UNIVERSIDADE FEDERAL DE ALAGOAS

INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

WENDELL SILVA SOARES

Aplicabilidade e Desempenho do Global Media Transmission Protocol (GMTP)

Maceió
2016

Wendell Silva Soares

Aplicabilidade e Desempenho do Global Media Transmission Protocol (GMTP)

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas, como requisito parcial para obtenção do grau de Mestre em Informática

Orientador: Prof. Dr. Leandro Melo de Sales-
Coorientador: Prof. Dr. André Lage Freitas

Maceió
2016

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecário Responsável: Valter dos Santos Andrade

S676a Soares, Wendell Silva.
Aplicabilidade e desempenho do Global Media Transmission Protocol (GMTP) / Wendell Silva Soares. – 2016.
126 f. : il.

Orientador: Leandro Melo de Sales.
Coorientador: André Lage Freitas.
Dissertação (Mestrado em Informática) – Universidade Federal de Alagoas. Instituto de Computação. Programa de Pós-Graduação em Informática. Maceió, 2016.

Bibliografia: f. 112-114.
Apêndice: f. 115-122.
Anexos: f.123-126.

1. GMTP (Protocolo de rede de computação). 2. Redes P2P. 3. Redes multimídia. I. Título.

CDU: 004.738.5.057.4

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PpgI
Instituto de Computação
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

Membros da Comissão Julgadora da Dissertação de Mestrado de Wendell Silva Soares, intitulada: "*Aplicabilidade e Desempenho do Global Media Transmission Protocol (GMTP)*", apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas em 01 de abril de 2016, às 09h00min, no Miniauditório do Instituto de Computação da UFAL.

COMISSÃO JULGADORA

Prof. Dr. Leandro Melo de Sales
UFAL – Instituto de Computação
Orientador

Prof. Dr. André Lage Freitas
UFAL – Instituto de Computação
Coorientador

Prof. Dr. Heitor Soares Ramos Filho
UFAL – Instituto de Computação
Examinador

Prof. Dr. Marco Aurélio Spohn
Universidade Federal da Fronteira Sul – UFFS
Examinador

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Juvenil e Erinete, pelo amor e dedicação que me concederam. Sei que sem o sacrifício, dedicação e esforço de vocês eu não seria ninguém. Pai, agradeço especialmente pelas suas orações e pelo seu amor. Mãe, agradeço especialmente por sua dedicação e seu amor. Agradeço aos meus irmãos, Whanderson, Wevillyn, Wemilly e Wemerson, que amo tanto. Agradeço também a todos os meus familiares.

À minha esposa Gilmara, pelo seu apoio, pelo seu carinho, por sua paciência (ou mesmo a falta dela), mesmo quando eu passava as noites no computador ao invés de estar ao seu lado. Não tenho palavras para descrever minha gratidão e meu amor por você.

Agradeço aos meus colegas da Pró-Reitoria de Graduação da UFAL - PROGRAD, por me proporcionarem um ambiente de trabalho extremamente agradável e, acima de tudo, pela amizade cultivada durante esses anos. Em especial, agradeço a Alexandre Marques, Felipe Sarmento e Thiago Prudente, por me apoiarem durante todo a minha jornada profissional na UFAL e, especialmente, quando precisei me afastar das atividades profissionais para me dedicar ao mestrado.

Agradeço também aos meus professores no mestrado, pela dedicação ao curso e aos seus alunos. Em especial, agradeço a meu orientador, Leandro Sales, pela compreensão, paciência, comprometimento e orientação, numa parceria que vem desde a graduação, sendo fundamental para meu crescimento acadêmico. Agradeço também ao meu coorientador, André Lage, pela inestimável colaboração para o meu trabalho e sua contribuição para meu desenvolvimento acadêmico.

Agradeço aos meus colegas do curso, em especial àqueles que foram meus companheiros de grupo nas diversas avaliações pelas quais passei e me ajudaram a sobreviver a algumas matérias. Agradeço também a todos os meus amigos e colegas do COMPE/UFAL. Por fim, agradeço especialmente aos meus colegas orientandos do PIBITI: Joilnen Leite e Mário André. A ajuda de vocês foi imprescindível para que este trabalho fosse concluído. Certamente este trabalho não teria sido concluído sem a ajuda de todas essas pessoas.

RESUMO

Neste trabalho, apresenta-se um estudo do Global Media Transmission Protocol (GMTP), identificando-se limitações e propondo-se melhorias, através da sua implementação no sistema operacional Linux. As aplicações para transmissão de mídias ao vivo na Internet atuais dependem de vários *middlewares* incompatíveis entre si, projetados para prover serviços de rede específicos dessa classe de aplicações, cada um focado em um determinado requisito da aplicação. Como consequência, não há a possibilidade de que dois ou mais nós cooperem entre si quando estão conectados através de aplicações distintas, mas interessados em obter o mesmo fluxo de dados a partir de um servidor, resultando no aumento do consumo de recursos de rede. Por esse motivo foi proposto o GMTP, um protocolo de distribuição de mídias ao vivo pela Internet que desacopla a forma que o conteúdo é reproduzido pela aplicação da forma que tal conteúdo é transportado através da rede. Isto ocorre com a utilização de algoritmos P2P em nível de *socket* a fim de construir uma rede de favores entre roteadores. As parcerias entre os roteadores são determinadas pelos nós servidores, conforme medições das capacidades de transmissão dos canais já em uso para disseminar o conteúdo, obtidas por meio de um algoritmo de controle de congestionamento assistido pela rede. Neste trabalho, propõem-se melhorias no GMTP através de sua implementação no núcleo do Linux. Avaliou-se a implementação do GMTP no Linux por meio de experimentos executados em máquinas virtuais. Através dos experimentos, constatou-se que a abordagem do protocolo GMTP, acrescido das modificações propostas neste trabalho, é viável e praticável para implantação em um sistema operacional e para uso em aplicações de rede. Na perspectiva de desempenho, avaliou-se o GMTP-Linux segundo métricas de qualidade de serviço e constatou-se que tal implementação tem um desempenho compatível com os resultados apresentados no trabalho que define a especificação do GMTP.

Palavras-chaves: GMTP. redes P2P. redes multimídia.

ABSTRACT

This work presents a study about the implementation of the Global Media Transmission Protocol (GMTP) in the Linux operating system. Currently, Internet live streaming applications require various middlewares incompatible among them, as they have different network services requirements. As a consequence, two end-systems using distinct applications can not cooperate to share a given data flow from a streaming server, increasing network resource consumption. For that reason, the GMTP was proposed as a network protocol that decouples the functions used for rendering the media flow to the user from the functions used for disseminating the data through the network. The GMTP protocol uses P2P algorithms at the socket level in order to build a network of favors among cooperative routers. The network of favors is managed by streaming servers, which orchestrate router partnerships based on the capacity of each router transmission channel. The capacity of transmission channels are obtained through network assisted congestion control algorithms. We evaluated the correctness of GMTP-Linux based on experiments in virtual machines. The experiments show that GMTP approach can be used to deploy it in a operating system and can be used in network applications for transmitting live streaming content over the network. In the performance perspective, GMTP-Linux was studied according to some quality of service metrics and it shown that GMTP achieved a performance similar to the results presented in the definition work of GMTP.

Keywords: multimedia networks. P2P. GMTP.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

3WHS	*Three Way Hand Shake*
ALTO	*Application-Layer Traffic Optimization*
ALM	*Application Layer Multicast*
API	*Application Programming Interface*
BSD	*Berkley Software Distribution*
CCID	*Congestion Control IDentifier*
CCN	*Content Centric Networks*
DASH	*Dynamic Adaptive Streaming over HTTP*
DCCP	*Datagram Congestion Control Protocol*
ECN	*Explicit Congestion Notification*
GMTP	*Global Media Transport Protocol*
HLS	*HTTP Live Streaming*
HDS	*HTTP Dynamic Streaming*
IANA	*Internet Assigned Numbers Authority*
ICN	*Information Centric Networks*
IETF	*Internet Engineering Task Force*
IGMP	*Internet Group Management Protocol*
IP	*Internet Protocol*
NDN	*Named-Data Networks*
POSIX	*Portable Operating System Interface*
RCP	*Rate Control Protocol*
RTO	*Retransmission Timeout*
RTT	*Round Trip Time*

SCTP	*Stream Control Transmission Protocol*
TCP	*Transport Control Protocol*
TFMCC	*TCP Friendly Multicast Congestion Control*
TFRC	*TCP Friendly Rate Control*
UDP	*User Datagram Protocol*

SUMÁRIO

1 INTRODUÇÃO

A demanda dos usuários por conteúdo multimídia na Internet tem crescido a cada ano. O crescimento das aplicações multimídia da Internet é resultado do aumento da penetração do acesso residencial à banda larga e do acesso sem fio de alta velocidade. Essa popularização resulta em um aumento da demanda para os sistemas de distribuição de mídias ao vivo, elevando-se os custos com largura de banda dos distribuidores de conteúdo. No cenário atual, os desafios relacionados à escalabilidade, custos econômicos e do nível de satisfação do usuário ao consumir mídias ao vivo através da Internet ainda necessitam ser resolvidos.Por isto, faz-se necessário pesquisar e propor novas soluções para utilizar eficientemente as redes de computadores a fim de distribuir mídias ao vivo através da Internet, em escala e com qualidade.

Atualmente, os sistemas para transmissão de mídias ao vivo na Internet executam diversas funções que foram implementadas para suprir a ausência de serviços de rede para este fim. A disseminação de diferentes sistemas e protocolos com este propósito tem levado à pulverização de soluções para transporte de dados multimídia, gerando uma falta de padronização na forma como tais sistemas transportam seus dados na Internet, principalmente porque tais soluções são implementadas na camada de aplicação (LIU et al., 2008). Como consequência, não há a possibilidade de que dois ou mais nós, conectados em sistemas distintos, cooperem entre si, compartilhando o mesmo fluxo de dados transmitido por um servidor. Idealmente, os nós deveriam compartilhar o mesmo fluxo de dados e cada um apresentar o conteúdo da forma definida pela aplicação, desacoplando a forma de transportar os dados da forma como estes são apresentados pela aplicação aos seus respectivos usuários (SALES, 2014).

Nesse contexto, Sales (2014) propôs um protocolo de rede denominado *Global Media Transmission Protocol* (GMTP), que considera uma arquitetura híbrida P2P/CDN para distribuição de mídias ao vivo com base na constituição de uma rede de favores entre roteadores a fim de entregar o conteúdo multimídia de interesse comum aos seus usuários. As parcerias entre os roteadores são formadas com base na capacidade dos canais de transmissão já em uso, dado um fluxo multimídia de interesse. Nesse contexto, os clientes não recebem os pacotes de dados diretamente dos servidores, mas sim dos roteadores já envolvidos na transmissão da mídia, evitando-se múltiplas conexões ao servidor sempre que possível.

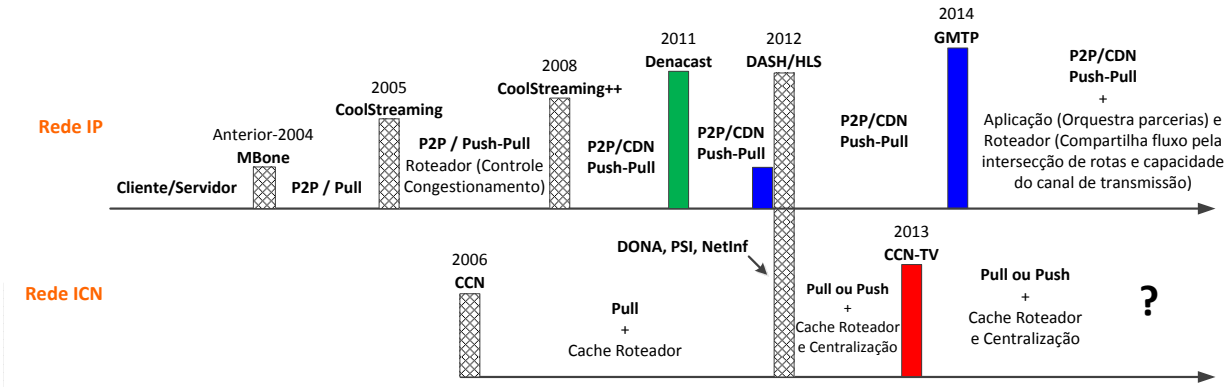
Essa solução difere da simples utilização do *multicast*, pois no GMTP os roteadores são capazes de manter estado sobre quais mídias estão sendo repassadas. Desta forma, os roteadores participam ativamente da distribuição do conteúdo multimídia e os repasses

são configurados dinamicamente, seguindo diversas estratégias a serem discutidas mais adiante neste documento.

A solução proposta no GMTP é inspirada nos recentes avanços de novas arquiteturas de rede, das quais destacam-se as Redes Centradas na Informação (*Information Centric Networks* – ICN) (XYLOMENOS et al., 2014). O objetivo das ICNs é facilitar a distribuição de conteúdo através do desacoplamento entre a fonte e o conteúdo. Nesse contexto, definem-se rotas, funções de transporte e decisões de repasse com base no nome do conteúdo a ser transmitido. Assim, um roteador pode fazer *cache* de pacotes de dados muito requisitados para retorná-los quando necessário, sem a necessidade de consultar o servidor. Essa estratégia pode melhorar sobremaneira o desempenho de aplicações de mídias ao vivo na Internet.

Na Figura 1, ilustram-se os principais representantes e marcos no contexto de distribuição de mídias ao vivo, bem como a evolução das arquiteturas e modelos de serviços adotados na Internet. Atualmente, existem duas vertentes para transmissão de dados na Internet: as Redes IP e as Redes Centradas na Informação. Para as soluções disponíveis nas redes IP, destacam-se as propostas DONet/CoolStreaming (ZHANG et al., 2005), Denacast (JAHROMI, 2012; SEYYEDI; AKBARI, 2011) e DASH (SODAGAR, 2011). Já para as Redes ICN, destaca-se a CCN (SEVERANCE, 2013), representada pelo sistema CCN-TV (CIANCAGLINI et al., 2013). Até 2004, predominaram soluções de transmissão de mídias ao vivo baseado no modelo de serviço cliente/servidor, com os sistemas sendo capazes de suportar algumas dezenas de usuários por sessão. Em seguida, com base nas pesquisas sobre compartilhamento de arquivos considerando o modelo de serviço P2P, estenderam-se tal modelo de serviço para distribuição de mídias ao vivo (SALES, 2014).

Figura 1 – Representantes e marcos importantes no contexto de distribuição de mídias ao vivo. Evolução das arquiteturas e modelos de serviços adotados na Internet.



Fonte – Sales (2014)

O GMTP utiliza redes IP para transmissão de dados na Internet, embora inspire-se

em algumas características das ICNs. No GMTP, nomeia-se o conteúdo multimídia a fim de entregá-lo aos seus usuários através da participação ativa dos roteadores, de forma que a fonte do conteúdo multimídia torna-se transparente à aplicação. Logo, no GMTP, considera-se um cenário onde os roteadores serão projetados com maior capacidade de processamento, oferecendo suporte a ICNs e visando a sua participação ativa na distribuição de conteúdo. Antes do GMTP, todas as soluções têm uma forte dependência dos sistemas finais para executar suas funções. No GMTP, os sistemas finais têm um papel coadjuvante na execução das funções, realizando-se parcerias entre os roteadores da rede (SALES, 2014).

1.1 Problemática

O GMTP (SALES, 2014) foi concebido a partir de evidências sólidas com base nos trabalhos anteriormente publicados e em simulações de rede, sendo uma solução discutida por meio de fundamentos matemáticos e testes com o uso de um simulador de rede. Pelos resultados das simulações, sugere-se que o protocolo é estável e eficiente para utilização em redes multimídia na Internet.

Contudo, visando a aplicabilidade do GMTP em grande escala (ex. Internet), faz-se necessário estudá-lo em uma instância de implementação mais próxima da realidade, ou seja, em um sistema operacional e com possibilidade de execução em uma rede real. Isto porque, quando se propõe um novo protocolo de rede, funções importantes podem se mostrar necessárias somente quando se realiza uma implementação possível de ser executada e implantada em um cenário de rede. Além disto, é igualmente importante destacar que uma das políticas da IETF para aprovação de um novo protocolo de Internet é que exista pelo menos duas implementações interoperáveis de tal protocolo (BRADNER, 1996).

A implementação do GMTP em um sistema operacional apresenta desafios, como por exemplo, a definição de escolha dos tipos de dados (variáveis de implementação, cabeçalhos do protocolo), que podem afetar a precisão e a dinâmica do protocolo. Este problema pode se agravar quando se trata da possibilidade de se ter o GMTP funcionando em múltiplos sistemas operacionais, cada um com diferentes formas de lidar com questões relacionadas ao uso de temporizadores e *threads*, bem como diferentes implementações da pilha TCP/IP. Nesse contexto, as funções definidas originalmente no GMTP podem não ser completamente compatíveis em determinados sistemas operacionais, necessitando ser adaptadas de acordo com a API disponível. Por exemplo, os algoritmos de controle de congestionamento do GMTP são compostos de alguns passos que dependem de operações em ponto-flutuante. Neste ponto, ainda não está claro como as soluções que visam contornar a ausência desse tipo de operação no núcleo do Linux afetariam a precisão e desempenho do

GMTP nesse sistema. Além disso, os resultados das simulações apresentadas em (SALES, 2014) não revelam como o desempenho do GMTP pode ser afetado por fatores do ambiente computacional, incluindo configurações do TCP/IP, compartilhamento de *link*, perdas devido a congestionamento e a presença de fluxos de outros protocolos, como TCP e UDP. É importante salientar que na Internet, o *feedback* de implementações reais de protocolos é mais importante que qualquer princípio arquitetural (CARPENTER, 1996).

Nesse contexto, a implementação do protocolo de rede, assim como o conjunto de ferramentas desenvolvidas, são necessários a fim de demonstrar se a abordagem do protocolo GMTP é viável e praticável para uso das aplicações de rede. Assim, pelo exposto, entende-se que é necessário testar o GMTP através da criação de aplicações, com o intuito de identificar eventuais problemas e avaliar possíveis correções e melhorias. Deve-se também realizar testes em diferentes cenários típicos de rede, considerando-se variação de largura de banda, latência e perda de pacotes.

1.2 Objetivos

O objetivo deste trabalho é implementar o GMTP no sistema operacional Linux, identificando-se limitações e propondo-se melhorias.

1.2.1 Objetivos específicos

Neste trabalho, tem-se os seguintes objetivos específicos:

1. Validar a especificação do GMTP no que diz respeito a sua implementação.
2. Realizar testes de desempenho sobre a referida implementação em diferentes cenários de rede.

1.3 Metodologia

Neste trabalho, definiu-se a seguinte metodologia:

1. Efetuar um estudo da API de desenvolvimento de novos protocolos de transporte na pilha de rede do sistema operacional Linux.
2. Efetuar um estudo das funções do protocolo GMTP e registrar quais funções serão implementadas tal como especificadas e quais serão adaptadas de acordo com as limitações do sistema operacional Linux ou das aplicações.
3. Implementar e configurar o GMTP no núcleo do Linux. A implementação deve prover uma abstração para a camada de aplicação de modo que os processos em execução

utilizem o referido protocolo, através de uma API compatível com as especificações de *socket* BSD e POSIX, facilitando o uso do GMTP nos atuais e futuros sistemas.

4. Validar a especificação do GMTP no que diz respeito a sua implementação. Para isto, deve-se implementar o GMTP sem utilizar como base nenhuma outra implementação existente.

5. Realizar testes básicos do acesso aos recursos do GMTP pela API de *socket* BSD e POSIX, através de aplicações de rede, utilizando-se máquinas virtuais.

6. Realizar testes de desempenho sobre a referida implementação em diferentes cenários de rede, considerando o acoplamento da implementação do GMTP no *Kernel* do Linux em um simulador de rede.

1.4 Relevância

A distribuição de áudio/vídeo ao vivo para muitos receptores é um tema relevante no contexto de redes de computadores devido ao grande crescimento de interesse dos usuários por esse serviço, em particular, na Internet. Nesse contexto, o projeto do GMTP suscita que um conjunto mínimo de serviços, quando organizado de forma mais apropriada, promove melhorias substanciais no projeto das aplicações para distribuição de áudio e vídeo ao vivo, otimizando o uso dos recursos de rede através da interoperabilidade entre os sistemas.

Deste modo, faz-se necessário estudar a aplicabilidade do GMTP em um sistema operacional e com possibilidade de execução em uma rede real. A implementação do protocolo de rede, assim como o conjunto de ferramentas desenvolvidas, tem como objetivo demonstrar se a abordagem do protocolo GMTP é viável e praticável para uso das aplicações. O compromisso com a utilização dos conceitos para construir soluções que possam ser aplicadas na indústria, torna o trabalho relevante em termos práticos, sobretudo em escala global na Internet.

1.5 Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

- Na Seção 2, apresentam-se os principais conceitos relacionados aos sistemas de distribuição de mídias ao vivo na Internet, com ênfase no GMTP.

- Na Seção 3, apresenta-se um estudo da aplicabilidade do GMTP, através da sua implementação no sistema operacional Linux, com a identificação das suas limitações e propostas de melhorias.

- Na Seção 4, apresentam-se os resultados sobre o desempenho da implementação do protocolo GMTP.
- Na Seção 5, apresentam-se as considerações finais, discutindo-se os principais tópicos elencados neste trabalho, conclusões e trabalhos futuros.
- No Apêndice A, apresentam-se alguns aplicações exemplo que utilizam o GMTP como protocolo de transporte.
- No Apêndice B, apresentam-se todas as opções de *socket* disponíveis para utilização no GMTP.
- No Apêndice C, apresentam-se detalhes a respeito dos experimentos realizados no contexto deste trabalho.
- No Anexo A, apresentam-se detalhes das estruturas que representam *sockets* no núcleo do Linux.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, apresentam-se as principais estruturas adotadas nos sistemas de distribuição de mídias ao vivo e as funções básicas empregadas nesses sistemas, com ênfase no funcionamento do GMTP.

