada um possui uma proposta diferente de descrição da mídia, predominando o uso de XML, porém não se mantendo o mesmo formato. A consequência disso é que a aplicação cliente, para suportar diversos sistemas, terá que implementar cada uma dessas propostas e usar XML gasta-se mais espaço se comparado ao SDP.

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

- *v*, a versão do SDP;
- *o*, a lista de nós s_a que a distribui;
- *s*, o nome da mídia, como discutido na Seção 1.4.1;
- *i*, o título da mídia;
- *u*, a URI que descreve detalhes sobre a mídia;
- *c*, as informações de conexão, como a versão do protocolo de rede e o endereço do nó r_d;
- *f*, o certificado digital emitido pelo nó s_a para verificação de autenticidade dos pacotes $p_x \in P$ (opcional). Este assunto será retomado na Seção 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.

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 [12] */
14 a=framerate:23.0
```

No exemplo apresentado no Quadro 5, utiliza-se a primeira versão do protocolo SDP e descreve-se a transmissão de dois fluxos de dados P (Linhas 10 e 11), sendo um de áudio e outro de vídeo. A distribuição dos fluxos de dados P ocorre com a geração dos pacotes de dados $p_x \in P$ em três nós s_a (Linha 2), dos quais dois são acessíveis através de endereços IPv4 e um através de um endereço IPv6. Os fluxos de áudio e vídeo são repassados por um nó r_d, acessível por um endereço IPv4 (Linha 6), através das portas 49170 e 51372, respectivamente (Linhas 9 e 10). As informações de endereço IP e porta do nó r_d são utilizadas para que os nós $c_f \in C_i(r_d)$ possam sintonizar seus sockets de conexão e iniciar a reprodução da mídia, através do modo de transmissão multicast (Linha 10). Em seguida, na Linha 7, observa-se uma URL do certificado digital a ser utilizado pelo nó r_d para verificar a autenticidade do conteúdo de pacote de dados $p_x \in P$ – na Seção 1.6, discute-se este assunto em mais detalhes. Por fim, entre as Linhas 11 e 14 especificam-se outros parâmetros para descrever a mídia, tais como o nível de qualidade da mídia, que varia entre 1 e 10, as taxas de bits para cada fluxo de dados, sendo para o áudio variando entre 16000 *Bytes* e 20000 *Bytes* e, para o vídeo, variando entre 156250 *Bytes* e 625000 *Bytes*. É importante salientar que os nós

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

1.4.2 Estabelecimento de conexão entre c_f e s_a para obter P

Divide-se o processo de estabelecimento de conexão em três fases. A Fase 1 acontece quando, por exemplo, um nó qualquer $c_1 \in C_i(r_d)$ deseja obter P transmitido por um nó s_1 e não existe nenhum outro nó $c_f \in C_i(r_d)$ em sua rede local recebendo P. Já a Fase 2 acontece quando um outro nó $c_2 \in C_i(r_d)$ precisa obter o mesmo fluxo de dados P, solicitado previamente pelo nó c_1. E, por fim, a Fase 3 acontece quando o nó r_d começa a buscar novos nós parceiros r_q a fim de obter P. Na Figura 1.9, ilustram-se um nó s_a, que gera um fluxo de dados P e 12 nós r_d, que constituem uma rede de diferentes domínios administrativos, sendo o nó r_1 o repassador de um desses domínios, composto por 6 nós $c_f \in C_i(r_1)$ (Rede Local).

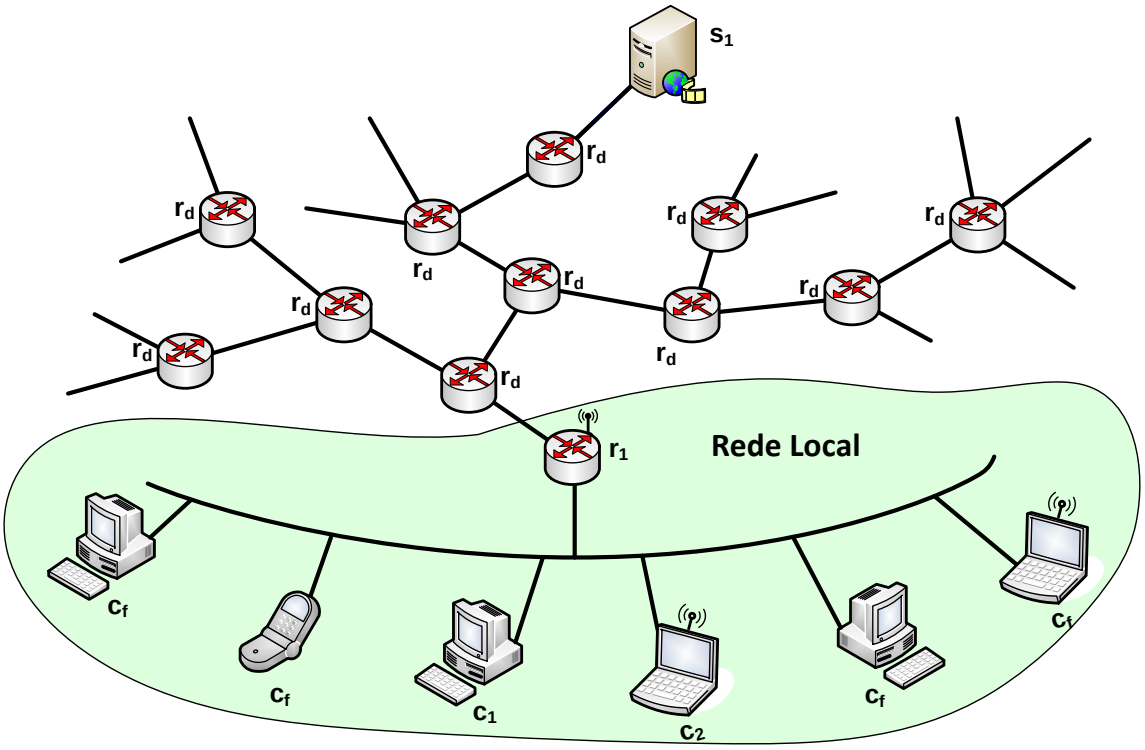

Figura 1.9: Exemplo de rede para o estabelecimento de conexão do GMTP.

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

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

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

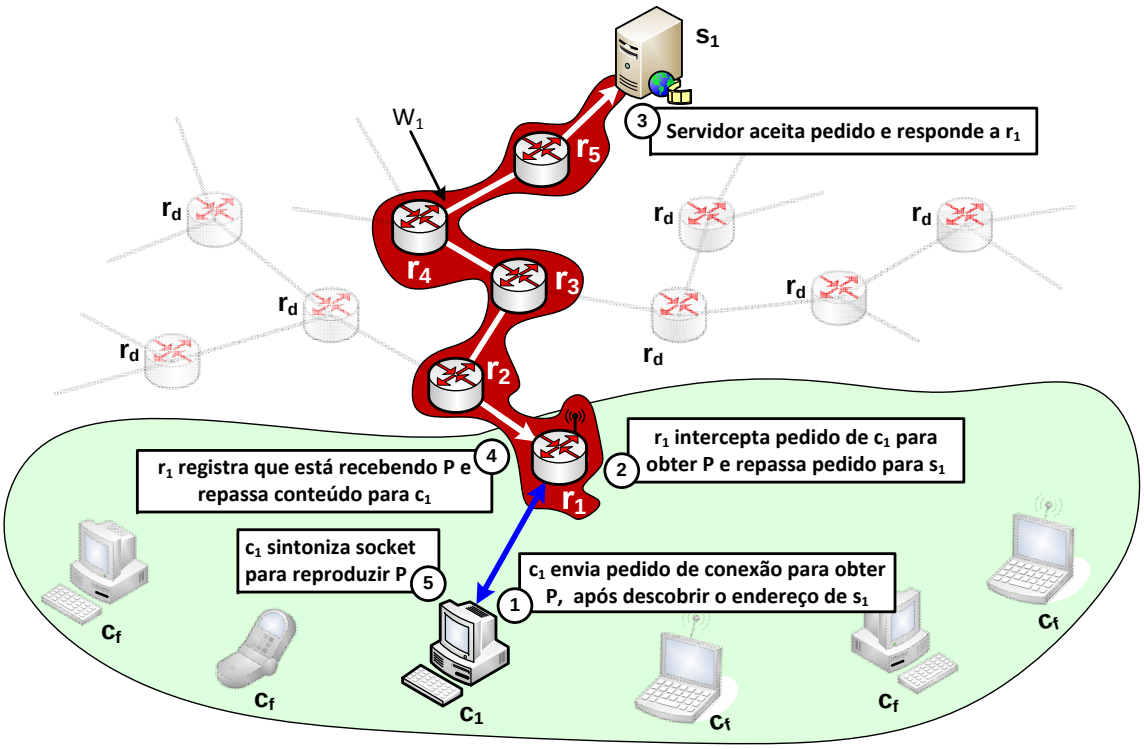

Figura 1.10: Passos do processo de estabelecimento de conexão do GMTP (Fase 1).

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

Para obter o fluxo de dados P, o nó c_1 inicia o canal de controle GMTP (detalhado na Seção 1.7.1) e transmite um pacote do tipo *GMTP-Request* (Figura 1.10, Passo 1). Para

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

Quando o pacote *GMTP-Request* alcançar o nó r_1 (Passo 2 da Figura 1.10), este consulta a tabela de recepção de fluxos de dados e constata que não há qualquer registro para o fluxo de dados P. Nesse instante, o nó r_d inicia um processo de registro de participação para obter o fluxo de dados P. Isto significa que a execução do procedimento *registerRelay(s_a, p_x)* (Seção 1.3.1), onde p_x é o pacote do tipo *GMTP-Request*, fará o nó r_1 transmitir um pacote do tipo *GMTP-Register* em direção ao nó s_1. À medida que os nós r_d repassam o pacote *GMTP-Register* até alcançar o nó s_1, constitui-se o caminho $W_1 = \{r_1,r_2,r_3,r_4,r_5,s_1\}$ (Passo 3 da Figura 1.10 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 1.10). Esta notificação permitirá os nós c_f, aguardando para obter P, "sintonizarem" seus respectivos sockets no canal multicast correspondente. No caso do exemplo supracitado, o nó c_1, após sintonizar o socket no canal multicast informado pelo nó r_1, começa a receber os pacotes de dados p_x do tipo *GMTP-Data* ou *GTMP-DataAck* (Passo 5 da Figura 1.10).

No Algoritmo 6, resume-se os passos descritos anteriormente para iniciar a transmissão dos pacotes de dados $p_x \in P$ aos nós $c_f \in C_i(r_d)$, após r_d receber o pacote do tipo *GMTP-RequestReply*. Note que, o nó r_d invoca tal procedimento nas Linhas 7 e 14 do Algoritmo 1 e nas Linhas 12 e 19 do Algoritmo 2 (Seção 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 1.11. Com base no exemplo citado, a tabela de recepção antes vazia, agora contém uma entrada que informa a ocorrência de recepção do fluxo de dados $P =$ *72c44591-7d82-427c-825f-722f015787c1*, originado no nó s_a, cujo endereço é *177.135.177.241*, com porta de recepção *49170*. Além disso, define-se o canal multicast no endereço *239.192.68.79* e porta *1900*, através do qual os nós $c_f \in C_i(r_d)$ podem receber os pacotes de dados $p_x \in P$.

