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

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

Leandro Melo de Sales

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

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

Angelo Perkusich e Hyggo Almeida
(Orientadores)

Campina Grande, Paraíba, Brasil

Resumo

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

Abstract

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

Conteúdo

Lista de Símbolos

3WHS - *Three Way Hand Shake*

BSD - *Berkley Software Distribution*

CCID - *Congestion Control IDentifier*

CPM - *Cooperative Peer Assists and Multicast*

DCCP - *Datagram Congestion Control Protocol*

ECN - *Explicit Congestion Notification*

GMTP - *Global Media Transport Protocol*

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

IANA - *Internet Assigned Numbers Authority* IETF - *Internet Engineering Task Force*

PDTP - *Peer Distributed Transfer Protocol*

POSIX - *Portable Operating System Interface*

PPETP - *Peer-to-Peer Epi-Transport Protocol*

PPSP - *P2P Streaming Protocol*

RTO - *Retransmission Timeout*

RTT - *Round Trip Time*

SCTP - *Stream Control Transmission Protocol*

Swift - *The Generic Multiparty Transport Protocol*

TCP - *Transport Control Protocol*

TTL - *Time-To-Live*

UDP - *User Datagram Protocol*

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Global Media Transmission Protocol (GMTP)

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

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

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

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

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

- Na Seção 1.1, apresenta-se uma visão geral do protocolo, como cenário de atuação, arquitetura, canais de comunicação e tipos de nós e pacotes.

- Na Seção 1.2, formalizam-se as definições e restrições do protocolo, que serão utilizadas nas seções subsequentes.

- 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 um novo conceito de sockets P2P. Detalham-se os aspectos inerantes à constituição de uma rede P2P, tais como o registro de participação de um nó e o processo de seleção de nós parceiros.

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

- Na Seção 1.5, apresentam-se detalhes de funcionamento dos algoritmos de controle de congestionamento utilizados no GMTP.

- Na Seção 1.6, discutem-se os aspectos relacionados a validação de autenticidade de um fluxo de dados transmitido através do GMTP.

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

1.1 Visão Geral do GMTP

O GMTP é composto por dois módulos chamados de *GMTP Intra* e *GMTP Inter*, que operam na camada de transporte e de rede, respectivamente. O *GMTP Intra* fornece serviços às aplicações de rede a fim de abstrair a complexidade na execução de tarefas comuns a qualquer sistema, tais como conexão multi-ponto, multiplexação/demultiplexação de segmentos IP e controle de congestionamento, quando em modo multicast. O *GMTP Inter* é responsável por constituir uma rede de sobreposição P2P composta por roteadores, os quais funcionam como pontes de acesso aos servidores de uma rede CDN. Sendo assim, para viabilizar a disseminação de conteúdos multimídia, cada nó roteador no caminho entre o servidor que transmite a mídia e o cliente interessado em obtê-la, pode replicar o conteúdo sendo roteado para seus respectivos clientes locais também interessados em receber o fluxo de dados correspondente e assim sucessivamente. No caso do GMTP, um roteador atende à demanda dos clientes diretamente conectados a ele, ao passo que ajuda os outros a fazer o mesmo, evitando-se duplicações de transmissão.

Na Figura 1.1, observa-se o cenário global de atuação do protocolo GMTP, onde ilustram-se os nós *Clientes GMTP* interessados em obter o conteúdo de um determinado evento ao vivo. Neste caso, observa-se também um *Servidor GMTP*, que está conectado a uma rede CDN e atua como fonte geradora de dados. Na prática, os nós *Clientes GMTP* são aplicações de rede capazes de iniciar uma sessão GMTP, que transmitem, recebem e reproduzem dados multimídia de um determinado evento. Os nós *Clientes GMTP* estão conectados a um nó *Repassador GMTP*, que é executado em um roteador de rede e, junto com outros nós *Repassadores GMTP*, efetivamente constituem a rede de sobreposição P2P. Os *Repassadores GMTP* também podem se conecta a um ou mais *Servidores GMTP*. Os *Servidores GMTP* são as fontes de conteúdos multimídia, obtidos através de três formas: i) diretamente

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

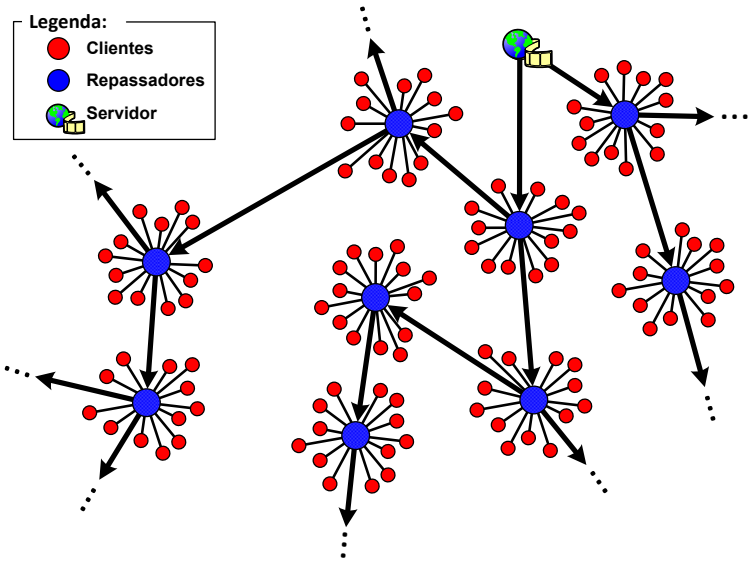

Figura 1.1: Cenário global de atuação do GMTP.

Quando um nó *Cliente GMTP* deseja reproduzir um determinado evento, este envia uma requisição destinada ao nó *Servidor GMTP* que está transmitindo o conteúdo de interesse, como atualmente acontece em qualquer conexão na Internet. A diferença é que um pedido de conexão é interceptado por algum nó *Repassador GMTP* durante o trajeto do pedido de conexão até o nó *Servidor GMTP*, que então determina os melhores parceiros para atendê-la. Em geral, isto ocorre já no roteador de borda do *Cliente GMTP*, que funciona como nó *Repassador GMTP* de origem. Caso o nó *Repassador GMTP* não encontre nenhum nó parceiro capaz repassar a mídia de interesse, este encaminha tal requisição ao nó *Servidor GMTP* que transmite a mídia correspondente, já que o pedido de conexão é intencionamente endereço ao *Servidor GMTP* que transmite o fluxo de dados de interesse. Em todo caso, sempre o nó *Repassador GMTP* de origem assumirá o controle da requisição do *Cliente GMTP*, habilitando-se como candidato a parceiro para outros nós *Repassadores GMTP*, quando motivados por requisições originadas pelos seus *Clientes GMTP*.

O posicionamento dos nós *Repassadores GMTP* e suas habilidades permitem a redução

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

As requisições de conexão podem ser originados não apenas por nós *Clientes GMTP* para seu respectivo nó *Repassador GMTP*, mas também estas podem ocorrer entre nós *Repassadores GMTP* que, motivados pelos interesses dos seus nós *Clientes GMTP*, podem formar parcerias entre si. Isto significa que um nó *Repassador GMTP* pode agir como se fosse um nó *Servidor GMTP*, respondendo às requisições originadas por seus nós *Clientes GMTP* ou de outros nós *Repassadores GMTP*, como se a requisição estivesse alcançado o *Servidor GMTP* que oficialmente transmite o conteúdo, o que ocorre de forma transparente para a aplicação.

Na Figura 1.2, observam-se detalhes do cenário supracitado, introduzindo-se o conceito de um grupo especial de nós chamados de nós *Relatores GMTP*. Estes nós são responsáveis por enviar relatórios periódicos sobre o estado da transmissão ao seu nó *Repassador GMTP*, que os utiliza para regular a taxa de transmissão de um ou mais fluxos de dados, impedindo-se que a rede entre em colapso de congestionamento.

1.1.1 Terminologias e Convenções

Nesta seção, apresentam-se algumas definições, terminologias e convenções utilizadas no restante deste documento, de acordo com a Figura 1.3.

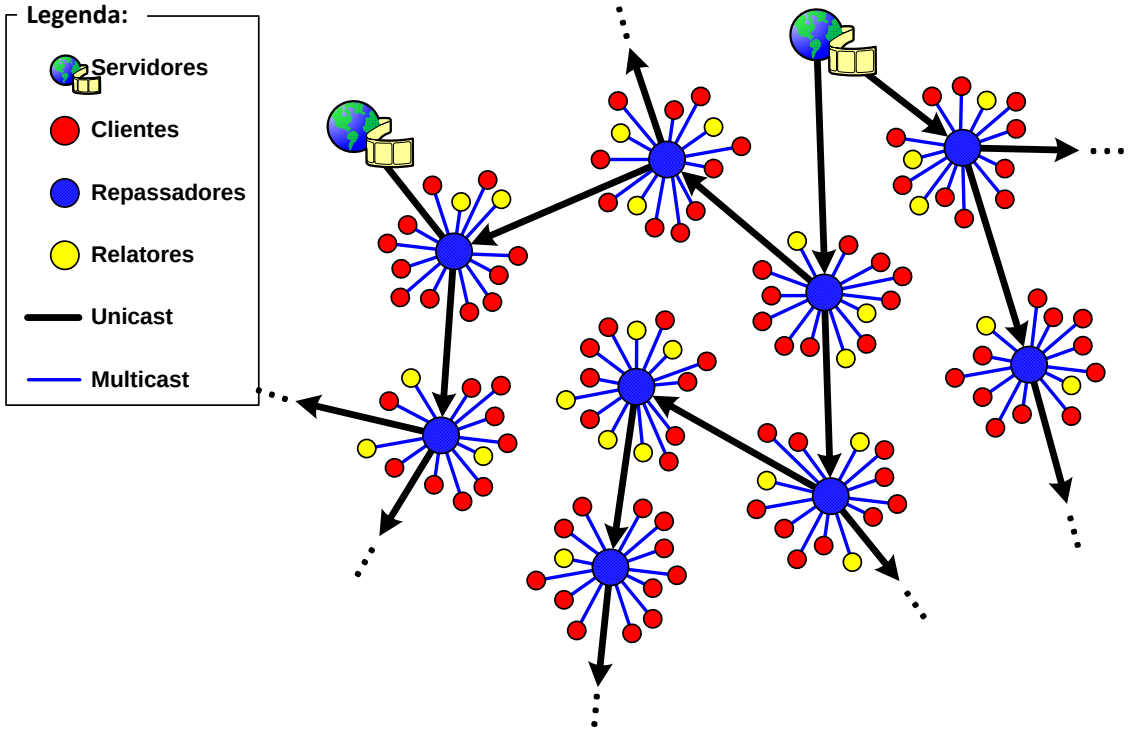

Figura 1.2: Rede de sobreposição construída dinamicamente pelo GMTP com a presença de nós repassadores e relatores.

Tipos de Nós:

- **Nó GMTP ou Processador GMTP:** qualquer processador de rede que implementa o protocolo GMTP. É um sistema computacional que implementa parte ou todo do protocolo GMTP, sendo capaz de interpretar os cabeçalhos dos pacotes definidos pelo GMTP e realizar ações pré-definidas. Não há restrições de qual tipo de processador de rede pode implementar qual(is) parte(s) do GMTP.

- **Cliente GMTP:** é um *nó GMTP* capaz de reproduzir e gerar conteúdos multimídia ao vivo. Em geral, um *Cliente GMTP* é um sistema final que executa um processo a nível de sistema operacional, representando uma aplicação manipulada pelo usuário final. A maioria dos *Clientes GMTP* funciona apenas de forma passiva, recebendo o fluxo de dados de um conteúdo multimídia e entregando para um processo em execução, sendo alguns contribuidores na execução do algoritmo de controle de congestionamento.

- **Servidor GMTP:** é um *nó GMTP* capaz de capturar um evento ao vivo e gerar conteúdos digitais de áudio e vídeo ou ainda, receber de um *Cliente GMTP* tais conteúdos.

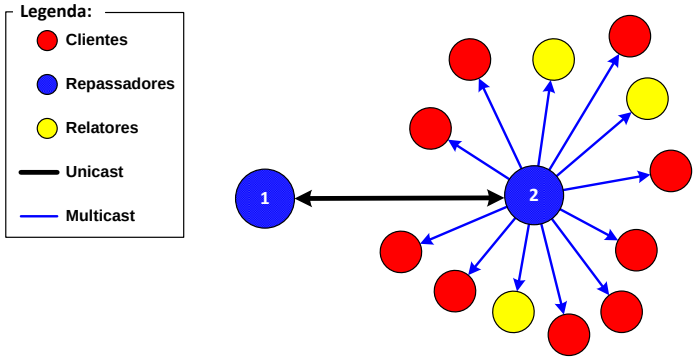

Figura 1.3: Tipos de Nós e modos de conexões do GMTP.

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

- **Repassador GMTP:** é um *Nó GMTP* com habilidades de repassar os fluxos de dados originados de um ou mais *Servidores GMTP* ou de um outro nó *Repassador GMTP*. Este nó sempre é um roteador de rede.

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

Modos de Transmissão:

- ***Unicast*:** ocorre em toda comunicação entre dois nós *Repassadores GMTP*, com a interpretação do conceito definido por *unicast* em sua forma tradicional no contexto de redes de computadores.

- ***Multi-unicast*:** é a combinação de dois ou mais fluxos de dados *unicast*.

- ***Multicast*:** ocorre em toda comunicação entre um nó *Repassador GMTP* e seus respectivos *Clientes GMTP*, com a interpretação do conceito definido por *multicast* em sua forma tradicional no contexto de redes de computadores.

O modo *multicast* sempre é utilizado para a transmissão dos datagramas correspondentes ao fluxo de dados multimídia. É mandatório que o modo *multicast* seja utilizado para transmissões entre um nó *Repassador GMTP* e seus *Clientes GMTP* diretos. O modo *unicast* é utilizado para que *Clientes GMTP* estabeleçam uma conexão com um *Servidor GMTP* ou um *Repassador GMTP* e passe a distribuir o conteúdo de dados multimídia em sua rede local.

Deste ponto em diante, os termos *Nó GMTP*, *Cliente GMTP*, *Servidor GMTP*, *Repassador GMTP* e *Relator GMTP* serão utilizados em sua forma simplificada, ou seja, *nó*, *cliente*, *servidor*, *repassador* e *relator*, respectivamente. Além disso, estes termos não serão mais formatados em itálico, bem como os termos *socket*, *unicast*, *multi-unicast* e *multicast*. Ademais, quando o termo *transmissão* ou *transmissão de um evento* for mencionado, denotar-se-á a transmissão de um fluxo de datagramas IP correspondente a um evento ao vivo, utilizando-se o protocolo GMTP.

Embora alguns autores considerem os termos "repasse" e "roteamento" como conceitos distintos, neste trabalho ambos os termos são considerados sinônimos e devem ser interpretados como a capacidade que um nó GMTP tem de receber dados em uma interface de rede de entrada e encaminhar estes dados através de uma interface de rede de saída, permitindo-se que uma mesma interface de rede seja utilizada como entrada e saída ao mesmo tempo.

As palavras "deve", "não deve", "requerido", "pode", "não pode", "recomendado" e "opcional", incluindo suas variações morfológicas, devem ser interpretadas como descrito na RFC 2119 [2].