2.1 Distribuição de Mídia ao Vivo através da Internet

Em um sistema de distribuição multimídia, uma aplicação cliente reproduz o conteúdo multimídia ao usuário final à medida que recebe pacotes de dados contendo partes da mídia, de um ou mais servidores.

A transmissão de áudio e vídeo na Internet é classificada em duas grandes categorias: áudio e vídeos pré-armazenados, enviados sob demanda, e áudio e vídeos ao vivo. Na primeira categoria, clientes têm a flexibilidade de requisitar, no momento que for conveniente, arquivos de áudio e vídeo comprimidos que estão armazenados em servidores. Por outro lado, um conteúdo ao vivo é transmitido no mesmo momento em que o fluxo é gerado. Essa classe de aplicação é semelhante à transmissão tradicional de rádio e televisão, exceto que a transmissão é realizada pela Internet. Logo, todas as aplicações devem estar sincronizadas e devem reproduzir o fluxo de vídeo ao mesmo tempo. Isto significa que esses sistemas têm requisitos críticos de tempo e a retransmissão baseada no controle de erro não é adequado em cenários de altos atrasos. Por isso, tais sistemas utilizam suas próprias funções para tolerar perda de pacotes, como mascarar erros e adaptação de conteúdo, em transmissões de fluxos de dados sem garantia de entrega, em geral utilizando UDP.

A abordagem mais simples para o envio do fluxo de áudio e vídeo na Internet é a utilização do modelo cliente-servidor. Nesse modelo, há um único servidor de áudio e vídeo e cada cliente cria uma conexão com o servidor, que por sua vez envia o conteúdo diretamente ao cliente. Existem algumas variantes deste modelo, mas as soluções baseadas em cliente-servidor demandam uma larga capacidade de transmissão nos canais de comunicação utilizados pelo servidor, o que gera um alto custo operacional (LIU; GUO; LIANG, 2008).

A utilização de redes de distribuição de conteúdo (*content distribution networks* — CDNs) pode amenizar a demanda para o servidor. Em um cenário onde utiliza-se a CDN, entrega-se o conteúdo para servidores CDN localizados em regiões geográficas diferentes a fim de reduzir a carga na fonte do fluxo de áudio e vídeo. A implementação de uma rede CDN é, em geral, uma estrutura de servidores que se comunicam com um servidor gerador do conteúdo multimídia a ser distribuído. Quando um cliente solicita um fluxo de áudio e vídeo, o servidor CDN mais próximo entregará o conteúdo solicitado fazendo-se passar

como a fonte do fluxo de mídia. Essa arquitetura tem como característica alta estabilidade, mas fatores econômicos como custo de *hardware* e limitações de escalabilidade dificultam sua adoção (MESKOVIC; BAJRIC; KOS, 2012).

Outra maneira eficiente de se estruturar e enviar um fluxo de áudio e vídeo a um grupo de usuários na Internet seria a utilização de *multicast* no nível de IP. O problema é que manter uma estrutura de IP *multicast* requer configurações específicas da rede, o que exige intervenção humana e combinação de diferentes domínios administrativos (SALES, 2014; LIU; GUO; LIANG, 2008). Para contornar a falta de suporte de *multicast* no nível de IP, implementa-se a função equivalente no nível da camada de aplicação. Os servidores de vídeo e os clientes formam uma rede sobreposta à rede real e, assim, organizam-se para distribuir o fluxo de vídeo através da estratégia entre pares (P2P). Neste tipo de sistema, as aplicações são encorajadas a atuarem como clientes e servidores (SALES, 2014).

A distribuição de áudio/vídeo ao vivo é comumente realizada por meio de *multicast* na camada de aplicação (utilizando P2P, CDN ou uma solução híbrida P2P/CDN) ou através de múltiplos fluxos *unicast* separados (KUROSE, 2010). No estado da prática, diversos sistemas definem uma arquitetura de rede híbrida P2P/CDN, ou seja, par-a-par e cliente-servidor, com suporte de uma rede de distribuição de conteúdos (MESKOVIC; BAJRIC; KOS, 2012). Como resultado, consegue-se escalabilidade do número de usuários e redução de custos com infraestrutura de rede, por meio das redes P2P; ao passo que facilita-se o gerenciamento e obtém-se maior estabilidade de disponibilização dos serviços, por meio das CDNs.

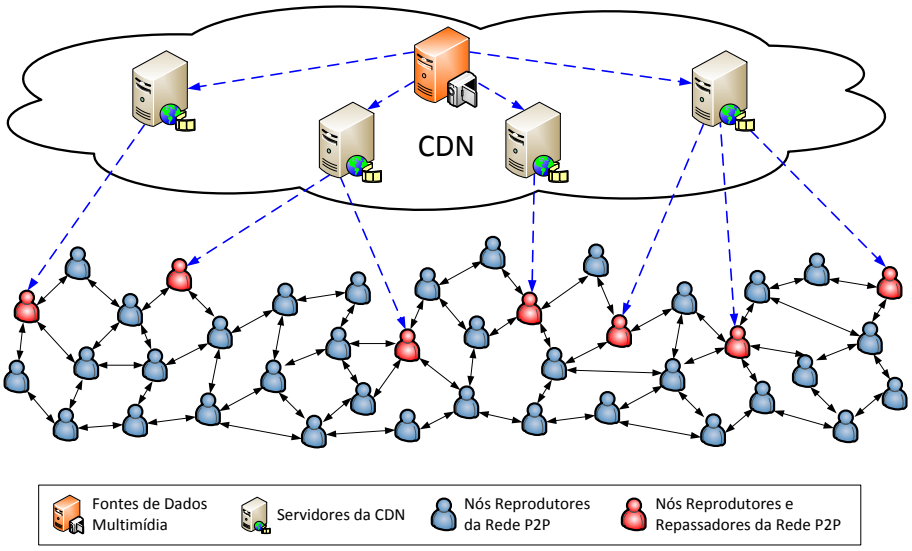
Conforme ilustra-se na Figura 2, nos sistemas que definem uma arquitetura híbrida P2P/CDN, um servidor gerador de datagramas multimídia transmite-os para um conjunto de servidores espalhados geograficamente em diferentes *backbones* que formam a CDN. Com o suporte dos nós constituindo uma rede P2P, os servidores da CDN enviam os datagramas para os nós da rede P2P, que os disseminam para outros nós interessados pelo mesmo conteúdo à medida que novas partes (*chunks*) do conteúdo são geradas e obtidas. Nesse tipo de arquitetura de rede, os servidores da CDN atuam como super nós para a rede P2P, ao passo que os nós da rede P2P cooperam entre si a fim de disseminar mais rapidamente os datagramas, reproduzindo-os também na rede local (SALES, 2014).

2.2 Global Media Transmission Protocol (GMTP)

O *Global Media Transmission Protocol* (GMTP) (SALES, 2014) é um protocolo projetado para operar na Internet em sistemas de distribuição de mídias ao vivo. Com o GMTP, busca-se otimizar a distribuição de conteúdos multimídia ao vivo em larga escala, abstraindo-se a complexidade e promovendo-se a interoperabilidade entre os sistemas.

O GMTP é um protocolo *cross-layer* atuando nas camadas de transporte e de rede

Figura 2 – Estrutura de rede de uma arquitetura híbrida P2P/CDN para distribuição de conteúdo multimídia. Nesse tipo de arquitetura, os nós finais podem, além de reproduzir o conteúdo multimídia, atuar como repassadores para ouros nós parceiros interessados pelo mesmo conteúdo.



Fonte – Sales (2014)

da pilha TCP/IP. Trata-se de um protocolo baseado em uma arquitetura de rede híbrida P2P/CDN, através da qual transmitem-se e compartilham-se pacotes de um ou mais sistemas independente do controle da aplicação. No GMTP, constituem-se redes de favores formadas por roteadores de rede, que cooperam entre si a fim de entregar o conteúdo multimídia de interesse comum aos seus clientes. Nesse cenário, os servidores atuam como super nós para os nós da rede P2P, orquestrando as parcerias entre os roteadores da rede. Uma aplicação cliente reproduz o conteúdo multimídia ao usuário final à medida que recebe os pacotes de dados contendo partes da mídia, gerada por um ou mais servidores. Os clientes não necessariamente recebem os pacotes de dados diretamente dos servidores, mas podem recebê-los de roteadores já envolvidos na transmissão da mídia.

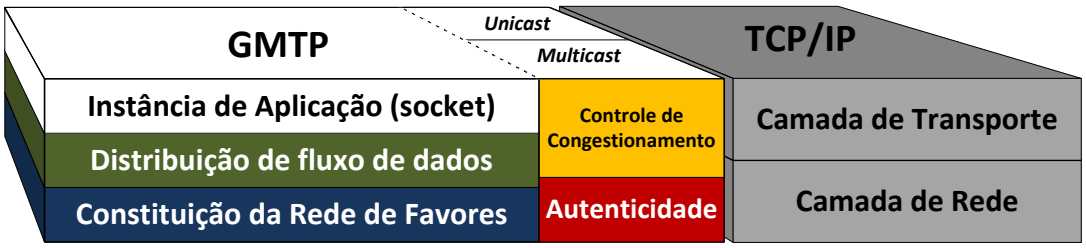
Os servidores instruem os roteadores a efetivarem as parcerias com outros roteadores que possuem clientes também interessados no mesmo conteúdo multimídia. As parcerias entre os roteadores são formadas com base na capacidade dos canais de transmissão atualizadas periodicamente por meio de uma versão adaptada do algoritmo controle de congestionamento assistido pela rede chamado *Rate Control Protocol* (RCP) (DUKKIPATI, 2007).

A transmissão de dados entre os nós GMTP ocorre através do envio de recebimento de partes de uma mídia. Os servidores transmitem os dados para os roteadores em modo *unicast*, que transmitem para os seus clientes na rede local em modo *multicast*. Em ambos os modos de transmissão, realiza-se controle de congestionamento sem garantia de entrega.

O princípio básico do GMTP é a introdução do conceito de *socket* P2P, onde a aplicação envia um pedido de conexão ao servidor, similar ao que ocorre com protocolos de transporte tradicionais. A diferença é que, no GMTP, se um roteador presente no trajeto de um fluxo de dados entre um cliente requisitante e o servidor já estiver repassando a mídia requisitada, este pode interceptar o referido pacote de requisição e responder ao cliente, como se fosse o próprio servidor.

O modo de funcionamento do protocolo GMTP é composto de quatro etapas básicas, conforme ilustra-se na Figura 3:

Figura 3 – Blocos funcionais do GMTP e suas relações com a pilha de protocolos TCP/IP.



Fonte – Sales (2014)

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

2. **Distribuição de fluxos de dados através de uma camada de *socket***: conectar os clientes interessados em receber um fluxo de dados de um evento ao vivo, bem como transmitir tal fluxo de dados através das redes de favores;

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

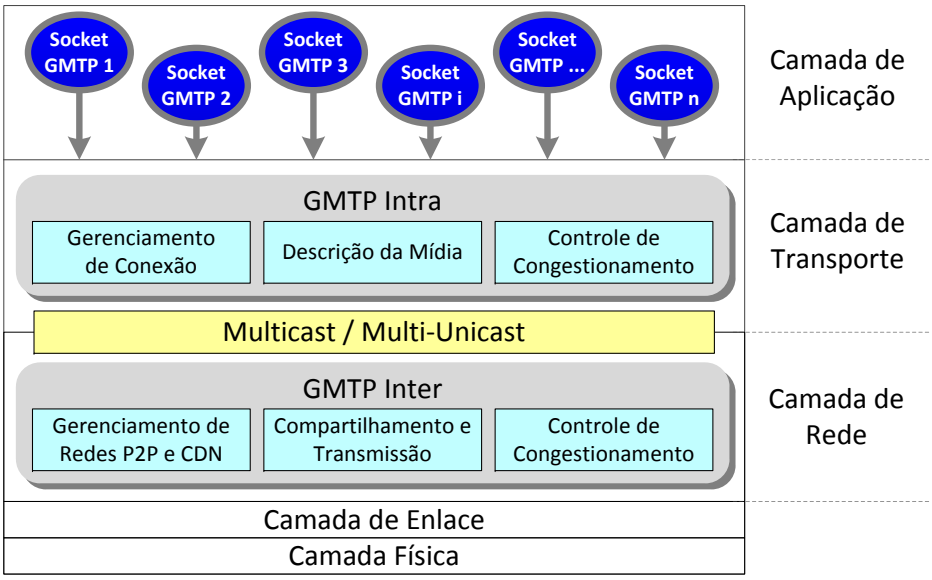
4. **Autenticidade do conteúdo**: verificar a autenticidade do fluxo de dados antes de repassá-los aos clientes, evitando-se ataques de poluição.

2.2.1 Visão Geral

O protocolo GMTP é composto por dois módulos chamados de *GMTP-Intra* e *GMTP-Inter*, que operam nas camadas de transporte e de rede, respectivamente, definindo assim sua arquitetura, ilustrada na Figura 4.

A principal responsabilidade do GMTP-Intra é fornecer serviços às aplicações de rede a fim de abstrair a complexidade na execução de tarefas comuns a qualquer sistema

Figura 4 – Arquitetura do Protocolo GMTP.



Fonte – Sales (2014)

final, tais como conexão multi-ponto, multiplexação/demultiplexação de datagramas IP entre as camadas de transporte/rede/aplicação e controle de congestionamento. Este módulo compreende a instância do GMTP em execução no sistema operacional do cliente ou do servidor, acessível através de uma API de *socket* GMTP.

Já a responsabilidade do GMTP-Inter é de constituir as redes de favores P2P compostas por roteadores, os quais funcionam como pontes de acesso aos servidores de uma rede CDN. Trata-se do módulo em execução nos roteadores, que cooperam entre si a fim de constituírem as redes de favores, com base no interesse comum por um determinado conteúdo, aceitando conexões originadas por seus clientes, bem como instruções dos servidores sobre formar parcerias com outros roteadores a fim de disseminar um determinado conteúdo (SALES, 2014).

No contexto dos tipos de nós do GMTP, definem-se 4 tipos, definidos a seguir:

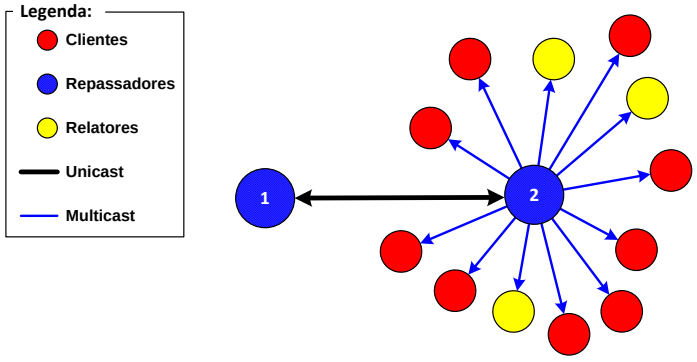
1. **Cliente**: sistema final capaz de reproduzir e gerar conteúdos multimídia ao vivo, executa um processo em nível de sistema operacional, representando uma aplicação manipulada pelo usuário final.

2. **Relator**: um *cliente* com habilidades de enviar relatórios periódicos ao nó *repassador* sobre o estado da transmissão.

3. **Repassador**: roteadores que participam efetivamente da rede de favores, com a responsabilidade de repassar os datagramas, originados em um ou mais *servidores*, para outros *repassadores* até que os datagramas alcancem os *clientes*.

4. **Servidor**: sistema final que participa de uma rede CDN e obtém a mídia a ser transmitida.

Figura 5 – Tipos de nós e modos de conexões do GMTP.



Fonte – Sales (2014)

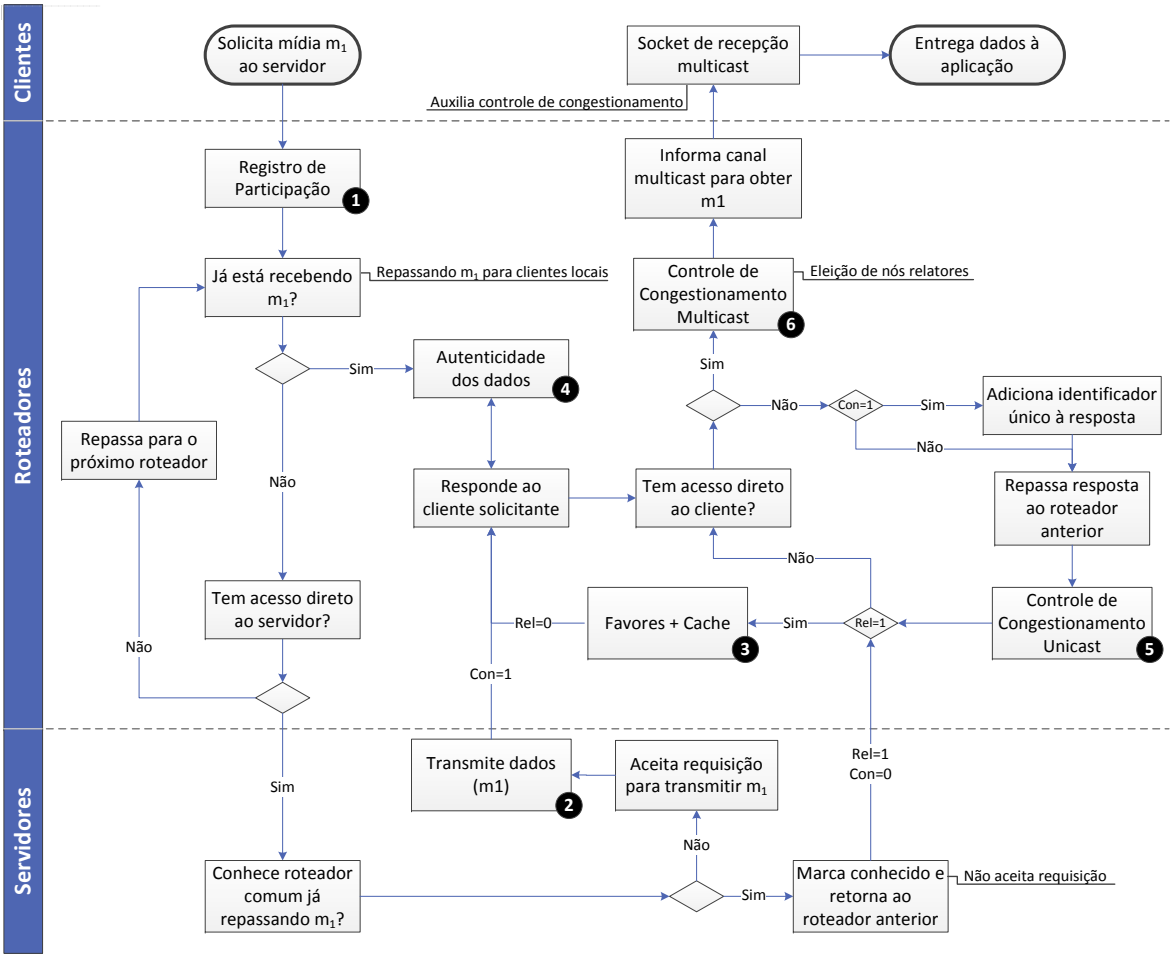
2.2.1.1 Fluxo Básico

O fluxo básico de comunicação do GMTP se inicia quando um cliente deseja reproduzir um conteúdo multimídia. O cliente envia uma requisição em direção ao servidor que está transmitindo o conteúdo multimídia de interesse, como em qualquer conexão na Internet. Na Figura 6, ilustra-se o funcionamento do GMTP. O protocolo GMTP executa 6 passos principais, ilustrados na figura e descritos a seguir.

1. **Registro de Participação**: é o primeiro passo do processo de funcionamento do GMTP. O cliente envia uma requisição em direção ao servidor que está transmitindo o conteúdo multimídia de interesse. Em seguida, um repassador intercepta a requisição durante seu trajeto até o servidor, que é capaz de determinar os melhores parceiros para atendê-la. Em geral, isto ocorre já no roteador de borda do cliente, que funciona como repassador de origem. Caso o repassador não encontre nenhum repassador parceiro capaz de transmitir a mídia de interesse, a mensagem de requisição alcança o servidor que transmite a mídia correspondente.

   Todo repassador define, uma única vez, seu código de identificação único, definido como um código de *hash* MD5 no formato UUID de 128 bits passando como parâmetro a *string* resultante da concatenação de todos os endereços MAC de todas as interfaces de rede do repassador. Esse código de identificação é enviado ao servidor no ato do registro de participação. Desta forma, define-se o único identificador do repassador, que é posteriormente utilizado pelo servidor para determinar parcerias entre os demais repassadores

Figura 6 – Fluxograma geral do GMTP, desde a solicitação do cliente por uma mídia ao vivo até receber uma resposta, executando-se ações importantes como registro de participação, controle de congestionamento, eleição de relatores e verificação de autenticidade dos dados.



Fonte – Sales (2014)

2. **Transmissão de Dados**: caso o servidor aceite o pedido de conexão do cliente, este inicia a transmissão do fluxo de dados solicitado ao cliente.

   Ao iniciar a transmissão, o servidor passa a conhecer a rota utilizada para transmitir o conteúdo até o cliente, que é definida pela lista de roteadores existentes do servidor até ao cliente, nessa direção e ordem.

3. **Formação de Rede de Favores e Cache**: em vez de aceitar o pedido de conexão do cliente, o servidor pode delegar tal requisição a um repassador, com base no conhecimento de quais repassadores já estão encaminhando o conteúdo de interesse para alguma rede. Neste caso, o servidor recusa o pedido de conexão e determina que um repassador comece a servir o repassador mais próximo ao cliente, determinando-se uma parceria entre um repassador já em uso para disseminar o conteúdo de interesse e o repassador que solicitou o registro de participação inicial.

Tanto no Passo 2 como no Passo 3, sempre o repassador de origem assumirá o controle de uma requisição do cliente, habilitando-se como candidato a parceiro para outros repassadores, quando motivados por requisições originadas pelos seus respectivos clientes.

4. **Verificação de Autenticidade de Dados**: baseia-se na assinatura digital dos dados transportados nos pacotes e na capacidade que os repassadores têm de validar se o conteúdo foi intencionalmente alterado por usuários maliciosos no trajeto até o cliente, com base no certificado digital emitido pelo servidor – esta ação é opcional e decidida pela aplicação no início da transmissão.