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

Figura 1.11: Tabela de recepção de fluxos de dados após a Fase 1.

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

```
/* A r_d node executes this Algorithm to respond to clients
   waiting for receiving a flow P. This algorithm is
   invoked in Lines 7 and 14 of Algorithm 1 and in
   Lines 12 and 19 of the Algorithm 2.                    */
1  destAddress ← getCtrlChannel();     /* 238.255.255.250:1900 */
2  setPacketFieldValue(p_x, 'destinationAddress', destAddress);
3  P ← getPacketFieldValue(p_x, 'flow');     /* Extracts P in p_x */
4  errorCode ← getPacketFieldValue(p_x, 'errorCode');
5  if errorCode ≠ NULL then
6      removeClientsWaitingForFlow(P);   /* See Algorithm 1 */
7      sendPkt(p_x);
8      return 0;
9  end
10 channel ← getPacketFieldValue(p_x, 'channel');
11 if channel ≠ NULL then
       /* Node r_d is already receiving P and clients C_i(r_d)
          must know the media description.                */
12     mediaDescription ← getMediaDescription(P);
13     setPacketFieldValue(p_x, 'data', mediaDescription);
       /* In Algorithm 1, Line 5, c_f nodes are added in a list
          of clients waiting for flow P. Now, r_d notifies
          them, wait confirmation (ACKs) from them and start
          relaying p_x ∈ P to them through given channel.  */
14     sendPkt(p_x);
15     C_i(r_d) ← getClientsWaitingForFlow(P);
16     waitAck(C_i(r_d), P);
17 else    /* Let C_i(r_d) know r_d is waiting for registration. */
18     setPacketFieldValue(p_x, 'waitingRegistration', true);
19     sendPkt(p_x);
20 end
21 return 0;
```

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 1.12 (Passo 1 da Figura 1.12). Para isto, o nó r_1 intercepta a requisição do nó c_2, consulta a tabela de recepção de fluxos de dados e dessa vez constata a recepção do fluxo de dados P, criando o pacote do tipo *GMTP-Request-Reply* (Passo 2). Este procedimento ocorre no registro de participação, especificamente no trecho de código definidos entre as Linhas 2-8 do Algoritmo 1. Em seguida, transmite-se o pacote do tipo *GMTP-Request-Reply* ao nó c_2, como descreve-se no trecho de código entre as Linhas 10-16 do Algoritmo 6, que então "sintoniza" seu socket para o canal multicast informado por r_1 (Passo 3). Tal procedimento se repete para cada novo nó $c_f \in C_i(r_1)$ interessado em obter P.

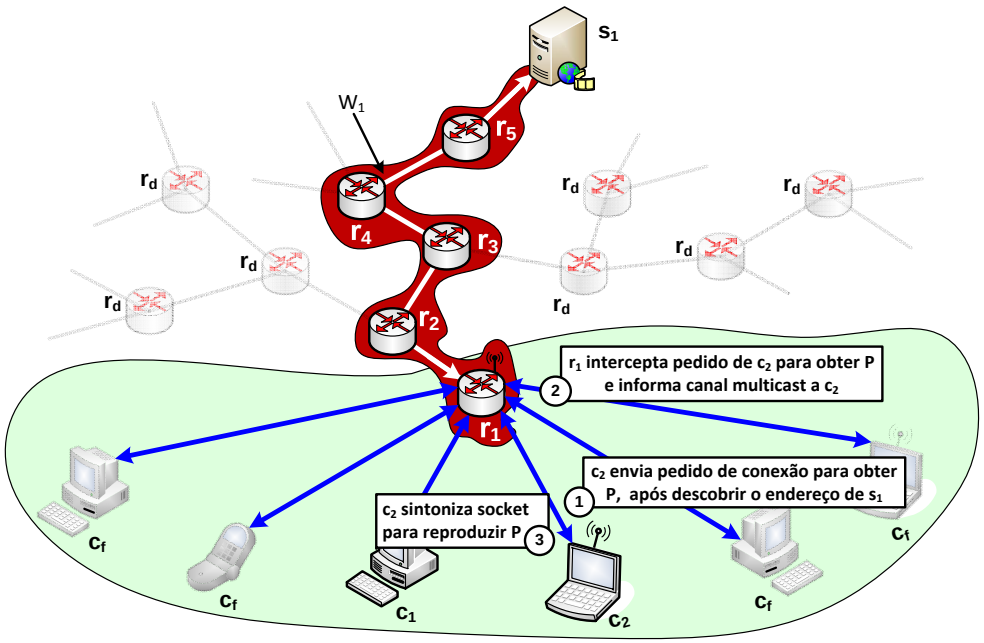

Figura 1.12: Passos do processo de estabelecimento de conexão do GMTP (Fase 2).

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 $p_x \in P$, através de caminhos W_v alternativos. Nesse contexto, seja um nó r_3 que esteja recebendo P originado em um nó s_a. Como ilustra-se na Figura 1.13, 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. O nó s_a constrói a lista de nós parceiros e envia ao nó r_d, funcionando como um indexador (*tracker*) de nós parceiros r_q, pré-selecionando parcerios para r_d. No caso do exemplo supracitado, essa pré-seleção ajuda o nó r_3 a escolher os melhores parcerios disponíveis, de acordo com os seguintes critérios de prioridade:

1. Maior capacidade de transmissão do caminho W_v. Define-se este critério com base na menor taxa de transmissão disponível entre todos os nós $w_m \in W_v$ em um determinado instante *t*. Na Seção 1.5, discute-se os algoritmos de controle de congestionamento do GMTP e o procedimento para determinar a taxa de transmissão de um caminho W_v;

2. Se for um caminho for $W_v^{\bullet}$, determinado através da verificação da condição $|W_v| = ttl(r_d,W_v)$, onde *ttl* é uma função que determina o número de saltos entre o nó r_d até o nó s_a. Este critério é importante porque quanto mais nós GMTP estiverem no caminho, maior será a possibilidade de interceptação para obter um fluxo de dados P;

3. Escolha aleatória de W_v entre os caminhos W conhecidos. Note que, por exemplo, no CoolStreaming, a escolha aleatória é feita em nível de nó cliente, ao passo que no GMTP a escolha é com base na capacidade de transmissão dos caminhos $W_v \in W$.

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

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

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

- Um nó r_d inicia uma nova parceria se a quantidade atual de parcerias reduzir por desconexão de um nó parceiro r_q ou se o buffer de recepção estiver com menos de $\frac{1}{3}$ de sua capacidade. O objetivo é evitar o esvazeamento do buffer para o fluxo de dados P, mantendo continuamente o repasse de pacotes de dados $p_x \in P$ aos nós $c_f \in C_i(r_d)$.
- A quantidade de parcerias em um determinado instante *t* é inversamente proporcional a quantidade de pacotes de dados $p_x \in P$ que chegam repetidos ao nó r_d. Nesse caso, se um mesmo pacote p_x chegar repetidamente na mesma quantidade de parcerias estabelecidas, o nó r_d desconecta-se daquele nó parceiro r_q cujo pacote p_x chegou por último.

2. Após obter a lista de candidatos a parceiros (Passo 1), o nó r_3 forma uma parceria com um dos candidatos da lista de possíveis parceiros ao enviar requisições do tipo *GMTP-Request* em direção a outro nó r_q que já esteja recebendo um fluxo de dados P. Como ilustra-se na Figura 1.14, este procedimento ocorre da seguinte forma: após o passo anterior, o nó r_3 envia uma requisição do tipo *GMTP-Request* para o nó r_2 contendo uma chave de autorização conhecida por ambos e informada pelo nó s_a. Caso a chave de autorização esteja correta, o nó r_2 deve enviar um resposta do tipo *GMTP-Response* ao nó r_3 e então começar a repassar os pacotes $p_x \in P$. O uso da chave de autorização é importante para evitar que um nó r_d se conecte a outro nó r_q sem que o nó s_a seja notificado sobre isto. As chaves de autorização são geradas pelo nó s_a e transmitidas como resposta no pacote do tipo *GMTP-Register-Reply*. Cada entrada disponível no pacote do tipo *GMTP-RelayQuery-Reply* contém endereço IP do nó repassador candidado a parceiro e sua respectiva chave de autorização;

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

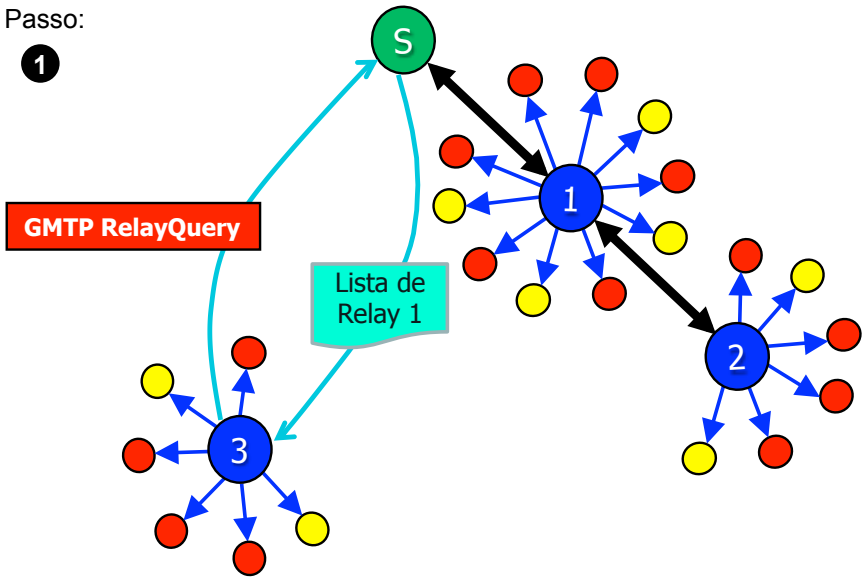

Figura 1.13: Fase 3 de conexão do GMTP (Passo 1).

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

Com a execução da Fase 3 do processo de conexão do GMTP, pode-se expandir a quantidade de parcerias e se registra tais informações na tabela de recepção de fluxos de dados, tal como ilustra-se Figura 1.15. Nesse exemplo, observa-se que um nó r_d está recebendo e repassando aos seus nós $c_f \in C_i(r_d)$ quatro fluxos de dados diferentes, originados em quatro nós s_a, porém recebendo fluxos de dados de diferentes nós parceiros r_q. Por exemplo, dentre os fluxos de dados que o nó r_d está recebendo, um deles é o $P =$ *72c44591-7d82-427c-825f-722f015787c1*, cujos pacotes de dados $p_x \in P$ são transmitidos por três nós r_q identificados pelos endereços IP e porta *182.111.88.21:49170*, *90.39.135.46:62242* e *83.67.132.41:53434*. Para esse fluxo de dados P, o nó r_d repassa os pacotes de dados p_x para os nós $c_f \in C_i(r_d)$ através do canal multicast *239.192.68.79:1900*.