1.1.2 Arquitetura

Na Figura 1.4, ilustra-se a arquitetura geral do GMTP.

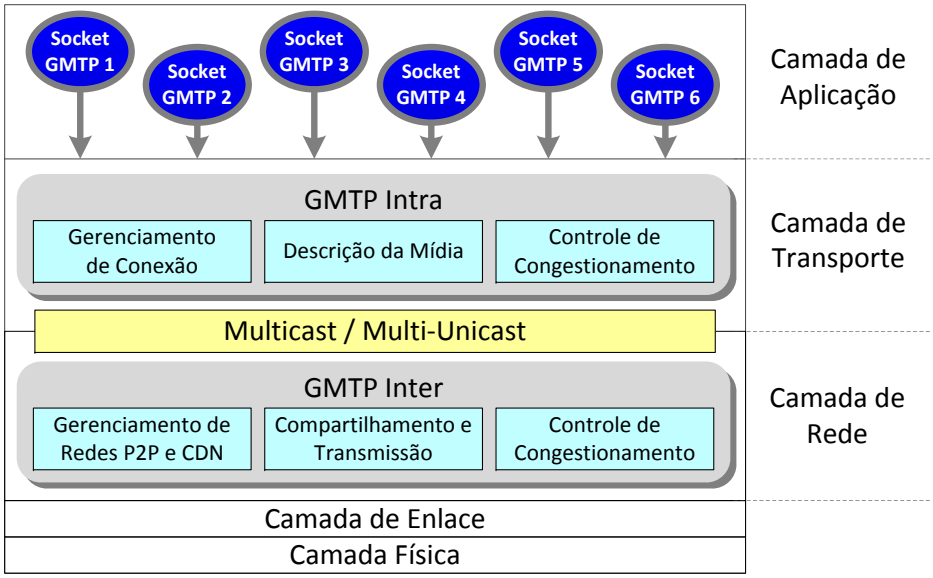

Figura 1.4: Arquitetura do Protocolo GMTP.

1.1.3 GMTP Intra e GMTP Inter

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

- **GMTP Intra:** parte do protocolo GMTP executada na camada de transporte da pilha de protocolos TCP/IP, corresponde ao módulo interno de uma rede, composta por nós clientes e relatores, disponibilizado no sistema operacional e utilizado pela aplicação através de uma API de socket GMTP. Um socket GMTP é a representação de uma instância do protocolo GMTP em execução, sendo responsável por gerenciar todas as atividades de comunicação da aplicação correspondente ao meio externo (outros processos GMTP). No contexto de uma conexão, o GMTP Intra mantém diversas variáveis de estado que representam uma instância e executa algoritmos para gerenciamento de conexão (estabelecimento e desconexão), controle de congestionamento, multiplexação e demultiplexação dos datagramas, eleição de nós parceiros, determinação do formato e preenchimento dos parâmetros que definem uma determinada mídia, permitindo-se que a aplicação defina os valores de tais parâmetros ou obtenha acesso aos valores dos mesmos. O GMTP não faz verificação de conteúdo por mecanismo de soma de verificação, diferentemente do TCP, UDP, DCCP, SCTP e outros.

- **GMTP Inter:** parte do protocolo GMTP executada na camada de rede da pilha de protocolos TCP/IP e corresponde ao módulo externo de uma rede, composta por vários nós repassadores que cooperam entre si. É executado por um roteador de rede e aceita conexões oriundas de um nó cliente ou de um nó repassador. No contexto de uma conexão, o GMTP Inter mantém variáveis de estado relacionadas às funções de sua responsabilidade, tais como estabelecimento de conexão com nós servidores ou repassadores, seleção de nós repassadores parceiros, eleição de nós relatores, controle de congestionamento assistido pela rede e controle de compartilhamento de fluxos multimídia.

  No GMTP Inter, permite-se a configuração de parâmetros iniciais de configuração da rede de favores e da integração com servidores de uma ou mais CDN, como ilustrado na Figura 1.5. Nesse caso, o usuário administrador de um nó repassador pode definir os seguintes parâmetros:

- registro de participação em uma ou mais redes redes CDN;
- largura de banda (*download* e *upload*) que deseja compartilhar;
- o período (faixa de dias e horários) que o roteador funcionará como nó repassador;
- quantidade máxima de parcerias que podem ser realizadas;
- quantidade máxima de fluxos de dados que podem ser compartilhados;
- parâmetros avançados relacionados aos algoritmos de controle de congestionamento;
- download automático ou não do certificado digital de um nó servidor; e
- realização de cache ou não dos certificados digitais obtidos.

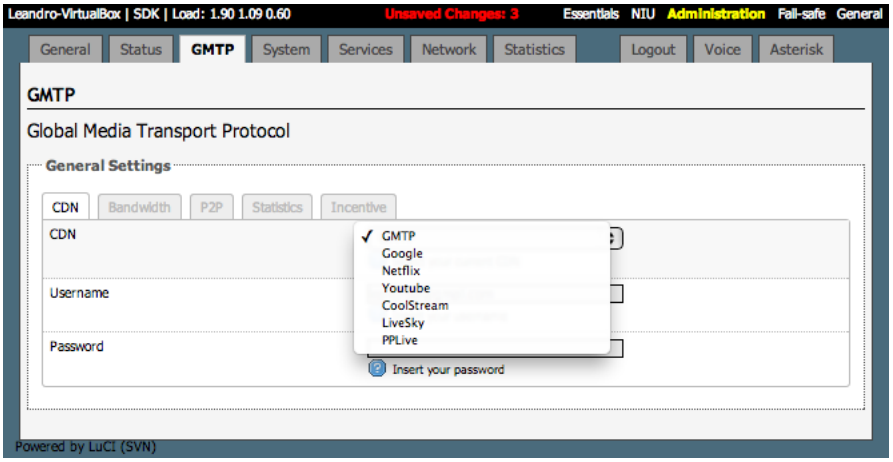

Figura 1.5: Tela da ferramenta de administração do OpenWRT [1] com suporte ao GMTP. Nessa tela, permitir que o administrador do roteador configure registros de participação em uma ou mais redes CDN.

1.1.4 Principais funções do GMTP

- Registro de participação de um nó repassador em um nó servidor. Isto permite o suporte à pré-seleção de nós parceiros filtrados por métricas que influenciam na qualidade de serviço oferecido aos sistemas finais, consequentemente reduzindo-se o número de fluxos de dados na rede.

- Acesso a uma transmissão através de um processo de conexão em três-vias (*3WHS*), com a requisição de conexão transmitida ao servidor e podendo ser interceptada por um nó repassador em seu trajeto ao servidor, com suporte automático de detecção e uso dos modos de transmissão suportados pelo nó repassador.

- Descoberta de nós parceiros entre redes distintas e negociação de parcerias, com suporte a formação de parcerias baseadas em métricas de rede, tais como atraso e largura de banda fim-a-fim.

- Envio e recebimento de fluxos de dados compartilhados entre nós da mesma rede através de multicast e uso de fluxos unicast entre redes distintas, porém sem a relação de uma conexão por cliente e assim evitando o fenômeno da tragédia dos bens comuns, discutido na Seção **??**.

- Suporte a algoritmos de controle de congestionamento assistidos pela rede e de fluxos multicast.

- Múltiplas taxas de transmissão definida pela segmentação de rotas entre um nó servidor e os nós repassadores, de acordo com a capacidade de transmissão do um canal. Este assunto é discutido em mais detalhes na Seção 1.5.

- Eleição de nós relatores com suporte a tolerância a desconexões de nós, com notificação e reeleição de novos nós.

- Permite que os nós clientes verifiquem a autenticidade das partes de uma mídia, por meio do uso de certificado digital determinado pelo nó servidor para impedir ataques de poluição de conteúdo.

1.1.5 Canais de Comunicação

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

Canal de Controle

Quando um nó repassador iniciar uma instância do protocolo GMTP, este deve criar um socket multicast no endereço IP 238.255.255.250 e na porta 1900, em toda interface de rede local, ou seja, nas interfaces por onde se fornece acesso aos nós clientes. Através desse socket, um nó GMTP é capaz de enviar e receber pacotes de controle utilizados para negociar as funções de transmissão de um determinado fluxo de dados de mídia ao vivo. Por exemplo, utiliza-se este canal para permitir que um nó cliente envie pedidos de conexão e descubrir 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 [3], que define o escopo administrativo do uso dos endereços multicast entre 239.0.0.0 e 239.255.255.255. O endereço *238.255.255.250* é definido no escopo de uso global e sua alocação deve ser confirmada pela *Internet Assigned Numbers Authority* – IANA[1] 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.

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

Canal de Repasse Multicast

Além do canal de controle, define-se no protocolo GMTP um canal de repasse utilizado por um nó repassador para encaminhar datagramas vindos de um servidor ou de outro repassador para a rede local. Esse canal de repasse, na prática, é um socket multicast criado pelo nó repassador para transmitir os datagramas para todos os seus clientes com interesse em reproduzir o mesmo fluxo de dados de um evento ao vivo.

O *socket de repasse multicast* deve ser criado quando um nó repassador passa a obter um determinado fluxo de dados correspondente a um determinado evento de interesse de pelo menos um dos seus clientes. Na prática, quando isto acontece, o repassador deve criar um socket multicast em um endereço IP e número de porta escolhida aleatoriamente para repassar os dados vindos do servidor ou de outro repassador para dentro de sua rede. A faixa de endereços IP multicast que o nó repassador deve utilizar para criar seu socket de repasse para um determinado fluxo de dados é a de escopo local 239.192.0.0/14, definida na RFC 2365 [3]. Como é uma faixa de endereços IP multicast de domínio local, não se faz necessário registrar o uso desses endereços. Isto significa que para todo fluxo de dados de um evento ao vivo, deve-se alocar um endereço IP e uma porta. No caso do esquema de endereçamento IPv4, com isto, será possível definir a transmissão de exatamente de 17.179.607.040 (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.

1.1.6 Tipos de Pacotes

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

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

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:* Cliente envia requisição para obter um fluxo de dados multimídia dado um nome do fluxo de interesse;
1. *GMTP-RequestNotify:* Repassador notifica um cliente que um fluxo de dados está prestes a ser transmitido ou já está sendo transmitido em um determinado canal de repasse multicast. Campo de dados contém a descrição da mídia a ser reproduzida;
2. *GMTP-Response:* Repassador confirma o estabelecimento de uma parceria com outro nó repassador, dado um determinado fluxo de dados;
3. *GMTP-Register:* Repassador registra participação no servidor para funcionar como distribuidor de um ou mais fluxos de dados;
4. *GMTP-Register-Reply:* Servidor responde ao repassador sobre seu pedido de registro de participação;
5. *GMTP-RelayQuery:* Repassador pode solicitar ao servidor uma lista de possíveis nós repassadores parceiros;
6. *GMTP-Data:* Qualquer nó utiliza para transmitir dados da aplicação;
7. *GMTP-Ack:* Qualquer nó utiliza para confirmar a recepção de um determinado pacote, seja pacotes previamente contendo dados ou não;
8. *GMTP-DataAck:* Combinação dos pacotes GMTP-Data e GMTP-Ack (*PiggyBack*);
9. *GMTP-MediaDesc:* Servidor transmite esse pacote para descrever informações sobre a mídia sendo transmitida em uma determinada transmissão;
10. *GMTP-DataPull-Request:* Repassador envia um pedido para obter o mapa de buffer atual de um outro repassador parceiros;
11. *GMTP-DataPull-Response:* Resposta ao pedido para obtenção de um mapa de buffer;

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

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

14. *GMTP-Close:* Servidor, repassador ou cliente solicita o término de uma conexão;

15. *GMTP-Reset:* Determina, incondicionalmente, a finalização de uma conexão;

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

No Apêndice **??**, apresenta-se detalhes acerca do uso dos tipos de pacotes do GMTP, sendo seu teor bastante técnico e portanto dedicado aos leitores interessados em sua implementação.

1.2 Definições, Relações e Restrições do GMTP

Nesta seção, descrevem-se as definições, relações e restrições do protocolo GMTP. Para isto, faz-se uso de fundamentos de algebra booleana, lógica proposicional, teoria de conjuntos e teoria dos grafos [4–7].

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 $R \rightarrow B$ que determina a sobreposição dos nós repassadores $r_d \in R$ sob os roteadores em B (*rede de sobreposição*).

3. Seja o conjunto finito dos nós servidores, definido por $S = \{s_1, s_2, s_3, \ldots, s_a\}$, 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 $(\mathbb{P}, \prec) = \{p_1, p_2, p_3, \ldots, p_h\}$, onde $h \in \mathbb{N}$. Note que o símbolo $\prec$ é utilizado para representar precedência entre dois elementos diferentes.

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

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

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

      i. Seja P, o conjunto *parcialmente ordenado (poset)* dos pacotes de dados p_x transmitidos por um nó s_a ou repassados por um nó r_d, também chamado de fluxo de pacotes de dados ou apenas fluxo de dados, definido por $(P, \prec) = \{p_1, p_2, p_3, \ldots, p_x\}$, 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, definida por $C_i : r_d \rightarrow 2^C$, $\forall r_d, r_{d+1} \in R$, $C_i(r_d) \cap C_i(r_{d+1}) = \{\emptyset\}$;

      iii. Seja L, o conjunto finito dos nós relatores, definido por $L = \{l_1, l_2, \ldots, l_w\}$. 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, que determina que dois nós c_f não podem estar relacionados a mais de um nó r_d, tem-se portanto que $L_\theta \subset L$ e $L_\theta \cup C_i(r_d) = C_i(r_d)$.

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

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

ii. Um caminho W_v é dito *caminho semi-completo*, representado por W_v°, 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 que transmite os pacotes de dados $p_x \in P$, definido por $T = \{t_u \mid \varphi(t_u,\, P)\ = 1\}$, sendo $u \in \mathbb{N}$ e φ uma função booleana que determina se um nó $t_u \in T$ transmite os pacotes $p_x \in P$ para $c_f \in C_i(t_u)$, ou seja:

A. $\varphi : (t_u,\, P) \rightarrow \{0, 1\}, \forall (t_u,\, P) \in \{T \times \{P\}\}$, onde 0 e 1 denotam, respectivamente, *falso* e *verdadeiro*.