5. **Controle de Congestionamento Unicast**: Executa-se um algoritmo de controle de congestionamento em cada repassador, que calcula sua capacidade de transmissão atual com base apenas no tamanho da fila de repasse e no atraso fim-a-fim marcado em pacote de dados a ser roteado. Em seguida, o repassador compara sua capacidade de transmissão com a menor capacidade de transmissão de todos os repassadores anteriores, transportada em cada pacote de dados. Se sua capacidade de transmissão for menor que a capacidade de transmissão informada no pacote de dados, o repassador o altera informando sua capacidade de transmissão, caso contrário, transmite-o para o próximo repassador.

6. **Controle de Congestionamento Multicast**: executa-se uma versão adaptada do algoritmo de controle de congestionamento *TCP-Friendly Multicast Congestion Control* (TFMCC) (WIDMER; HANDLEY, 2006) para fluxo de dados *multicast*, regulando-se a taxa de transmissão na rede local com base nos relatórios transmitidos por nós eleitos como relatores.

O posicionamento dos repassadores e suas habilidades permitem a redução do número de fluxos de dados na rede correspondentes a um mesmo evento, maximizando a quantidade de clientes interessados em receber o mesmo fluxo.

Cada repassador mantém uma tabela chamada *Tabela de Recepção de Fluxos de Dados*, conforme ilustra-se na Figura 7. Um repassador utiliza tal tabela para registrar todos os fluxos de dados que estão sendo repassados para seus clientes e os respectivos canais *multicast* utilizados, mantendo-se as seguintes informações:

- *Nome do Fluxo de Dados F:* é uma sequência de 128 *bits* que determina o nome de um fluxo de dados F. No GMTP, um fluxo de dados tem um nome único que o identifica em qualquer nó. Na prática, cada fluxo de dados corresponde a uma mídia gerada a partir de um evento ao vivo. Portanto, define-se um nome para F por um código de *hash* MD5 no formato UUID (*Universally Unique IDentifier*) de 128 bits (LEACH; MEALLING; SALZ, 2005). Para determinar F, disponibilizado por

Figura 7 – Exemplo de uma tabela de recepção de fluxo mantida por um repassador.

Fonte – Sales (2014)

um nó servidor s_a, utiliza-se MD5(IP$_{sa}$ + PORTA$_{sa}$). Opcionalmente, o nó servidor pode divulgar o nome do fluxo de dados através do serviço DNS.

- *Servidores s_a:* o endereço IP do servidor que gera o fluxo de dados F;
- *Repassadores r_q:* o endereço IP do repassador parceiro, que está transmitindo o fluxo de dados F para repassador. Se nulo, significa que F está sendo recebido diretamente do servidor;
- *Porta de Recepção de F:* o número da porta do nó remoto que está transmitindo F para o repassador. Nesse caso, o nó remoto pode ser o servidor, em caso de conexão direta com o servidor, ou um parceiro do repassador;
- *Endereço do Canal Multicast:* o endereço IP *multicast* utilizado pelo repassador para repassar F para os clientes; e
- *Porta do Canal Multicast:* o número da porta *multicast* utilizada pelo repassador para repassar F para os clientes.

O protocolo GMTP é flexível para permitir que um repassador atue somente encaminhando conteúdos multimídia entre duas ou mais redes distintas, mesmo que este não tenha demandas explícitas dos seus clientes por tal conteúdo. Desta forma, maximiza-se o uso dos canais de transmissão ociosos, em particular das redes residenciais, principalmente quando seus usuários estão ausentes e portanto sem fazer uso dos recursos disponíveis de rede.

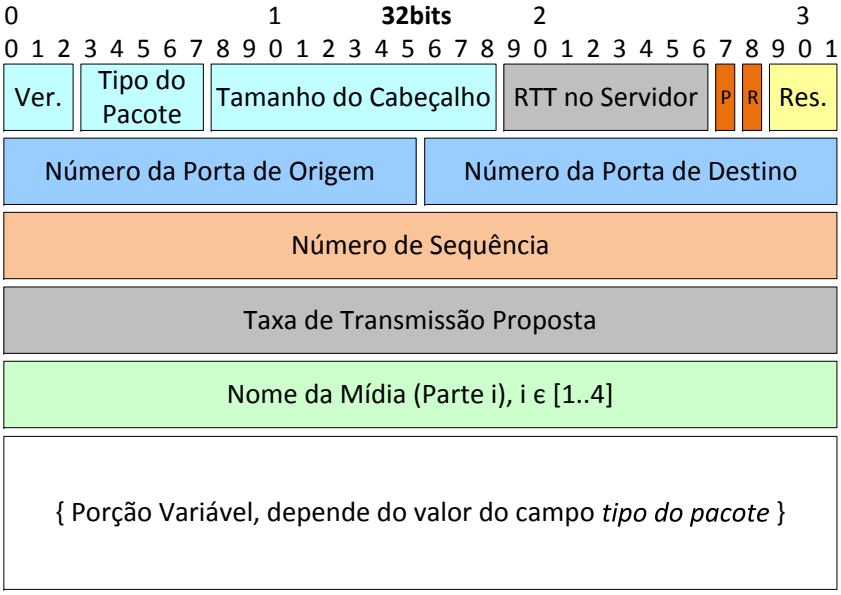
2.2.1.2 Cabeçalho geral e tipos de pacotes

A comunicação entre dois ou mais nós GMTP ocorre através da troca de datagramas IP, que podem carregar sinalizações de controle ou dados da aplicação.

Conforme ilustra-se na Figura 8, o cabeçalho do GMTP foi organizado em uma parte fixa e uma parte variável. A parte fixa foi definida em 32 bits, sendo organizada da seguinte forma: 3 *bits* para a *Versão do Protocolo*, 5 *bits* para o *Tipo de Pacote*, 11 *bits* para o *Tamanho do Cabeçalho*, 8 *bits* para o *RTT no Servidor*, 1 *bit* para o campo P (*Pull*), 1 *bit* para o campo R (*Relay*), 3 *bits* reservados para uso futuro, 16 *bits* para o *Número da*

Porta de Origem, 16 *bits* para o *Número da Porta de Destino*, 32 *bits* para os *Números de Sequência*, 32 *bits* para a *Taxa Proposta de Transmissão* e 128 *bits* para o *Nome do Fluxo de Dados*. É importante entender que os campos *RTT no servidor* e *taxa de transmissão* estão relacionados com o mecanismo de controle de congestionamento adotado no GMTP, ao passo que o campo *Nome do Fluxo de Dados* é uma forma de identificar unicamente a transmissão de um evento ao vivo e, com base nessa informação, formam-se as redes de favores entre os repassadores (SALES, 2014).

Figura 8 – Porção fixa do cabeçalho dos pacotes do GMTP.



Fonte – Sales (2014)

No contexto deste trabalho, propõem-se mudanças na organização da parte fixa do cabeçalho do GMTP, conforme discute-se na Seção 3.2.

Com a utilização de um cabeçalho de tamanho variável, permite-se a flexibilização das funções do protocolo. Por exemplo, no GMTP permite-se que uma aplicação injete informações na porção variável do cabeçalho para que sejam lidas pelos repassadores localizados entre os sistemas finais. Um exemplo do uso da porção variável do cabeçalho é no processo de conexão do GMTP, onde cada repassador na rota entre o cliente e o servidor adiciona seu identificador no cabeçalho de um pacote GMTP, o que permite aos servidores conhecerem as rotas sendo utilizadas para distribuir a mídia até seus clientes. Conhecendo-se as rotas pode-se sugerir parcerias de melhor qualidade. Dependendo do valor do campo *tipo do pacote*, um nó interpretará o restante do cabeçalho (parte variável) de forma diferente. O campo *Tamanho do Cabeçalho* permite a flexibilidade da parte variável do cabeçalho e seu espaço de 11 *bits* permite informar um cabeçalho com tamanho total de 2047 *Bytes*.

Na especificação original do GMTP, não constam as definições de organização da porção variável do cabeçalho do GMTP para cada tipo de pacote. No contexto deste trabalho, propõe-se uma organização para a porção variável do cabeçalho para cada tipo de pacote do GMTP, conforme descrito na Seção 3.2.2.

O campo *tipo do pacote* determina quais informações serão encapsuladas em cada tipo de pacote para, posteriormente, serem consumidas pelos nós da rede (clientes, servidores, repassadores e relatores), a fim de executarem as ações definidas no protocolo GMTP e disseminarem os pacotes de dados de um fluxo. A seguir, apresentam-se brevemente os tipos de pacote utilizados no GMTP.

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

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

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

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

4. *GMTP-Register-Reply:* o servidor responde sobre o pedido de registro de participação enviado por um repassador. Além disso, transporta o identificador de todos os repassadores entre o servidor e o cliente;

5. *GMTP-Route-Notify:* contém uma rota entre o repassador e um servidor. O repassador envia esse tipo de pacote ao servidor, após receber o pacote do tipo *GMTP-Register-Reply*;

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

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

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

9. *GMTP-Ack:* em geral, qualquer nó utiliza esse tipo de pacote para confirmar a recepção de um determinado pacote;

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

Visando a execução de ações definidas no protocolo GMTP, além de ações não previstas no proposta inicial do protocolo (SALES, 2014), no contexto deste trabalho, propõe-se a adição de novos tipos de pacote além dos tipos já definidos originalmente no GMTP, conforme descrito na Seção 3.2.2.

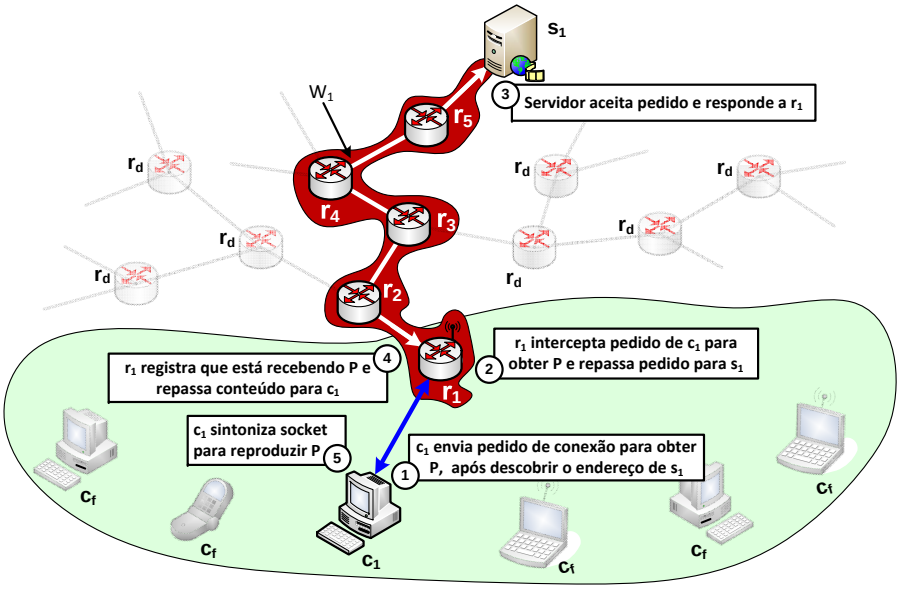
2.2.2 Estabelecimento de conexão entre cliente e servidor

Seja c um nó cliente, r um nó repassador e s um nó servidor de mídias. Divide-se o processo de estabelecimento de conexão em três fases. A Fase 1 acontece quando, por exemplo, um cliente c qualquer deseja obter o fluxo de mídia F transmitido por um nó servidor s e não existe nenhum outro cliente em sua rede local recebendo F. Já a Fase 2 acontece quando um outro nó cliente precisa obter o mesmo fluxo de dados F, solicitado previamente pelo primeiro cliente. E, por fim, a Fase 3 acontece quando o repassador r começa a buscar novos nós parceiros a fim de obter F (SALES, 2014).

2.2.2.1 Fase 1

Como ilustra-se na Figura 9, para obter o fluxo de dados F, o cliente c transmite um pacote do tipo *GMTP-Request* endereçado ao servidor s, com o valor para o campo do cabeçalho IP $TTL = 1$. Além dos valores para o IP de destino e para o TTL, o cliente também deve informar o nome do fluxo de dados F que o usuário deseja reproduzir, presente no cabeçalho de transporte do pacote do tipo *GMTP-Request*.

Figura 9 – Passos do processo de estabelecimento de conexão do GMTP (Fase 1).



Fonte – Sales (2014)

Quando o pacote *GMTP-Request* alcançar o repassador r (Passo 2), este consulta a tabela de recepção de fluxos de dados e constata que não há qualquer registro para o fluxo de dados F. Nesse instante, o repassador inicia um processo de registro de participação para obter o fluxo de dados F. Em seguida, ao receber o pacote do tipo *GMTP-Register-Reply*, como resposta ao registro de participação, o repassador cria um canal *multicast* e envia um pacote do tipo *GMTP-Request-Notify* para os nós clientes que solicitaram F.

Como resultado da Fase 1, o repassador insere uma nova entrada na sua tabela de recepção de fluxos de dados, tal como ilustra-se na Figura 10. Na referida figura, a tabela de recepção agora contém uma entrada que informa a ocorrência de recepção do fluxo de dados $P = 72c44591 - 7d82 - 427c - 825f - 722f015787c1$, originado no servidor, cujo endereço é *177.135.177.241*, com porta de recepção *49170*. Além disso, define-se o canal *multicast* no endereço *239.192.68.79* e porta *1900*, através do qual os clientes podem receber os pacotes de dados do fluxo F.

Figura 10 – Exemplo de configuração da tabela de recepção de fluxos de dados.

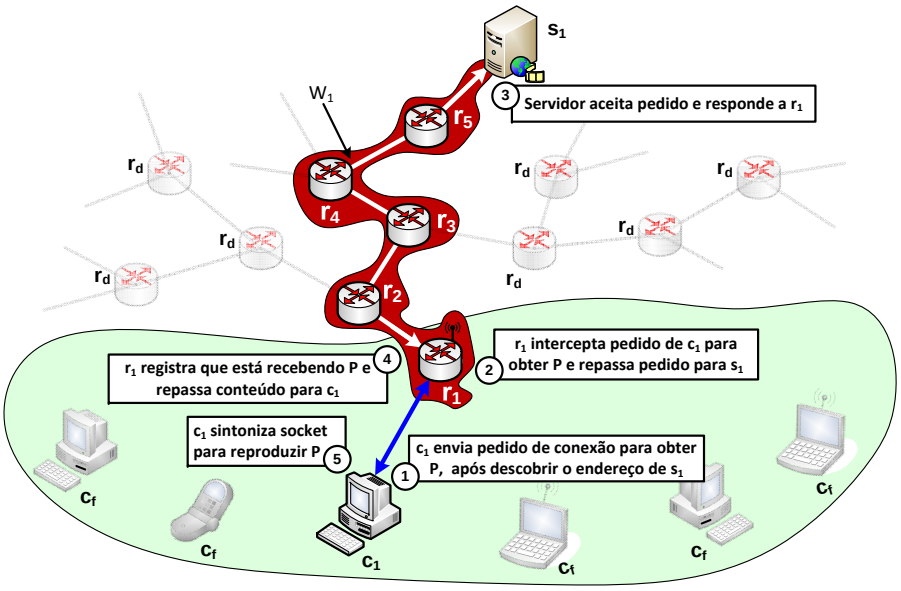
#	Nome do Fluxo de Dados (P)	Servidores s_a	Repassadores r_d	Porta de Recepção de P	End. do Canal Multicast	Porta do Canal Multicast
1	72c44591-7d82-427c-825f-722f015787c1	177.135.177.241	nulo	49170	239.192.68.79	1900

Fonte – Sales (2014)

2.2.2.2 Fase 2

A Fase 2 de conexão ocorre quando futuras requisições para obter o fluxo de dados F são originadas por qualquer cliente do repassador. Nesse caso, ao consultar a tabela de recepção de fluxos de dados, o repassador constatará que já existe um registro para o fluxo de dados F, não sendo necessário realizar o registro de participação no servidor nem criar um novo canal de repasse *multicast*. Nesse caso, o repassador apenas envia um pacote do tipo *GMTP-Request-Notify* para os nós clientes que solicitaram F.

Figura 11 – Tabela de recepção de fluxos de dados após a Fase 1.



Fonte – Sales (2014)

2.2.2.3 Fase 3

Na Fase 3, o repassador inicia um processo de aumentar suas parcerias a fim de obter mais rapidamente os pacotes através de caminhos alternativos. Para isto, o servidor constrói uma lista de nós parceiros e envia ao repassador, funcionando como um indexador de nós parceiros, pré-selecionando parceiros para o repassador. Não necessariamente quanto mais parceiros um nó repassador tem, melhor será a qualidade do fluxo de dados. Por isso,

um repassador sempre mantém uma lista de candidatos a parceiros fornecida pelo servidor, porém não necessariamente estabelece parcerias com todos. Para isto, o repassador envia periodicamente um pacote do tipo *GMTP-RelayQuery* para o servidor, a fim de descobrir melhores parceiros e aumentar o número de parcerias. O nó s constrói uma lista de nós parceiros e envia ao nó r, através de um pacote do tipo *GMTP-RelayQuery-Reply*. Cada entrada disponível no referido pacote contém o identificador do repassador candidato a parceiro e sua respectiva chave de autorização.

Após obter a lista de candidatos a parceiros, o nó r forma uma parceria com um dos candidatos da lista de possíveis parceiros. Para isto, o nó r envia requisições do tipo *GMTP-Request* em direção a outros nós repassadores sugeridos por s e que já estejam recebendo o fluxo de dados. As requisições *GMTP-Request* contêm uma chave de autorização conhecida por ambos e informada pelo nó s. Caso a chave de autorização esteja correta, o nó parceiro deve enviar uma resposta do tipo *GMTP-Response* ao nó r e então começar a repassar os pacotes do fluxo de mídia. O uso da chave de autorização é importante para evitar que um nó r se conecte a outro nó repassador sem que o nó s seja notificado sobre isto.

2.2.3 Constituição da Rede de Favores

No GMTP, a constituição da rede de favores ocorre por meio do registro de participação de um ou mais nós repassadores de um fluxo de mídia, a um ou mais nós servidores de mídia. O registro de participação permite que um nó repassador se registre no servidor para sinalizar interesse em funcionar como repassador de um fluxo de dados. O registro de participação pode ocorrer antes do servidor iniciar a transmissão do fluxo de mídia ou durante a transmissão.

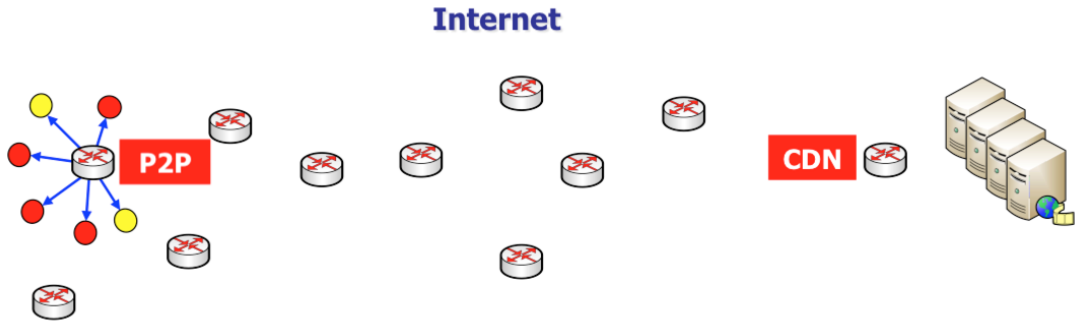
Conforme ilustra-se na Figura 12, para realizar o registro de participação, o nó r deve enviar uma mensagem para o nó s utilizando um pacote do tipo *GMTP-Register*. Este, por sua vez, responde com um pacote do tipo *GMTP-Register-Reply*. O pacote do tipo *GMTP-Register-Reply* contém a **chave de autorização** única de r, gerada por s.

Conforme ilustra-se na Figura 12c, o uso do pacote do tipo *GMTP-Register-Reply* permite a descoberta de uma rota entre s e r. Isto porque todos os nós repassadores existentes na rota entre s e r devem adicionar seu identificador no pacote *GMTP-Register-Reply*, antes de roteá-lo para o próximo salto da rota em direção ao nó r. Quando o pacote *GMTP-Register-Reply* alcança o nó r, este envia a lista ordenada de repassadores entre r e s de volta ao nó s, utilizando um pacote do tipo *GMTP-Route-Notify*. A partir desse ponto, o nó s conhece a rota que utilizará para enviar qualquer fluxo de dados até alcançar r, organizando-o em uma estrutura de dados do tipo grafo. Esse procedimento de três vias confirma o registro de participação do repassador r no servidor s, ao passo que s poderá utilizar a rota armazenada para instruir os nós repassadores a realizarem parcerias, a fim

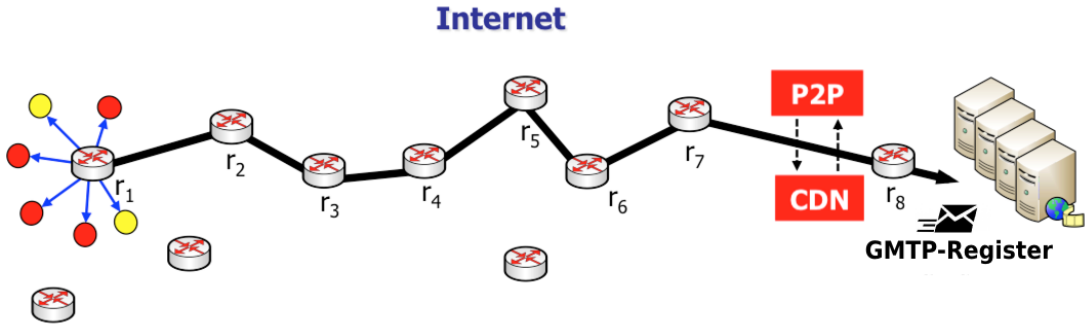
de distribuir um fluxo de dados.

Figura 12 – Registro de participação do repassador. Neste exemplo, o repassador r_1 registra-se no servidor de mídia.