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

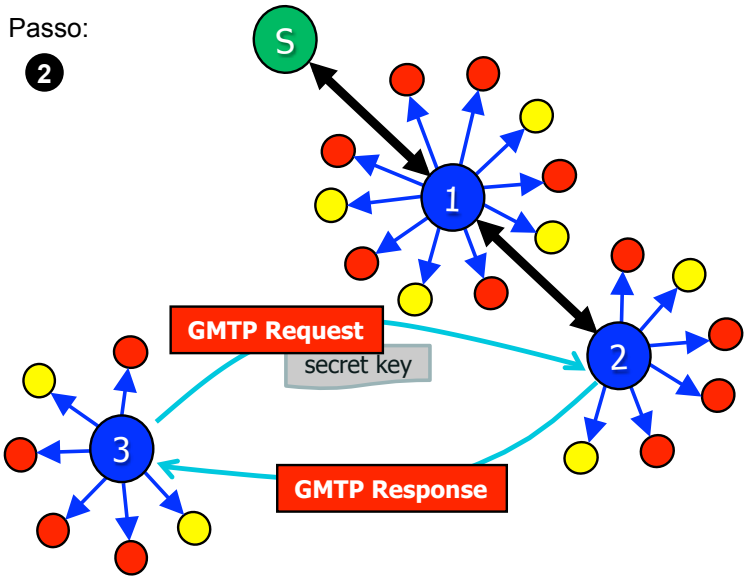

Figura 1.14: Fase 3 de conexão do GMTP (Passo 2).

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

Figura 1.15: Tabela de recepção de fluxos de dados após a Fase 3.

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

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

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

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

Buffer de Envio e Recepção:

A transmissão de um evento $\mathcal{E}$ consiste no processo de disseminação dos pacotes $p_x \in P$ através dos nós interessados em obtê-lo. Para isto, cada nó GMTP controla um buffer de envio e recepção definido por uma estrutura de dados do tipo array circular (*ring buffer*), onde cada posição é utilizada para armazenar um pacote p_x, como ilustra-se na Figura 1.16. Ao receber p_x, um nó GMTP armazena-o no buffer e posteriormente o entrega para a aplicação, que o reproduz para o usuário final. Para o envio ou repasse de um pacote, o nó GMTP consome os pacotes p_x do buffer e transmite para o(s) nó(s) interessado(s), seja em modo unicast e/ou em modo multicast. Isto porque é possível que um nó r_1 repasse p_x para um outro nó r_2 (unicast) ao mesmo tempo que r_1 pode repassar P para seus nós c_f (multicast).

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

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

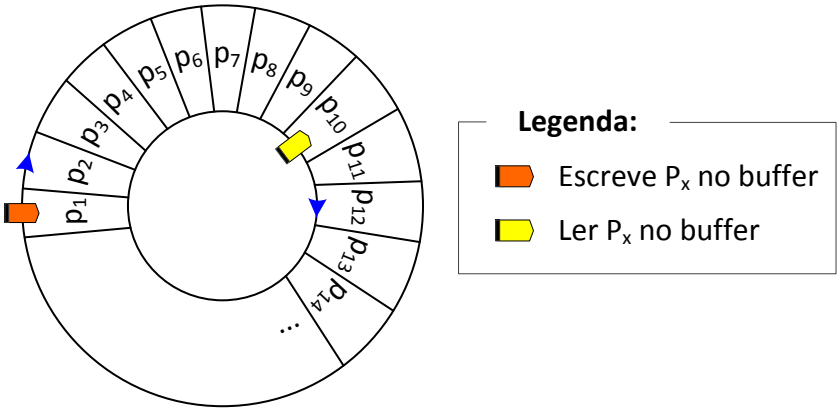

Figura 1.16: Exemplo da estrutura do buffer de envio e recepção de um nó GMTP com dois ponteiros, um para escrever e outro para ler pacotes p_x.

Mapa de *Buffer*:

O mapa de buffer do GMTP descreve o estado atual do buffer de envio e recepção de um nó GMTP. Como ilustrado na Figura 1.17, trata-se de uma estrutura de dados que determina se um pacote p_x está ou não presente no buffer de um respectivo nó GMTP. O conteúdo de cada posição é o número de sequência do pacote, que determina a ordem que um pacote foi gerado e transmitido pelo nó s_a, mas o nó r_d não os ordena, pois tal responsabilidade é delegada aos nós c_f no momento de sua reprodução ao usuário final.

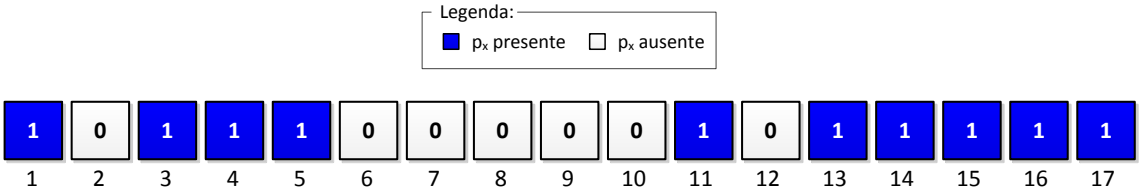

Figura 1.17: Exemplo do mapa de buffer de um nó GMTP com tamanho de 17 p_x.

Um nó GMTP utiliza o mapa de buffer para sinalizar seu atual estado com relação a um determinado fluxo de dados P. Um nó GMTP pode enviar o mapa de buffer completo, como ilustrado na Figura 1.17, ou o mapa de buffer apenas dos p_x presentes ou ausentes. Na prática, quando deseja indicar a sua atual disponibilidade, um nó r_d envia para um nó parceiro r_q o mapa de buffer dos p_x presentes e, quando desejar obtê-los, envia o mapa de buffer dos p_x ausentes. Para diferençar o tipo de requisição, utiliza-se uma sinalização binária (*flag*) chamada *request-type*, onde 0 significa que o mapa de buffer contém pacotes disponíveis e 1, pacotes ausentes. Note que, quando um nó r_d transmite um mapa de buffer para um outro nó qualquer r_q, caracteriza-se automaticamente o uso do método *pull*, em vez do método *push*, que é o modo padrão do GMTP. Salienta-se ainda que deve-se evitar o método pull devido à transitoriedade dos pacotes de dados p_x (transmissão de dados ao vivo) e um nó r_d deve apenas realiza tal procedimento após completar a Fase 3 do processo de estabelecimento de conexão. Isto porque um nó r_d pode nunca receber a resposta para uma requisição do tipo *pull*. Essa sinalização ocorre através do uso do pacote do tipo *GMTP-DataPull-Request*, que é preenchido com o mapa de buffer dos pacotes ausentes e transmitido aos respectivos nós parceiros. Ao receber esse tipo de requisição, um nó parceiro avalia seu conteúdo e responde com o pacote do tipo *GMTP-DataPull-Response*, o qual contém o mapa de buffer dos pacotes disponíveis, seguido dos pacotes p_x do tipo *GMTP-Data*. Note que os pacotes do tipo *GMTP-DataPullRequest* e *GMTP-DataPull-Response* são transmitidos com garantia de entrega.

Na prática, o mapa de buffer utilizado para sinalizar a presença ou ausência de p_x é representado por faixas de acordo com o índice do buffer. Por exemplo, para representar o mapa de buffer dos pacotes ausentes ilustrados na Figura 1.17, o nó GMTP preenche o pacote do tipo *GMTP-DataPull-Request* com a sequencia *2;6-10;12*. Ao receber esta sequência, o

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

Descarte de pacotes:

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

1. **Por transbordo do buffer:** o transbordo do buffer pode ocorrer devido ao mecanismo de controle de congestionamento empregado no GMTP, que pode reduzir a taxa de transmissão enquanto novos pacotes precisam ser alocados no buffer. Sendo assim, deve-se descartar os primeiros pacotes p_x recebidos se o buffer alcançou seu limite, mesmo que ainda não tenham sido repassados. Uma otimização não explorada neste trabalho, mas que é possível de ser realizada, é o descarte seletivo de pacotes, primeiro os que tenham menos impacto na qualidade da mídia, por exemplo, pacotes de dados contendo quadros B (codificação MPEG4, tipo 2). O descarte seletivo de pacotes não impede que o vídeo seja reproduzido, porém com perda de qualidade, mas ao menos se permite a transmissão dos pacotes de dados p_x de acordo com a largura de banda disponível;

2. **Por duplicação:** ocorre quando o pacote p_x já foi recebido anteriormente. Tal verificação é feita de acordo com o número de sequência presente em cada pacote p_x. Note que essa contagem é importante e pode determinar que um nó r_d desconecte de um nó parceiro r_q, tal como explicou-se na Seção 1.4.5.

1.5 Controle de Congestionamento em η

No GMTP, executa-se um algoritmo para controle de congestionamento híbrido, cujo comportamento dependerá do modo de transmissão sendo utilizado para transportar os pacotes de dados $p_x \in P$(unicast ou em multicast). Como ilustra-se na Figura 1.18, trata-se de dois algoritmos para controle de congestionamento, um que atua em transmissões unicast, chamado de *GMTP Unicast Congestion Control* (GMTP-UCC) e outro que atual em transmissões multicast, chamado de *GMTP Multicast Congestion Control* (GMTP-MCC). Nas próximas seções, discute-se o funcionamento de cada um desses algoritmos.

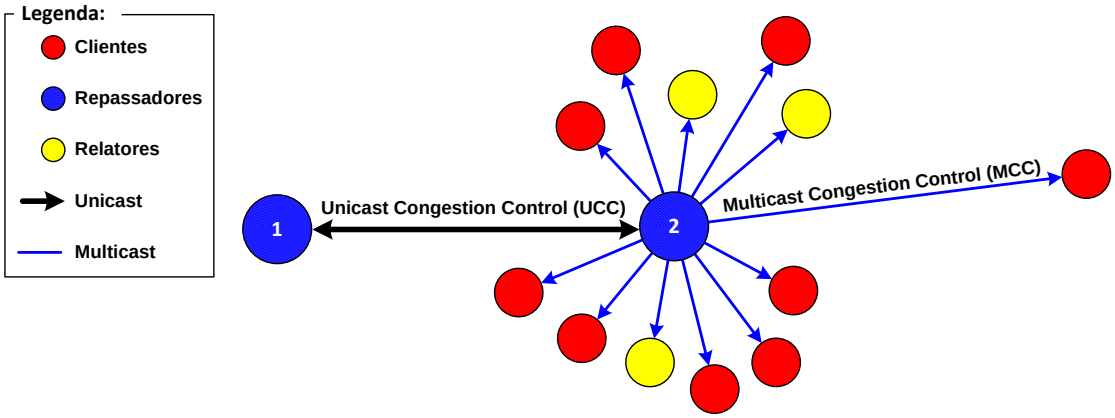

Figura 1.18: Organização do algoritmo de controle de congestionamento no GMTP.

Em modo de transmissão unicast, utilizado na comunicação entre os nós r_d, define-se a taxa de transmissão de um nó GMTP através de uma versão modificada do protocolo RCP [13]. Já em modo de transmissão multicast, executa-se uma versão modificado do TFRC [14], utilizando-se relatórios transmitidos por nós relatores $l_w \in C_i(r_d)$, eleitos em cada rede e controlados por um nó r_d.

1.5.1 Controle de congestionamento unicast

O GMTP-UCC funciona de forma similar ao protocolo RCP, porém com alguns diferenciais a serem discutidos a seguir. O RCP é um protocolo para controle de congestionamento assistido pela rede que tenta emular um Comutador Compartilhado (*Processor Sharing* – PS), por exemplo, um roteador [15]. Nesse contexto, entende-se que se um roteador pudesse obter a informação exata sobre o número de fluxos de entrada em um instante $\hat{t}$, a taxa de transmissão ideal para cada fluxo de dados seria $R_{ps}(\hat{t}) = \frac{C}{N(\hat{t})}$, onde C corresponde à capacidade do *enlace* e $N(\hat{t})$ o número de fluxos no instante $\hat{t}$.