(d) Seja $\sim$, uma função reversa de um conjunto *toset*, tal que $\sim : (W_v, \prec) \rightarrow (W_v, \succ)$. Isto é, para um conjunto $(W_v, \prec) = \{w_m \mid s_a, r_1, r_2, \ldots, r_d\}$, 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 $(W_v, \prec) = \{t_{u+1}, t_{u+1}, t_u, t_{u-1}, t_{u-2}, \ldots, t_2, t_1\}$, $\delta(t_u, W_v) = W_v^{\triangleleft} = \{t_u, t_{u-1}, t_{u-2}, \ldots, t_2, t_1\}$. Neste caso, como δ faz um corte no conjunto W_v, pode-se obter *caminho semi-completo* ou *completo*, representados por $W_v^{\triangleleft\circ}$ e $W_v^{\triangleleft\bullet}$, respectivamente;

(f) Seja 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, o custo é calculado pelo RTT entre dois nós distintos, mas podendo-se utilizar outras métricas, como número total de saltos no caminho W_v;

(g) *Conjectura 1*: $\forall r_d \in R$ e $\forall c_f \in C$, r_d é mais estável que qualquer c_f com relação a sua disponibilidade e participação em uma rede de favores η. Em uma rede comutada por pacotes IP, um nó $b_e \in B$, portanto para o GMTP um nó r_d, fica menos indisponível se comparado aos seus nós $C_i(r_d)$. Por exemplo, nas transmissões de dados na Internet, a participação de um roteador no processo de

transmissão de um fluxo de dados P é fundamental, mesmo que seja apenas para rotear os respectivos pacotes. Apesar de óbvia, tal observação é importante porque para qualquer nó c_f receber os pacotes de dados $p_x \in P$, primeiramente os pacotes de dados p_x devem, obrigatoriamente, passar pelo roteador de c_f, ou seja, o seu roteador padrão. Sendo assim, quando um nó r_d se desconecta, todos seus nós $C_i(r_d)$ tornam-se capazes de receber P, mas a recíproca não é verdadeira – se um nó c_f se tornar indisponível, não necessariamente r_d também se torna indisponível. Com base na aceitação dessa conjectura, especificamente para a rede η, pretende-se permitir que outros nós c_f possam continuar recebendo P, mesmo ocorrendo a desconexão de um nó c_f que esteja recebendo P durante a recepção de um fluxo de dados P. No GMTP, adota-se tal estratégia quando um nó r_d passa a manter estado sobre tal transmissão e não mais por qualquer nó c_f, antes prática comumente adotada em soluções tradicionais de distribuição de conteúdos multimídia baseado em uma arquitetura P2P ou em qualquer protocolo de transporte e rede disponível no estado da arte;

(h) *Conjectura 2*: as tabelas de roteamento dos nós $w_m \in W_v$ não mudam frequentemente e são independentes umas das outras. Em redes comutadas por pacotes IP, as rotas entre quaisquer nós c_{f_1} e $c_{f_2} \in C$ não se alteram com um nível de frequência que desestabilize a comunicação entre estes. Mesmo se estas mudanças ocorrerem em uma rota de um caminho W_v, o impacto causado é temporário e insignificante para a transmissão de um evento $\mathcal{E}$quando se utiliza um conjunto de algoritmos que tratem essas mudanças. Com base na aceitação dessa conjectura, é possível antecipar a formação de parcerias entre os nós em Z antes da efetiva transmissão de um fluxo de dados P. Essa estratégia é adotada no GMTP.

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

Nas próximas seções, detalham-se os aspectos teóricos e computacionais empregados do GMTP a fim de construir η, em três partes distintas de acordo com os blocos funcionais

ilustrados na Figura 1.6:

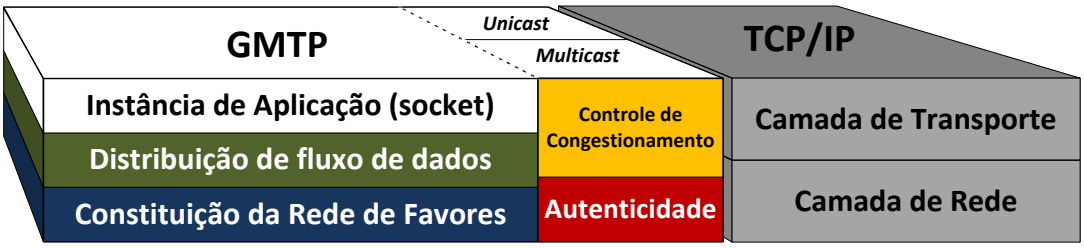

Figura 1.6: Blocos funcionais do GMTP e as relações com a pilha de protocolos TCP/IP.

1. *Constituição da rede de favores η:* descobrir, definir, efetivar e desfazer parcerias entre os nós $r_d \in R$ de acordo com o evento $\mathcal{E}$ a ser transmitido (Seção 1.3);

2. *Distribuição do fluxo de dados P em η, através das Instâncias de Aplicações (sockets):* conectar os nós $c_f \in C$, $r_d \in R$ e $s_a \in S$, bem como transmitir os pacotes de dados $p_x \in P$ através da rede de sobreposição constituída na Fase 1 (Seção 1.4);

3. *Controle de congestionamento em η:* controlar a taxa de transmissão dos fluxos de dados P transmitidos em η na Fase 1 (Seção 1.5); e

4. *Segurança em η:* verificar a autenticidade do conteúdo de P antes de entregar aos nós $c_f \in P$ (Seção 1.6).

1.3 Constituição da Rede de Favores 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

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 ou mais fluxos de dados P. O registro de participação pode ocorrer antes do nó s_a iniciar a transmissão de um fluxo de dados P, ou durante sua transmissão. 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 evento $\mathcal{E}$ qualquer, será possível mapeá-lo antecipadamente e selecionar um subconjunto de possíveis nós parcerios r_q para executar a distribuição de um fluxo de dados P. Neste caso, pode-se utilizar r_d para repassar pacotes de dados p_x mesmo quando $C_i(r_d) = \{\emptyset\}$, ou seja, mesmo se o nó repassador não tiver nós clientes para repassar o fluxo de dados P. Assim, os nós r_d passam a funcionar como se fossem servidores de uma rede CDN, porém dinâmicos, que podem ser acionados quando fosse conveniente.

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

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

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

Note que, no GMTP, toda transferência de pacotes de controle entre nós r_d ocorre com garantia de entrega, representando-se tais ações pelas funções com nomes contendo o sufixo *Rdt* (*Reliable data transfer*). Uma outra decisão importante tomada no GMTP é que um nó r_d deve periodicamente sinalizar sua participação na rede de favores η através de uma função tradicionalmente conhecida por *keep-alive*, comumente utilizado em outros protocolos de rede consolidados, como o TCP. Nesse aspecto, o GMTP segue a RFC 1122, *Requirements for Internet Hosts - Communication Layers* [8].

Além disso, um nó r_d pode sinalizar explicitamente sua desconexão a $s_a$quando não desejar mais participar da rede de favores η ou receber um fluxo de dados P. Para isto, deve-se enviar um pacote do tipo *GMTP-Close*. Em qualquer um dos casos de desconexão, por expiração do tempo (devido ao procedimento de *keep-alive*) ou explicitamente através do envio do pacote do tipo *GMTP-Close*, o nó s_a deve desconsiderar r_d no processo de formação de parcerias e enviar para este um pacote do tipo *GMTP-Reset*.

Algoritmo 1: registerRelay(s_a: PeerServer, p_x = *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); /* See Section 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));
            return 0;
        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.      */
isWaitingRegisterReply(P, false);
if p_x = OK then                          /* s_a confirmed registration */
    s_a ← getPacketFieldValue(p_x, 'server');    /* Gets s_a in p_x */
    P ← getPacketFieldValue(p_x, 'flow');        /* Gets P in p_x */
    if P ≠ NULL then                  /* Reply to C_i(r_d), waiting for P */
        if s_a enabled security layer then                /* Section 1.6.4 */
            getAndStoreServerPublicKey(s_a);
        end
        channel ← createMulticastChannel(s_a, P);
        updateFlowReceptionTable(channel);  /* Section 1.3.2 */
        /* Let c_f ∈ C_i(r_d) know the multicast channel to
           receive P (Section 1.4.2).                        */
        respondToClients(GMTPRequestReply(channel));
        /* Start to relay P to clients (Section 1.4.6).     */
        startRelay(channel);
    end
    /* It was just a reply of a registration of
       participation.  Update flow reception table.        */
    updateFlowReceptionTable(s_a);          /* Section 1.3.2 */
else
    /* s_a refused to accept the registration of
       participation.  This r_d must notify the clients
       waiting for receiving P.                            */
    errorCode ← getPacketFieldValue(p_x, 'error');
    respondToClients(GMTPRequestReply(errorCode, P));
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 1.5. Nesses casos, as redes de distribuição podem fazer uso do roteador desse usuário em momentos ociosos de recepção e transmissão de dados através da Internet. Como consequência, relações comerciais podem ser construídas entre o usuário e os provedores de rede, mas essa discussão está fora do escopo deste trabalho.

Manutenção do registro de participação:

Após o registro de participação, o nó r_d deve enviar periodicamente sinalizações de controle (*polling*) sobre sua participação na rede de favores η. Este procedimento deve ser feito usando o pacote do tipo *GMTP-Ack* em um tempo $t = max(300, t_{user})$, onde t_{user} é definido em segundos e corresponde a um tempo definido pelo administrador do nó r_d, caso deseje um tempo menor que 300 *s* para mantém o registro de participação ativo. Quando s_a receber um pacote do tipo *GMTP-Ack* do nó r_d, este deve enviar um pacote do mesmo tipo. Caso r_d não receba *GMTP-Ack* no período de $4 \times RTT$, deve-se repetir tal procedimento por no máximo 3 vezes, quando r_d deve considerar a conexão finalizada por tempo de expiração (*timeout*) e enviar um pacote do tipo *GMTP-Reset*. Na RFC 5482 [9], discute-se sobre outros aspectos de expiração no tempo que podem ser adaptadas 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 1.7. O nó r_d utiliza tal tabela para registrar todos

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

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

Figura 1.7: Exemplo de uma tabela de recepção de fluxo mantida por um nó r_d.

- **Nome do Fluxo de Dados** P**:** é uma sequência de 128 *bits* que determina o nome de um fluxo de dados, como descrito na Seção 1.4.1;
- **Servidores** s_a**:** o endereço IP do nó s_a que gera o fluxo de dados P;
- **Repassadores** r_q**:** o endereço IP do nó r_q, parceiro de r_d, que está transmitindo o fluxo de dados P para r_d. Se nulo, significa que o fluxo de dados P está sendo recebido diretamente do nó s_a;
- **Porta de Recepção de** P**:** o número da porta do nó remoto que está transmitindo o fluxo de dados P para r_d. Nesse caso, o nó remoto pode ser o nó s_a, em caso de conexão direta com o servidor, ou um nó r_q, parceiro de r_d;
- **Endereço do Canal Multicast:** o endereço IP multicast utilizado pelo nó r_d para repassar o fluxo de dados P para os nós clientes $c_f \in C_i(r_d)$; e
- **Porta do Canal Multicast:** o número da porta multicast utilizada pelo nó r_d para repassar o fluxo de dados P para os nós clientes $c_f \in C_i(r_d)$.

Um nó r_d consulta a tabela de recepção de fluxos de dados quando recebe um pedido de conexão (*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 10 do Algoritmo 2, Seção 1.3.1. Mais adiante, na Seção 1.4.2, discute-se em mais detalhes as ações de consulta e atualização da tabela de recepção de fluxos de dados.

1.3.3 Formação de parcerias

No GMTP, as parcerias ocorrem entre os nós $r_d \in R$ e não entre os nós $c_f \in C$, como em solução existente baseada no modelo de serviço P2P. A formação de parcerias consiste em determinar intersecções de caminhos W_v, considerando o nó *pivot* s_a e diversos nós r_d interessados em obter P, a pedido de seus nós $c_f \in C_i(r_d)$. Este processo pode ocorrer antes e durante a transmissão de um fluxo de dados P gerado por um no s_a, de forma transparente para a aplicação em execução em c_f, durante seu pedido de conexão transmitido em direção ao nó s_a. Como consequência, constitui-se um ou mais caminhos $W_v \in W$, os quais interconectam um nó s_a e os nós $c_f \in C_i(w_m)$, de modo que $\exists W_v \mid w_m \in W_v$. Como regra geral para formação de parcerias, definem-se três critérios:

1. o melhor nó s_a para servir um nó r_d é aquele que está especificado em seu pedido de registro de participação, respeitando-se as funções de balanceamento de carga definida pela CDN;

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 1.8. 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

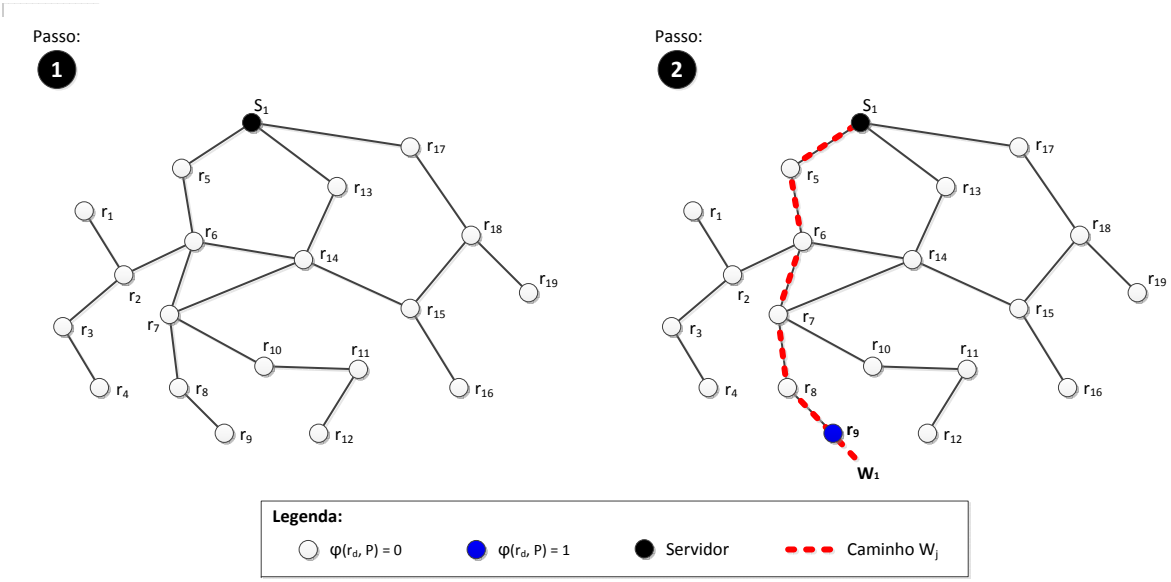

Figura 1.8: Cenário e passos para seleção de nós (exemplo 1).