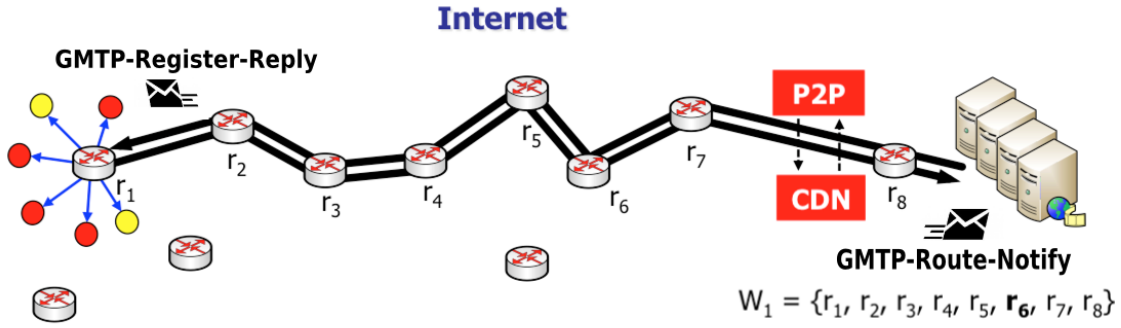
(a) Antes do registro de participação, o servidor de mídia não tem conhecimento da topologia da rede.



(b) Para realizar o registro de participação, o nó r_1 envia uma mensagem para o nó servidor utilizando um pacote do tipo *GMTP-Register*.



(c) Quando o pacote *GMTP-Register-Reply* alcança o nó r, este envia a rota W_1 de volta ao nó servidor, utilizando um pacote do tipo *GMTP-Route-Notify*.

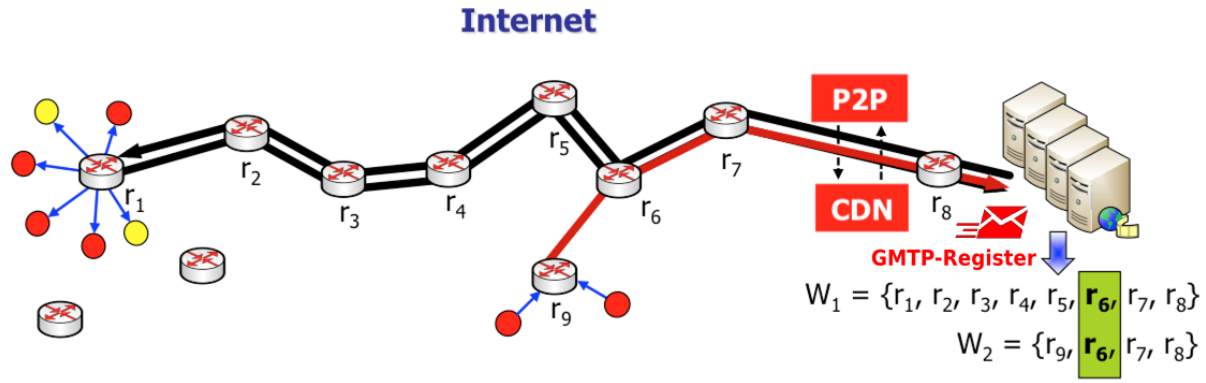


No GMTP, as parcerias ocorrem entre os nós repassadores, e não entre os nós clientes. A formação de parcerias consiste em determinar intersecções de caminhos entre repassadores e servidores. Dessa forma, se um nó repassador r for um nó comum entre dois caminhos conhecidos por s, será necessário apenas enviar um fluxo de dados até r, que repassará o referido fluxo de dados para os demais nós parceiros, mais distantes de s, utilizando *multicast*. Conforme ilustra-se na Figura 13b, é possível que um repassador atue

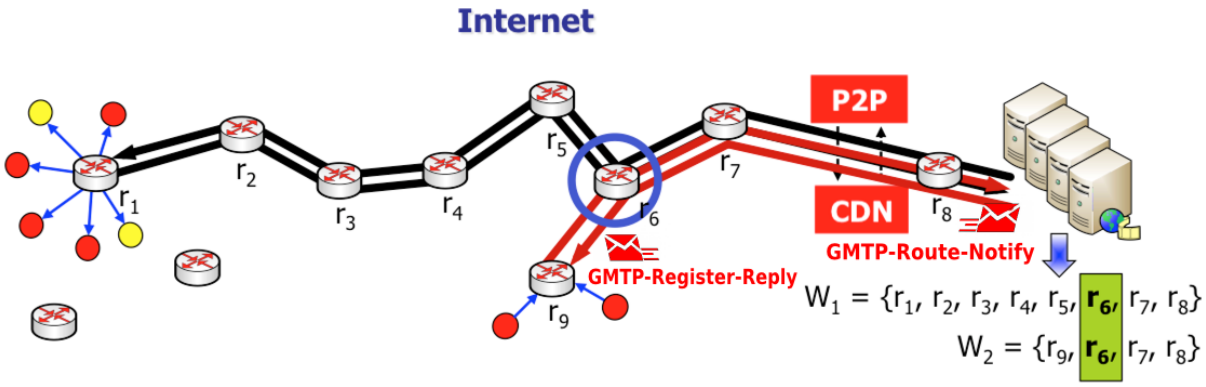
somente encaminhando conteúdos multimídia entre duas ou mais redes distintas, mesmo que este não tenha demandas explícitas dos seus clientes por tal conteúdo. Na referida figura, o nó r_6 atua dessa forma.

Figura 13 – Formação de parcerias entre repassadores. Neste exemplo, o nó r_9 forma parceria com o nó r_6.

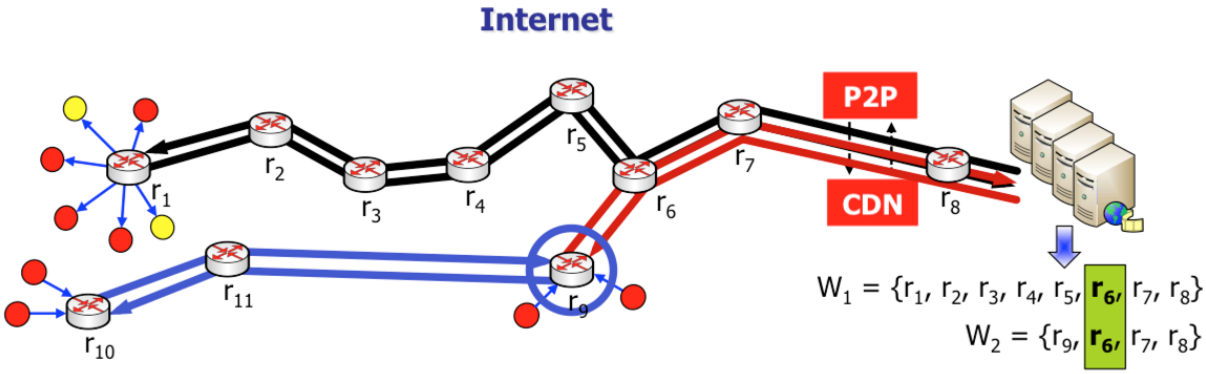
(a) O nó r_9 realiza o registro de participação no servidor. Após o registro, o servidor conhece a rota W_2 até r_9.



(b) Através do algoritmo de formação de parcerias do GMTP, identifica-se uma possível parceria entre r_6 e r_9, que passam a colaborar entre si.



(c) O nó r_9 intercepta o registro de participação de r_{10}, transmitindo a mídia como se fosse o próprio servidor.



Um repassador pode interceptar o registro de participação de outros repassadores

durante seu trajeto até o servidor. Caso o repassador que enviou a requisição não encontre nenhum repassador parceiro capaz de transmitir a mídia de interesse, a mensagem de requisição alcança o servidor que transmite a mídia correspondente.

Sendo assim, quanto mais nós repassadores se registrarem em nós servidores, mais caminhos serão conhecidos. Quanto mais caminhos forem conhecidos, mais parcerias poderão ser formadas entre os nós repassadores. Quanto mais parcerias forem formadas, maior será o número de nós clientes capazes de receber um fluxo de dados originado nos servidores (SALES, 2014).

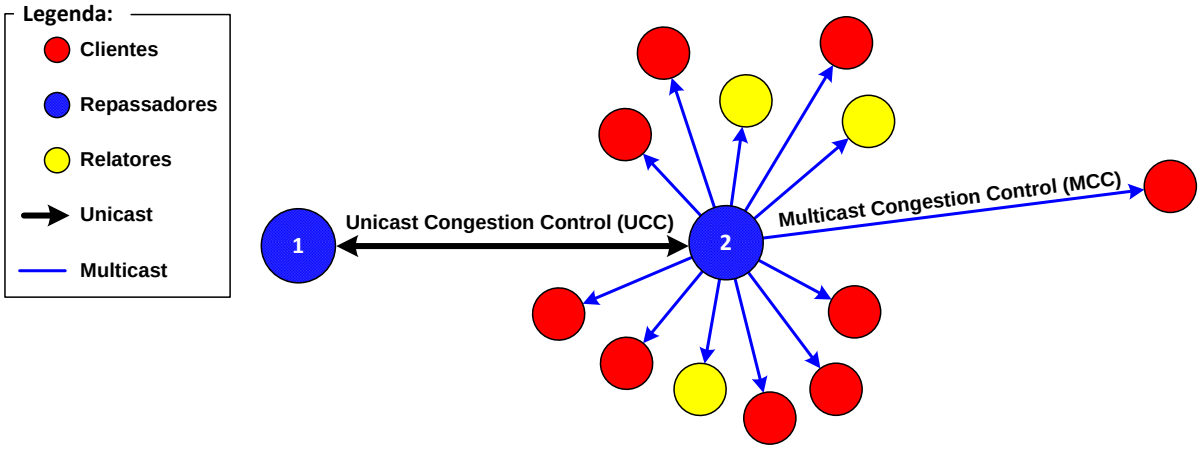
Desta forma, permite-se a configuração dinâmica de canais *multicast* aliada à formação de uma rede de favores constituída entre roteadores. Constitui-se a rede de favores através da formação de parcerias pela intersecção de rotas que se tornam conhecidas à medida que novos clientes se interessam por uma mesma mídia, sendo este processo regido por um servidor *pivot* transmissor da mídia, que sugere as parcerias com base na capacidade de transmissão dos canais em uso.

2.2.4 Envio e recebimento do fluxo de mídia

Após o estabelecimento de conexão, os nós repassadores trocam dados entre si em modo *unicast* a fim de distribuir os pacotes de dados do tipo *GMTP-Data* e *GMTP-DataAck*. De forma similar, os nós repassadores utilizam os mesmos tipos de pacotes para enviar os dados para os nós clientes, porém em modo *multicast*.

Conforme ilustra-se na Figura 14, no GMTP, executa-se um algoritmo para controle de congestionamento híbrido, cujo comportamento dependerá do modo de transmissão sendo utilizado para transportar os pacotes de dados (*unicast* ou em *multicast*).

Figura 14 – Algoritmos de controle de congestionamento do GMTP.



Fonte – Sales (2014)

2.2.5 Controle de congestionamento *unicast*

O controle de congestionamento *unicast* do GMTP (GMTP-UCC) funciona de forma similar ao protocolo RCP, porém com alguns diferenciais. 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 (DUKKIPATI et al., 2005). Nesse contexto, entende-se que, se um roteador pudesse obter a informação exata sobre o número de fluxos de entrada em um instante $\hat{t}$, a taxa de transmissão ideal para cada fluxo de dados seria $R_{ps}(\hat{t}) = \frac{C}{N(\hat{t})}$, onde C corresponde à capacidade do enlace e $N(\hat{t})$ o número de fluxos no instante $\hat{t}$.

Com base nesse princípio, Dukkipati (2007) argumentou que para um roteador funcionar de forma equânime e reduzir o tempo de duração de um fluxo, deve-se oferecer a mesma taxa de transmissão para todos os fluxos transmitidos através deste roteador. Com base nisso, determinou-se a Equação 2.1, onde $R(\hat{t})$ é a taxa de transmissão que deve ser oferecida para cada fluxo de dados que passa pelo roteador, e:

$$R(\hat{t}) = R(\hat{t} - h_0) + \frac{\alpha(C - y(\hat{t})) - \beta\frac{q(\hat{t})}{h_0}}{\hat{N}(\hat{t})} \tag{2.1}$$

- h_0 é a média móvel dos valores de RTT_s, calculada através da Equação 2.2, onde θ é o ganho e corresponde a 0,02. RTT_s é o tempo de ida e volta calculado entre o nó transmissor e o receptor;

$$h_0 = \theta \times RTT_s + (1 - \theta) \times h_0 \tag{2.2}$$

- $R(\hat{t} - h_0)$ é a última taxa de transmissão medida, em *bytes/milissegundos*;
- C é a capacidade total do canal;
- $N(\hat{t})$ é o número de fluxos no instante $\hat{t}$;
- $y(\hat{t})$ é a taxa de tráfego de entrada medida no intervalo entre a última atualização de $R(\hat{t})$ e o instante H;
- $q(\hat{t})$ é o tamanho instantâneo da fila de repasse, em *bytes*. No GMTP esse valor é obtido pela soma de todos os pacotes p_x presentes nos *buffers*, para todos os fluxos de dados registrados na tabela de repasse. Nesse caso, um nó repassador mantém um *buffer* para cada fluxo de dados do GMTP e um *buffer* geral, para pacotes de dados que não precisam de tratamento especial, por exemplo, pacotes TCP..
- α e β, tal que $0 < \alpha, \beta \leq 1$, determinam, respectivamente, a estabilidade e o desempenho do algoritmo. Com base em discussões apresentadas em (DUKKIPATI, 2007), os valores de α e β dependem do tamanho médio dos fluxos comparados com

o produto largura de banda - atraso. Quando o tamanho médio dos fluxos estão próximos do produto largura de banda - atraso (fluxos longos), sugere-se um valor alto para α e um valor baixo para γ, uma vez que nesse caso, prefere-se maximizar a taxa de transmissão de cada fluxo a minimizar o atraso na fila. Por outro lado, para fluxos de curta duração, recomenda-se um valor baixo para α e um valor alto para γ, pois esta combinação ajuda a manter um baixo atraso de fila. Para um equilíbrio entre estabilidade e desempenho, recomendam-se valores de $\alpha \in (0.4, 0.6)$ e $\beta \in (0.2, 0.6)$.

Pela Equação 2.1, estima-se a largura de banda disponível em um determinado canal, representada pela porção $\alpha(C - y(\hat{t})) - \beta\frac{q(\hat{t})}{h_0}$ e divide-se por $\hat{N}(\hat{t})$. Porém, como é impossível determinar o valor exato de $N(\hat{t})$, estima-se $\hat{N}(\hat{t}) = \frac{C}{R(\hat{t}-h_0)}$. Além disso, para atualizar $R(\hat{t})$ com mais frequência do que o intervalo de um RTT (h_0), definiu-se $H = min(H_{user}, h_0)$ e, para manter a estabilidade do restante da equação, escala-se a mudança agregada por $\frac{H}{h_0}$, resultando na Equação 2.3, onde:

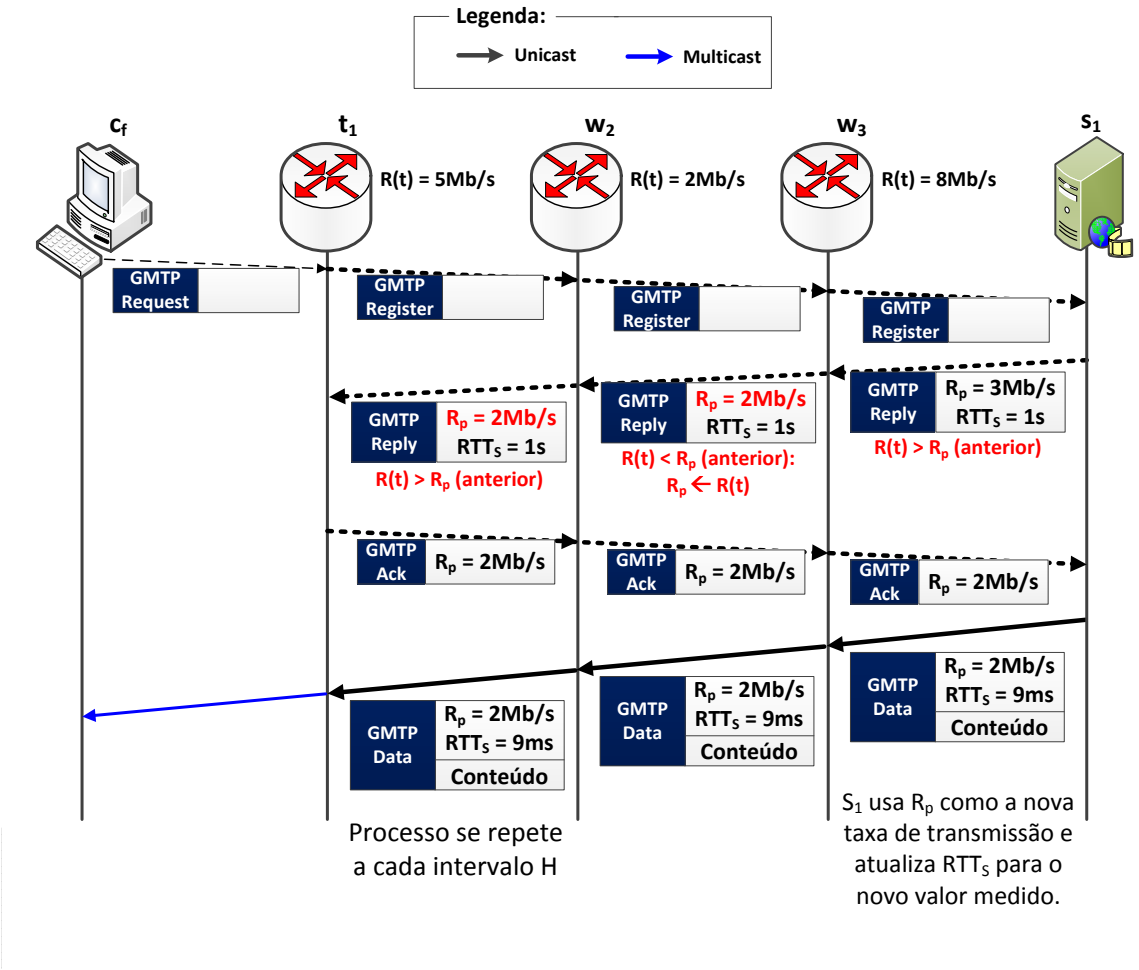
$$R(\hat{t}) = R(\hat{t} - H)\left[1 + \frac{\frac{H}{h_0}(\alpha(\gamma C - y(\hat{t})) - \beta\frac{q(\hat{t})}{h_0}}{\gamma C}\right] \tag{2.3}$$

- $H = min(H_{user}, h_0)$, sendo H_{user} um tempo definido pelo usuário (por exemplo, o administrador do roteador), caso seja necessário atualizar $R(\hat{t})$ em um intervalo de tempo menor que h_0. O valor para H é definido em *milissegundos*;
- $R(\hat{t} - H)$ é a última taxa de transmissão medida, em *bytes/milissegundos*;
- γ, tal que $0 < \gamma \leq 1$, controla o pico de utilização do canal. Por exemplo, se desejar utilizar no máximo 95% do canal em um certo instante $\hat{t}$, basta definir $\gamma = 0,95$.

Seja r um nó repassador, s um nó servidor de mídias e W uma rota entre r e s. Conforme ilustra-se passo a passo na Figura 15, o algoritmo adotado no GMTP-UCC, adaptado do RCP, funciona da seguinte forma: Cada nó r mantém uma única taxa de transmissão $R(\hat{t})$ que é atribuída no cabeçalho de todos os pacotes transmitidos do nó s aos nós repassadores em W. À medida que o pacote passa em cada repassador em W, se a taxa atual $R(\hat{t})$ no roteador for menor do que R_p informada no pacote sendo processado, $Rp \leftarrow R(\hat{t})$. Quando o pacote alcançar o último nó do caminho, este envia para s o valor de R_p, que é a máxima taxa de transmissão suportada na rota W. Ao receber o valor de R_p, s atualiza $R(\hat{t})$ para transmitir os próximos pacotes de dados. Este procedimento se repete a cada intervalo de tempo H.

O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão de um nó, o que pode limitar a taxa de transmissão em uma rota. Utilizando o RCP,

Figura 15 – Funcionamento básico do protocolo RCP (*Rate Control Protocol*).

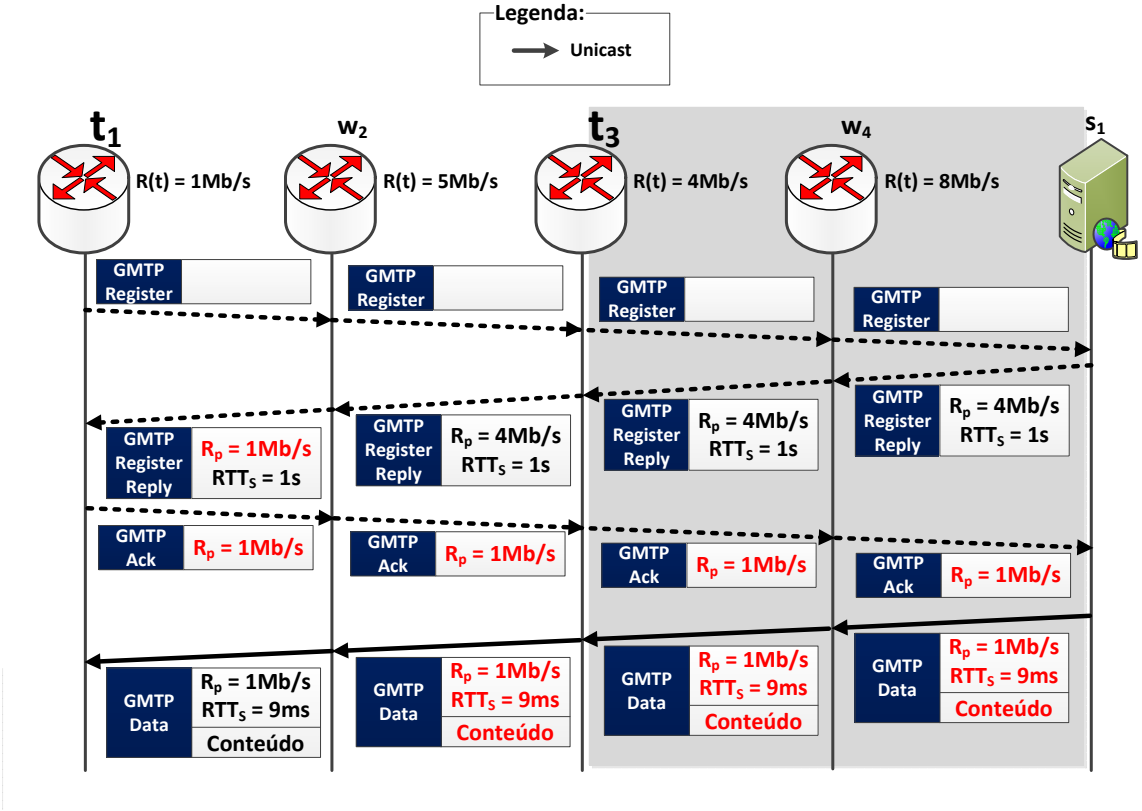


Fonte – Sales (2014)

dado um determinado caminho W, a taxa de transmissão $R(t)$ de W será igual à menor taxa máxima de transmissão informada por um nó em W. Por exemplo, na Figura 16, ilustra-se um cenário de transmissão onde observa-se que a taxa de transmissão de toda a rota é limitada pela capacidade de um dos nós do caminho.