Partindo desse ponto, Nandita et. al [13] argumentou em sua tese de doutorado[3] que para um roteador funcionar de forma equânime e reduzir o tempo de duração de um fluxo (seja de curta ou de longa duração, proporcional à capacidade e independente da topologia da rede), deve-se oferecer a mesma taxa de transmissão para todos os fluxos transmitidos através dele. Com isso, objetiva-se manter o número de pacotes na fila de roteamento perto de zero e evitar que apenas os fluxos mais antigos, ou seja, com a taxa de transmissão mais

[3]O vídeo do dia da defesa de Nandita Dukkipati, autora do RCP, está disponível através da referência [16].

próxima da equânime, utilizem mais largura de banda se comparado aos fluxos mais recentes (discussões sobre este aspecto mais adiante, ainda nessa seção).

Com base nisso, determinou-se a Equação 1.1, onde $R(\hat{t})$ é a taxa de transmissão que deve ser oferecida para cada fluxo de dados que passa pelo roteador. Pela Equação 1.1, estima-se a largura de banda disponível em um determinado canal, representada pela porção $\alpha(C - y(\hat{t})) - \beta\frac{q(\hat{t})}{h_0}$ (mudança agregada) e a divide por $N(\hat{t})$. Porém, como é impossível determinar o valor exato de $N(\hat{t})$, estima-se[4] $\hat{N}(\hat{t}) = \frac{C}{R(\hat{t}-h_0)}$. Além disso, para atualizar $R(\hat{t})$ com mais frequência do que no intervalo de um RTT (h_0), definiu-se $H = min(H_{user}, h_0)$ e, para manter a estabilidade do restante da equação, escala-se a mudança agregada por $\frac{H}{h_0}$, resultando na Equação 1.2, onde:

$$R(\hat{t}) = R(\hat{t} - h_0) + \frac{\alpha(C - y(\hat{t})) - \beta\frac{q(\hat{t})}{h_0}}{\hat{N}(\hat{t})} \tag{1.1}$$

$$R(\hat{t}) = R(\hat{t} - H)\left[1 + \frac{\frac{H}{h_0}\left(\alpha(\gamma C - y(\hat{t})) - \beta\frac{q(\hat{t})}{h_0}\right)}{\gamma C}\right] \tag{1.2}$$

- h_0, é a média móvel dos valores de RTT_s, calculada através da Equação 1.3, onde θ é o ganho e corresponde a 0.02. Note que quanto maior o valor de θ, mais rápida será a convergência de h_0 ao valor de RTT_s. RTT_s é o tempo de ida e volta calculado entre o nó transmissor e o receptor.

$$h_0 = \theta \times RTT_s + (1 - \theta) \times h_0 \tag{1.3}$$

- $H = min(H_{user}, h_0)$, sendo H_{user} um tempo definido pelo usuário (por exemplo, o administrador do roteador), caso seja necessário atualizar $R(\hat{t})$ em um intervalo de tempo menor que h_0. Por exemplo, se a fila estiver enchendo, é desnecessário esperar um tempo de h_0 para processar a fila. O valor para H é definido em *milissegundos*;

- $R(\hat{t} - H)$, é a última taxa de transmissão medida, em *bytes/milissegundos*;

- $y(\hat{t})$, é a taxa de tráfego de entrada medida no intervalo entre a última atualização de $R(\hat{t})$ e o instante H;

[4]Mais detalhes sobre a estimativa do valor de $\hat{N}(\hat{t})$ está disponível na Seção 3.2.1 da referência [13].

- $q(\hat{t})$, é o tamanho instantâneo da fila de repasse, em bytes. No GMTP, esse valor é obtido pela soma de todos os pacotes p_x presentes nos buffers, para todos os fluxos de dados P registrado na tabela de repasse. Nesse caso, um nó r_d mantém um buffer geral e um buffer para cada fluxo de dados P, que esteja repassando aos seus nós $c_f \in C_i(r_d)$. Utiliza-se o buffer geral para pacotes de dados que não precisam de tratamentos especiais, por exemplo, pacotes TCP;

- α e β, tal que $0 < \alpha, \beta \leq 1$, determinam a estabilidade e o desempenho do algoritmo, respectivamente. Com base em discussões apresentadas em [13], os valores de α e β dependem do tamanho médio dos fluxos comparado com o produto largura de banda – atraso. Quando o tamanho médio dos fluxos estão próximo do produto largura de banda – atraso (fluxos longos), sugere-se um alto valor para α e um baixo valor para β (por exemplo: $\alpha = 0.9$ e $\beta = 0.1$), uma vez para fluxos longos, prefere-se maximizar a taxa de transmissão de cada fluxo a minimizar o atraso na fila. Por outro lado, para fluxos de curta duração, recomenda-se um valor baixo de α e um valor alto de β (por exemplo: $\alpha = 0.1$ e $\beta = 1$), pois esta combinação ajuda manter um baixo atraso de fila. Para um meio termo, recomenda-se valores de $\alpha \in (0.4, 0.6)$ e $\beta = (0.2, 0.6)$;

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

- C, é a capacidade (largura de banda) do canal.

Antes de detalhar o funcionamento do GMTP-UCC, a ideia básica é a seguinte: para quaisquer dois nós t_1,$t_2 \in W_v$, a taxa de transmissão a ser utilizada por t_1 e t_2 será definida pela menor taxa de transmissão oferecida pelos nós $w_m \in W_v$ posicionados entre t_1 e t_2. Isto significa que o GMTP-UCC segmenta um caminho W_v em vários sub-caminhos $W_v^{\triangleleft}$. Com isto, se existir largura de banda disponível entre t_1 e t_2, ou seja, $C - y(\hat{t}) > 0$, então o GMTP-UCC compartilhará igualmente o canal entre todos os fluxos, inclusive para o fluxo partindo de t_1 em direção a t_2. Caso contrário, ou seja, se $C - y(\hat{t}) < 0$, considera-se o canal saturado e o GMTP-UCC reduzirá a taxa de transmissão igualmente para todos os fluxos, inclusive

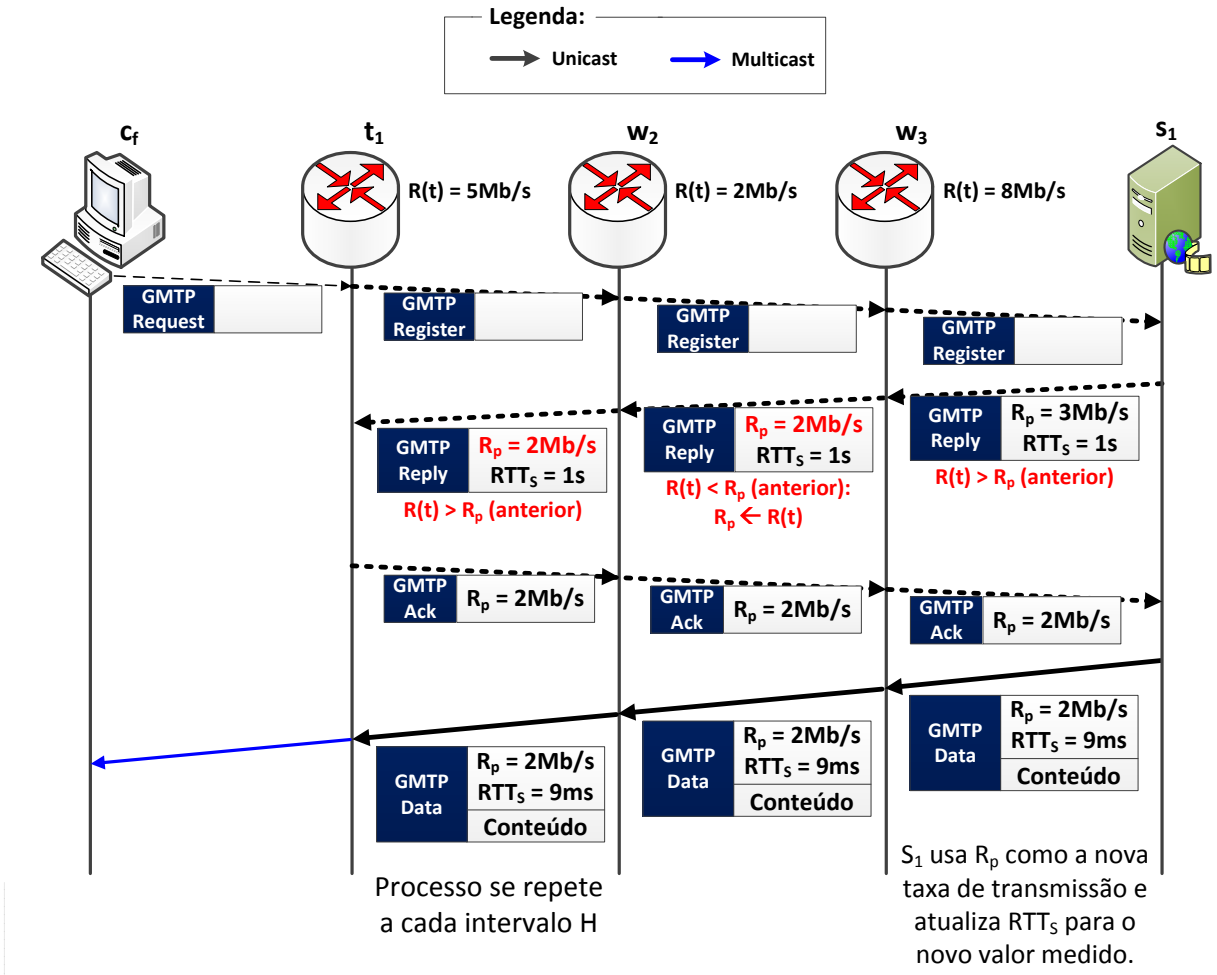

Figura 1.19: Cada r_d mantém uma única taxa de transmissão $R(\hat{t})$ que é atribuída no cabeçalho de todos os pacotes transmitidos do nó s_a aos nós $w_m \in W_v$. À medida que o pacote passa em cada w_m, se a taxa atual $R(\hat{t})$ no roteador for menor do que R_p, informada no pacote sendo processado, $R_p \leftarrow R(\hat{t})$. Quando o pacote alcançar o último nó w_m, este envia para s_a o valor de R_p, que é a máxima taxa de transmissão suportada no caminho W_v. Ao receber o valor de R_p, s_a atualiza $R(\hat{t})$ e o valor de h_0, utilizando $R(\hat{t})$ para transmitir os próximos pacotes de dados. Este procedimento se repete a cada intervalo de tempo H.

para o fluxo partindo de t_1 para t_2. Por este motivo, o tempo H é definido entre dois nós t_u e t_{u+1} contidos em um caminho W_v. A consequência dessa estratégia de segmentar um caminho é muito importante e por esse motivo o GMTP-UCC é relativamente diferente se comparado ao RCP, como se discute a seguir.