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 1.9, considera-se a formação de parceria por intersecção do fluxo de dados P, a partir do Passo 2 da Figura 1.8. 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 1.9 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 1.9), o nó s_1 descobrirá

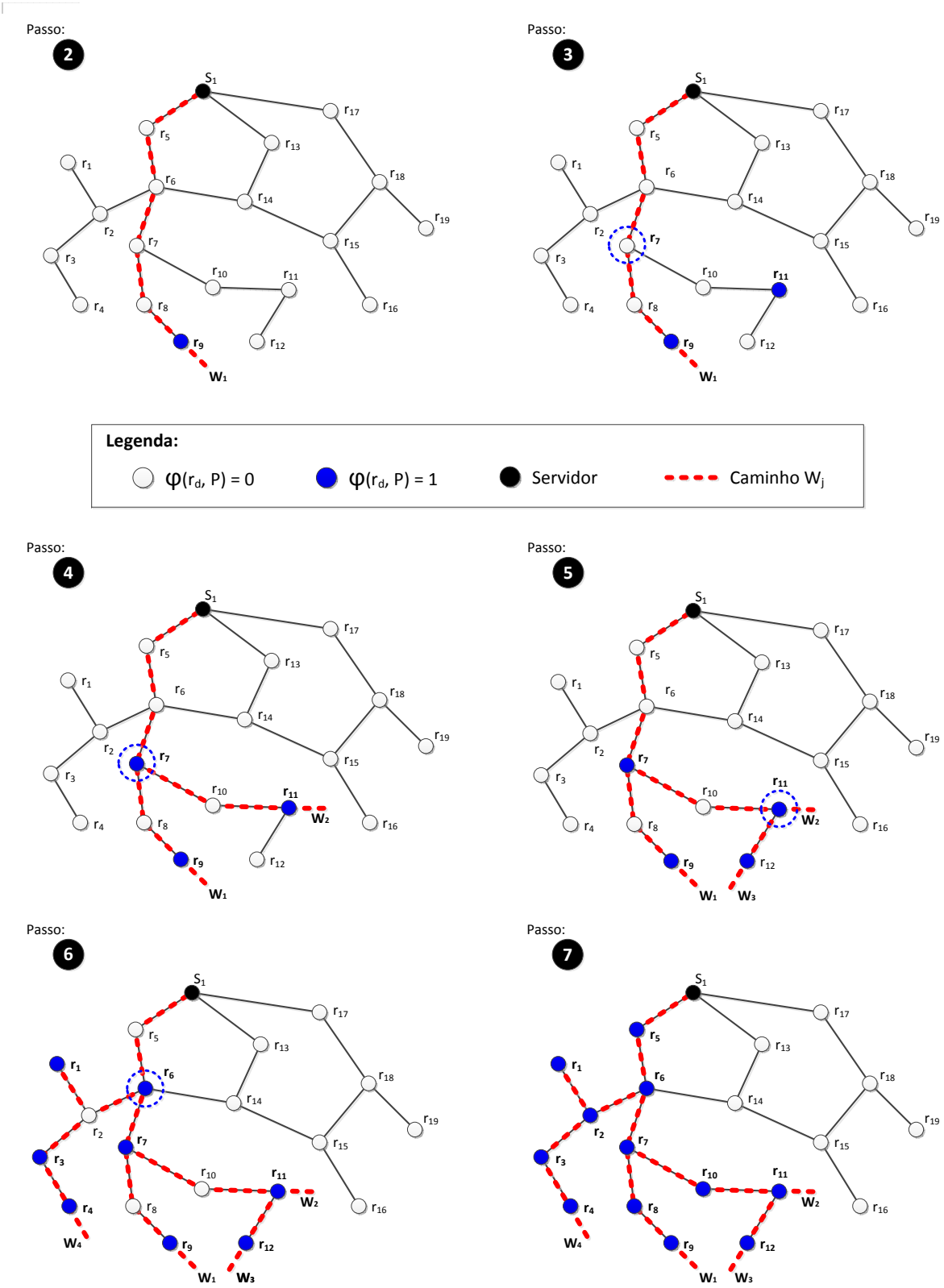

Figura 1.9: Cenário para seleção de nós por interseção de caminhos W_v.

o caminho $W_2 = \{r_5, r_6, r_7, r_{10}, r_{11}\}$. 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} (Passo 4). 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 1.9, 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). 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 $r_d \in W_4$ (Passo 6). 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 o nó r_{11} tem autonomia para responder ao nó r_{12}, ambos como se fossem 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 para 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. Na Seção 1.5, discute-se em mais detalhes este aspecto.

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

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

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

```
/* s_a executes this algorithm to finds the first node w_m
   common to a known path W_v and the path W_{r_d}. W_v is
   already used for transporting P to node in 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 foi o caso dos nós r_6 e r_7 do exemplo anterior. Além disso, como a transmissão de um fluxo de dados P entre um nó r_d e seus nós $c_f \in C_i(r_d)$ ocorrerá sempre em modo multicast, faz-se necessária apenas uma entrada na tabela de recepção de fluxos de dados sobre P. Com essa estratégia, deve-se esperar uma quantidade significativa de nós c_f capazes de reproduzir um fluxo de dados P, sem sobrecarregar a rede com demasiadas transmissões do mesmo fluxo de dados P, além da reduzir o tempo de inicialização para reproduzir o fluxo de dados P (*startup delay*). Ademais, apresentou-se procedimentos que não são adotados em nenhum protocolo de rede pesquisa no estado da arte. Trata-se da primeira solução em que o nó servidor dar suporte aos roteadores no processo de formação de parcerias, delegando-se para estes a responsabilidade de distribuir um determinado fluxo de dados P, tudo de forma transparente para as aplicações. Como resultado, pode-se afirmar que os roteadores passam a funcionar como se fossem servidores de uma CDN, só que participando dinamicamente sempre que conveniente.

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* como 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*, utiliza-se 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, seguindo o princípio das redes centradas no conteúdo. Na prática, cada fluxo de dados P corresponde a uma mídia gerada a partir de um evento real $\mathcal{E}$, por exemplo, a transmissão de um jogo de futebol, corrida de fórmula 1 etc.

No GMTP, define-se um nome de um fluxo de dados P por um código de *hash* no formato UUID (*Universally Unique IDentifier*) de 128 bits [10]. 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* e *GMTP-Data*.

Na prática, para gerar o nome para um fluxo de dados P, utiliza-se uma função de *hash* do tipo SHA1. Sendo assim, para determinar o nome de um fluxo de dados P, disponibilizado por um servidor s_a, utiliza-se MD5(IP_{s_a} + : + PORTA_{s_a}). Por exemplo, suponha que um servidor esteja disponibilizando um fluxo de dados P qualquer através do endereço 200.17.113.98, na porta 21200. O nome do fluxo de dados P será definido por *MD5("200.17.113.98:21200") = f8ea01fd-4d71-5d95-89ec-35646e11d7fe*. Opcionalmente, o nó s_a pode divulgar o nome do fluxo de dados através do serviço DNS. Já com relação ao título do conteúdo e sua descrição, tais informações podem ser divulgadas por meio de um serviço web, ou por meio de uma busca de diretório via um *Web Services*. Independente da forma que o nó s_a disponibilize os nomes dos fluxos de dados P, os nós r_d mantêm a tabela de recepção de fluxo de dados (como discutido na Seção 1.3.2) que estão repassando para seus clientes $c_f \in C_i(r_d)$ e, sendo assim, podem compartilhá-la para outros nós repassadores. Dessa forma, o GMTP não requer alteração na camada de aplicação para informar o fluxo de dados de interesse – a aplicação continua informando endereço IP e número da porta, mantendo-se a compatibilidade com as aplicações existentes.

De posse de um identificador de um fluxo de dados P, um nó GMTP poderá solicitar os pacotes de dados $p_x \in P$. No caso do uso do DNS, o nó s_a divulga os identificadores de todos os eventos sendo transmitido por meio de um mecanismo de atualização dinâmica de registro de DNS, como especificado na RFC 2136 [11]. Para o GMTP, criou-se um novo tipo

de registro de DNS chamado de SID (*Streaming IDentifier*).

No Quadro 4, ilustra-se um exemplo de uma requisição DNS, utilizando a ferramenta *dig*, um comando de terminal para Linux. Nesse exemplo, apresenta-se a lista dos nomes dos fluxos de dados transmitidos pelo domínio administrativo *globo.com*. Por ser uma consulta simples de DNS, qualquer sistema final conectado à Internet pode realizar tal procedimento, enaltecendo-se a facilitar de adaptar aplicações multimídia existentes para utilizar o GMTP. Ao indexar o conteúdo através de um serviço de DNS, permite-se desacoplar a forma de indexar um determinado conteúdo e a forma de obtê-lo, que passa a ser de responsabilidade da infra-estrutura de rede e não de uma ou mais aplicações isoladamente. Isto pode permitir o aumento em grandes proporções das aplicações multimídia sem se preocupar como localizar um determinado conteúdo, extrapolando-se as barreiras administrativas de cada sistema de geração de conteúdos multimídia, bastante para isso apenas todos passarem a adotar o GMTP.

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

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

Descrição de um fluxo de dados P

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

um mesmo fluxo de dados P. Para isto, incorporou-se ao GMTP o protocolo o SDP (*Session Description Protocol*), definido na RFC 2327 [12], permitindo-se que as aplicações consigam obter mais detalhes sobre a mídia transmitida, flexibilizando-se o acesso a um determinado conteúdo, descrevendo para a aplicação o formato do conteúdo, que então o decodifica e reproduz ao usuário através da aplicação final. Com esta decisão, torna-se mais fácil implementar novas aplicações multimídia, ao passo que também fica mais fácil adaptar aplicações existentes para fazer uso do GMTP, uma vez que, em sua grande maioria, utiliza-se o protocolo SDP. Do ponto de vista de engenharia de software, isto evitará a repetição de esforço com implementações já consolidadas e que, com o passar dos anos, provou-se funcionar a contento, como foi o caso do SDP. Consequentemente, caso seja necessário a atualização do referido padrão, tal atualização será realizada internamente no GMTP e todas as aplicações automaticamente já poderão usufruir dos novos recursos disponibilizados.

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

- *v*, a versão do SDP;
- *o*, a lista de nós s_a que a distribui;
- *s*, o nome da mídia, como discutido na Seção 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 [13] */
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 P (Linhas 10 e 11), sendo um de áudio e outro de vídeo. A distribuição dos fluxos de dados P ocorre com a geração dos pacotes de dados $p_x \in P$ em três nós s_a (Linha 2), dos quais dois são acessíveis através de endereços IPv4 e um através de um endereço IPv6. Os fluxos de áudio e vídeo são repassados por um nó r_d, acessível por um endereço IPv4 (Linha 6), através das portas 49170 e 51372, respectivamente (Linhas 9 e 10). As informações de endereço IP e porta do nó r_d são utilizadas para que os nós $c_f \in C_i(r_d)$ possam sintonizar seus sockets de conexão e iniciar a reprodução da mídia, através do modo de transmissão multicast (Linha 10). Em seguida, na Linha 8, observa-se uma URL do certificado digital a ser utilizado pelo nó r_d para verificar a autenticidade do conteúdo de pacote de dados $p_x \in P$ – este assunto será discutido com mais detalhes na

Seção 1.6. Por fim, entre as Linhas 11 e 17 especificam-se outros parâmetros para descrever a mídia, tais como o nível de qualidade da mídia, que varia entre 1 e 10, as taxas de bits para cada fluxo de dados, sendo para o áudio variando-se entre 16000 *Bytes* e 20000 *Bytes* e, para o vídeo, variando-se entre 156250 *Bytes* e 625000 *Bytes*. É importante salientar que os nós r_d utilizam as informações de taxa de bits para determinar o tamanho do buffer necessário para permitir a transmissão da mídia, 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

No GMTP, divide-se o processo de estabelecimento de conexão em três fases. A Fase 1 acontece quando, por exemplo, um nó qualquer $c_1 \in C_i(r_d)$ deseja obter P transmitido por um nó s_1 e não existe nenhum outro nó $c_f \in C_i(r_d)$ em sua rede local recebendo P. Já a Fase 2 acontece quando um outro nó $c_2 \in C_i(r_d)$ precisa obter o mesmo fluxo de dados P, solicitado previamente pelo nó c_1. E, por fim, a Fase 3 acontece quando o nó r_d começa a buscar novos nós parceiros r_q a fim de obter P. Na Figura 1.10, ilustram-se um nó s_a, que gera um fluxo de dados P e 12 nós r_d, que constituem uma rede de diferentes domínios administrativos, sendo o nó r_1 o repassador de um desses domínios, composto por 6 nós $c_f \in C_i(r_d)$ (Rede Local).

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

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

- P, um fluxo de dados;
- s_1, o nó servidor que gera os pacotes de dados $p_x \in P$;

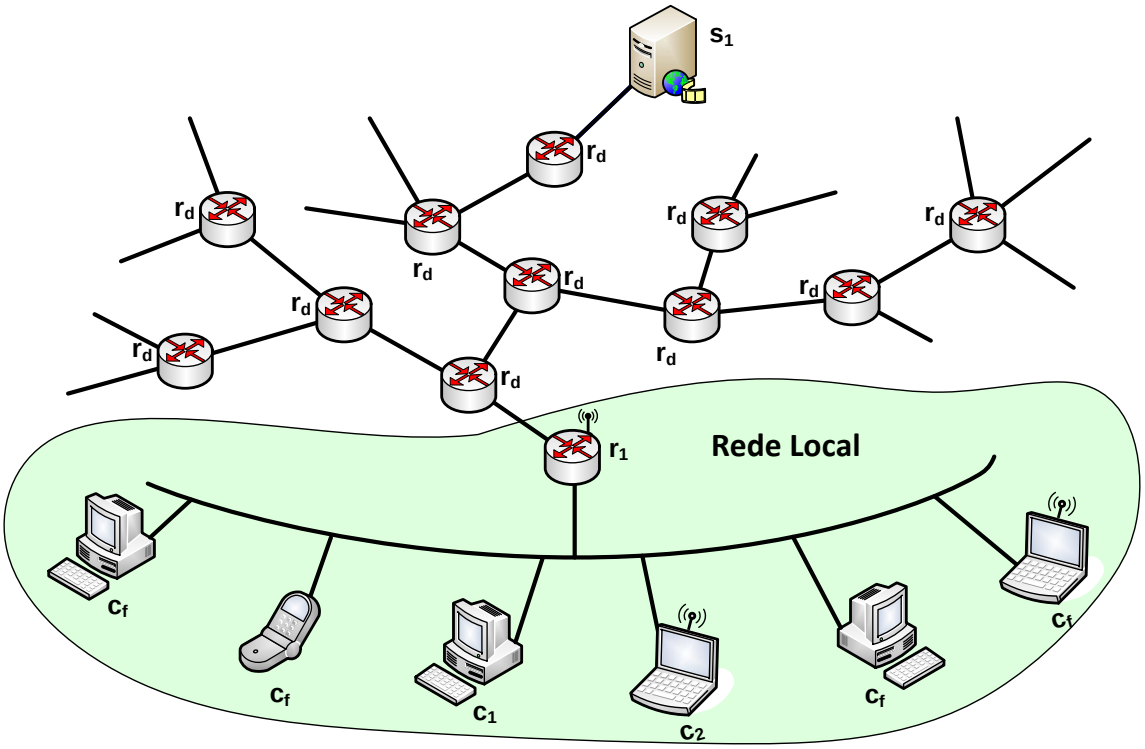

Figura 1.10: Exemplo de rede para o estabelecimento de conexão do GMTP.

- 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 (Seção 1.1.5) e transmite um pacote do tipo *GMTP-Request* (Figura 1.11, 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*. Alternativamente, o nó c_f pode especificar o endereço IP de destino como sendo *0.0.0.0*. 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*. Ao definir o endereço IP de destino como sendo o IP de s_a, permite-se que o roteador padrão de c_1, no caso o nó r_1, obter o endereço IP do nó s_a sem precisar consultar o servidor DNS. Caso contrário, quando o IP especificado for *0.0.0.0*, o nó r_1 deverá realizar tal consulta, como discutiu-se na Seção 1.4.1. Além disso, pelo valor de *TTL=1*, permite-se interceptar o referido pacote de requisição, evitando-se extrapolar o domínio administrativo de tal rede local.

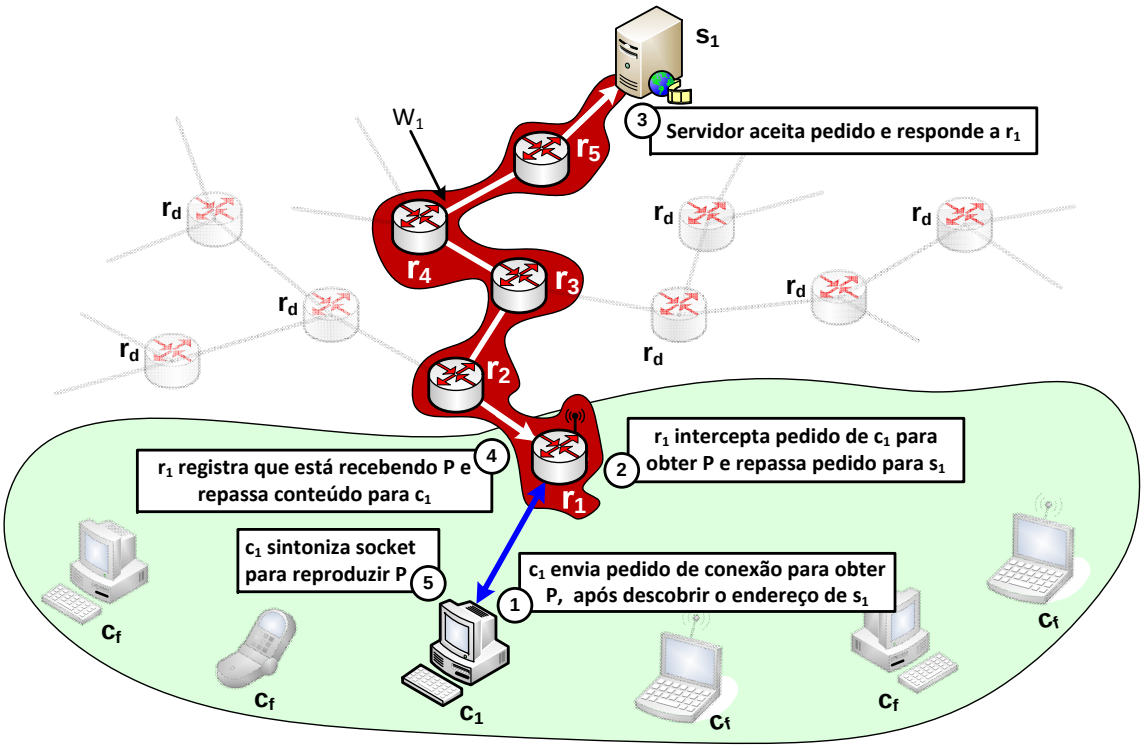

Figura 1.11: Passos do processo de estabelecimento de conexão do GMTP (Fase 1).

Quando o pacote *GMTP-Request* alcançar o nó r_1 (Passo 2 da Figura 1.11), o nó r_1 consulta a tabela de recepção de fluxos de dados e constata que não há qualquer registro para o fluxo de dados P. Nesse instante, o nó r_d inicia um processo de registro de participação para obter o fluxo de dados P. 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 1.11 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 1.11). Esta notificação sinalizará aos nós clientes c_f qual canal multicast seus respectivos sockets devem ser "sintonizado". No caso do exemplo supracitado, o no 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 1.11). Na Seção 1.4, discute-se em detalhes

sobre os aspectos de transmissão de pacotes do tipo *GMTP-Data* ou *GTMP-DataAck*.

No Algoritmo 6, resume-se os passos descritos anteriormente para iniciar a transmissão dos pacotes de dados $p_x \in P$ aos nós $c_f \in C_i(r_d)$, após r_d receber o pacote do tipo *GMTP-RequestReply*. Note que, o nó r_d invoca tal procedimento nas Linhas 7 e 15 do Algoritmo 1 e nas Linhas 11 e 17 do Algoritmo 2 (Seção 1.3.1). Como resultado da Fase 1, gera-se uma nova entrada na tabela de recepção de fluxos de dados do nó r_d, tal como ilustra-se na Figura 1.12. Com base no exemplo citado, a tabela de recepção antes vazia, agora contém uma entrada que informa a ocorrência de recepção do fluxo de dados $P =$ *72c44591-7d82-427c-825f-722f015787c1*, originado no nó s_a, cujo endereço é *177.135.177.241*, com porta de recepção *49170*. Além disso, define-se o canal multicast no endereço *239.192.68.79* e porta *1900*, por onde os nós $c_f \in C_i(r_d)$ podem receber os pacotes de dados $p_x \in P$.

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