Visando contornar as limitações do RCP, o GMTP-UCC segmenta a rota, não limitando a taxa de transmissão de um nó capaz de receber um fluxo de dados em uma taxa de transmissão maior. Sendo assim, considerando o mesmo exemplo ilustrado na Figura 16, mas adotando essa estratégia de segmentar a rota W, tal cenário corresponde ao ilustrado na Figura 17. Como resultado dessa estratégia e ainda considerando o exemplo em discussão, o nó t_3 deverá solicitar ao nó s, o fluxo de dados com codificação compatível com a taxa de bits de 1 *Mbps* ou inferior e então servir ao nó t_1. A partir deste ponto, o nó t_3 deve receber a mídia codificada em duas taxas de bits: 1024 *Kbps* e 3072 *Kbps*. Dessa forma, t_3 torna-se capaz de servir a outros nós repassadores tanto a uma taxa de 1 *Mbps* quanto a uma taxa de 4 *Mbps*.

Figura 16 – O RCP utiliza uma abordagem fim-a-fim para determinar a taxa de transmissão de um nó. Neste exemplo, o nó t_3 tinha capacidade para receber o conteúdo a uma taxa de transmissão de 4 *Mbps*, porém a taxa de máxima relatada por t_1 é de 1 *Mbps*, fazendo com que todos os nós na rota W recebam o fluxo de dados a 1 *Mbps*.



Fonte – Sales (2014)

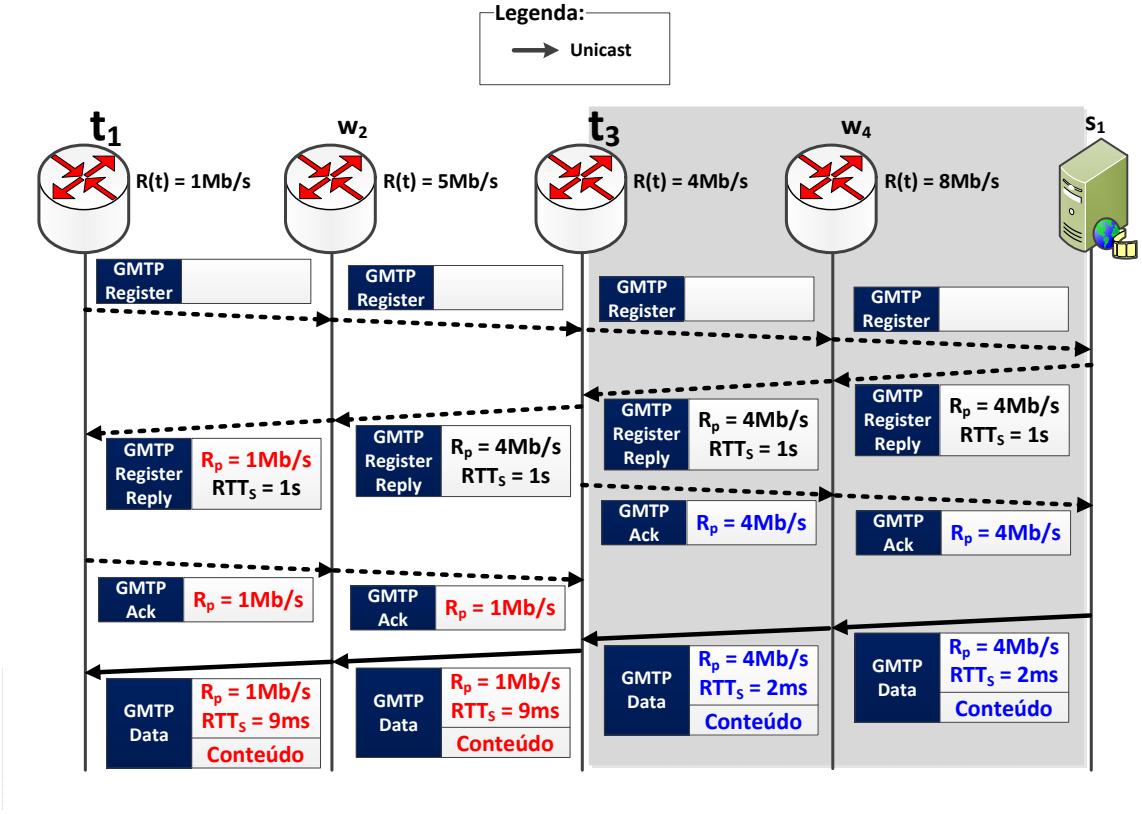
O GMTP permite descrever a mídia em múltiplas codificações, ao mesmo tempo que os nós repassadores podem acessar tal informação para implementar a solução discutida anteriormente. Como resultado dessa estratégia, permite-se que a rede seja capaz de selecionar fluxos codificados pelo servidor de forma segmentada, de acordo com a capacidade de transmissão nos sub-caminhos observados entre os nós repassadores e o servidor.

2.2.6 Controle de congestionamento *multicast*

O controle de congestionamento *multicast* do GMTP, denominado GMTP-MCC, tem como objetivo determinar uma taxa de transmissão equânime entre os fluxos de dados transmitidos pelo GMTP e por outros protocolos, como o TCP, em modo de transmissão *multicast*.

No algoritmo de controle de congestionamento *TCP-Friendly Rate Control protocol* (TFRC) (HANDLEY et al., 2003), o nó receptor envia para o transmissor relatórios sobre as perdas observadas e, com base nesse relatório, o transmissor calcula a nova taxa de

Figura 17 – O GMTP-UCC segmenta a rota e dessa forma não limita a taxa de transmissão de um nó capaz de receber um fluxo de dados em uma taxa de transmissão maior.



Fonte – Sales (2014)

transmissão. O TFRC é categorizado como um protocolo de controle de congestionamento baseado em uma equação matemática (*Equation Based Congestion Control*) e algoritmos desse tipo são adotados em diversos protocolos, como no DCCP (SALES, 2014).

O algoritmo do TRFC não é eficiente em transmissões *multicast*, devido ao problema conhecido por implosão de confirmações (*feedback implosion*). Esse problema ocorre quando há muitos receptores enviando relatórios de perdas para o mesmo transmissor, o que resulta em uma inundação de relatórios, os quais o transmissor é incapaz de processar em tempo hábil (SALES, 2014). Visando contornar esse problema, Widmer e Handley (2006) propuseram o *TCP-Friendly Multicast Congestion Control* (TFMCC), uma versão do TFRC adaptada para fluxo de dados *multicast*. No TFMCC, propõe-se um mecanismo de taxa de supressão (*suppression rate*), onde somente receptores com uma taxa calculada inferior à taxa de supressão são elegíveis para dar *feedback*, a menos que seu RTT seja maior do que o RTT máximo, caso em que também são elegíveis para dar *feedback*. O receptor que, em seu *feedback*, informar a menor taxa é denominado *current limiting receiver* (CLR).

No GMTP, utiliza-se um algoritmo de controle de congestionamento próprio, adaptado do TRFC, para fluxo de dados *multicast*. Diferente do TFMCC, que utiliza uma taxa de supressão, no GMTP-MCC elege-se um subconjunto dos nós clientes para atuar como relatores. Apenas os relatores devem calcular a taxa de transmissão baseado na taxa de eventos de perda de pacotes e enviar os *feedbacks* ao emissor. Assim, regula-se a taxa de transmissão na rede local com base nos relatórios transmitidos por nós eleitos como relatores.

Resumidamente, o algoritmo do TRFC funciona da seguinte forma:

1. O receptor mede a taxa de perda de pacotes e a envia para o nó transmissor;
2. O nó transmissor usa esse relatório para medir o RTT até o receptor;
3. O nó transmissor utiliza a Equação 2.4 para determinar qual será a sua próxima taxa de transmissão em função do relatório de perdas e o RTT obtidos;
4. O nó transmissor ajusta sua taxa de transmissão para o valor calculado no passo anterior.

$$R(s,p) = \frac{s}{RTT \times \left(\sqrt{\frac{2\times p}{3}} + \left(12 \times \sqrt{\frac{3\times p}{8}}\right) \times p \times (1 + 32 \times p^2)\right)} \tag{2.4}$$

Na Equação 2.4 (PADHYE et al., 1998), $R(s,p)$ é a taxa de transmissão medida em bytes/segundo definida em função de s, que é o tamanho do pacote medido em bytes e p, que corresponde a taxa de perda de pacotes observado pelo nó receptor; RTT é o tempo de ida-volta entre o nó transmissor e o receptor, medido em segundos.

O GMTP-MCC, sendo uma adaptação da versão original do TFRC, funciona da seguinte forma:

1. O nó repassador executa um algoritmo de eleição de nós relatores, selecionando um subconjunto de seus clientes.
2. Os nós relatores calculam a taxa de transmissão utilizando a Equação 2.4, em vez do transmissor realizar este cálculo, como na versão original do TFRC;
3. Os nós relatores determinam a taxa de eventos de perda, e não todos os receptores do grupo *multicast*. Para calcular o evento de perda p, utiliza-se o mesmo procedimento feito pelo TFRC, onde um intervalo de perda é determinado por consecutivas perdas de pacotes, desde do primeiro pacote perdido até o último pacote perdido, seguido de um pacote recebido com sucesso (HANDLEY et al., 2003; PADHYE et al., 1998);

4. O RTT é calculado entre o nó relator e o nó repassador, com o temporizador controlado pelos nós relatores e não pelo nó repassador. Isto evita que o nó repassador tenha que manter estado de temporizador para cada fluxo de dados transmitido para os nós clientes. Para determinar o valor do parâmetro RTT e calcular a taxa de transmissão através da Equação 2.4, o GMTP-MCC utiliza a Equação 2.2, que também é utilizada no GMTP-UCC, porém com $\theta = 0.25$, valor igual ao utilizado no TCP e no DCCP;

5. A taxa de transmissão a ser utilizada pelo nó repassador é a média aritmética de todas as taxas enviadas pelos nós relatores;

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.

Para todo fluxo de dados, o primeiro nó relator será o cliente que iniciar a primeira conexão para obter o referido fluxo. Os seguintes relatores serão os próximos clientes que se conectarem para receber o fluxo de dados, até atingir um parâmetro que determinará a quantidade máxima de relatores por fluxo de dados. Tal parâmetro pode ser determinado pelo administrador do repassador. Por padrão, utiliza-se $\frac{1}{6}$ dos nós clientes como sendo relatores para a transmissão de um fluxo de dados.

Sendo assim, à medida que um nó repassador recebe pacotes do tipo *GMTP-Request*, no pacote de resposta *GMTP-RequestReply*, o repassador ativa um indicador sinalizando que o referido nó cliente em processo de conexão deverá se comportar como um nó relator, passando a enviar relatórios da taxa de transmissão calculada.

2.2.7 Autenticidade do Fluxo de Dados

Em uma solução baseada em um modelo de serviço P2P, é possível que nós repassadores mal-intencionados poluam o sistema com conteúdos que não foram gerados pelo servidor. Para evitar esse tipo de ataque, executa-se um procedimento para verificar a autenticidade de um fluxo de dados. Para isto, os próprios nós repassadores verificam se o conteúdo de um pacote de dados foi alterado por algum nó repassador anterior durante o procedimento de repasse. Apenas após comprovar a autenticidade de um pacote, o repassador encaminha tal pacote de dados para o próximo nó, transmitindo-os também para seus clientes, se houver demanda. Este procedimento evita que todos os clientes que receberem o fluxo de dados tenham que verificar a autenticidade dos pacotes, evitando-se que a rede repasse conteúdo multimídia errados, consequentemente não consumindo recursos computacionais desnecessários (SALES, 2014).

Considerando que os repassadores processam cada pacote de dados para decidir sobre seu repasse e para executar os algoritmos de controle de congestionamento, se o

processo de verificação de autenticidade fosse aplicado em todos os pacotes de dados, haveria oneração dos recursos computacionais de cada repassador, ocasionando aumento do tempo de entrega dos pacotes aos clientes. Para reduzir a sobrecarga de verificação de autenticidade de um fluxo de dados em cada repassador, no GMTP definiram-se regras para decidir quais nós devem realizar a verificação de autenticidade e para determinar a quantidade de pacotes que se deve realizar tal procedimento.

A função de verificação de autenticidade de um fluxo de dados 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 servidor tem o controle de habilitar tal funcionalidade, e este procedimento requer sinalizar os repassadores para que estes executem o procedimento de verificação de autenticidade.

Por fornecer a função de verificação da porção de dados transportado em um pacote, no GMTP não se realiza checagem de erro por soma de verificação (*checksum*), tal como em protocolos como TCP, UDP, DCCP e SCTP.

3 IMPLEMENTAÇÃO DO GLOBAL MEDIA TRANSMISSION PROTOCOL (GMTP)

Nesta seção, apresentam-se pontos importantes relacionados à implementação e configuração do GMTP no núcleo do Linux, bem como um registro de quais funções do GMTP foram implementadas tal como especificadas e quais foram adaptadas para o correto funcionamento do protocolo no Linux e nas aplicações estudadas.

Antes de entrar em detalhes em alguns aspectos importantes, a seguir, apresenta-se um resumo das adições e adaptações ao GMTP que foram propostas e implementadas no contexto deste trabalho:

1. Mudanças na organização do cabeçalho fixo (Seção 3.2.1);
2. Adição de novos tipos de pacote (Seção 3.2.2);
3. Especificação da organização dos cabeçalhos variáveis (Seção 3.2.2);
4. Definição da estrutura das tabelas dos clientes (Seção 3.3.2.1);
5. Definição da estrutura das tabelas dos servidores (Seção 3.3.2.1);
6. Alterações na estrutura da tabela de repasse (Seção 3.3.3.1);
7. Especificação de códigos para o pacote do tipo *RequestNotify* (Seção 3.4.1);
8. Definição do mecanismo de *keep-alive* dos clientes através dos relatores (Seção 3.4.2);
9. Alteração no mecanismo de *keep-alive* dos repassadores (Seção 3.4.3.1);
10. Alteração no mecanismo do GMTP-UCC (Seção 3.4.4.2);
11. Adição de um novo mecanismo de ampliação de parcerias entre os roteadores, baseado na iniciativa do servidor (Seção 3.4.5);
12. Alteração dos mecanismos de finalização de conexão (Seção 3.4.6)

Além disso, outras adições e adaptações foram identificadas no contexto do GMTP, as quais estão descritas na Seção 5.2.

3.1 Protocolos de Rede no Sistema Operacional Linux

Nesta seção, discute-se como organiza-se a infraestrutura de rede na arquitetura do núcleo do Linux, com ênfase na camada de transporte, realizando-se ponderações com relação à implantação do GMTP no referido núcleo.

No núcleo de um sistema operacional, não realiza-se qualquer tarefa relacionada às camadas acima da camada de transporte. As funções típicas das camadas de sessão, apresentação e de aplicação devem ser implementadas por aplicações do espaço do usuário. Essencialmente, é tarefa da pilha de rede do núcleo do Linux a passagem de pacotes da camada de *enlace* para a camada de rede (geralmente IPv4 ou IPv6) e então para os protocolos da camada de transporte, caso o pacote seja destinado ao sistema final local, ou de volta à camada de *enlace*, caso o pacote precise ser encaminhado a outro sistema final. Pacotes gerados localmente passam da camada de transporte para a camada de rede e então para a camada de *enlace*, transmitindo-os através do dispositivo de rede (ROSEN, 2013).

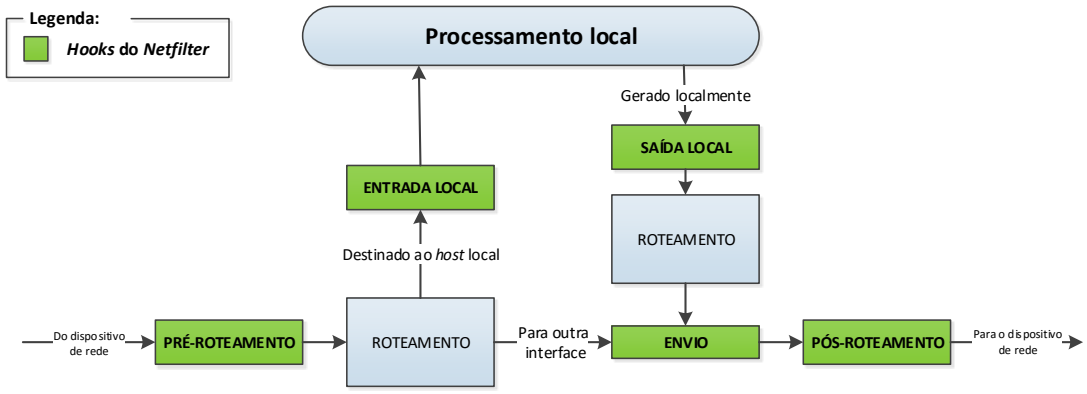
Para cada pacote de rede, seja ele enviado ou recebido, realiza-se uma busca no subsistema de roteamento. O resultado dessa busca determina se o pacote será encaminhado ou se deverá ser tratado localmente. Além da busca no subsistema de roteamento, outros fatores podem determinar a rota de um pacote na pilha de rede do Linux, como por exemplo, o valor do campo *TTL* do cabeçalho dos pacotes IPv4 ou *callbacks* do Netfilter. O Netfilter (WELTE; AYUSO, 2016) é um subsistema do núcleo do Linux que permite o registro de funções de *callback* a fim de interceptar e manipular os pacotes de rede. Conforme ilustra-se na Figura 18, o Netfilter oferece várias opções para filtragem de pacotes, tradução de endereços de rede e de portas. Estas funções geralmente são aplicadas ao tráfego na forma de regras de filtragem e de modificação, sendo invocadas para cada pacote que atravessa o respectivo *hook*[1] dentro da pilha de rede, fornecendo as funcionalidades necessárias para modificar pacotes, direcioná-los através de uma rede, bem como proibi-los de alcançar determinados nós da rede.

3.1.1 Sockets

No Linux, utiliza-se uma API compatível com as especificações de *socket* BSD e POSIX como interface entre as aplicações do espaço de usuário e o seu subsistema de redes. Conforme o paradigma do Unix onde "tudo é um arquivo", todo *socket* é associado a um arquivo. A utilização de uma API padrão de *sockets* facilita a portabilidade de aplicações. Por exemplo, as opções de configuração do GMTP podem ser acessadas e modificadas de

[1] Em programação de computadores, o termo *hooking* (enganchar) cobre uma série de técnicas utilizadas para modificar ou melhorar o comportamento de um sistema operacional, aplicações ou outros componentes de software através da interceptação de chamadas de funções, mensagens ou eventos passados entre componentes de software. Fonte: https://pt.wikipedia.org/wiki/Hooking.

Figura 18 – Caminhos possíveis no processamento de pacotes e locais onde os *hooks* podem ser configurados.



forma uniforme em qualquer sistema que siga a API padrão de *sockets*, através de opções de *socket*, utilizando-se a função *setsockopt*. A lista completa de todas as opções de *socket* do GMTP encontra-se no Anexo B.

No núcleo do Linux, há duas estruturas que representam um *socket*: a primeira é `struct socket`, que provê uma interface para o espaço de usuário. A segunda é `struct sock`, que provê uma interface para a camada de rede.

Além das variáveis relacionadas ao estado da conexão, na estrutura `struct socket` armazena-se um apontador para a estrutura `struct proto_ops`, que consiste basicamente de *callbacks* para o *socket*, como `connect()`, `listen()`, `sendmsg()`, `recvmsg()` e outros. Esses *callbacks* são interfaces para as aplicações do espaço de usuário. Cada protocolo da camada de transporte define uma estrutura `proto_ops` de acordo com seus requisitos. Por exemplo, no TCP, sua estrutura `proto_ops` inclui um *callback* para a função `listen()` e outro para a função `accept()`. Já no UDP, que não funciona no modelo orientado a conexão, os *callbacks* para as funções `listen()` e `accept()` são apenas métodos que retornam um código de erro indicando que o protocolo não suporta a operação (ROSEN, 2013). O detalhamento dos principais campos da estrutura `struct socket` encontra-se no Anexo A.1.

A estrutura `struct sock` é a representação de *sockets* na camada de transporte. É nessa estrutura onde se armazena a fila de recepção e envio de pacotes, por exemplo. Além dos campos definidos pelo Linux, cada protocolo da camada de transporte estende essa estrutura para armazenar as variáveis de estado que lhe são necessárias. O detalhamento dos principais campos da estrutura `struct sock` encontra-se no Anexo A.2.