O algoritmo adotado no GMTP-UCC, adaptado do RCP, funciona da seguinte forma (acompanhe os passos de acordo com a Figura 1.19):

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

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

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

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

3° No início da transmissão de um fluxo de dados P por parte de s_a, o nó w_m, motivado por um ou mais nós $c_f \in C_i(w_m)$, transmite um pedido de registro de participação ao nó s_a (Seção 1.3.1). Ao receber o pacote *GMTP-Register*, o nó s_a envia um pacote *GMTP-Register-Reply* com o campo R_p contendo a taxa de transmissão necessária para transmitir o fluxo de dados P, com o campo RTT_s igual a $1\,s$ [17]. O valor inicial de $RTT_s = 1\,s$ é bastante conservador, mas se decidiu utiliza-lo por ser o valor inicial adotado no protocolo TCP. Além disso, inicia-se um temporizador para medir o próximo RTT instantâneo, que de fato será o primeiro valor correto e alimentará a média móvel h_0.

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

(a) se o pacote for do tipo *GMTP-Data*, repassa-se o pacote para seus nós $c_f \in C_i(t_u)$ através do canal multicast, a uma taxa de transmissão definida pelo algoritmo GMTP-MCC, como se discute na próxima seção; e também repassa-o em direção ao próximo nó $t_{u+1} \in W_v$, se houver. A transmissão dos pacotes *GMTP-Data* partindo de t_u em direção ao nó t_{u+1} (ou seja, *down-streaming*), ocorre a uma taxa de $R(\hat{t})$ atualmente definida em t_u;

(b) cria-se um pacote *GMTP-Ack* informando o valor de R_p e o envia de volta em direção a t_{u-1}, se $t_u \neq s_a$. Ao receber esse pacote de *GMTP-Ack*, t_{u-1} utilizará R_p como a nova taxa para transmitir os próximos pacotes de dados $p_x \in P$, pois se trata da menor taxa de transmissão oferecida ao longo do sub-caminho $W_v^{'} \subset W_v$, tal que $W_v^{'} = \{t_{u-1}, ..., w_m, w_{m+1}, w_{m+2}, ..., t_u\}$.

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

5° A cada instante H, o nó w_m atualiza h_0 de acordo a Equação 1.3, utilizando como parâmetro o valor do campo RTT_s do último pacote recebido. Em seguida, recalcula-se a taxa de transmissão local $R(\hat{t})$ usando a Equação 1.2.

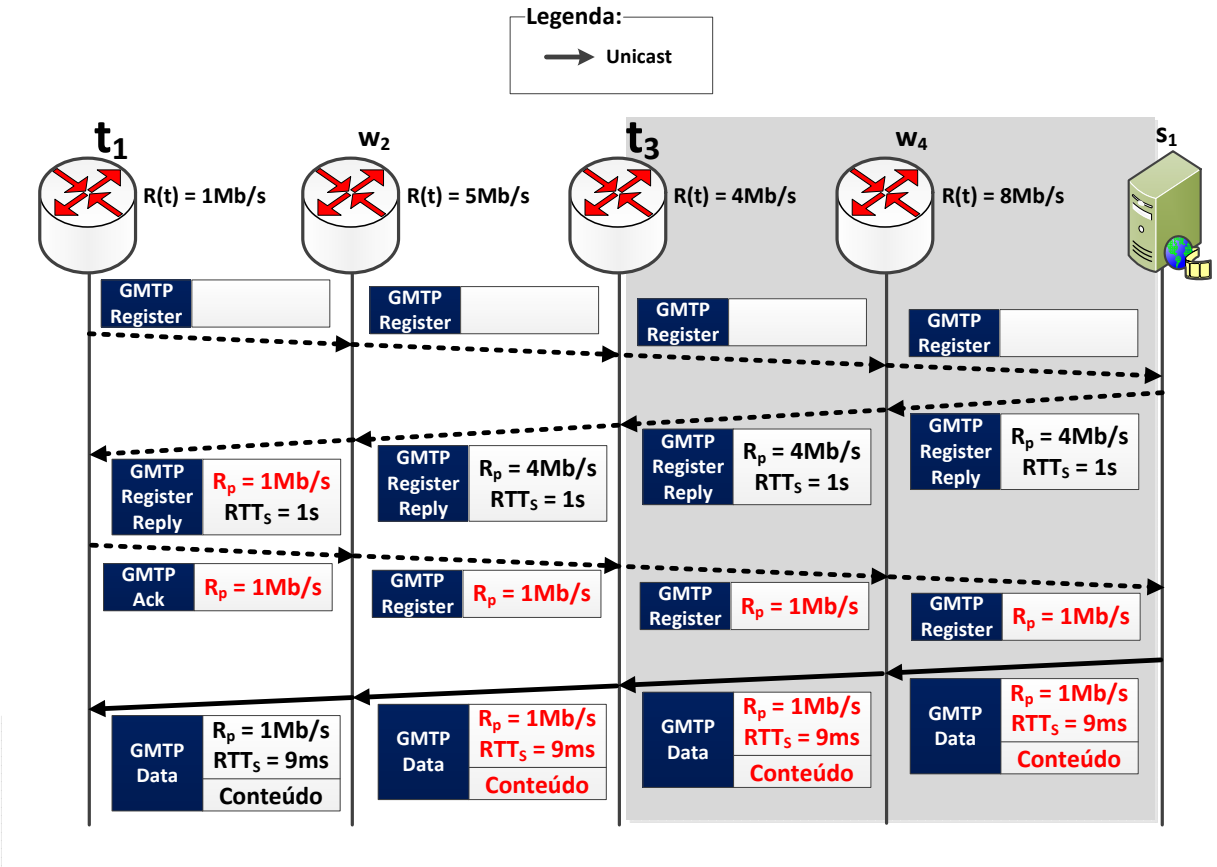

Figura 1.20: O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão, porém isto pode limitar alguns nós clientes a receberem os pacotes de dados em uma taxa maior. Nesse caso, o nó t_3 tinha capacidade para receber o conteúdo a uma taxa de transmissão de 4 *Mb/s*, porém a taxa de máxima relatada por t_1 é de 1 *Mb/s*, fazendo com que todos os nós no caminho W_v recebam o fluxo de dados P a 1 *Mb/s*.

Segmentação de um caminho W_v:

O RCP considera todo o caminho entre o nó transmissor e o nó receptor para determinar o novo valor da taxa de transmissão do nó transmissor, especificado por $R(\hat{t})$ e atualizado a cada instante H utilizando o novo valor medido de R_p e h_0. Porém, essa estratégia pode limitar alguns nós c_f a receberem os pacotes de dados $p_x \in P$ em uma taxa maior, quando disponível. Por exemplo, na Figura 1.20, ilustra-se um cenário de transmissão usando apenas o RCP, abstraindo-se os nós $c_f \in C_i(w_m)$. Nesse cenário, observa-se um caminho $W_v = \{t_1, w_2, t_3, w_4\}$. Isto significa que existem nós $c_f \in (C_i(t_1) \cup C_i(t_3))$ interessados em receber os pacotes de dados p_x. Ao utilizar apenas o RCP, o nó s_a transmitirá pacotes de dados p_x

a uma taxa de transmissão de 1 *Mb/s* tanto para os nós $c_f \in C_i(t_1)$ quanto para os nós $c_f \in C_i(t_3)$. Porém, isso faz sentido apenas para os nós $c_f \in C_i(t_1)$ e não para os nós $c_f \in C_i(t_3)$, visto que em t_3 o valor de $R(\hat{t})$ é igual a 4 *Mb/s* e o nó w_4 não limita o uso dessa taxa de transmissão para os nós $c_f \in C_i(t_3)$, uma vez que em w_4 o valor de $R(\hat{t})$ corresponde a 8 *Mb/s*. No caso do GMTP-UCC, esta limitação foi superada ao determinar que se $\varphi(w_m, P) = 1$, ou seja, $w_m = t_u \in T$, então a taxa de transmissão informada em R_p será utilizada por w_m, porém não será considerada para determinar a taxa de transmissão do próximo nó t_{u+1}. Por isso, o GMTP-UCC segmenta o caminho W_v de acordo com a menor taxa de transmissão entre dois nós t_u e t_{u+1}, como já foi discutido anteriormente.

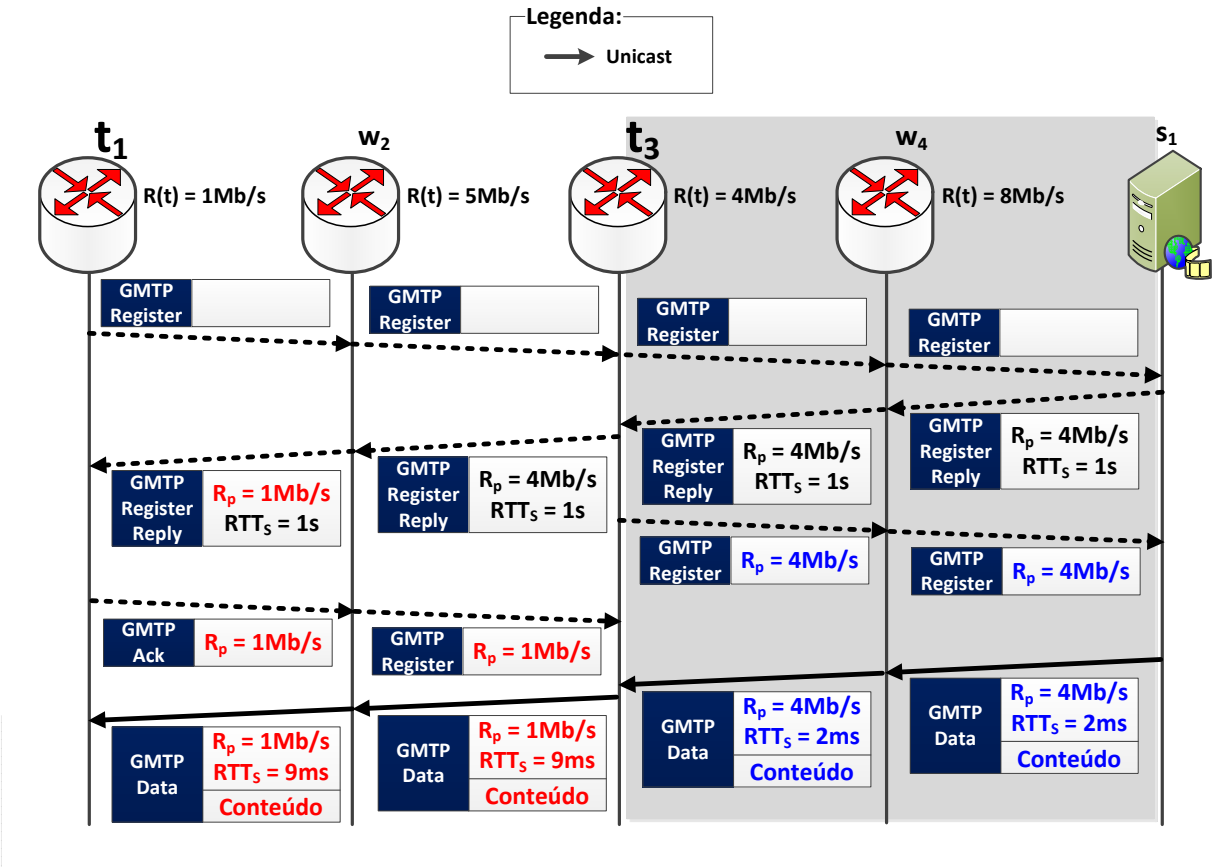

Figura 1.21: O GMTP-UCC segmenta o caminho e dessa forma não limita a taxa de transmissão de um fluxo de dados para certos nós capazes de receber em uma taxa de transmissão maior.