```
/* A r_d node executes this Algorithm to respond to clients
   waiting for receiving a flow P. This algorithm is
   invoked in Lines 7 and 15 of Algorithm 1 and in
   Lines 11 and 17 of the Algorithm 2.                    */
1  destAddress ← 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;
```

#	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 1.12: Tabela de recepção de fluxos de dados após a Fase 1.

1.4.4 Fase 2: próximas requisições para obter P

A Fase 2 de conexão ocorre quando futuras requisições para obter o fluxo de dados P são originadas por qualquer nó $c_f \in C_i(r_1)$. Considerando o exemplo anterior, citado na Fase 1, se um nó $c_2 \in C_i(r_1)$ também solicitar P, o nó r_1 simplesmente informará o canal multicast correspondente ao fluxo de dados P, como ilustra-se na Figura 1.13. 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*. Este procedimento ocorre no registro de participação, especificamente no trecho de código definidos entre as Linhas 2-8 do Algoritmo 1. Em seguida, o pacote do tipo *GMTP-Request-Reply* é transmitido para o nó c_2, como descreve-se no trecho de código entre as Linhas 10-16 do Algoritmo 6. Tal procedimento se repete para cada novo nó $c_f \in C_i(r_1)$ interessado em obter P.

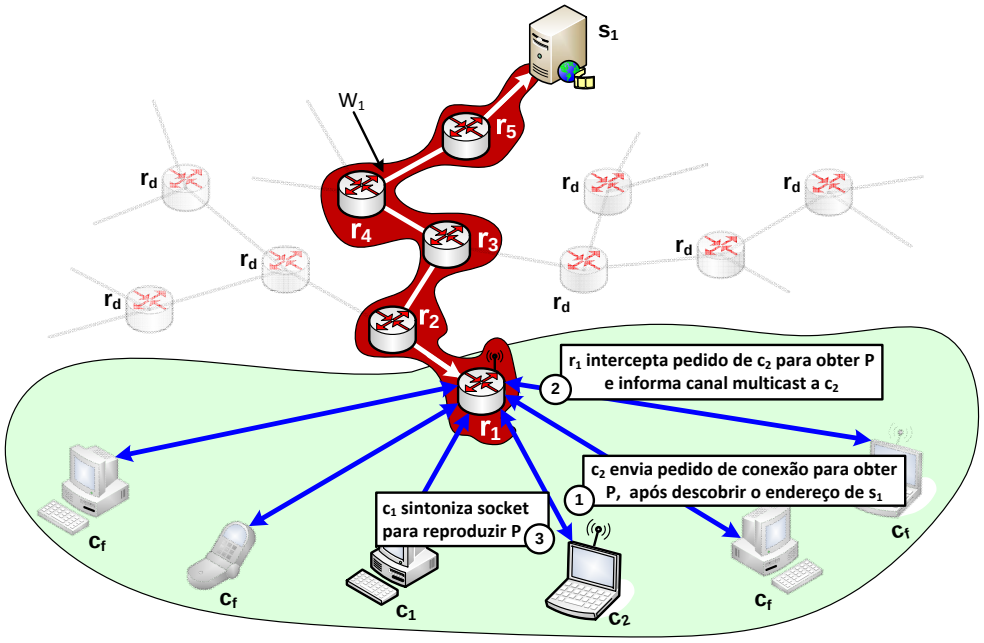

Figura 1.13: Passos do processo de estabelecimento de conexão do GMTP (Fase 2).

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. Para conseguir mais nós parceiros r_q, o nó r_3 envia uma requisição do tipo *GMTP-RelayQuery* para s_a e obtém um subconjunto de nós r_q candidatos a parceiro de r_3, como ilustra-se na Figura 1.14. O nó s_a constrói a lista de nós parceiros e envia ao nó s_a, funcionando como um indexador (*tracker*) de nós parceiros r_q, pré-selecionando 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 taxa de transmissão disponível do nó $w_m \in W_v$ com menor capacidade de transmissão 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.

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 uma maior quantidade de parcerios de um nó r_d significa uma melhor qualidade do fluxo de dados P. Por isso, um nó r_d sempre mantém uma lista de candidatos a parcerios r_q, porém não estabelece parcerias com todos eles. Em vez disso, utiliza-se os seguintes critérios:

- 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 $c_f \in C_i(r_d)$. Este passo se repete continuamente.
- A quantidade de parcerias em um determinado instante *t* é inversamente proporcional a quantidade de pacotes de dados $p_x \in P$ que chegam repetidos ao nó r_d. Nesse caso, se um mesmo pacote p_x chegar repetidamente na mesma quantidade de parcerias estabelecidas, o nó r_d desconecta-se daquele nó parceiro r_q que recebeu o pacote p_x repetido por último. Este passo se repete continuamente.

2. Após obter a lista de candidatos a parceiros (Passo 1), o nó r_3 envia requisições do tipo *GMTP-Request* em direção a outro nó r_2 que já esteja recebendo um fluxo de dados P. Como ilustra-se na Figura 1.15, este procedimento ocorre da seguinte forma: após o passo anterior, o nó r_3 envia uma requisição do tipo *GMTP-Request* para o nó r_2 contendo uma chave de autorização conhecida por ambos e informada pelo nó s_a. Caso a chave de autorização esteja correta, o nó r_2 deve enviar um resposta do tipo *GMTP-Response* ao nó r_3 e então começar a repassar os pacotes $p_x \in P$. O uso da chave de autorização é importante para evitar que um nó r_d se conecte a outro nó r_q sem que o nó s_a seja notificado sobre isto. As chaves de autorização são geradas pelo nó s_a e transmitidas como resposta no pacote do tipo *GMTP-Register-Reply*;

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

Com a execução da Fase 3 do processo de conexão do GMTP, pode-se expandir a quantidade de parcerias e essas informações são registradas na tabela de recepção de fluxos de dados, tal como ilustra-se na Figura 1.16. Nesse exemplo, nota-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 $P = 72c44591$-

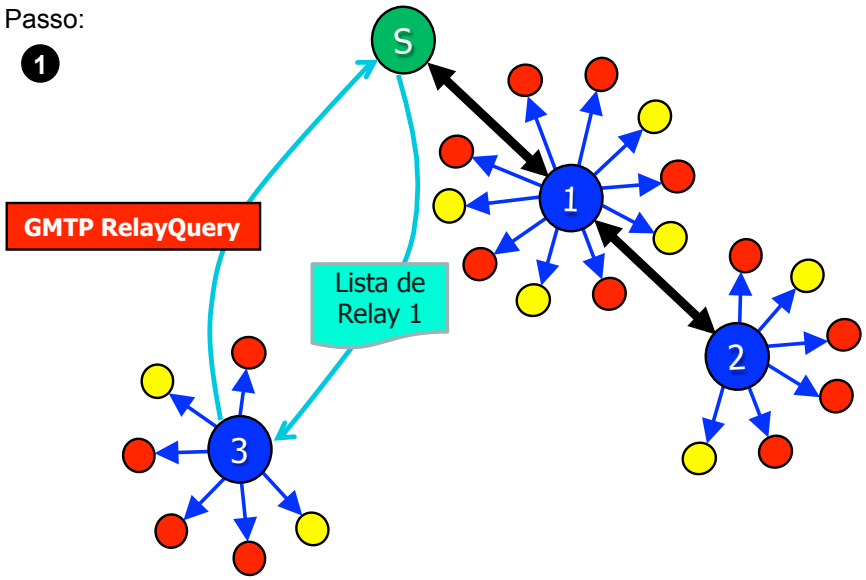

Figura 1.14: Fase 3 de conexão do GMTP (Passo 1).

7d82-427c-825f-722f015787c1, cujos pacotes de dados $p_x \in P$ são transmitidos por três nós r_q identificados pelos endereços ip e porta *182.111.88.21:49170*, *90.39.135.46:62242* e *83.67.132.41:53434*. Para esse fluxo de dados P, os pacotes de dados p_x são repassados para os nós $c_f \in C_i(r_d)$ através do canal multicast *239.192.68.79:1900*.

Desta forma, o processo de conexão do GMTP é fundamental para a efetiva distribuição de mídias ao vivo, pois permite que as aplicações compartilhem fluxos de dados entre si, mesmo que estas não tenham sido desenvolvidas pela mesma equipe. Esta unificação ajuda no processo de distribuição do fluxo de dados P, pois, na prática, até mesmo uma aplicação *standalone* e um objeto de vídeo imbutido em uma página Web podem obter o mesmo fluxo de dados sem que estas conheçam um a outra. Como resultado, reduz-se para 1 o número de transmissões para um mesmo fluxo de dados P originado em s_a e destinados a uma mesma rede ou para um subconjuntos de redes adjacentes. Além dessa diferença, a forma de conexão do GMTP supre uma antiga deficiência das soluções tradicionais de transmissão multicast, as quais os nós clientes, camada de aplicação, tinham que se adaptar às configurações estáticas dos canais multicast definidos pelo administrador de rede, e até os próprios administradores de rede tinham que fazer tal configuração de forma manual, que obrigatoriamente tinha que ser realizada em todos os nós roteadores de um determinado caminho.

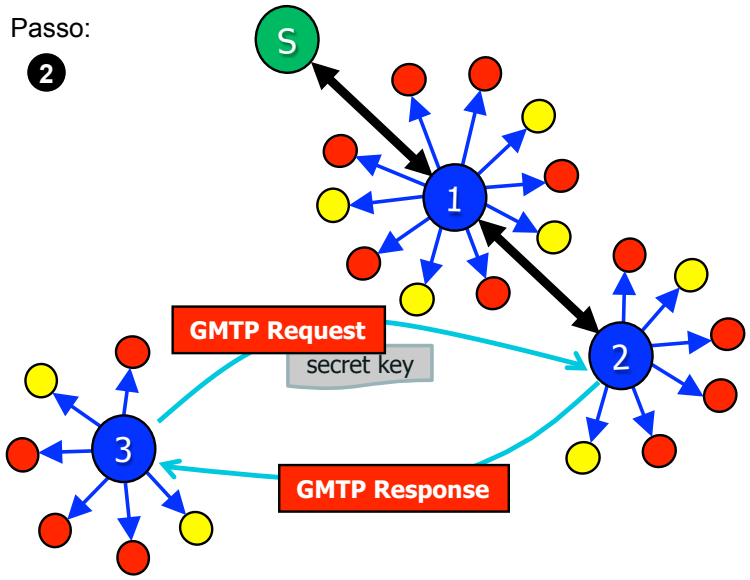

Figura 1.15: Fase 3 de conexão do GMTP (Passo 2).

#	Nome do Fluxo de Dados (P)	Servidores s_a	Repassadores r_d	Porta em r_d	Endereço do Canal Multicast	Porta do Canal Multicast
1	72c44591-7d82-427c-825f-722f015787c1	177.135.177.241	182.111.88.21	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 1.16: Tabela de recepção de fluxos de dados após a Fase 3.

Até o presente momento, não se conhece nenhuma solução que permita configuração dinâmica de canais multicast aliado a formação de uma rede de sobreposição constituída entre roteadores, com benefícios diretos para a aplicação e para a rede, fazendo-se uso dos recursos computacionais e de rede de forma mais apropriada, como será discutido 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 obter P, constituído de pacotes p_x do tipo *GMTP-Data* e *GMTP-DataAck*. De forma similar, os nós r_d utilizam os mesmos tipos de pacotes para enviar $p_x \in P$ para os nós c_f, porém em modo multicast. 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 $p_x \in P$, um s_a deve transmitir pacotes do tipo *GMTP-Data* ou o *GMTP-DataAck* em direção aos nós r_d de acordo com os registros de partipação. Nesta seção, detalha-se como o GMTP executa os procedimentos de transmissão e recepção desses pacotes de dados.

Buffer de Envio e Recepção:

A transmissão de um evento $\mathcal{E}$ consiste no processo de disseminação dos pacotes $p_x \in P$ através dos nós interessados em obtê-lo. Para isto, cada nó GMTP controla um buffer de envio e recepção definido por uma estrutura de dados do tipo array circular (*ring buffer*), onde cada posição é utilizada para armazenar um pacote p_x (Figura 1.17). Ao receber p_x, um nó GMTP armazena-o no buffer e posteriormente o entrega para a aplicação, que o reproduz para o usuário final. Para o envio ou repasse de um pacote, o nó GMTP consome os pacotes p_x do buffer e transmite para o(s) nó(s) interessado(s), seja em modo unicast e/ou em modo multicast. Isto porque é possível que um nó r_1 repasse p_x para um outro nó r_2 (unicast) ao mesmo tempo que r_1 pode repassar P para seus nós c_f (multicast).

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

transporte de um determinado fluxo de dados P. O tamanho do buffer é especificado pelo nó s_a e propagado para os demais nós em um caminho W no cabeçalho do pacote do tipo *GMTP-Register-Reply* ou *GMTP-MediaDesc*.

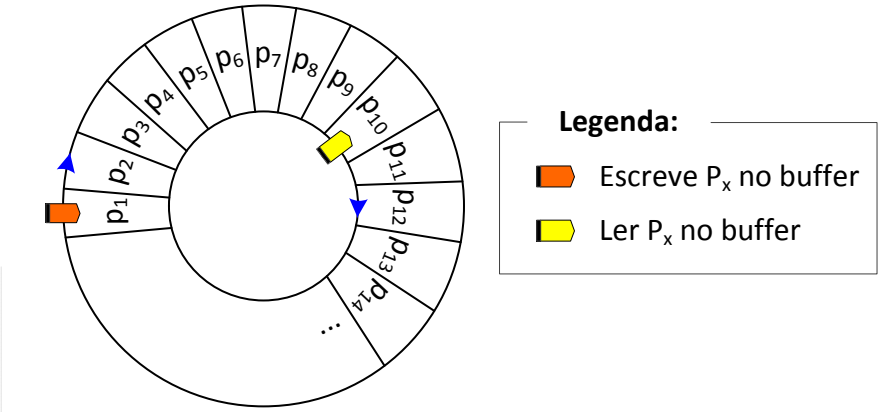

Figura 1.17: Exemplo da estrutura do buffer de envio e recepção de um nó GMTP com dois ponteiros, um para escrever e outro para ler pacotes p_x.

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

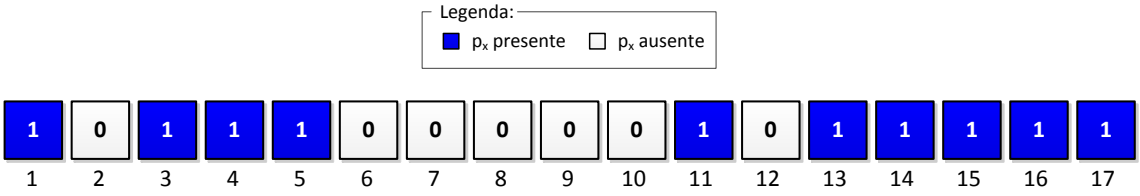

Figura 1.18: Exemplo do mapa de buffer de um nó GMTP com tamanho de 17 p_x.

O mapa de buffer é utilizado por um nó GMTP para sinalizar seu atual estado com relação a um determinado fluxo de dados P. Um nó GMTP pode enviar o mapa de buffer completo, como ilustrado na Figura 1.18, ou o mapa de buffer apenas dos p_x presentes ou ausentes. Na prática, um nó r_d envia para um nó parceiro r_q o mapa de buffer dos p_x presentes quando

deseja indicar a sua atual disponibilidade; ao passo que envia o mapa de buffer dos p_x ausentes quando desejar obtê-los. Para 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 automaticamente o uso do método *pull*, em vez do método *push*, que é o modo padrão do GMTP. Salienta-se que se evita o método pull devido à transitoriedade dos pacotes de dados p_x (transmissão de dados ao vivo) e um nó r_d apenas realiza tal procedimento após completar a Fase 3 do processo de estabelecimento de conexão.

Quando um nó r_d percebe a falta de um ou mais pacotes p_x, este pode solicitar a um ou mais nós r_d os pacotes p_x ausentes e então obtê-los usando o método *pull*. Para isso, um nó r_d enviar aos seus nós parceiros r_q o mapa de buffer dos pacotes p_x ausentes e aguarda as respostas sobre tal disponibilidade. Essa sinalização ocorre através do uso do pacote do tipo *GMTP-DataPull-Request*, que é preenchido com o mapa de buffer dos pacotes ausentes e transmitido aos respectivos nós parceiros. Ao receber esse tipo de requisição, um nó parceiro avalia seu conteúdo e responde com o pacote do tipo *GMTP-DataPull-Response*, o qual contém o mapa de buffer dos pacotes disponíveis, seguido dos pacotes p_x do tipo *GMTP-Data*. Note que os pacotes do tipo *GMTP-DataPullRequest* e *GMTP-DataPull-Response* são transmitidos com garantia de entrega, ou seja, caso sejam perdidos, o GMTP garante sua retransmissão. Para isto, o GMTP utiliza o mecanismo básico de envio e confirmação utilizando o pacote do tipo *GMTP-DataAck* ou *GMTP-DataAck*. No caso de falha na execução de uma requisição utilizando o método *pull*, o nó GMTP pode reavaliar a necessidade de retransmitir o pedido, pois é possível que os p_x ausentes já tenham expirados e requisitá-los novamente não fará mais sentido.