Uma aplicação no espaço de usuário cria um *socket* através da chamada de sistema

socket:

```
sockfd = socket(int socket_family, int socket_type, int protocol);
```

O argumento *socket_family* pode ser, por exemplo, `AF_INET` para IPv4 ou `AF_INET6` para IPv6; *socket_type* pode ser `SOCK_STREAM` para TCP ou `SOCK_DGRAM` para UDP; e *protocol*, que pode ser 0 para TCP ou UDP, ou um número de protocolo válido conforme definido pela *Internet Assigned Numbers Authority* (IANA)[2].

3.1.2 Registro de novos protocolos

Apenas os protocolos UDP, ICMP e TCP são tão importantes para o funcionamento do subsistema de redes do Linux que são compilados estaticamente no núcleo do sistema operacional. Por outro lado, existe uma segunda categoria de protocolos de transporte que podem ser adicionados ou removidos do núcleo do Linux de forma dinâmica. Os protocolos desse segundo grupo são implementados através de *Linux Kernel Modules* (LKM). Nesse grupo, encontram-se os protocolos IGMP, SCTP, DCCP e outros.

Os protocolos da camada de transporte baseados no IPv4 são definidos através da estrutura `net_protocol`, que consiste dos seguintes campos:

- *int (*handler)(struct sk_buff *skb)*: Função invocada toda vez que chegar um pacote do protocolo em questão.
- *void (*err_handler)(struct sk_buff *skb, u32 info)*: Função invocada pelo ICMP para informar ao protocolo de transporte a respeito de alguma mensagem de erro.
- *int no_policy*: Este campo é consultado em alguns pontos da pilha de rede e é utilizado para isentar protocolos da checagem do IPsec, caso seu valor seja igual a 1.

Registram-se novos protocolos de transporte através da função `inet_add_protocol`, e removem-se os protocolos através da função `inet_del_protocol`. Todas as estruturas `net_protocol` registradas no núcleo são inseridas em uma tabela chamada `inet_protos`, que é implementada como um *array* comum de 256 posições, sendo cada posição equivalente a um protocolo registrado (BENVENUTI, 2005).

Ao receber um pacote da rede, a função `ip_local_deliver_finish` procura o protocolo de transporte correto para efetuar a chamada da função `handler` correspondente ao pacote recebido. A função lê o valor do campo *Protocolo*, no cabeçalho IP, e acessa a tabela de protocolos registrados, lendo o *array* na exata posição correspondente ao valor do referido campo. Se a tabela não contiver um protocolo válido para esse número, significa que o pacote recebido pertence a um protocolo de transporte que não foi registrado no

[2] Todos os códigos de protocolos constam no documento *Protocol Numbers*, disponível em <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

kernel. Nesse caso, o pacote é descartado e uma mensagem de erro ICMP é enviada de volta ao remetente (BENVENUTI, 2005).

3.2 Novo cabeçalho e tipos de pacotes

Tanto a ordem quanto o tamanho de cada campo da porção fixa do cabeçalho GMTP foram definidos criteriosamente para:

- Permitir manutenção (adição, atualização e remoção) das funcionalidades existentes;
- Flexibilizar o uso do protocolo por diferentes tipos de aplicação multimídia e;
- Otimizar a leitura e escrita dos campos de cada pacote, buscando-se reduzir o atraso nodal.

No entanto, ao estudar a organização do cabeçalho do GMTP a fim de implementá-lo no núcleo do Linux, percebeu-se a necessidade de mudanças na organização dos campos. Além disso, na especificação original do GMTP não constam as definições de organização da porção variável do cabeçalho do GMTP para cada tipo de pacote. No contexto deste trabalho, propõem-se mudanças na organização da parte fixa do cabeçalho do GMTP, bem como uma organização para a porção variável do cabeçalho para cada tipo de pacote do GMTP.

3.2.1 Novo cabeçalho fixo do GMTP

Em um cenário de rede onde há congestionamento, o valor do RTT tende a aumentar. Por essa razão, tanto o GMTP-UCC quanto o GMTP-MCC utilizam o valor do RTT como um dos parâmetros indicativos de congestionamento na rede. Os algoritmos de controle de congestionamento utilizados no GMTP consideram que quanto maior o valor medido do RTT, mais congestionada está a rede, e menor deveria ser a taxa de transmissão de dados. Por essa razão, no GMTP propôs-se o campo *RTT no Servidor*, originalmente com 8 *bits*, possibilitando informar um valor máximo de RTT de 255 *ms*.

No entanto, durante os testes realizados como parte da implementação do GMTP, averiguou-se que, em um cenário de congestionamento de rede durante uma transmissão multimídia, o valor do RTT pode superar o limite de 255 *ms*. Logo, na versão original do GMTP, em uma transmissão ao vivo, caso houvesse um congestionamento e o valor do RTT superasse os 255 *ms*, não seria possível informar o novo valor utilizando o campo *RTT no Servidor* de 8 *bits*, prejudicando o desempenho dos algoritmos de controle de congestionamento.

No contexto deste trabalho, estudaram-se três estratégias para melhorar a precisão da medição do RTT, uma vez que tal informação é amplamente utilizada pelos algoritmos de controle de congestionamento do GMTP:

1. Escalar o valor do campo *RTT no Servidor* de forma que cada unidade corresponda a um período de tempo maior que 1 *ms*. Por exemplo, pode-se considerar que cada unidade do campo *RTT no Servidor* corresponde a 4 *ms*. Assim, se em determinado momento o valor do RTT for de 400 *ms*, o valor armazenado no campo *RTT no Servidor* será igual a 100 . Não utilizou-se essa estratégia porque isso diminuiria a precisão da informação relativa ao RTT, afetando o desempenho dos algoritmos de controle de congestionamento.

2. Utilizar a técnica de ponto flutuante. No protocolo TFMCC, utiliza-se um campo de 8 *bits* em ponto flutuante para representar o RTT máximo de uma transmissão, sendo 4 *bits* para o expoente e 4 *bits* para a mantissa. Assim, representam-se RTTs no valor de 1 *ms* a 64 *s*, com um erro de cerca de 6% (WIDMER; HANDLEY, 2006). Não utilizou-se essa estratégia porque isso diminuiria a precisão da informação relativa ao RTT e aumentaria a carga de processamento dos nós que leem ou escrevem a informação no campo *RTT no Servidor*. Esses dois fatores causariam uma redução no desempenho dos algoritmos de controle de congestionamento.

3. Aumentar a quantidade de bits do campo *RTT no Servidor*, reorganizando o cabeçalho fixo do GMTP. Vale salientar que o cabeçalho do RCP (Seção 2.2.5) dispõe de 16 *bits* para armazenar o valor do RTT (DUKKIPATI, 2007). O RCP é o algoritmo de controle de congestionamento que serve de referência para o GMTP-UCC. Através dessa opção, haveria um aumento do valor máximo do RTT sem perda de precisão e sem aumentar a carga de processamento dos nós que lidam com o referido campo. No entanto, é necessário evitar que o aumento do tamanho do campo acarretasse no aumento do tamanho do cabeçalho fixo do GMTP.

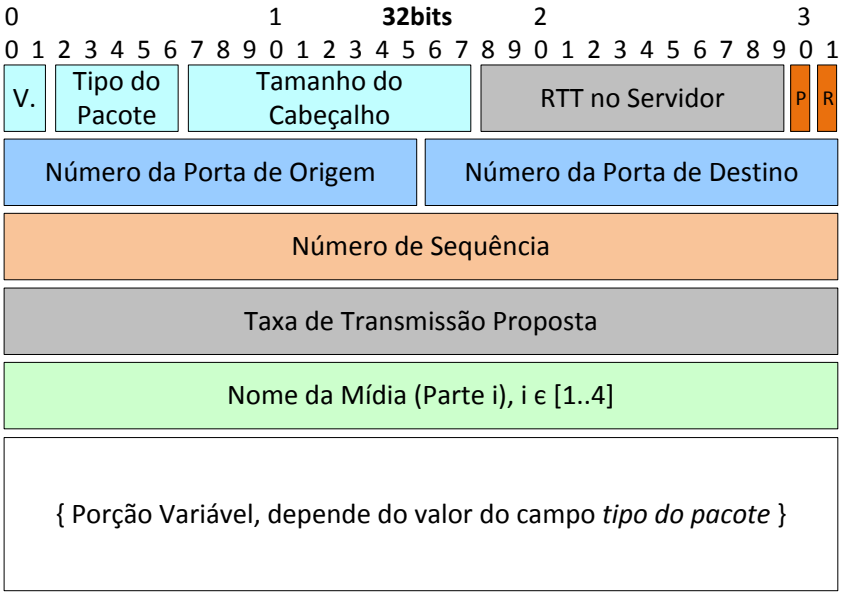
Das opções elencadas acima, escolheu-se a Opção 3. O tamanho do campo *RTT no Servidor* mudou de 8 *bits* para 12 *bits*, tornando-se possível registrar RTT de até 4095 *ms*, que é um valor satisfatório para utilização nos algoritmos de controle de congestionamento do GMTP. Para conseguir ampliar o tamanho do campo *RTT no Servidor* e ao mesmo tempo evitar o aumento do tamanho do cabeçalho fixo do GMTP, reduziu-se o tamanho do campo *Versão do Protocolo* de 3 *bits* para 2 *bits*, e suprimiram-se os 3 *bits* reservados para uso futuro.

A ampliação do campo *RTT no Servidor* permitiu o aumento do valor máximo do RTT sem perda de precisão e sem aumento da carga de processamento dos nós que lidam com o referido campo. Além disso, nos casos onde o valor do RTT supera os 255 *ms*, os

algoritmos de controle de congestionamento do GMTP passaram a receber o valor correto do RTT, e a reagir conforme esperado.

Após as mudanças na organização da parte fixa do cabeçalho do GMTP, este passa a ser organizado conforme ilustra-se na Figura 19.

Figura 19 – Nova organização da porção fixa do cabeçalho de pacotes GMTP.



Definiu-se a nova organização da parte fixa do cabeçalho do GMTP da seguinte forma: 2 *bits* para a *Versão do Protocolo*, 5 *bits* para o *Tipo de Pacote*, 11 *bits* para o *Tamanho do Cabeçalho*, 12 *bits* para o *RTT no Servidor*, 1 *bit* para o campo P (*Pull*), 1 *bit* para o campo R (*Relay*), 16 *bits* para o *Número da Porta de Origem*, 16 *bits* para o *Número da Porta de Destino*, 32 *bits* para os *Números de Sequência*, 32 *bits* para a *Taxa Proposta de Transmissão* e 128 *bits* para o *Nome do Fluxo de Dados*.

As mudanças em relação à organização original foram as seguintes: reduziu-se o tamanho do campo *Versão do Protocolo* de 3 *bits* para 2 *bits*, disponibilizando 1 *bit*. Os 3 *bits* reservados para uso futuro mais o bit que foi retirado do campo *Versão do Protocolo* totalizam 4 bits, que foram alocados para o campo *RTT no Servidor*, que antes tinha 8 *bits* e agora tem 12 *bits*. Originalmente, o campo *RTT no Servidor* com seu espaço de 8 *bits* permitia informar um RTT de no máximo 255 *ms*, que poderia restringir o funcionamento das aplicações em cenários de rede com atrasos acima desse valor, o que é possível na Internet. Com a mudança no cabeçalho fixo do GMTP, o campo *RTT no Servidor* passa a dispor de 12 *bits*, tornando-se possível registrar um RTT de até 4095 *ms*.

3.2.2 Tipos de pacotes e cabeçalhos variáveis

Na especificação original do GMTP não constam as definições de organização da porção variável do cabeçalho do GMTP para cada tipo de pacote. Nesta seção, propõe-se uma organização para a porção variável do cabeçalho para cada tipo de pacote do GMTP.

- *GMTP-RequestNotify*: 2 *bits* para o código do *GMTP-RequestNotify*, 32 *bits* para o endereço IP do canal *multicast* de dados, 16 *bits* para a porta de recepção do canal *multicast* de dados, 128 *bits* para o número de identificação único do repassador que interceptou a requisição do cliente, 32 *bits* para o endereço IP do relator responsável pelo cliente, 16 *bits* para a porta do relator do cliente e 8 *bits* para o número de clientes que estarão registrados no relator (campo *max_nclients*), caso o cliente em questão seja um relator. Logo, o tamanho total da parte variável de um pacote do tipo *GMTP-RequestNotify* é de 234 *bits*.

  O referido pacote serve para que um repassador notifique um cliente que um fluxo de dados está prestes a ser transmitido ou já está sendo transmitido em um determinado canal de repasse *multicast*. Assim, é necessário que o referido pacote contenha um código que informe o estado da conexão, o endereço e porta do canal *multicast*, o identificador do próprio repassador, e os dados referentes ao nó relator vinculado ao cliente. Caso o cliente seja um relator, é necessário informar o número máximo de clientes que lhe serão vinculados.

- *GMTP-Register*: 128 *bits* para o número de identificação único do repassador que enviou a mensagem de registro.

  O referido pacote serve para que um repassador registre sua participação no servidor para funcionar como distribuidor de um fluxo de dados. Logo, é necessário que o repassador informe seu identificador único ao servidor, que servirá como chave de uma tabela *hash* que o servidor utiliza para armazenar informações sobre os repassadores registrados.

- *GMTP-Register-Reply*: 2 *bits* para o tipo de controle de congestionamento *unicast*, 8 *bits* para a quantidade de repassadores que adicionou seu próprio identificador no pacote, e até 69 identificadores de repassadores, sendo cada identificador composto de 160 *bits* (chave MD5 + endereço IP), num total de 11040 *bits* ou 1380 *Bytes*. Logo, o tamanho da parte variável de um pacote do tipo *GMTP-Register-Reply* varia entre 10 *bits* e 11050 *bits*.

  O referido pacote serve para que o servidor responda sobre o pedido de registro de participação enviado por um repassador. Além disso, transporta o identificador de todos os repassadores entre o servidor e o cliente. Logo, é necessário que o servidor informe o tipo de controle de congestionamento (se os roteadores devem ou não

utilizar algoritmos de adaptação de fluxo para reduzir a taxa de transmissão). Além disso, os repassadores incrementam o valor do campo que armazena a quantidade de repassadores registrados no pacote e adicionam seu próprio identificador ao final do referido pacote.

- *GMTP-Route-Notify*: idêntico à parte variável dos pacotes do tipo *GMTP-Register-Reply*.

  O referido pacote serve para informar a rota entre um repassador e servidor. O repassador envia esse tipo de pacote ao servidor, após receber o pacote do tipo *GMTP-Register-Reply*. Assim, a sua parte variável é idêntica à parte variável do pacote do tipo *GMTP-Register-Reply*.

- *GMTP-Data*: 32 *bits* para a marca de tempo do emissor.

  O referido pacote serve para transportar os dados da mídia transmitida. Com o intuito de calcular o RTT entre o servidor e os repassadores, é necessário armazenar no referido pacote a marca de tempo do momento em que o pacote foi enviado.

- *GMTP-Ack*: 32 *bits* para a marca de tempo do pacote que está sendo confirmado.

  O referido pacote serve para confirmação de recebimento e controle de congestionamento, a depender do contexto. Quando utilizado para controle de congestionamento, é necessário armazenar a marca de tempo que estava gravada no último pacote de dados recebido. Assim, quando o servidor receber o pacote do tipo *GMTP-Ack*, calculará o RTT subtraindo a marca de tempo de quando recebeu o ACK pela marca de tempo gravada no ACK (que corresponde à marca de tempo do pacote de dados que originou o ACK).

- *GMTP-DataAck*: 32 *bits* para a marca de tempo do emissor e 32 *bits* para a marca de tempo do pacote que está sendo confirmado, totalizando 64 *bits*.

  O referido pacote é uma combinação dos pacotes *GMTP-Data* e *GMTP-Ack* (*Piggy-Back*).

- *GMTP-Elect-Request*: 128 *bits* para o número de identificação único do repassador de origem do cliente e 8 *bits* para o número de clientes que estarão registrados no futuro relator.

  O referido pacote serve para que o repassador envie para um cliente uma solicitação para que este atue como relator. Logo, é necessário um campo para que o repassador seja identificado (através do seu identificador único), e um campo informando quantos clientes podem ser vinculados ao novo relator, caso este aceite o novo papel.

- *GMTP-Elect-Response*: 2 *bits* para a resposta do cliente, que aceita ou não a responsabilidade de ser um relator.

O referido pacote serve para que o cliente envie ao repassador a confirmação de que aceita ou não atuar como relator. Logo, é necessário um campo que contenha um código para a resposta ao repassador.

- *GMTP-Reset:* 8 *bits* para o código da mensagem de *RESET* e 24 *bits* para os dados adicionais da mensagem, totalizando 32 *bits*.

  O referido pacote serve para determinar, incondicionalmente, a finalização de uma conexão. Assim, são necessários campos para que o emissor justifique apropriadamente o fim da conexão ao receptor.

Nem todos os tipos de pacote necessitam da porção variável do cabeçalho GMTP. Os pacotes dos tipos *GMTP-Request*, *GMTP-Response*, *GMTP-RelayQuery*, *GMTP-RelayQuery-Reply*, *GMTP-MediaDesc*, *GMTP-DataPull-Request*, *GMTP-DataPull-Response* e *GMTP-Close* não necessitam da porção variável.

Visando a execução de ações definidas no protocolo GMTP, além de ações não previstas na proposta inicial do protocolo, propõe-se a adição de novos tipos de pacote além dos tipos já definidos originalmente no GMTP. Os tipos adicionais são definidos a partir do número 18, conforme se segue:

18. *GMTP-Feedback*: o relator envia seu relatório de acordo com o GMTP-MCC. Sua parte flexível é organizada sendo 32 *bits* para a marca de tempo do último pacote *GMTP-Data* recebido e 8 *bits* para o número de clientes que estão ativos e registrados no relator, conforme explicado na Seção 3.4.2, totalizando 40 *bits*.

    Assim como nos pacotes do tipo *GMTP-Ack*, é necessário armazenar a marca de tempo que estava gravada no último pacote de dados recebido para que o repassador calcule o RTT da rede local apropriadamente. Pode-se considerar que um pacote do tipo *GMTP-Feedback* é um pacote *GMTP-Ack* estendido, com a informação adicional sobre o número de clientes vinculados ao relator e ativos.

19. *GMTP-Delegate*: o servidor delega a um repassador a responsabilidade de repassar um determinado fluxo de dados a outro repassador parceiro. Sua parte flexível é organizada sendo 128 *bits* para o número de identificação único do repassador que está sendo delegado ao destinatário, 32 *bits* para o endereço IP do repassador delegado e 16 *bits* para a porta do repassador delegado, totalizando 176 *bits*.

    Para que o repassador destinatário saiba quem será o seu novo parceiro (repassador delegado), o referido pacote precisa carregar as informações sobre a identificação única do repassador delegado, bem como seu endereço IP e porta de conexão.

20. *GMTP-Delegate-Reply*: o repassador envia ao servidor a confirmação do pacote *GMTP-Delegate* recebido. Não foi necessário definir parte flexível para este tipo de pacote.

21. *Reservado:* a partir do identificador 21 até o 31, tratam-se de valores reservados para uso futuro e os pacotes com esses valores devem ser descartados pelos nós que os processam.

3.3 Implementação dos Módulos do GMTP

Implementaram-se os módulos através de *Linux Kernel Modules* (LKM). Assim, é possível carregar os módulos do GMTP no *kernel*, descarregar parte do GMTP e futuramente estender o GMTP desenvolvendo novos LKMs.

3.3.1 Consideração sobre a arquitetura *cross-layer* do GMTP no Linux

Ao longo da pesquisa sobre o GMTP, questionou-se se o GMTP é um protocolo *cross-layer* (SRIVASTAVA; MOTANI, 2005). Argumentou-se que parte das funcionalidades do GMTP-Inter, que deveria estar na camada de rede, pertencem à camada de transporte. De fato, funcionalidades do GMTP-Inter como controle de congestionamento e transmissão de dados utilizando *unicast*/*multicast* IP são típicas da camada de transporte, considerando estritamente a organização de protocolos na arquitetura TCP/IP.

No entanto, existem funções do módulo GMTP-Inter que interferem na rota dos pacotes de controle ou de mídia, modificando os endereços IP e MAC de origem ou de destino, quando necessário. Ao alterar o endereço de destino de um pacote, o GMTP-Inter precisa encaminhar o pacote para a *interface* de rede correta, a fim de transmiti-lo ao nó de destino. Essa não é uma tarefa dos protocolos da camada de transporte. Isso pode ocorrer antes ou depois dos pacotes serem roteados, conforme discute-se na Seção 3.3.3. Essas funções visam oferecer a configuração dinâmica de canais *multicast* e o redirecionamento de fluxos *unicast*, viabilizando o gerenciamento de parcerias entre os roteadores. Na prática, esta configuração dinâmica de quando se deve utilizar *multicast* e/ou *unicast* é um dos itens que diferencia o GMTP de outros protocolos existentes.

Nesse contexto, o GMTP ultrapassa os limites de atuação de protocolos da camada de transporte ao acessar diretamente as camadas inferiores para modificar o endereço IP ou MAC de origem ou destino dos pacotes com o intuito de melhorar o desempenho em uma transmissão de datagramas através da rede. O GMTP-Inter acessa a camada de rede sem a utilização de *sockets* ou funções disponíveis para os protocolos de transporte do Linux. Esse enfraquecimento das restrições e responsabilidades exclusivas de cada camada da

pilha TCP/IP é típica de protocolos *cross-layer*. Portanto, o GMTP deve ser considerado um protocolo *cross-layer* atuando nas camadas de transporte e de rede da pilha TCP/IP.

3.3.2 GMTP-Intra

O GMTP-Intra compreende a instância do GMTP em execução no sistema operacional do cliente ou do servidor, acessível através de uma API de *socket* GMTP. Implementou-se o módulo GMTP-Intra através de dois *Linux Kernel Modules* (LKM) distintos, denominados *gmtp.ko* e *gmtp_ipv4.ko*. No primeiro módulo, encontram-se as funções e variáveis de estado relativas ao GMTP que são independentes da versão do *Internet Protocol* (IPv4 ou IPv6). O segundo módulo diz respeito às funções e variáveis de estado relativas ao GMTP que são dependentes do IPv4. No contexto deste trabalho, o LKM do GMTP específico para o IPv6 não foi desenvolvido.