Sendo assim, considerando o mesmo exemplo ilustrado na Figura 1.20, mas adotando essa estratégia de segmentar o caminho W_v, tal cenário corresponde ao ilustrado na Fi-

gura 1.21. Note que, no sub-caminho $W_1^{\triangleleft} = \{t_3, w_2, t_1\}$, a taxa de transmissão de t_3 em direção a t_1 será de 1 *Mb/s*, ao passo que no sub-caminho $W_2^{\triangleleft} = \{s_1, w_4, t_3\}$ será de 4 *Mb/s*. Sendo assim, os nós $c_f \in C_i(t_1)$ receberão o fluxo de dados P em uma taxa de 1 *Mb/s*, ao passo que os nós $c_f \in C_i(t_3)$ receberão os pacotes de dados p_x a uma taxa de 4 *Mb/s*.

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

Na Seção 1.4.5, discutiu-se que na Fase 3 de conexão do GMTP, um nó s_a pode sugerir possíveis nós r_q como candidatos a parceiros de um nó solicitante r_d. O primeiro critério para sugerir nós parceiros r_q é priorizar os que fazem parte de um caminho W_v com maiores capacidade de transmissão. No GMTP, isto é possível porque os nós s_a conhecem a capacidade de transmissão de todo o caminho, inclusive os pontos onde se termina um sub-caminho e se inicia o subsequente, obtidos pelo Passo 4b do algoritmo GMTP-UCC.

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

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

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

transmissão $R_{rcp}(\hat{t})$ e $R_{xcp}(\hat{t})$ é bastante diferente, alocando-se suas taxas para cada fluxo de dados a fim de emular $R_{ps}(\hat{t})$. Dessa forma, foi fundamental decidir qual dos dois protocolos seria mais adequado ao GMTP-UCC e, para tomar tal decisão, estudou-se as diferenças entre tais protocolos, com base no que se apresenta a seguir e detalhado em [13, 15].

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

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

Para se ter uma visão mais adequada, nos gráficos da Figura 1.22, compara-se o TCP e o XCP com um PS ideal com base em uma rede simulada, com taxa de entrada de pacotes de dados definida em *Poisson* e tamanhos dos fluxos em distribuição *Pareto*, com média de 30 pacotes (1000 *bytes/pacote*), *shape* igual a 1.4, capacidade do link igual a 2.4 *Gbps* e RTT igual a 100 *ms*, com carga ofertada igual a 0.9. No gráfico da esquerda, ilustra-se o tempo médio de duração de um fluxo (quanto tempo o respectivo protocolo gasta para completar o fluxo) em função do tamanho do fluxo. No gráfico da direita, ilustra-se o número de fluxos ativos em função do tempo. Os valores de PS foram calculados a partir de expressões analíticas [18] e mostram que os fluxos poderiam ser finalizados uma ordem de magnitude mais rápida do que o TCP.

Com base nos gráficos da Figura 1.22, observa-se que os fluxos TCP demoram para fi-

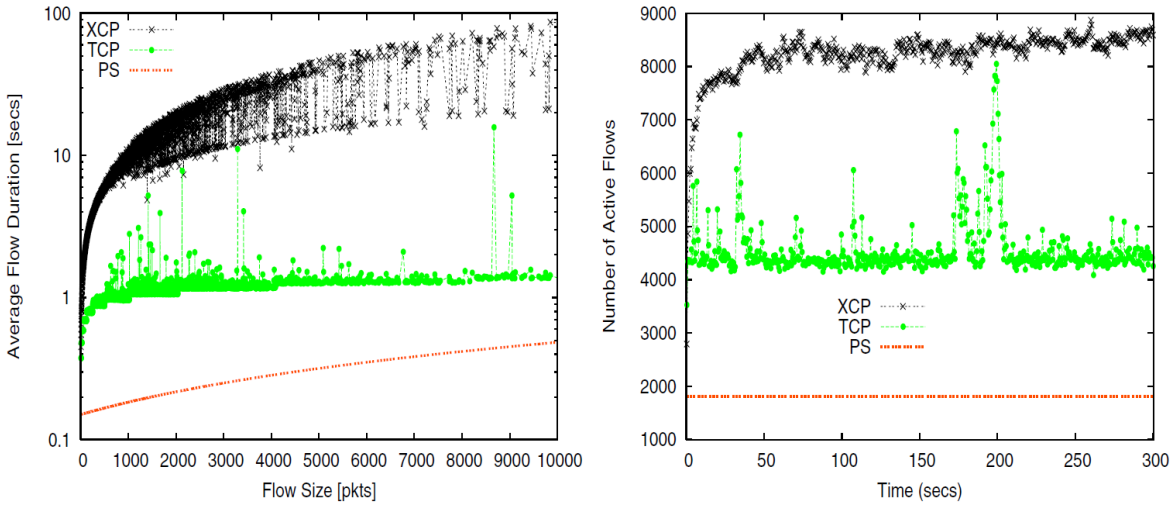

Figura 1.22: No gráfico da esquerda, ilustra-se o tempo médio de duração (quanto tempo leva para finalizar o fluxo) de um fluxo versus o tamanho do fluxo utilizando o TCP e o XCP. No gráfico da direita, ilustra-se o número de fluxos ativos versus o tempo. Ambos os gráficos são resultados de simulações com chegada de fluxo em *Poisson* e tamanhos do fluxo em distribuição *Pareto* com média de 30 pacotes (1000 *bytes/pacote*), *shape* igual a 1.4, capacidade do *enlace* igual a 2.4 *Gbps* e RTT igual a 100 *ms*, com carga ofertada igual a 0.9. Os valores de PS foram calculados a partir de expressões analíticas. Extraído de [13].

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

Um outro exemplo comparativo e importante entre o RCP, XCP e TCP se observa na Figure 1.23. Nesses gráficos, ilustra-se dois fluxos de tamanhos distintos, um considerado de curta duração (260 pacotes) e outro de longa duração (3600 pacotes). Note que no primeiro gráfico o fluxo do TCP finalizou primeiro (enquanto ainda estava na fase de partida lenta) do que mesmo transmitido usando o XCP. Já no segundo gráfico, observa-se o impacto causado no TCP quando houve uma perda de pacote na fase de partida lenta, forçando-o a entrar na fase de AIMD, retardando a finalização do fluxo. Em ambos os casos, o RCP obteve um melhor desempenho se comparado ao TCP e ao XCP porque o roteador oferece uma taxa de transmissão inicial muito próxima ao PS, sendo eficiente em rapidamente perceber largura de banda ociosa e que pode ser oferecida aos fluxos.

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

Já no RCP, todos os fluxos (novos e antigos) recebem a mesma taxa de transmissão $R_{rcp}(t)$ baseada no estado atual do nó r_d com menor largura de banda disponível em um certo instante t. Isto permite que um fluxo de dados de curta duração termine o mais rápido possível, ao passo que os fluxos de dados mais longos não influenciam diretamente no compartilhamento equânime do PS, sem permitir que parte da largura de banda disponível fique ociosa por muito tempo. Este procedimento ocorre em um intervalo de tempo definido por H (vide Equação 1.2). Isso ocorre ao preço de ocorrer uma super-utilização temporária do canal (quando ocorrer um aumento substancial no número de fluxos no intervalo menor do que H – *flash crowd*), mas para fluxos de mídias ao vivo pode-se tolerar perdas circunstanciais.

Já ao observar o gráfico da Figura 1.24, percebe-se que a estratégia do RCP de compartilhar uma única taxa de transmissão para qualquer fluxo com base no estado atual do roteador saturado, produz um resultado satisfatório no que diz respeito a melhor utilizar o canal de transmissão (seja quando em altos níveis de utilização quanto de ociosidade). Com base no gráfico, percebe-se que, em comparação ao XCP e a outras soluções tradicionais como o

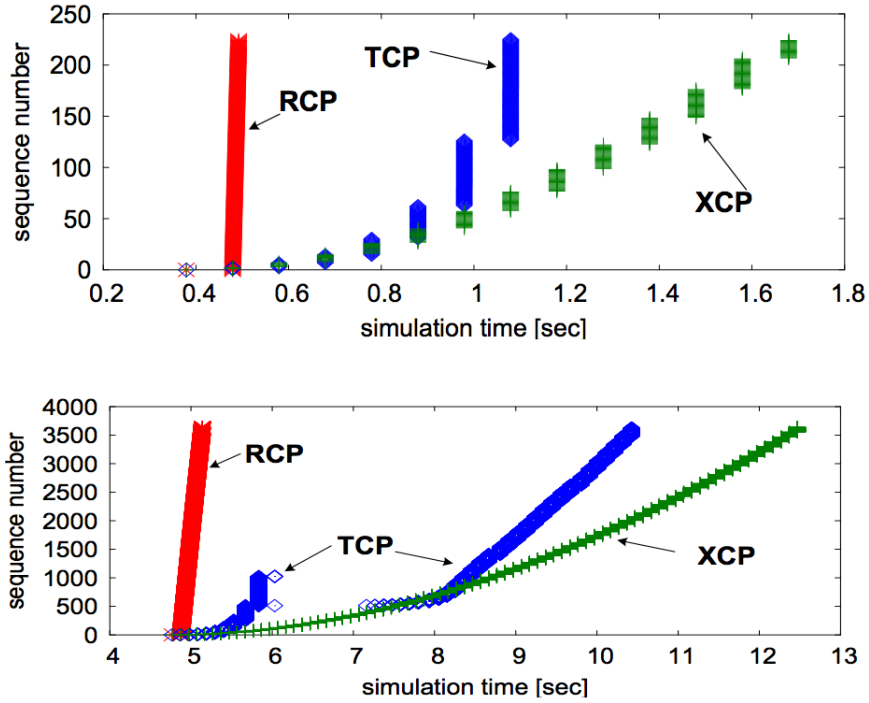

Figura 1.23: Evolução dos números de sequência de fluxos de dados quando utilizando TCP (Reno), XCP e RCP. O tamanho do fluxo no primeiro gráfico foi de 230 pacotes, e no segundo gráfico foi de 3600 pacotes. Extraído de [13].

TCP, o RCP emula melhor um PS e por isso acompanha o tempo médio de finalização de um fluxo de dados à medida que se aumenta o tamanho do fluxo. Note que, para o cenário descrito, o XCP teve um desempenho pior se comparado, inclusive, ao TCP.