Na prática, o mapa de buffer utilizado para sinalizar a presença ou ausência de p_x é representado por faixas de acordo com o índice do buffer. Por exemplo, para representar o mapa de buffer dos pacotes ausentes ilustrados na Figura 1.18, 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:** descartar os primeiros pacotes p_x recebidos se o buffer alcançou seu limite, mesmo que ainda não tenham sido repassados. Uma otimização não explorada neste trabalho, mas que é possível de ser realizada, é o descarte seletivo de pacotes, primeiro os que tenham menos impacto na qualidade da mídia, por exemplo, pacotes de dados contendo quadros B (codificação MPEG4, tipo 2). Isto não impede que o vídeo seja reproduzido, porém com perda de qualidade, ao passo que se permite a transmissão dos pacotes de dados p_x com a largura de banda de rede disponível;

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

1.5 Controle de Congestionamento em η

COLOCAR O CAMPO RTT? ISSO AJUDARÁ MUITO E SEGMENTARÁ O CAMINHO, EVITANDO-SE QUE A TRANSMISSÃO DE UM FLUXO NÃO SEJA DEGRADADA POR UM NÓ EM GARGALO QUE ESPECIFICARÁ UMA VELOCIDADE DE TAXA DE TRANSMISSÃO MENOR DO QUE SUPORTADO POR UM CAMINHO MAIS PRÓXIMO DO SERVIDOR:

S1 R1 R2 R3 R4 R5 (COMO O GMTP USA O RCP, SE R1,R5 RECEBEM P, O USO DO RCP PURO IMPACTARIA QUE A TAXA DE TRANSMISSÃO DE R1 SERIA GOVERNADA POR R5. GUARDANDO TAMBÉM, PODE-SE TER TX INFLUENCIADA APENAS PELOS NÓS DO SUBCAMINHO.)

No GMTP, define-se dois algoritmos para controle de congestionamento, um para transmissões em modo unicast e outro para transmissões em modo multicast. Como ilustra-se na Figura 1.19, para a parte do algoritmo que funciona em modo unicast, dar-se o nome de *GMTP Unicast Congestion Control* (GMTP-UCC), ao passo que para a parte do algoritmo

que funciona em modo multicast, dar-se o nome de *GMTP Multicast Congestion Control* (GMTP-MCC). Nesta seção, discute-se o funcionamento de cada um desses algoritmos.

Na prática, definiu-se um algoritmo para controle de congestionamento híbrido, cujo comportamento dependerá se o nó que o executa está transmitindo em modo unicast ou em multicast. Em modo de transmissão unicast, utilizado na comunicação entre os nós r_d, define-se a taxa de transmissão de um nó GMTP através de uma versão modificada do protocolo RCP (Rate Control Protocol) [14]. Já em modo de transmissão multicast, executa-se um algoritmo baseado em relatórios (*feedbacks*) transmitidos pelos nós clientes l_w, eleitos em cada rede e controlados por um nó r_d, tal que $l_w \in C_i(r_d)$.

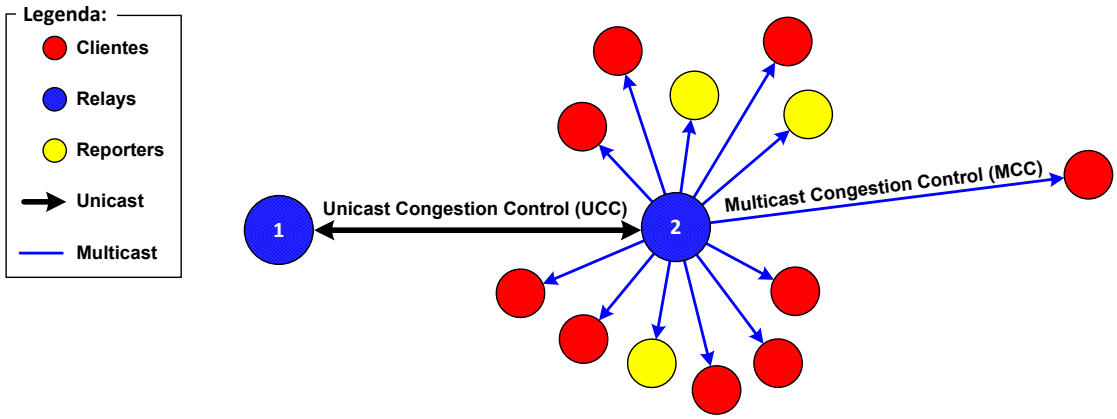

Figura 1.19: Organização do algoritmo de controle de congestionamento no GMTP.

1.5.1 Controle de Congestionamento Unicast

O GMTP-UCC funciona de forma similar ao protocolo RCP, 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 t, a taxa de transmissão ideal para cada fluxo de dados seria $R_{ps}(t) = \frac{C}{N(t)}$, onde C corresponde à capacidade do link e $N(t)$ ao número de fluxos no instante t.

Partindo desse ponto, Nandita et. al [14] argumenta que para um roteador funcionar de forma equânime, tal roteador deve oferecer a mesma taxa de transmissão para todos os fluxos transmitidos através dele, mantendo-se o número de pacotes na fila de roteamento perto de

zero, a fim de evitar que apenas os fluxos que tem pacotes na fila de repasse compartilhem a largura de banda disponível. Com base nisso, Nandita et. al [14] determinou a Equação 1.1, onde $R(t)$ é a taxa de transmissão que deve ser oferecida para cada fluxo de dados que passa pelo roteador. Pela Equação 1.1, estima-se a largura de banda disponível em um determinado canal, representado pela porção $\alpha(C-y(t))-\beta\frac{q(t)}{d_0}$ (mudança agregada) e a divide por $N(t)$. Porém, como é impossível determinar o valor exato de $N(t)$, estima-se $\hat{N}(t) = \frac{C}{R(t-T)}$ e para atualizar $R(t)$ com mais frequência do que no tempo de um RTT, escala-se a mudança agregada por $\frac{T}{d_0}$, resultando na Equação 1.2, onde:

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

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

- d_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 d_0 ao valor de RTT_s.

$$d_0 = \theta \times RTT_s + (1-\theta) \times d_0 \tag{1.3}$$

- $T = min(RTT_{user}, d_0)$, sendo RTT_{user} um tempo definido pelo usuário (administrador do roteador) caso seja necessário atualizar $R(t)$ mais rápido do que o tempo de d_0;

- $R(t-T)$, é a última taxa de transmissão medida;

- $y(t)$, é a taxa de tráfego de entrada medida no intervalo entre a última atualização da taxa de transmissão e d_0;

- $q(t)$, é o tamanho instantâneo da fila de repasse, em bytes. Note que no GMTP esse valor é obtido pela soma de todos os pacotes p_x presentes em todos os buffers de recepção e envio para cada fluxo de dados P;

- α e β, são parâmetros pré-definidos que determinam a estabilidade e o desempenho;

- C, é a capacidade do link.

No GMTP-UCC, o algoritmo para controle de congestionamento, adaptado do RCP, funciona da seguinte forma (acompanhe os passos de acordo com a Figura 1.20):

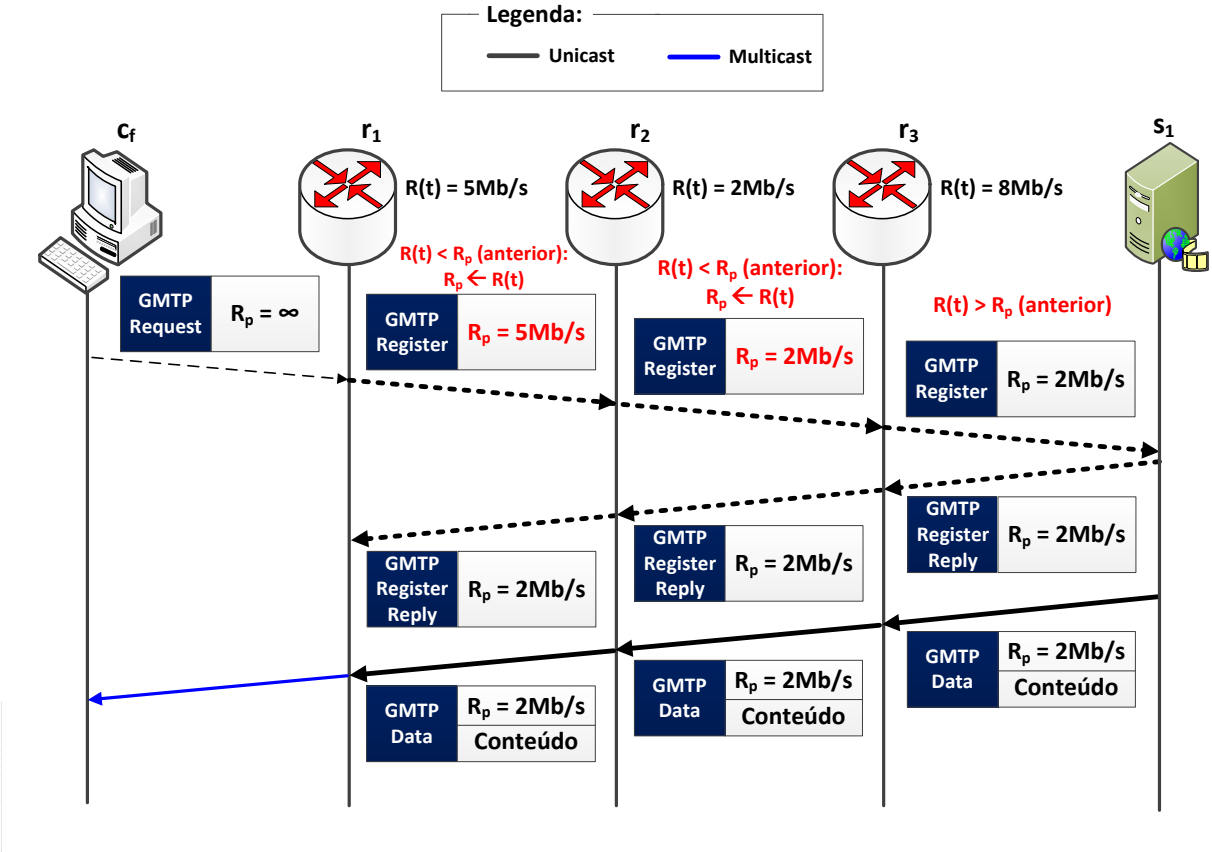

Figura 1.20: Cada r_d mantém uma única taxa de transmissão $R(t)$ que é atribuída em todos os pacotes de dados p_x no cabeçalho de qualquer pacote GMTP. A medida que o pacote passa através dos roteadores em um caminho W_v, se a taxa atual $R(t)$ no roteador for menor do que R_p informado no pacote sendo processado, o roteador o sobrescreve com sua taxa atual $R(t)$. Quando o pacote alcançar o destino, o nó $s_a$utiliza R_p para transmitir o fluxo de dados, pois trata-se da menor taxa de transmissão do roteador mais congestionado. Este procedimento se repete periodicamente utilizando os pacotes de dados *GMTP-Data* e *GMTP-Ack*.

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

2° Todo pacote GMTP carrega duas informações de controle (campo no cabeçalho):

- *taxa de transmissão proposta* (R_p): corresponde à taxa de transmissão necessária para transmitir um fluxo de dados P, em geral, calculada pelo nó s_a;
- *RTT na fonte* (RTT_s): corresponde ao RTT estimado entre quaisquer nós t_u,t_{u+1} $\in W_v$, ou seja, o RTT entre dois nós t_u e t_{u+1} que processam o respectivo pacote p_x de um fluxo de dados P, a fim de repassar aos seus nós $c_f \in (C_i(t_u) \cup C_i(t_{u+1}))$.

3° No início de uma transmissão de um fluxo de dados P, o nó c_f transmite um pacote p_x com o valor de $R_p = \infty$ em direção a s_a.

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

(a) repassa p_x para seus nós c_f em modo multicast (Seção 1.4.6);

(b) cria um pacote *GMTP-Ack* contendo R_p e o envia de volta para seu nó parceiro $r_q = w_{m-1}$. O pacote *GMTP-Ack* também carrega um campo de RTT_s. Quando w_m receber um pacote *GMTP-Ack*, deve-se utilizar RTT_s para atualizar a média móvel do RTT no intervalo de d_0 (Equação 1.3).

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

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

Sendo assim, no caso do GMTP, a ideia básica é a seguinte: para quaisquer dois nós t_1,$t_2 \in W_v$, a taxa de transmissão a ser utilizada por t_1 e t_2 será definida pela menor taxa de transmissão oferecida pelos nós $w_m \in W_v$ posicionados entre t_1 e t_2. Com isto, se existir largura de banda disponível entre t_1 e t_2, ou seja, $C - y(t) > 0$, então o GMTP compartilhará

igualmente o canal entre todos os fluxos, inclusive para o fluxo entre t_1 e t_2. Caso contrário, ou seja, se $C - y(t) < 0$, considera-se o canal saturado e o GMTP reduzirá a taxa de transmissão igualmente para todos os fluxos, inclusive para o fluxo entre t_1 e t_2. Especificamente, a largura de banda necessária para repassar todos os pacotes $p_x \in P$ que estão na fila de roteamento no intervalo de um RTT corresponde à $\frac{q(t)}{d_0}$ [15].

Segmentação de um caminho W_ve definição de d_0:

Falar do problema de usar o RCP (end-to-end pode ser um problema) e o GMTP segmenta isso

que, no caso do GMTP, não é definido entre o nó s_a e o nó c_f, como na versão original do RCP. Em vez disso, o tempo d_0 é definido entre dois nós r_d e r_q parceiros entre si contidos em um caminho W_v, tal que $\varphi(r_d, P) = 1$ e $\varphi(r_q, P) = 1$.

Diferença entre o GMTP-UCC e o RCP:

- quem inicia é o cliente, não o servidor - segmentação do caminho - o cliente sabe a qual taxa os pacotes irão chegar (pode adaptar fluxo)

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

LEMBRAR DE MENCIONAR O ITEM QUE PRIORIZA ISSO, FASE 3 DE CONEXÃO

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

Essa seção foi escrita com base na referência [15], incluindo seus gráficos, que foram extraídos da mesma referência.

A motivação para o RCP é identificar um algoritmo para controle de congestionamento simples e prático para emular um PS independente da característica do tráfego e das condições da rede. A abordagem adotada no RCP é diferente se comparada ao TCP e ao XCP. No RCP, em vez de monitorar a mudança de uma janela deslizante a cada tempo de RTT, busca-se 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 [1] são os