Uma vez que os módulos do GMTP estejam habilitados no sistema, o desenvolvedor de aplicações deve abrir um *socket* do tipo GMTP utilizando qualquer linguagem de programação que forneça uma API compatível com as especificações de *socket* BSD e POSIX.

O GMTP automaticamente gera o identificador do fluxo de dados (identificador da mídia) a partir do IP e porta fornecidos no momento da criação do *socket*, conforme descrito na Seção 2.2.1.1. Opcionalmente, o identificador da mídia pode ser definido pelo desenvolvedor da aplicação através do uso da função *setsockopt*. A lista completa de todas as opções de *socket* do GMTP encontra-se no Apêndice B.

3.3.2.1 Tabelas do GMTP-Intra

No módulo GMTP-Intra, utilizam-se algumas tabelas *hash* para diferentes propósitos, armazenando-se informações úteis para clientes e servidores GMTP.

A fim de armazenar informações sobre os clientes GMTP em determinado sistema final, implementou-se uma tabela *hash* cuja chave é o identificador do fluxo de mídia F. Cada entrada dessa tabela de clientes contém, além da chave F, o endereço IP e porta do canal *multicast* para recepção dos dados e a lista de todos os clientes que solicitaram o fluxo de mídia F no sistema final.

No GMTP-Intra, há outra tabela *hash*, desta vez para os servidores, cuja chave também é o identificador do fluxo de mídia F. Além da chave F, a tabela do servidor contém a lista de rotas W do servidor aos repassadores, a quantidade de repassadores registrados e uma tabela *hash* que armazena informações de cada repassador registrado. Na tabela *hash* de repassadores, utiliza-se o identificador único do repassador r como chave. Além de sua chave, a tabela de repassadores contém um ponteiro para a rota W_r do servidor até r, o endereço IP e um apontador para o *socket* correspondente à conexão

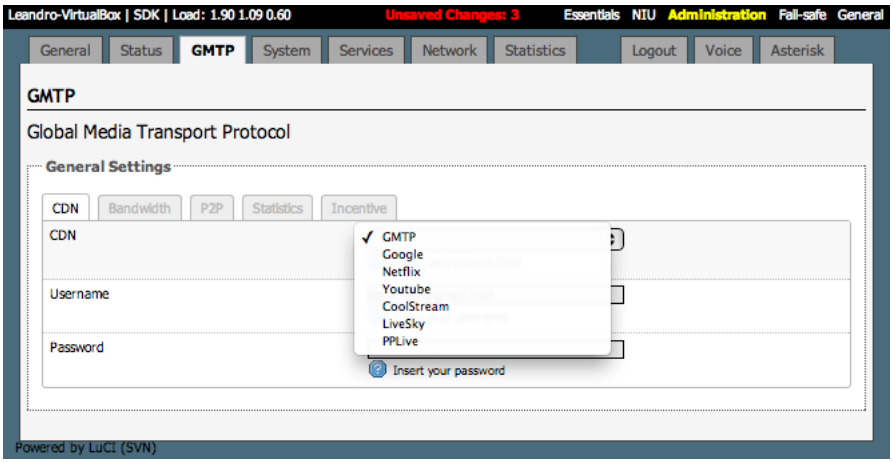
entre o servidor e o repassador r. Através do registro da lista de rotas e da tabela de repassadores, permite-se que o servidor utilize essas informações para instruir os nós repassadores a realizarem parcerias, a fim de distribuir um fluxo de dados F.

3.3.3 GMTP-Inter

O GMTP-Inter tem como responsabilidade constituir as redes de favores P2P compostas por roteadores. Trata-se do módulo em execução nos roteadores, que cooperam entre si a fim de constituírem as redes de favores, com base no interesse comum de um determinado conteúdo. Através do GMTP-Inter, os roteadores aceitam conexões originadas por seus clientes, bem como instruções dos servidores para formar parcerias com outros roteadores a fim de disseminar um determinado conteúdo (SALES, 2014).

Nesse contexto, implementou-se o GMTP-Inter como um roteador baseado em *software*. A vantagem é que não foi necessário nenhum suporte especial do *hardware*, visto que utilizou-se a infraestrutura disponível no núcleo do Linux (DUKKIPATI, 2007). Desta forma, torna-se possível integrar o GMTP-Inter em uma distribuição Linux direcionada a roteadores como, por exemplo, o OpenWrt[3]. Na Figura 20, ilustra-se uma sugestão para o *layout* da tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP.

Figura 20 – Tela da ferramenta de administração da distribuição Linux OpenWRT com suporte ao GMTP. Nessa tela, permite-se que o administrador do roteador configure parâmetros do módulo GMTP-Inter.



Fonte – Sales (2014)

A operação de um roteador pode ser dividida em duas partes: o plano de dados e o plano de controle. No plano de dados, os pacotes recebidos são processados e encaminhados ao seu destino. As tarefas dessa parte incluem verificar o *checksum* do cabeçalho IP, extrair

[3] OpenWrt é uma distribuição do GNU/Linux, personalizável, direcionada a sistemas embarcados (geralmente roteadores *wireless*). Mais informações em <https://openwrt.org/>.

o endereço de destino do cabeçalho IP, encontrar seu próximo salto na tabela de roteamento, decrementar o *time-to-live* do pacote, atualizar o *checksum* e encaminhar o pacote pela interface de rede correta. No plano de controle, realizam-se tarefas menos frequentes, como a atualização das tabelas de roteamento (DUKKIPATI, 2007).

Um roteador habilitado com o GMTP-Inter precisa oferecer a mesma taxa de transmissão para todos os fluxos transmitidos através deste roteador. Essa taxa é calculada através da Equação 2.3 e gravada no cabeçalho GMTP dos pacotes, conforme descrito na Seção 2.2.5. Para calcular a taxa, o roteador precisa manter uma estimativa média do RTT, utilizando a informação sobre o RTT disponível no cabeçalho do GMTP. Estatísticas, como o tráfego total recebido e o RTT médio são calculadas durante cada intervalo de controle e utilizados para calcular a taxa de transmissão.

As funcionalidades do GMTP-Inter são, resumidamente:

1. Plano de dados (processamento por pacote):
   - identificar os pacotes GMTP recebidos;
   - atualizar estatísticas sobre o tráfego recebido e enviado;
   - escrever a taxa de transmissão ao encaminhar um pacote;
   - encaminhar os pacotes recebidos aos seus destinos.
2. Plano de controle (processamento periódico):
   - calcular a taxa de transmissão, conforme Equação 2.3;
   - atualizar o RTT médio, conforme descrito na Seção 2.2.5;
   - enviar pacotes de *keep-alive* aos servidores ou repassadores parceiros, conforme descrito na Seção 3.4.3.1.

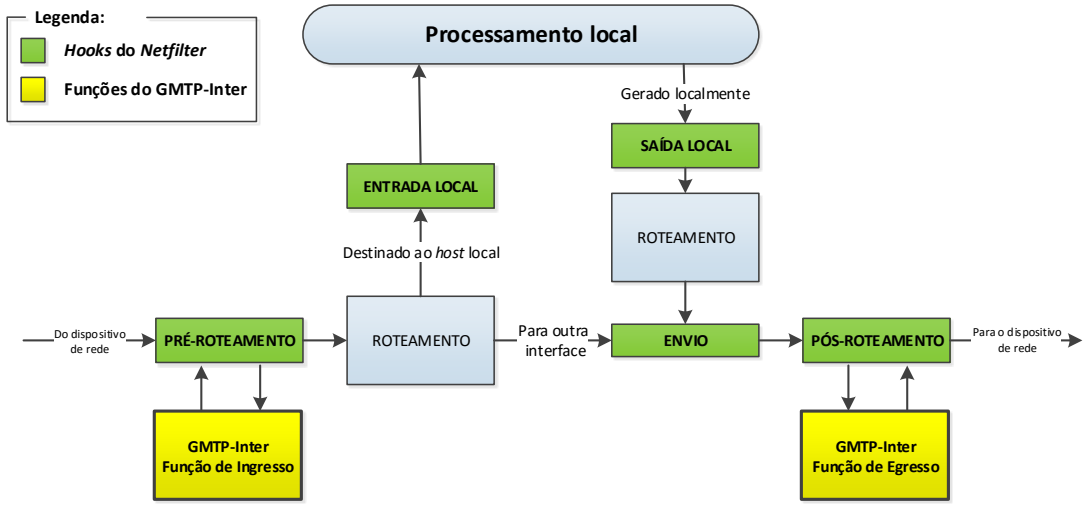
Implementou-se o GMTP-Inter em um sistema Linux padrão, através de um *Linux Kernel Module* (LKM), denominado *gmtp_inter.ko*. Desta forma, é possível ativar ou desativar o GMTP-Inter em um roteador baseado em *software*, apenas inserindo ou removendo o LKM.

As opções de configuração do GMTP-Inter podem ser acessadas e modificadas através de opções de *socket*, utilizando-se a função *setsockopt*. Para tanto, deve-se instanciar um *socket* GMTP comum e configurar a opção de *socket* `GMTP_SOCKOPT_ROLE_RELAY` com o valor 1. A partir desse momento, o *socket* GMTP torna-se exclusivo para configuração do GMTP-Inter na máquina local, sendo vedada a utilização de tal *socket* para envio e recebimento de dados.

O plano de controle do GMTP-Inter diz respeito aos *timers* que controlam as funções relacionadas ao controle de congestionamento e mecanismos de *keep-alive* e manutenção de parcerias.

Para a implementação do plano de dados, utilizou-se a ferramenta do Linux denominada *Netfilter*, a qual permite desenvolver operações personalizadas para cada pacote em processamento no *kernel*. As operações do plano de dados do GMTP-Inter são efetuadas através dos *hooks* do Netfilter. Apenas os pacotes GMTP são manipulados pelo GMTP-Inter. Nele, registram-se as funções de ingresso de pacotes através do *hook* `NF_INET_PRE_ROUTING` e as funções de egresso através do *hook* `NF_INET_POST_ROUTING`, conforme ilustra-se na Figura 21.

Figura 21 – GMTP-Inter: Locais onde os *hooks* do Netfilter foram configurados.



3.3.3.1 Tabelas do GMTP-Inter

Cada repassador mantém uma tabela chamada *Tabela de Recepção de Fluxos de Dados*, conforme ilustra-se na Figura 7. Além das informações definidas na especificação original do GMTP, os repassadores precisam armazenar informações a respeito do estado da transmissão do fluxo de dados.

Dentre as informações, as mais relevantes são:

- *Estado da transmissão*: define o estado atual da transmissão, que pode ser: aguardando o *GMTP-Register-Reply* do servidor, recebendo o *GMTP-Register-Reply*, transmitindo a mídia, encerrando transmissão, e transmissão encerrada.
- *Endereço do repassador*: endereço IP pelo qual o repassador recebe os pacotes do servidor.
- *Porta do repassador*: porta pela qual o repassador recebe os pacotes do servidor.

- *Taxa de transmissão*: taxa nominal de transmissão do servidor
- *Taxa de recepção*: taxa real de recepção de dados oriundos do servidor, medida periodicamente.
- *Timer do GMTP-MCC*: *timer* para efetuar as operações do GMTP-MCC, conforme descrito na Seção 3.4.2.2.
- *Número de feedbacks*: número de pacotes *GMTP-Feedback* recebidos. Esse contador é zerado a cada intervalo de tempo, visando implementar os mecanismos descritos na Seção 3.4.2.2.
- *Somatório dos valores de feedback*: somatório das taxas de transmissão informadas nos pacotes *GMTP-Feedback* recebidos. Esse somatório é zerado a cada intervalo de tempo, visando implementar os mecanismos descritos na Seção 3.4.2.2.
- *Lista de relatores*: lista dos relatores registrados no repassador para receber o fluxo de dados F.
- *Número de relatores*: número de relatores registrados no repassador.
- *Número de clientes*: número de clientes registrados no repassador que estão recebendo o fluxo de dados F. Atualizado periodicamente conforme relatório dos relatores.
- *Buffer de recepção*: por se tratar de um roteador baseado em *software*, o módulo GMTP-Inter isola o *buffer* de recepção de cada fluxo de dados F.
- *Timer do GMTP-UCC*: *timer* para envio da taxa de transmissão ideal calculada via GMTP-UCC.
- *Timer de Renovação de Registro*: *timer* para efetuar a renovação do registro de participação no servidor.
- *Lista de repassadores "clientes"*: lista dos repassadores parceiros registrados para receber o fluxo de mídia. Cada entrada da lista contém o endereço IP, porta, endereço MAC, estado da conexão com o parceiro e a taxa de transmissão que o parceiro solicita (tornando possível a definição de sub-rotas do GMTP-UCC).
- *Número de repassadores "clientes"*: número de repassadores parceiros registrados para receber o fluxo de mídia.

3.4 Funcionamento do Global Media Transmission Protocol

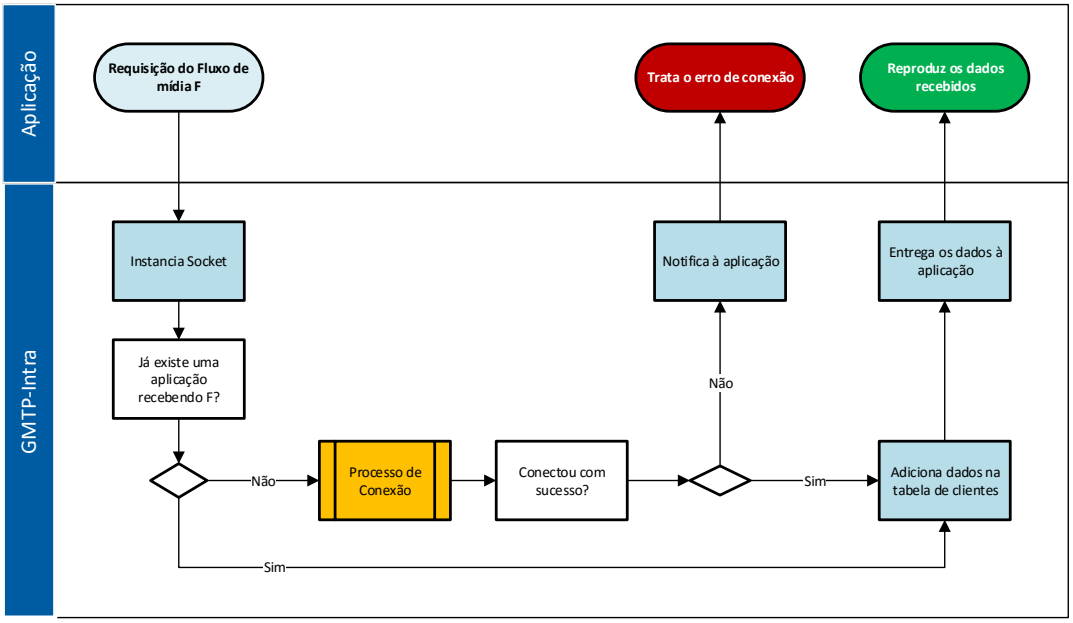
O fluxo básico de comunicação do GMTP ocorre quando um cliente deseja reproduzir um conteúdo multimídia. A seguir, descreve-se detalhadamente o funcionamento prático do GMTP, conforme implementação no núcleo do Linux.

3.4.1 Requisição da mídia ao vivo pelos clientes

Os clientes GMTP sempre se conectam ao servidor através do seu endereço IP (*unicast*) e porta. Para tanto, instancia-se um *socket* cliente GMTP como se fosse um *socket unicast*. Após o estabelecimento da conexão, um cliente GMTP pode ler os dados enviados pelo servidor utilizando as funções comuns de leitura de dados via *socket*. A aplicação cliente reproduz o conteúdo multimídia ao usuário final à medida que recebe os pacotes de dados contendo partes da mídia, gerada por um ou mais servidores.

Conforme ilustra-se na Figura 22, ao requisitar um fluxo de mídia F, o GMTP-Intra consultará a tabela de clientes para verificar se algum cliente local requisitou com sucesso a mídia. Caso alguma aplicação cliente do mesmo sistema final já requisitou o mesmo fluxo de mídia F, o GMTP-Intra não enviará um novo pacote *GMTP-Request*, mas aproveitará a primeira requisição para servir ao novo cliente. Caso contrário, o GMTP-Intra enviará um pacote do tipo *GMTP-Request* endereçado ao servidor, configurando o campo *Nome da mídia* do cabeçalho do GMTP com o valor de F e configurando o campo *TTL* do cabeçalho IP com o valor 1. Em ambos os casos, o GMTP-Intra registrará os dados do cliente em sua tabela de clientes, utilizando a chave F.

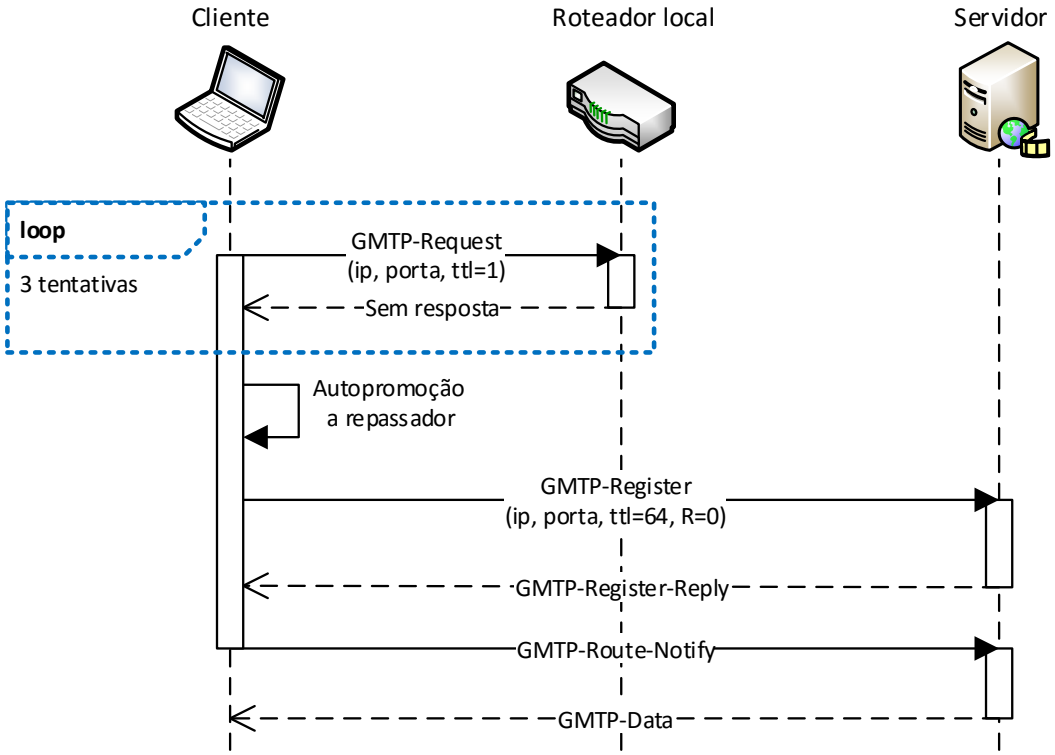
Figura 22 – Fluxograma do processo de requisição de mídia pela aplicação no nó local.



Caso não haja nenhum nó repassador na rota entre o cliente e o servidor, o cliente se "autopromove" a repassador. Conforme ilustra-se na Figura 23, o cliente GMTP deve fazer três tentativas de conexão utilizando um pacote do tipo *GMTP-Request*, endereçado ao servidor e com TTL=1. Caso as tentativas falhem, assume-se que o roteador do cliente não está habilitado com o GMTP e o cliente envia ao servidor pacote do tipo *GMTP-Register*, com o TTL padrão do sistema operacional. É importante salientar que nesse caso o cliente não assume todas as responsabilidades de um repassador, como por exemplo, repasse

do fluxo de mídia a outros repassadores ou a clientes através do canal *multicast*. Caso o servidor aceite a conexão, enviará um pacote do tipo *GMTP-Register-Reply* de volta ao cliente. O servidor sabe se a requisição partiu de um cliente que se autopromoveu porque o campo R do cabeçalho GMTP é igual a zero. Quando um cliente se conecta diretamente ao servidor, sem intermédio de um repassador, recebe os pacotes de mídia que são diretamente endereçados pelo servidor, através do seu IP e porta de requisição.

Figura 23 – Procedimento para autopromoção de cliente para repassador.



O repassador de origem do cliente deve interceptar pacotes *GMTP-Request* com TTL=1 para fins de conexão. Caso o repassador ainda não esteja recebendo a mídia F solicitada pelo cliente, ele deve registrar F em sua tabela de repasse e encaminhar um pacote do tipo *GMTP-Register* ao servidor. Nesse caso, o pacote *GMTP-Register* contém o valor 1 (um) no campo R do cabeçalho GMTP e o identificador único do repassador que está se registrando na parte variável de seu cabeçalho, conforme descrito na Seção 3.2.2.

O repassador, ao receber a requisição do cliente, envia um pacote do tipo *GMTP-RequestNotify* como resposta à requisição. O conteúdo do campo *Código* da parte variável do cabeçalho do pacote *GMTP-RequestNotify* pode conter os seguintes valores:

0. *OK:* o repassador envia o pacote *GMTP-RequestNotify* com este valor apenas após o pleno estabelecimento da conexão com o servidor. Ao receber o pacote com este

valor, o cliente deve considerar que a conexão está aberta e deve responder com um pacote do tipo *GMTP-Ack*. O cliente então deve aceitar os pacotes endereçados ao endereço *multicast* informado no pacote recebido. Após receber o pacote *GMTP-Ack*, o repassador passa a encaminhar os dados de mídia provenientes do servidor ao endereço *multicast* correspondente.

1. *WAIT:* o repassador envia o pacote *GMTP-RequestNotify* com este valor apenas enquanto aguarda a resposta do servidor ao seu pedido de conexão. Apenas os clientes que requisitaram a mídia antes do registro de participação do repassador é que recebem o pacote com o valor *WAIT*. Ao receber o pacote com este valor, o cliente deve apenas aguardar o recebimento do pacote *GMTP-RequestNotify* com o valor do campo *Código* igual a *OK*.

2. *ERROR:* indica um erro no formato da requisição do cliente ou a recusa do servidor em atender à requisição. Ao receber este pacote, o cliente pode tentar requisitar a mídia novamente, até o limite de 3 tentativas.