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

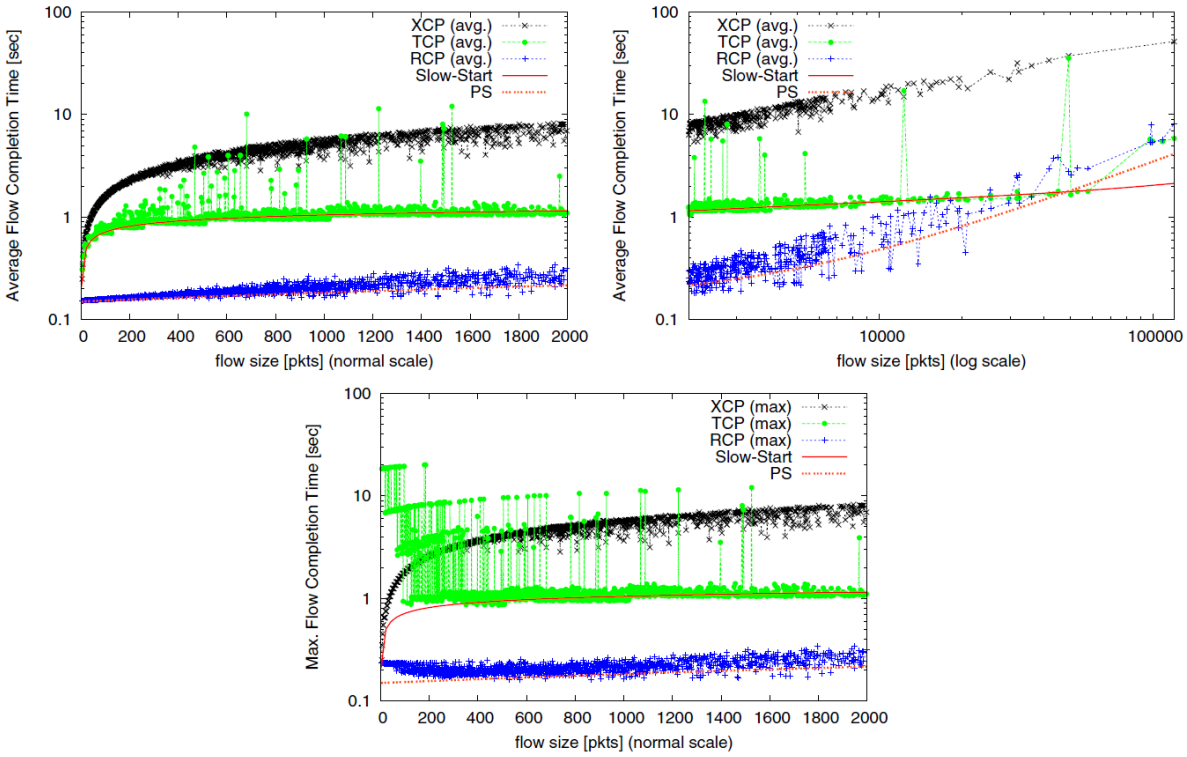

Figura 1.24: Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP como resultados de simulações com taxa de entrada de dados em *Poisson* e tamanhos do fluxo em distribuição Pareto com média de 25 pacotes (1000 *bytes/pacote*), *shape* igual a 1.2, capacidade do link igual a 2.4 *Gbps* e RTT igual a 100 *ms*, com carga ofertada igual a 0.9. Os valores de PS foram calculados a partir de expressões analíticas. Extraído de [13].

matemáticas por pacote de dados.

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

rede, evitando-se sobrecarga nos nós s_a.

Apesar das observações anteriores, não é foco desse trabalho aprofundar as discussões acerca do desempenho do XCP comparado ao RCP, mas para não deixar a impressão que o XCP sempre é ruim comparado ao TCP, vale ressaltar que também há cenários onde o XCP tem um desempenho superior ao TCP, incluindo as variantes de *High Speed* TCP. É possível observar um melhor desempenho do XCP quando se aumenta o tamanho médio de duração de um fluxo para se aproximar ao produto largura da banda – atraso. Nesses casos, aproxima-se de um regime onde existem fluxos consumindo praticamente toda a largura de banda disponível de um canal, com o valor equânime da taxa de transmissão alocada individualmente para cada fluxo. Nesse tipo de cenário, o TCP-SACK apresenta vários problemas de desempenho, que foram aprimorados pelo HS-TCP [13]. Em todo caso, sabe-se que em cenários reais, o tamanho dos fluxos variam, existindo fluxos de curta e longa duração multiplexados a todo instante em um roteador.

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 $\eta_{local} = r_d \cup C_i(r_d)$. Na prática, os nós da rede η_{local} formam um grupo multicast para a transmissão e recepção de um ou mais fluxos de dados P, onde o nó r_d sempre será o transmissor e os nós $c_f \in C_i(r_d)$ os receptores. A estratégia é que o valor a ser utilizado pelo GMTP-MCC para a taxa de transmissão de fluxo de dados P seja tão próximo ao valor da taxa de transmissão que o TCP usaria se este fosse utilizado para transmitir P, tornando-se o GMTP-MCC um algoritmo *TCP-Friendly*. Um fluxo de dados é considerado *TCP-Friendly* quando este não degrada a taxa de transmissão de um fluxo de dados TCP mais do que outro fluxo TCP degradaria se começasse a ser transmitido na rede.

O GMTP-MCC foi inspirado em um protocolo publicado pela IETF chamado *TCP-Friendly Rate Control protocol (TFRC)* (RFC 3448 [19]). O TFRC é um mecanismo para controle de congestionamento de fluxos unicast que tenta prever a taxa de transmissão de um fluxo TCP e utilizá-la em protocolos diferentes do TCP [14]. Trata-se de uma abordagem

diferente da utilizada em algoritmos baseados em janela deslizante e que utilizam pacotes de confirmação para determinar a taxa de transmissão de uma conexão, como acontece no TCP/NewReno. No TFRC, o receptor envia para o transmissor relatórios sobre as perdas observadas e, com base nesse relatório, o transmissor calcula a nova taxa de transmissão. O TFRC é categorizado com um protocolo de controle de congestionamento baseado em uma equação matemática (*Equation Based Congestion Control*) e algoritmos desse tipo são adotados em diversos protocolos, como no CCIDs 3 e 4 do DCCP [20, 21]. Em resumo, o algoritmo TFRC funciona da seguinte forma:

1° o receptor mede a taxa de perda de pacotes e a envia para o nó transmissor;

2° o nó transmissor usa esse relatório para medir o RTT até o receptor;

3° o nó 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;

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

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

Na Equação 1.4 [22], $R(s,p)$ é a taxa de transmissão medida em bytes/segundo definida em função de s, que é o tamanho do pacote medido em bytes e p, que corresponde a taxa de perda de pacotes observado pelo nó receptor; RTT é o tempo de ida-volta entre o nó transmissor e o receptor, medido em segundos.

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

Nesse contexto, para evitar o problema da *explosão de retorno*, determinou-se que apenas alguns nós c_f são obrigados a enviar tais relatórios ao nó r_d. Estes nós são chamados de nós

relatores e representados por l_w. No GMTP-MCC, a versão original do TFRC foi alterada e funciona da segunte forma:

1° O nó r_d executa um algoritmo de eleição de nós relatores $l_w \in C_i(r_d)$. Na Seção 1.7.3, 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, em vez do transmissor realizar este cálculo, como na versão original do TFRC;

3° Os nós l_w determinam a taxa de eventos de perda, e não todos os receptores do grupo multicast. Para calcular o evento de perda p, utiliza-se o mesmo procedimento feito pelo TFRC, onde um intervalo de perda é determinado por consecutivas perdas de pacotes, desde do primeiro pacote perdido até o último pacote perdido, seguido de um pacote recebido com sucesso [19, 22];

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, que também é utilizada no GMTP-UCC, porém com $\theta = 0.25$, valor igual ao utilizado no TCP e no DCCP;

5° A taxa de transmissão a ser utilizada pelo nó r_d é a média aritmética de todas as taxas enviadas pelos nós l_w;

6° Repete-se todos os passos a partir do passo 2 a cada intervalo igual ao RTT ou quando um intervalo de perda p é determinado.

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

Um aspecto importante na medição do RTT está relacionado com o início de uma conexão GMTP, pois não se sabe o valor para inicial para RTT até o final do processo de estabelecimento de uma conexão. Nesse caso, deve-se utilizar um valor consideravelmente alto para evitar taxas de transmissões maiores do que a rede tem capacidade de suportar. No GMTP, utiliza-se o valor inicial de RTT igual a 64 *ms*, que é um valor relativamente alto se considerar apenas redes locais em condições normais, que é o caso aqui. Quando um nó c_f envia um pedido de conexão utilizando o pacote do tipo *GMTP-Request*, o mesmo deve realizar a sua primeira medição do valor de RTT, iniciando-se o marcador de tempo para o cálculo do RTT quando enviar o primeiro *GMTP-Request* e parando-o quando receber o pacote do tipo *GMTP-Response*. Em seguida, deve-se acionar o mecanismo de cálculo da taxa de transmissão através da Equação 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. Para evitar esse tipo de ataque, executa-se um procedimento para verificar a autenticidade de um fluxo de dados P. Para isto, os próprios nós $w_m \in W_v$ verificam se o conteúdo de um pacotes de dados $p_x \in P$ foi alterado por algum nó w_m anterior durante o procedimento de repasse. Apenas após comprovar a autenticidade de um pacote p_x, o nó w_m repassa tal pacote de dados p_x para o próximo nó w_{w+1}, transmitindo-os também para seus nós $c_f \in C_i(w_m)$, se houver demanda. Este procedimento evita que todos os nós c_f que receberem o fluxo de dados P tenham que verificar a autenticidade dos pacotes p_x, evitando-se que a rede repasse conteúdo multimídia errados, consequentemente não consumindo recursos computacionais desnecessários.

Na prática, o ideal seria que todos nós w_m verificassem a autenticidade de cada pacote p_x, porém, tal ação pode onerar os recursos computacionais de cada nó w_m e aumentar o tempo de entrega de p_x aos nós $c_f \in C_i(w_m)$. Isto porque os nós w_m também processam cada pacote de dados p_x para decidir sobre seu repasse e para executar os algoritmos de controle de congestionamento, como discutiu-se nas Seções 1.3, 1.4 e 1.5.

Para reduzir a sobrecarga de verificação de autenticidade de um fluxo de dados P em

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

1. apenas os nós w_m, tal que $\varphi(w_m, P) = 1$ devem realizar o procedimento de verificação de autenticidade do fluxo de dados P; e

2. os nós w_m, definidos pela Regra 1, não devem verificar todos os pacotes $p_x \in P$, mas apenas uma quantidade $pc(t)$ de pacotes de dados $p_x \in P$, em um instante t. Nesse caso, define-se $pc(t)$, apresentada na Equação 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 $p_x \in P$;

$$pc(t) = \left\lfloor bs(t, P) \times \left(1 - \frac{1}{\mid W_v^{\triangleleft} \mid -1}\right)\right\rfloor \tag{1.5}$$

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

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 $p_x \in P$ adotado no GMTP, utilizando-se a mesma técnica apresentada na Seção **??**.