protocolos mais conhecidos do estado da arte que tentam emular um PS e, por este motivo, suas equações de controle de congestionamento são similares. Porém, o modo que o RCP e o XCP tentam convergir suas respectivas taxas de transmissão $R_{rcp}(t)$ e $R_{xcp}(t)$ é bastante diferente, alocando-se tais taxas para cada fluxo de dados a fim de emular a taxa de transmissão do PS, definida por $R_{ps}(t)$. Dessa forma, tornou-se 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.

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, 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. Por exemplo, nos gráficos da Figura 1.21, compara-se o TCP e o XCP com um PS ideal com base em uma rede simulada, com taxa de entrada de pacotes de dados de um fluxo definida em *Poisson* e tamanhos dos fluxos em distribuição *Pareto* com média de 30 pacotes (1000 bytes *pacote*), *shape* igual a 1.4, capacidade do link igual a 2.4 *Gbps* e RTT igual a 100 *ms*, com carga ofertada igual a 0.9. No gráfico esquerdo, ilustra-se o tempo médio de duração (quanto tempo o respectivo protocolo gasta para completar o fluxo) em função do tamanho do fluxo. No gráfico direito, ilustra-se o número de fluxos ativos em função do tempo. Os valores de PS foram calculados a partir de expressões analíticas [16] e mostram que os fluxos poderiam

ser finalizados uma ordem de magnitude mais rápida do que o TCP.

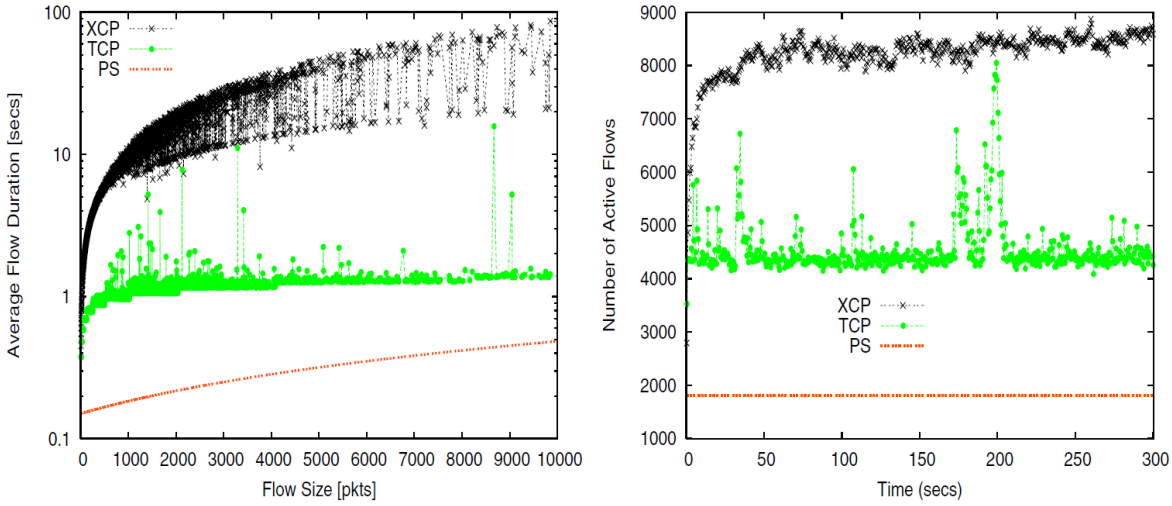

Figura 1.21: No gráfico esquerdo, ilustra-se o tempo médio de duração (quanto tempo leva para completar) de um fluxo versus o tamanho do fluxo utilizando o TCP e o XCP. No gráfico direito, ilustra-se o número de fluxos ativos versus o tempo. Ambos os gráficos são resultados de simulações com chegada de fluxo em Poisson e tamanhos do fluxo em distribuição Pareto com média de 30 pacotes (1000 bytes *pacote*), shape igual a 1.4, capacidade do link igual a 2.4 *Gbps* e RTT igual a 100 *ms*, com carga ofertada igual a 0.9. Os valores de PS foram calculados a partir de expressões analíticas.

Com base nos gráficos da Figura 1.21, observa-se que os fluxos TCP demoram para finalizar porque consome-se múltiplos RTTs na fase de partida lenta para encontrar uma taxa de transmissão equânime, além do mais, muitas vezes o fluxo acaba antes que tal taxa seja encontrada. Em seguida, quando o fluxo TCP entra no modo de prevenção de congestionamento, o TCP adapta-se lentamente devido ao método de aumento aditivo, o que aumenta o tempo de finalização do fluxo. Além disso, o TCP deliberadamente preenche o buffer dos roteadores saturados de modo a ajustar a taxa de transmissão com base nos descartes de pacotes, mas buffers adicionais resulta em aumento no tempo (atraso) para entregar um pacote de dados, impactando no tempo total de duração de um fluxo. Já o XCP funciona melhor em redes com altos produtos largura de banda–atraso. Os roteadores disponibilizam para as fontes transmissoras relatórios sobre as mudanças da janela de congestionamento, enviados em múltiplos RTTs, que funcionam a contento quando todos os fluxos são de longa duração. Por isso, em um ambiente dinâmico, o XCP pode aumentar a duração de cada fluxo em relação

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

Já no RCP, todos os fluxos (novos e antigos) recebem a mesma taxa de transmissão $R_{rcp}(t)$ baseada no estado atual do nó r_d com menor largura de banda disponível em um certo instante t. Isto permite que um fluxo de dados de curta duração termine o mais rápido possível ao passo que os fluxos de dados mais longos não influenciam diretamente no compartilhamento equânime do PS, sem permitir que parte da largura de banda disponível fique ociosa por muito tempo. Este procedimento ocorre em um intervalo de tempo definido por d_0 (vide Equação 1.2).

Como observa-se no gráfico da Figura 1.22, a estratégia do RCP de compartilhar uma única taxa de transmissão para qualquer fluxo com base no estado atual do roteador saturado, produz um resultado satisfatório no que diz respeito a melhor utilizar o canal de transmissão (seja quando em altos níveis de utilização quanto de ociosidade). Com base no gráfico, percebe-se que em comparação ao XCP e a outras soluções tradicionais como o TCP, o RCP emula melhor um PS e por isso acompanha o tempo médio de finalização de um fluxo de dados à medida que se aumenta o tamanho do fluxo de dados. Note que, para o cenário descrito, o XCP teve um desempenho pior se comparado inclusive ao TCP.

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

Desta forma, os aspectos que determinam o funcionamento do RCP são fundamentais quando se trata de transmissão de conteúdos multimídia ao vivo, aliado às outras estratégias

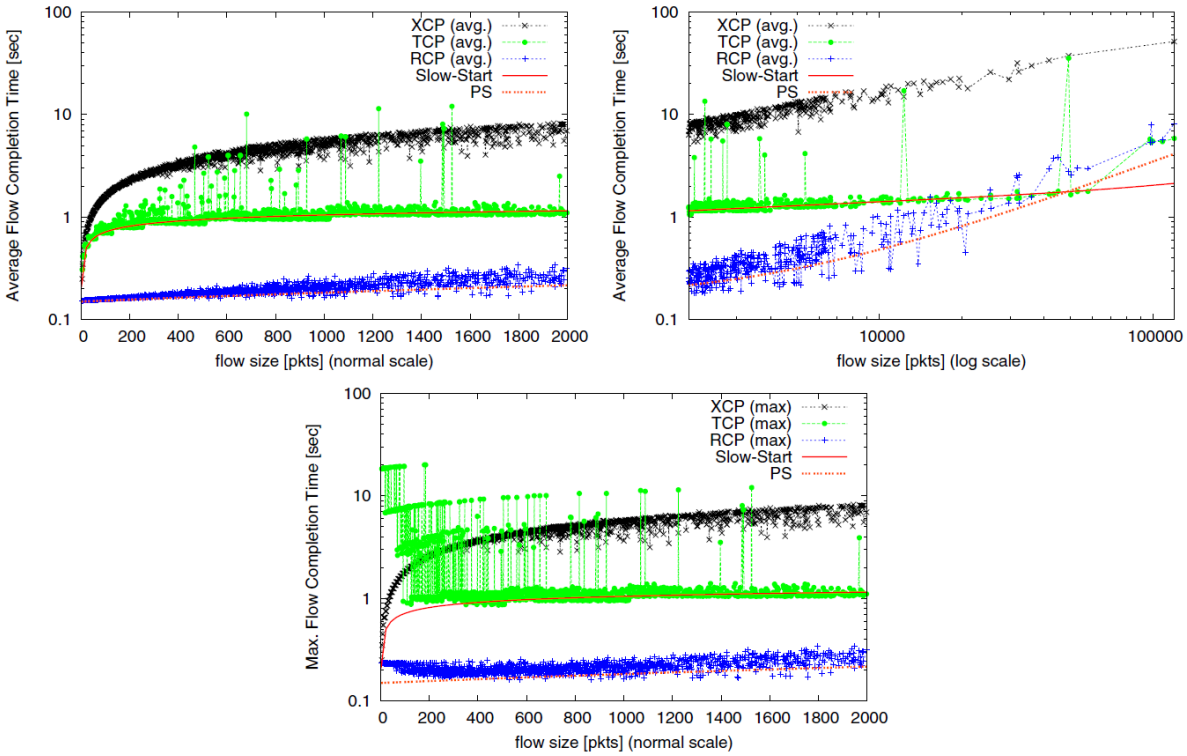

Figura 1.22: Tempo médio, em segundos, de finalização de um fluxo de dados, ao utilizar os protocolos XCP, TCP e RCP como resultados de simulações com taxa de entrada de dados em Poisson e tamanhos do fluxo em distribuição Pareto com média de 25 pacotes (1000 bytes *pacote*), shape igual a 1.2, capacidade 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.

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

1.5.2 Controle de Congestionamento Multicast

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

O GMTP-MCC foi inspirado em um protocolo publicado pela IETF chamado *TCP-friendly Rate Control protocol (TFRC)* (RFC 3448 [17]). O TFRC é um mecanismo para controle de congestionamento de fluxos unicast que tenta prevê a taxa de transmissão de um fluxo TCP e utilizá-la em protocolos diferentes do TCP [18]. Trata-se de uma abordagem diferente da utilizada em algoritmos baseados em janela deslizante e que utilizam pacotes de confirmação para determinar a taxa de transmissão de uma conexão, como acontece no TCP. No TFRC, o receptor envia para o transmissor relatórios sobre as perdas observadas e, com base nesse relatório, o transmissor calcula a nova taxa de transmissão. O TFRC é categorizado com um protocolo de controle de congestionamento baseado em uma equação matemática (*Equation Based Congestion Control*). Algoritmos desse tipo são adotados em diversos protocolos, como é o caso dos CCIDs 3 e 4 do DCCP [19, 20]. Em linhas gerais, o algoritmo TFRC funciona da seguinte forma:

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

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

3° o transmissor utiliza a Equação 1.4 para determinar qual será a sua próxima taxa de transmissão em função do relatório de perdas e o RTT obtidos anteriormente;

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

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

Na Equação 1.4 [21], T é a taxa de transmissão medida em bytes/segundo definida em função de *s*, que é o tamanho do pacote medido em bytes; RTT, é o RTT entre o nó transmissor e o receptor, medido em segundos e p, a taxa de perda de pacotes observado pelo nó receptor.

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

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

1° O nó r_d executa um algoritmo de eleição de nós relatores $l_w \in C_i(r_d)$. Na Seção 1.7.2, descreve-se o procedimento para eleger os nós l_w.

2° Os nós l_w calculam a taxa de transmissão utilizando a Equação 1.4, ao invés do transmissor realizar este cálculo, como na versão original do TFRC;

3° Os nós l_w determinam a taxa de eventos de perda, e não todos os receptores do grupo multicast. Para calcular o evento de perda p, utiliza-se o mesmo procedimento feito pelo TFRC [17, 21], onde um intervalo de perda é determinado por consecutivas perdas de pacotes, desde do primeiro pacote perdido até o último pacote perdido, seguido de um pacote recebido com sucesso;

4° O RTT é calculado entre o nó l_w e o nó r_d, com o temporizador controlado pelos nós l_w e não pelo nó r_d. Isto evita que o nó r_d tenha que manter estado de temporizador para cada fluxo de dados P transmitido para os nós $c_f \in C_i(r_d)$. Para determinar o valor do parâmetro RTT e calcular a taxa de transmissão através da Equação 1.4, o GMTP-MCC utiliza a Equação 1.3, com $\theta = 0.25$, padrão do TCP;

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

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

Teoricamente, o GMTP-MCC seria um protocolo *TCP-Friendly* se $R(RTT, p)$ fosse o valor máximo entre as taxas de transmissão relatadas pelos nós l_w. Porém, optou-se por utilizar a média 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 $150\,ms$. Quando um nó c_f envia um pedido de conexão utilizando o pacote do tipo *GMTP-Request*, o mesmo deve realizar a sua primeira medição do valor de RTT, iniciando-se o marcador de tempo para o cálculo do RTT quando enviar o primeiro *GMTP-Request* e parando-o quando receber o pacote do tipo *GMTP-Response*. Em seguida, deve-se acionar o mecanismo de cálculo da taxa de transmissão através da Equação 1.4, caso o respectivo nó c_f seja eleito um nó relator.

1.6 Autenticidade de P

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

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

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

Dada a quantidade $pc(t)$ de pacotes que w_m deve verificar suas respectivas autenticidades, o nó w_m escolhe aleatoriamente (distribuição uniforme) os pacotes p_x disponíveis no buffer de recepção, gerendo 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 ou desabilitar o procedimento de segurança

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

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

Note que quando um nó $c_f \in C_i(r_d)$ solicitar um fluxo de dados P, em resposta a tal pedido, o nó r_d retornará um pacote do tipo *GMTP-Request-Notify*. No cabeçalho desse pacote, o nó r_d deve também ativar a opção assinado (*signed*) para que o nó c_f seja notificado e entenda que seu nó r_d realizará a verificação de autenticidade do fluxo de dados P da forma descrita anteriormente na Seção 1.6.2. Este procedimento permitirá que a aplicação em execução no nó c_f possa informar ao usuário final que tal funcionalidade está habilitada, por exemplo.