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

```
/* s_a executes this algorithm to digital sign the packet
   content using its private key K_{s_a}^- and a pre-defined
   hash function, such as the well-know md5 or sha1
   function. s_a get the value of data field, which is the
   content that application wants to transport and
   generates a signature by encrypt the hash of the data
   using the s_a private key. After, put the generated
   signature in the signature field of the packet p_x. The
   signature field will be used later by a note r_d to
   verify the packet p_x authenticity executing the
   Algorithm 8.                                          */
1 data ← getPacketFieldValue(p_x, 'data');
2 hashValue ← hash(data);
3 signature ← 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 Pe 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' ⊂ P was modified. It marks
   each p_x ∈ 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^◁ = ∼(δ(w_{m+1}, W_v)).                    */
1 verifiedPackets ← array of boolean;
2 foreach p_x ∈ P do
3     signature ← getPacketFieldValue(p_x, 'signature');
4     data ← getPacketFieldValue(p_x, 'data');
5     verifiedPackets[x] ← (hash(data) = decrypt(K+_{s_a}, signature));
6 end
7 return verifiedPackets;
```

1.6.3 Habilitar / desabilitar a validação de pacotes $p_x \in P$

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

que estes executem o procedimento de verificação de autenticidade descrito na Seção 1.6.2. Para isto, o nó s_a ativa a opção assinado (*signed*), disponível no pacote de dados *GMTP-Register-Reply*, sinalizando que todos os pacotes de dados $p_x \in P$ conterá a assinatura da porção de dados sendo transportados naquele pacote de dados, podendo ser verificado pelos nós $w_m \in W_v$, desde que $\varphi(w_m, P) = 1$.

Note que quando um nó $c_f \in C_i(r_d)$ solicitar um fluxo de dados P, em resposta a tal pedido, o nó r_d retornará um pacote do tipo *GMTP-Request-Notify*. No cabeçalho desse pacote, o nó r_d deve também ativar a opção assinado (*signed*) para que o nó c_f seja notificado e entenda que seu nó r_d realizará a verificação de autenticidade do fluxo de dados P da forma descrita anteriormente na Seção 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. Por fornecer essa função de verificação da porção de dados transportado em um pacote, no GMTP não se realiza checagem de erro por soma de verificação (*checksum*), tal como em protocolos como TCP, UDP, DCCP e SCTP.

1.6.4 Obtenção da chave pública $K^+_{s_a}$ de s_a

Um nó r_d obtém a chave pública $K^+_{s_a}$ de s_a através do certificado digital disponível na URI especificada no parâmetro *f* da descrição da mídia, como ilustrou-se no Trecho de Código 5, Linha 7, da Seção 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 7, Seção 1.3.1.

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

1.7 Outras Considerações

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

1.7.1 Canais de comunicação

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

Canal de Controle:

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

A decisão do uso do endereço IP multicast *238.255.255.250* foi baseada na RFCs 2365 [23], que define o escopo administrativo do uso dos endereços multicast entre 239.0.0.0 e 239.255.255.255. O endereço *238.255.255.250* é definido no escopo de uso global e sua alocação deve ser confirmada pela IANA antes do uso massivo do GMTP na Internet.

Canal de transmissão unicast:

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

Do ponto de vista de roteamento, todo nó repassador deve avaliar os datagramas GMTP e realizar as ações apropriadas, definidas nas próximas seções deste capítulo. Por exemplo, no processo de estabelecimento de conexão, a ser detalhado na Seção 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

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

O *socket de repasse multicast* deve ser criado quando um nó repassador começa a receber um determinado fluxo de dados correspondente a um evento de interesse de pelo menos um dos seus clientes. Na prática, quando isto acontece, o repassador deve criar um socket multicast em um endereço IP e número de porta escolhida aleatoriamente na faixa de endereços IP de escopo local 239.192.0.0/14, definida na RFC 2365 [23]. Como se trata de uma faixa de endereçamento IP multicast de domínio local, não se faz necessário registrar o uso desses endereços. Isto significa que para todo fluxo de dado de um evento ao vivo, deve-se alocar um endereço IP e uma porta. No caso do esquema de endereçamento IPv4, será possível definir a transmissão de exatos 17.179.607.040 (dezessete bilhões, cento e setenta e nove milhões, seiscentos e sete mil e quarenta) diferentes fluxos de dados em uma rede local, o que é mais do que suficiente e escalável por vários séculos.

1.7.2 Procedimentos para desconexão de nós c_f, l_w e r_d

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

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

Desconexão de um nó l_w:

Como apresentado na Seção 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. Os candidados a se tornar nó l_w são os nós c_f já recebendo o fluxo de dados P, sendo que o nó l_w em procedimento de desconexão deve esperar que o procedimento de nova eleição seja concluído. Nesse ínterim, o nó l_w em processo de desconexão deve continuar enviando pacotes do tipo *GMTP-Ack* para o nó r_d.

Desconexão de um nó r_d:

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

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

de r_d.

1.7.3 Eleição de nós l_w

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

Sendo assim, à medida que um nó r_d recebe pacotes do tipo *GMTP-Request*, no pacote de resposta *GMTP-Response*, o nó r_d ativa um indicador sinalizando que o referido nó c_f em processo de conexão deverá se comportar como um nó l_w, passando a enviar relatórios da taxa de transmissão calculada, como discutiu-se na Seção 1.5.2.

Uma outra situação que se faz necessária a eleição de nós l_w é no procedimento de desconexão, como explicado na Seção 1.7.2. Para esse caso, quando o nó r_d receber o pacote do tipo *GMTP-Close*, este deve verificar se o referido nó c_f é um nó l_w. Em caso afirmativo, o nó r_d deve transmitir para um dos nós c_f que também recebe o referido fluxo de dados P (se houver), um pacote do tipo *GMTP-Elect-Request* e aguardar por um *GMTP-Elect-Response*. Este procedimento deve ocorrer com garantia de entrega.

1.8 Sumário do Capítulo

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

começam a conhecer todas as possíveis rotas para alcançar os clientes.

Quando um cliente requisita um fluxo de dados a um servidor, seu roteador de borda, participante da rede P2P, assume a responsabilidade de obter o fluxo de dados de interesse. Nesse ínterim, o roteador do cliente transmite um pedido ao servidor e receber como resposta os pacotes de dados referentes à mídia de interesse (em modo unicast/push). À medida que transmissões diretas (servidor $\rightarrow$ cliente) ocorrem, os roteadores presentes nas rotas entre o servidor e os clientes vão sendo "alimentados" pelos pacotes de dados referentes à mídia em questão, ao passo que os roteadores intermediários interceptam os mesmos pacotes de dados quando seus clientes locais também tem interesse pela mesma mídia. Isso permite o conceito de sub-fluxo.

Os roteadores formam parcerias entre si, diretamente por interceptação de pacotes de dados ou com base em instruções obtidas através dos servidores, que executam um algoritmo para determinar a intersecção de rotas usadas por outros roteadores para obter o mesmo fluxo a partir do servidor. Isto ocorre quando o servidor detecta pontos comuns nas rotas e, em vez de aceitar um novo pedido de conexão de um nó solicitante em obter o fluxo de dados, recusa-o e sugere ao nó solicitante uma lista de roteadores candidatos a parceiros, de modo a evitar múltiplas conexões em direção ao servidor. Um roteador pode também solicitar explicitamente uma lista de candidatos a parceiros a fim de obter mais rapidamente os pacotes de dados e, secundariamente, conhecer outros roteadores e contata-los em caso de desconexões dos seus parceiros atuais.

A distribuição do conteúdo em uma rede local sempre ocorre em modo multicast. Para isto, o roteador cria dinamicamente canais multicast e os divulga na rede à medida que recebem pedidos de conexão para obter fluxos de dados que já estão sendo recebidos. Sendo assim, não se transmite mais de um pedido de conexão ao servidor para uma mesma mídia, partindo da mesma rede. O GMTP executa três fases de conexão. A primeira fase de conexão ocorre quando o primeiro nó em uma rede local solicita receber um determinado fluxo de dados; a segunda fase ocorre através do compartilhamento dos pacotes de dados de um fluxo em modo multicast; e a terceira fase permite que os roteadores aumentem suas parcerias.

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

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

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

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

A decisão de utilizar roteadores para formar uma rede colaborativa a fim de disseminar o conteúdo multimídia pode melhorar o desempenho das transmissões de conteúdos multimídia ao vivo, pois reduz-se o impacto negativo causado por fatores que desestabilizam a rede e a qualidade de serviço, tais como a capacidade de processamento e memória dos dispositivos, mobilidade e dinâmica de entrada/saída dos nós clientes (*churn*). Esses fatores são os mais críticos em se utilizar uma rede P2P para distribuição de conteúdos multimídia, principalmente com a popularização dos dispositivos móveis. Por exemplo, usar dispositivos móveis como nós contribuidores de recurso não é uma estratégia apropriada, principalmente em redes híbridas IP/celular (por exemplo, 3G). A partir do uso do GMTP, bastará posicionar um roteador GMTP entre a rede IP e a rede celular para atender à demanda de todos os nós móveis.

Bibliografia

[1] S. Bradner. Keywords for Use in RFCs to Indicate Requirement Levels, 3 1997. `http://www.ietf.org/rfc/rfc2119.txt`. Último acesso: 14 de Março de 2014.

[2] Jonathan L. Gross and Jay Yellen. *Handbook of Graph Theory (Discrete Mathematics and Its Applications)*. CRC Press, 2 2003. ISBN: 978-1584880905.

[3] R Séroul. *Programming for Mathematicians*. Springer-Verlag, 2 2000. ISBN: 978-3540664222.

[4] 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.

[5] K. J. Devlin. *Fundamentals of Contemporary Set Theory*. Springer, 9 1979. ISBN: 978-0387904412.

[6] R. Braden. Requirements for Internet Hosts - Communication Layers, 10 1989. Último acesso: 14 de Março de 2014.

[7] L. Eggert and F. Gont. TCP User Timeout Option, 3 2009. `http://www.ietf.org/rfc/rfc5482.txt`. Último acesso: 14 de Março de 2014.

[8] S. Tanwir and H. Perros. A Survey of VBR Video Traffic Models. *Communications Surveys Tutorials, IEEE*, 15(4):1778–1802, Fourth 2013.

[9] 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: 14 de Março de 2014.

[10] 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: 14 de Março de 2014.

[11] M. Handley and V. Jacobson. SDP: Session Description Protocol, 4 1998. `http://www.ietf.org/rfc/rfc2327.txt`. Último acesso: 14 de Março de 2014.

[12] H. Alvestrand. Tags for the identification of languages, 2 1995. `http://www.ietf.org/rfc/rfc1766.txt`. Último acesso: 14 de Março de 2014.

[13] Nandita Dukkipati. *Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly*. PhD thesis, Stanford University, Stanford, CA, USA, 2008. AAI3292347.

[14] 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.

[15] Nandita Dukkipati, Masayoshi Kobayashi, Rui Zhang-Shen, and Nick McKeown. Processor Sharing Flows in the Internet. In *Proceedings of the 13th international conference on Quality of Service*, IWQoS'05, pages 271–285, Berlin, Heidelberg, 2005. Springer-Verlag.

[16] Nandita Dukkipati. Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly (Apresentação), 4 2007. `http://research.microsoft.com/apps/video/dl.aspx?id=104005`. Último acesso: 14 de Março de 2014.

[17] V. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's Retransmission Timer, 6 2011. `http://www.ietf.org/rfc/rfc6298.txt`. Último acesso: 14 de Março de 2014.

[18] W. Wolff. *Stochastic Modeling and the Theory of Queues*. PrenticeHall, 1989.

[19] 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: 14 de Março de 2014.

[20] 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: 14 de Março de 2014.

[21] 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.

[22] 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.

[23] D. Meyer. Administratively Scoped IP Multicast, 7 1998. `http://www.ietf.org/rfc/rfc2365.txt`. Último acesso: 14 de Março de 2014.