Além disso, como parâmetros de configuração, o usuário administrador do nó r_d pode habilitar ou desabilitar a opção de verificação de autenticidade dos fluxos de dados P, mesmo que o nó s_a possibilite tal verificação, como descrito anteriormente.

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

Um nó r_d obtem a chave pública $K^+_{s_a}$ de s_a através do certificado digital disponível na URI especificada no parâmetro *f* da descrição da mídia, como ilustrou-se no Trecho de Código 5, Linha 7, da Seção 1.4.1. Isto ocorre após o nó r_d receber o pacote *GMTP-Register-Reply*, que confirma o registro de participação ou a conexão para obter um fluxo de dados P, como apresentou-se no Trecho de Código 1, Linha X, Seção 1.3.1.

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

1.7 Outras considerações sobre o GMTP

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

1.7.1 Procedimentos para desconexão de nós c_f, l_w e r_d

O processo de finalização de uma conexão GMTP ocorre com algumas diferenças se comparado com outros protocolos orientados à conexão. Para sinalizar uma desconexão, um nó c_f transmite um pacote do tipo *GMTP-Close* pelo canal de controle, contendo o nome do fluxo que deseja se desconectar. Ao receber este tipo de pacote, o nó r_d transmite ao nó c_f um pacote do tipo *GMTP-Reset*, sinalizando que está ciente do fechamento da conexão. Nesse interim, os nós desalocam recursos relacionados à respectiva conexão. Este procedimento é suficiente para o pedido de finalização de uma conexão de um cliente GMTP, porém para finalizar uma conexão de um nó l_we r_d outros procedimentos são necessários.

Desconexão de um nó l_w

Como apresentado na Seção 1.5.2, um nó l_w é responsável por relatar ao nó r_d as condições de recepção de pacotes $p_x \in P$ em uma transmissão multicast e assim determinar a taxa de transmissão que deve ser utilizada para repassar o referido fluxo de dados. Sem os nós l_w, tal procedimento não seria possível. Sendo assim, deve-se realizar um procedimento para eleger um novo nó l_w quando um nó com tal responsabilidade solicite desconexão. São candidatos a nó l_w os nós c_f já recebendo o fluxo de dados P, e o nó l_w em procedimento de desconexão deve esperar que o procedimento de nova eleição seja concluído. Nesse interim, o nó l_w em processo de desconexão deve continuar enviando pacotes do tipo *GMTP-Ack* para o nó r_d.

Desconexão de um nó r_d

Um nó r_d realiza o procedimento de desconexão não por intervenção da aplicação, mas sim quando $C_i(r_d) = 0$ para um determinado fluxo de dados P. Neste caso, pode ocorrer uma situação crítica para todos os nós parceiros r_q de r_d, pois teoricamente estes não poderão mais receber os pacotes de dados $p_x \in P$. Para evitar um período de instabilidade na recepção

de P por parte dos nós parceiros de r_d, define-se no GMTP um parâmetro chamado de período de carência para novas parcerias (*grace period for new partnerships*). Trata-se de um parâmetro que determina o tempo que um nó r_d continuará repassando o fluxo de dados P para seus parceiros.

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

Falha de um nó r_d

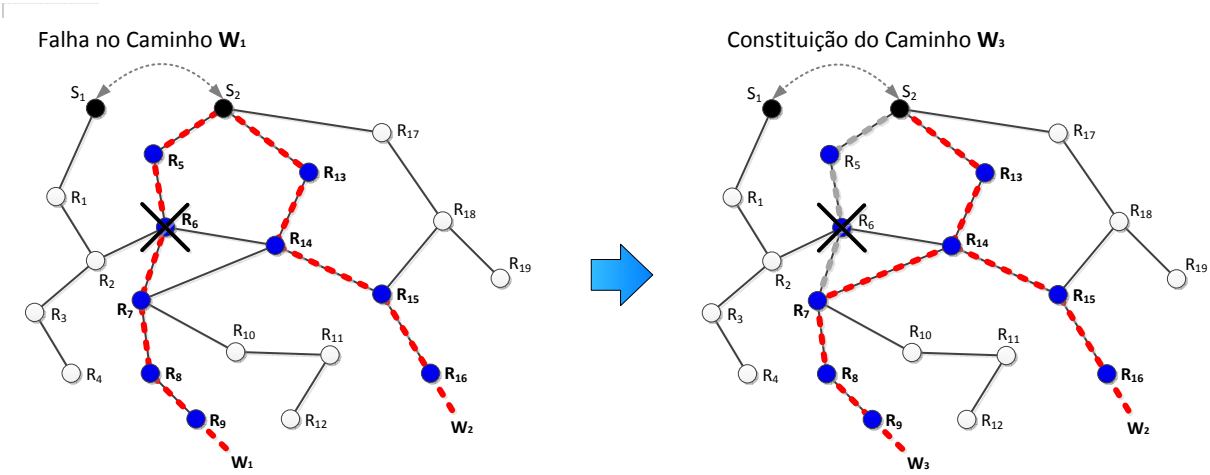

Figura 1.23: Cenário de falha do nó r_6 em um caminho W_1, seguida de constituição de um novo caminho W_3 formado pelo procedimento de formação de parceria intra W_v.

Além da desconexão explícita de um nó r_d, uma ação específica deve ser realizada se um nó $w_m \in W_v$ circunstancialmente falhar. Para tratar estes casos, definiu-se que o nó r_d pode formar parcerias com os próprios nós $r_d \in W_v^{\triangleleft}$, tal que $W_v^{\triangleleft} = \sim(\delta(w_{m+1}, W_v))$, através de uma outra rota de rede que também alcance o nó s_a – isto pode acontecer devido à execução de algoritmos de roteamento dinâmico *Intra-AS (Autonomous Systems)*, por exemplo, o OSPF, e *Inter-AS*, por exemplo, o BGP [22].

Para entender o comportamento do GMTP em caso de falha de um nó r_d, observe a Figura 1.23. Se o nó r_6 falhar e o nó r_{14} também estiver repassando P, uma nova parceria

é formada transparentemente entre o nó r_7 e o nó r_{14}. Isto porque o nó r_{14} interceptará os pacotes de controle de *keep-aline* que o nó r_7 está transmitindo para o nó s_a. Mas, pode acontecer o caso de que $\varphi(r_{14}, P) = 0$ e $\varphi(r_{13}, P) = 0$ para o fluxo de dados P, portanto o pedido de conexão enviado pelo nó r_9 alcançará o nó s_1 como antes. Isto resultará na constituição de um novo caminho $W_3 = W_1^{\lhd} \cup W_2^{\lhd}$, tal que $W_1^{\lhd} = \sim(\delta(r_6, W_1))$ e $W_2^{\lhd} = \delta(\sim(W_2), r_{15})$. Isto fará com que todo o caminho W_3 repasse o fluxo de dados P. Como consequência, aumenta-se a possibilidade de parcerias futuras com nós r_d cujo pedido de conexão para obter P seja roteado pelo caminho W_3. Para este caso, criou-se o procedimento de formação de parceria por intersecção de caminhos W_v, detalhado mais adiante.

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

1.7.2 Eleição de nós l_w

Para um fluxo de dados P, o primeiro nó l_w será o nó c_f que iniciar a primeira conexão unicast para obter o referido fluxo. Os seguintes nós l_w serão os próximos nós c_f que se conectar para receber o fluxo de dados P, até atingir um parâmetro que determinará a quantidade máxima de nós l_w por fluxo de dados P. Tal parâmetro pode ser determinado pelo administrador do nó r_d.

Sendo assim, à medida que um nó r_d recebe pacotes do tipo *GMTP-Request*, no pacote de resposta *GMTP-Response*, o nó r_d ativa um indicador sinalizando que o referido nó c_f em processo de conexão deverá se comportar como um nó l_w, passando a enviar relatórios da taxa de transmissão calculada por ele. Note que este modo de transmissão deve ser implementado com garantia de entrega, ou seja, com a confirmação de recepção de pacotes e retransmissão caso este tipo de pacote seja perdido. Assim, um nó r_d poderá ter controle sobre a quantidade de nós l_w e receber relatórios apenas dos nós $l_w \in L$.

Uma outra situação que se faz necessária a eleição de nós l_w é no procedimento de desconexão, como explicado na Seção 1.7.1. Para esse caso, quando o nó r_d receber o pacote do tipo *GMTP-Close*, este deve verificar se o referido nó c_f é um nó l_w. Em caso afirmativo, o nó r_d deve transmitir para um dos nós c_f que também recebe o referido fluxo de dados P (se

houver), um pacote do tipo *GMTP-Elect-Request* e aguardar por um *GMTP-Elect-Response*. Este procedimento deve ocorrer com garantia de entrega.

1.8 Sumário do Capítulo

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

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

Diversas estratégias adotadas no GMTP e apresentadas neste capítulo discutidas são diferenciais que permitem a disseminação mais rapidamente de um determinado fluxo de dados

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

No GMTP, os responsáveis por formar as parcerias P2P são os nós repassadores e não mais os nós clientes, como em soluções tradicionais de distribuição de conteúdo P2P. Como consequência, melhora-se o desempenho das transmissões de conteúdos multimídia ao vivo, pois o GMTP não é influenciado por fatores que impactam negativamente no funcionamento da rede P2P, tais como a capacidade de processamento, armazenamento (memória), mobilidade e dinâmica de conexão/desconexão (*churn*) dos nós clientes. Esses dois últimos fatores são mais críticos se comparados aos demais, principalmente com a popularização dos dispositivos móveis e usar esse tipo de cliente para compartilhar seus recursos em uma rede P2P não é apropriado.

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

transmissão que o roteador deverá utilizar para distribuir o conteúdo multimídia para os nós clientes diretamente conectados a ele.

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

Bibliografia

[1] TBE. Tbe, 3 2008.

[2] S. Bradner. Key words for use in rfcs to indicate requirement levels, 3 1997. `http://www.ietf.org/rfc/rfc2119.txt`. Último acesso: 24 de Janeiro de 2014.

[3] D. Meyer. Administratively scoped ip multicast, 7 1998. `http://www.ietf.org/rfc/rfc2365.txt`. Último acesso: 24 de Janeiro de 2014.

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

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

[6] 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.

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

[8] R. Braden. Requirements for Internet Hosts - Communication Layers, 10 1989. Último acesso: 24 de Janeiro de 2014.

[9] L. Eggert and F. Gont. TCP User Timeout Option, 3 2009. `http://www.ietf.org/rfc/rfc5482.txt`. Último acesso: 24 de Janeiro de 2014.

[10] 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: 24 de Janeiro de 2014.

[11] 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: 24 de Janeiro de 2014.

[12] M. Handley and V. Jacobson. Sdp: Session description protocol, 4 1998. http://www.ietf.org/rfc/rfc2327.txt. Último acesso: 24 de Janeiro de 2014.

[13] H. Alvestrand. Tags for the identification of languages, 2 1995. http://www.ietf.org/rfc/rfc1766.txt. Último acesso: 24 de Janeiro de 2014.

[14] Nandita Dukkipati. *Rate control protocol (rcp): congestion control to make flows complete quickly*. PhD thesis, Stanford, CA, USA, 2008. AAI3292347.

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

[17] 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: 24 de Janeiro de 2014.

[18] 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.

[19] 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: 24 de Janeiro de 2014.

[20] 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.

[21] 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.

[22] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Addison Wesley, trad. 3 ed. edition, 2006.