Após o estabelecimento da conexão com o repassador, o cliente então deve aceitar os pacotes de dados enviados ao endereço *multicast* informado no pacote *GMTP-RequestNotify* recebido. Para tanto, localiza-se o *socket* GMTP correspondente ao pacote recebido e o envia à aplicação.

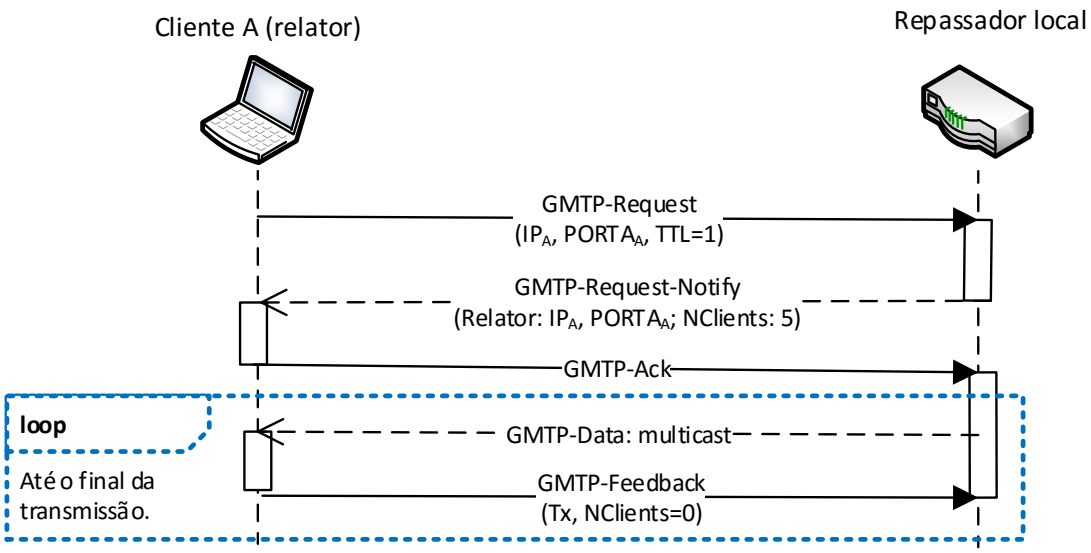
3.4.2 Relatores

Conforme descrito na Seção 2.2.6, o primeiro nó relator será o cliente que iniciar a primeira conexão para obter o fluxo de dados F, e os seguintes relatores serão os próximos clientes que se conectarem para receber o fluxo de dados, seguindo uma relação que por padrão é de um relator para cada seis clientes.

Conforme ilustra-se na Figura 24, ao enviar o pacote *GMTP-RequestNotify* ao cliente, o repassador sinaliza que o cliente em processo de conexão deverá se comportar como um relator. Tal sinalização se dá através dos campos *Endereço IP do Relator*, *Porta do Relator* e *Número máximo de clientes conectados* (doravante denominado *max_nclients*). Ao receber um pacote *GMTP-RequestNotify*, o cliente deve se comportar como relator apenas se os campos *Endereço IP do Relator* e *Porta do Relator* forem idênticos aos seus próprios endereço IP e porta, e no campo *max_nclients* constar um valor maior que zero. O número máximo de clientes conectados a um relator é informado através do campo *max_nclients*, de 8 *bits*, variando de de 0 a 255.

A regra é que se a proporção de clientes por relator for de $K : 1$, o número máximo de clientes registrados em um relator deve ser $K - 1$. Por exemplo, se a proporção de

Figura 24 – Eleição do primeiro nó relator.



clientes por relator for de 6 : 1, o número máximo de clientes conectados por relator será de 5 clientes e o valor que deve constar no campo *max_nclients* é 5.

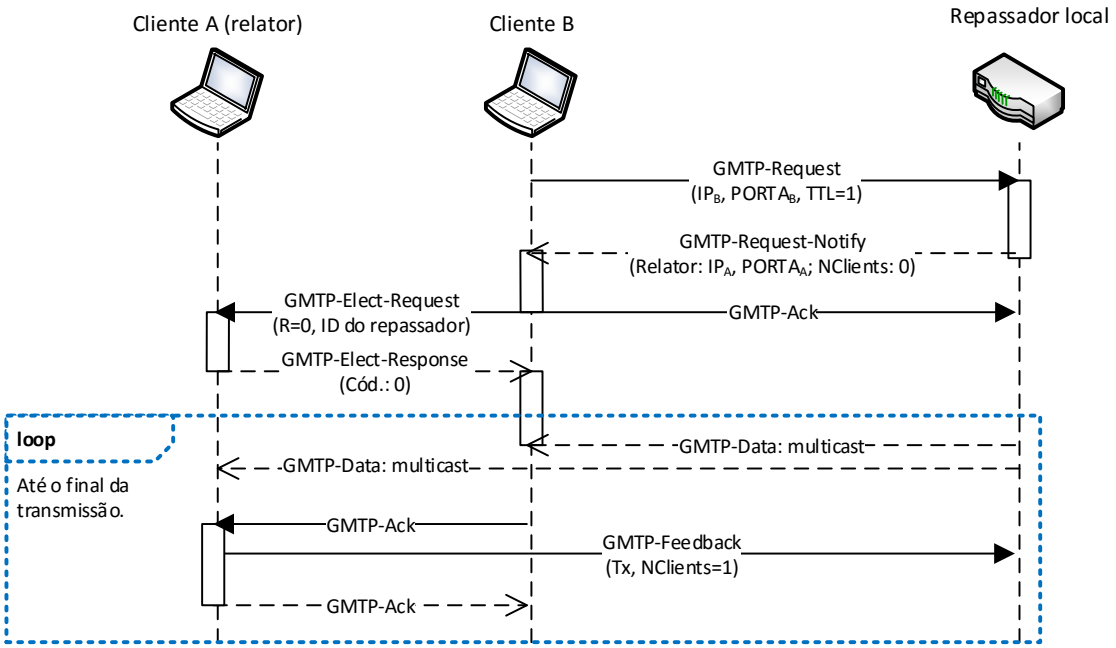
O nó relator, ao enviar seu relatório ao repassador, deve informar: (1) a marca de tempo do último pacote de dados recebido, (2) quantos clientes estão conectados ao relator e (3) a taxa de transmissão proposta, através do campo *Taxa de transmissão* do cabeçalho fixo do GMTP. Os relatórios dos relatores também servem como mecanismos de *keep-alive* para os seus repassadores.

3.4.2.1 Relação entre clientes e relatores

Conforme ilustra-se na Figura 25, os clientes devem se conectar ao relator indicado nos respectivos campos do pacote *GMTP-RequestNotify* recebido. Isso ocorre através do envio de um pacote do tipo *GMTP-Elect-Request* ao relator, onde o campo *Relay (R)* do cabeçalho fixo do GMTP deve ser igual a 0, indicando que o referido pacote foi enviado por um cliente na rede local. O campo *Identificação do Repassador* do cabeçalho da requisição *GMTP-Elect-Request* deve ser preenchido com o identificador do repassador do cliente, para que o relator, ao receber a requisição, confirme que o cliente está vinculado ao mesmo repassador do relator. O relator deve responder à requisição com um pacote do tipo *GMTP-Elect-Response*, configurando o campo *Código* com o valor 0, caso aceite a conexão, ou com o valor 1, caso a rejeite. O relator deve rejeitar a requisição caso já tenha atingido o seu número máximo de clientes registrados.

Um vez registrado no relator, o cliente deve enviar periodicamente um pacote do tipo *GMTP-Ack* ao relator, como forma de confirmação de presença, o chamado *keep-alive*. No GMTP, o intervalo de tempo padrão para envio do *keep-alive* é de 1 *s*. O relator então deve enviar um *GMTP-Ack* como resposta ao cliente. Caso o relator não receba

Figura 25 – Conexão entre clientes e relatores.



o *keep-alive* de um cliente no intervalo de 4 *s* (*timeout*), deve considerar que o cliente está inativo, e desfazer a conexão com o cliente. Assim, quando um cliente comum sair da rede, ele não precisa enviar nenhum tipo de notificação ao seu relator, visto que o relator realiza um procedimento de *timeout*. De forma semelhante, caso o cliente, após envio do *GMTP-Ack* ao relator, não receba a confirmação de recebimento em 4 *s*, deve se autopromover a relator, enviando um pacote do tipo *GMTP-Elect-Response* ao seu repassador.

Esse procedimento de registro dos clientes nos relatores se faz necessário porque o repassador precisa ter conhecimento da quantidade de clientes que está recebendo a mídia. Caso um repassador não esteja cooperando com nenhum outro repassador na rede e nenhum de seus clientes esteja mais recebendo a mídia, o repassador pode finalizar a conexão com o servidor. Para tanto, o repassador deve armazenar na sua tabela de repasse uma estrutura contendo a listagem de todos os relatores da rede local. O número máximo de relatores em uma rede local é de 255. A estrutura deve conter o endereço IP, porta e número de clientes conectados ao relator. Dessa forma, o repassador tem conhecimento da quantidade de clientes ativos, sem ter que armazenar informação sobre o estado de todos os clientes na rede local. No pior caso, o repassador terá que armazenar informação de 255 relatores, independentemente da proporção de relatores por clientes.

O relator, ao enviar o relatório periódico ao repassador, além da taxa de transmissão ideal, deve também informar a quantidade de clientes ativos e registrados nele, através do campo *Número de Clientes* do pacote *GMTP-Feedback*.

3.4.2.2 Implementação do GMTP-MCC

No GMTP-MCC, o próprio relator, baseado no relatório de perdas, calcula a taxa de transmissão adequada utilizando a Equação 2.4 e envia o relatório ao receptor. Para envio dos relatórios, utilizam-se pacotes do tipo *GMTP-Feedback*. Os relatórios são enviados a cada evento de perda ou, caso não haja perdas, a cada RTT. Cabe ao repassador, após receber os relatórios dos relatores, definir a nova taxa de transmissão, através da média aritmética de todas os valores de taxas de transmissão recebidas no intervalo de 4 RTTs.

Se o repassador não receber nenhum relatório no intervalo de 4 RTTs, deve reduzir a taxa de transmissão pela metade, até certa fração da taxa de transmissão requerida pelo servidor. Essa fração diz respeito à taxa mínima de envio de dados definida pelo servidor, considerando a menor qualidade de transmissão tolerável em determinada transmissão. No contexto deste trabalho, utilizou-se o valor de $\frac{1}{8}$ da taxa de transmissão requerida pelo servidor como a taxa mínima de envio de dados tolerável para a transmissão.

No protocolo TFMCC, definiu-se que caso o emissor não receba nenhum relatório do *current limiting receiver*(CLR) (No TFMCC, é a figura análoga ao relator GMTP) por pelo menos 10 RTTs, deve assumir que o CLR está inativo ou saiu do grupo *multicast* (WIDMER; HANDLEY, 2006). Nos testes realizados com repassadores e clientes GMTP, observou-se que a utilização de um *timeout* de 12 RTTs apresenta resultados melhores que a utilização de um *timeout* de 10 RTTs. Portanto, no GMTP-MCC definiu-se que se o repassador não receber nenhum relatório de determinado relator por pelo menos 12 RTTs, o repassador deve assumir que o relator não está mais ativo e deve selecionar um novo relator através do mecanismo de eleição de relatores descrito a seguir.

3.4.2.3 Eleição de novos relatores

Um nó relator pode sinalizar explicitamente sua desistência de participação de forma semelhante a um nó cliente, com a diferença que ele precisa continuar enviando os relatórios até que o repassador lhe envie um pacote do tipo *GMTP-Reset*, confirmando que outro relator foi escolhido pelo repassador. Para tanto, o pacote *GMTP-Close* oriundo do relator que está saindo deve ter a lista dos clientes que lhe foram designados, pois o repassador não armazena a lista de clientes de cada relator.

Na especificação original do GMTP, preconiza-se que o nó relator em processo de desconexão deve continuar enviando pacotes do tipo *GMTP-Feedback* para o nó repassador, até que um novo relator seja eleito. Consequentemente, o nó relator também deve manter a conexão com os seus clientes designados até a eleição de um novo relator.

Ainda de acordo com a especificação original, quando o nó repassador receber o pacote do tipo *GMTP-Close*, este deve verificar se o referido nó cliente é um nó relator. Em caso afirmativo, o repassador deve transmitir para um dos clientes que também recebe o

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

No contexto deste trabalho, implementou-se parcialmente o procedimento de eleição de um novo relator. Não implementou-se o procedimento de eleição de um novo relator quando este sai da rede. No entanto, implementou-se o mecanismo que possibilita que um cliente se autopromova a relator após determinado período, caso seu relator saia da rede.

3.4.3 Registro de participação no servidor

Ao receber um pacote do tipo *GMTP-Request* referente a um fluxo de mídia F pela primeira vez, o repassador deve registrar F em sua tabela de repasse e encaminhar um pacote do tipo *GMTP-Register* ao servidor. Na parte variável do cabeçalho do pacote *GMTP-Register* deve constar o identificador único do repassador.

Ao receber o pacote *GMTP-Register*, o servidor deve abrir um *socket* de conexão com o repassador, assim como no TCP. O servidor então compara se o valor do campo *Nome do fluxo* que consta no pacote recebido é idêntico ao identificador do fluxo F referente à mídia disponível no servidor. Caso seja diferente, o servidor rejeita a conexão, enviando um pacote do tipo *GMTP-Reset* ao repassador.

Uma vez aceita a conexão, o servidor responde ao repassador com um pacote do tipo *GMTP-Register-Reply*. Ao enviar o referido pacote, o servidor registra a marca de tempo no instante do envio a fim de, posteriormente, calcular o valor do RTT. Conforme descrito na Seção 2.2.3, o uso do pacote do tipo *GMTP-Register-Reply* permite a descoberta de uma rota entre o servidor e o repassador. Isto porque todos os demais nós repassadores existentes na rota entre o servidor e o repassador, e que já estejam recebendo a mídia F, devem adicionar seu identificador no pacote *GMTP-Register-Reply*, antes de roteá-lo para o próximo salto da rota em direção ao nó repassador.

Quando o pacote *GMTP-Register-Reply* alcança o nó repassador, este envia a lista ordenada de repassadores entre ele e o servidor de volta ao nó servidor, utilizando um pacote do tipo *GMTP-Route-Notify*, conforme ilustra-se na Figura 12c. Ao mesmo tempo, o repassador também notifica aos clientes que aguardavam o início da conexão com o servidor, enviando um pacote do tipo *GMTP-RequestNotify* a cada cliente que requisitou a mídia. Neste caso, o conteúdo do campo *Código* da parte variável do cabeçalho do pacote *GMTP-RequestNotify* deve conter o valor $OK(0)$, indicando sucesso na requisição ao servidor.

Ao receber o pacote *GMTP-Route-Notify*, o servidor armazena a rota W, isto é, a lista de repassadores entre repassador requisitante e o servidor, em uma tabela *hash*. O servidor armazena uma tabela *hash* para cada fluxo de mídia que transmite, e a chave da tabela é o identificador único do repassador. Cada entrada da tabela armazena a rota W,

o endereço IP e um apontador para o *socket* correspondente à conexão entre o servidor e o repassador.

Esse procedimento de três vias confirma o registro de participação do repassador no servidor. Após a realização do registro, o servidor passa a enviar os dados multimídia ao repassador, utilizando seu IP e porta. O servidor não se conecta diretamente aos clientes, a não ser que este tenha se autopromovido a repassador.

Um repassador pode interceptar o registro de participação de outros repassadores durante seu trajeto até o servidor, conforme ilustra-se na Figura 13c. Neste caso, o pacote *GMTP-Register* não alcançará o servidor. Em momento oportuno, o servidor será notificado sobre o novo repassador que solicitou o registro de participação, conforme descrito na Seção 3.4.3.1.

3.4.3.1 Renovação do registro de participação

No GMTP, um nó repassador deve periodicamente sinalizar sua participação na rede de favores através de um método *keep-alive*, comumente utilizado em outros protocolos de rede consolidados, como o TCP (SALES, 2014). Nesse aspecto, o GMTP segue, guardadas as devidas restrições e especificidades, a RFC 1122, *Requirements for Internet Hosts - Communication Layers* (BRADEN, 1989).

O procedimento de renovação do registro de participação visa principalmente atualizar a tabela de rotas do servidor, a fim de possibilitar a ampliação do número de parcerias entre os nós repassadores e melhorar a qualidade de transmissão para os clientes.

Para sinalizar a sua participação na rede de favores, o repassador deve enviar periodicamente um pacote do tipo *GMTP-Register* ao servidor. Este, por sua vez, responderá à requisição com um pacote do tipo *GMTP-Register-Reply* e, por fim, o repassador responderá com um pacote do tipo *GMTP-Route-Notify*. O servidor difere um registro de participação de uma renovação porque no registro, o campo *Taxa proposta de transmissão* do cabeçalho do GMTP tem valor igual a 4294967295, que corresponde ao valor máximo que o campo de 32 *bits* pode assumir. No ato da renovação, o valor do campo difere do valor máximo.

O procedimento de renovação da participação difere do procedimento de registro de participação em um aspecto fundamental: um repassador não pode interceptar a renovação de participação de outros repassadores durante seu trajeto até o servidor. Desta forma, o servidor passa a conhecer os repassadores que estejam sendo servidos por repassadores parceiros e cujos registros de participação foram interceptados. Caso um novo repassador entre na rede, o servidor tomará conhecimento ao receber pacotes *GMTP-Route-Notify* de outros repassadores, que o novo repassador poderá ser utilizado na formação de novas parcerias, conforme discutido na Seção 3.4.5. Caso um dos repassadores saia da rede, o

servidor tomará conhecimento (a) ao deixar de receber a renovação de participação do repassador ou (b) ao receber o pacote *GMTP-Route-Notify* contendo a rota atualizada.

Por padrão, adotou-se o intervalo de 1 s para cada renovação do registro. O intervalo entre os registros de participação não precisa e nem pode ser muito pequeno, porque isso causaria um consumo desnecessário de largura de banda. Utilizando o valor padrão de 1 s, o procedimento de renovação de participação gera um consumo adicional de pelo menos[4] 246 Bytes por segundo por repassador. Além disso, quando o servidor recebe o pacote *GMTP-Route-Notify*, ele deve atualizar a sua tabela de rotas para analisar a rota atualizada e identificar a possibilidade de novas parcerias. Se o intervalo de renovação do registro de participação for muito pequeno, o servidor poderá ser sobrecarregado, calculando rotas e parcerias desnecessariamente. Esse aspecto do GMTP pode ser revisto e aprimorado, tanto no que se refere ao intervalo ideal para a renovação do registro de participação, quanto aos algoritmos de cálculo e rotas e parcerias.

3.4.4 Transmissão de dados

Após o estabelecimento de conexão, os servidor envia dados aos repassadores em modo *unicast* a fim de distribuir os pacotes de dados do tipo *GMTP-Data*. De forma similar, os nós repassadores utilizam o mesmo tipo de pacote para enviar os dados para os nós clientes, porém em modo *multicast*.

Na implementação atual do GMTP, não é possível o envio de pacotes maiores que o MTU (*Maximum Transmission Unit*). Para evitar eventuais erros na aplicação, no GMTP implementou-se a descoberta de PMTU (*Path Maximum Transmission Unit*) (MOGUL; DECWRL; DEERING, 1990). Assim, o desenvolvedor da aplicação servidora poderá consultar o tamanho máximo do pacote de dados permitido, evitando possíveis falhas no envio dos dados.

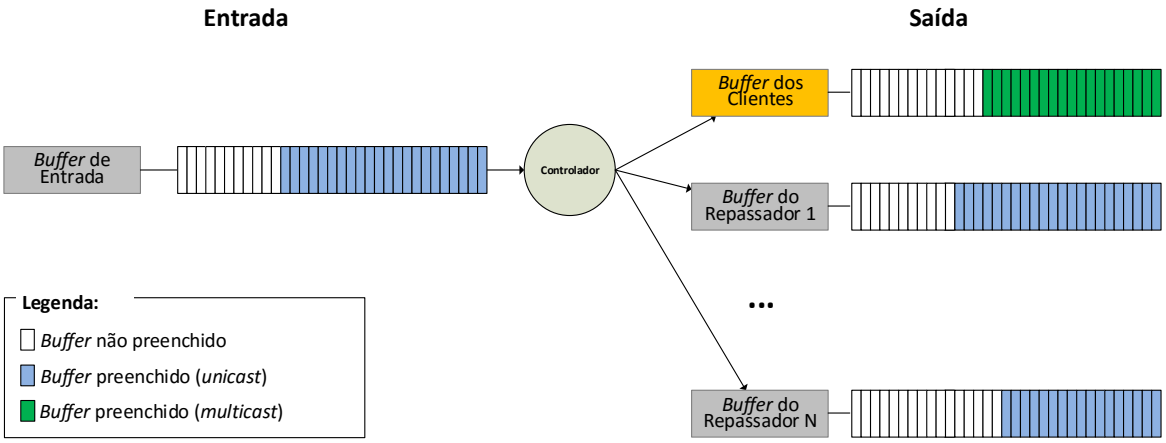
3.4.4.1 *Buffers* dos repassadores

Conforme ilustra-se na Figura 26, para cada fluxo de dados, um repassador GMTP gerencia um *buffer* de entrada e um ou mais *buffers* de saída. Utiliza-se o somatório dos espaços ocupados por cada *buffer* de entrada para obter o termo $q(\hat{t})$ da Equação 2.3, utilizada no GMTP-UCC.

Para cada fluxo de dados, um repassador GMTP gerencia pelo menos um *buffer* de saída referente ao canal *multicast* para os clientes locais. Se o repassador tiver parceria com outros repassadores, gerenciará um *buffer* de saída para cada repassador parceiro. Cada *buffer* de saída, seja *unicast* ou *multicast*, possui seu próprio temporizador para gerenciamento independente da taxa de transmissão.

[4] O tamanho dos pacotes *GMTP-Register-Reply* e *GMTP-Route-Notify* varia conforme a quantidade de repassadores na rota.

Figura 26 – *Buffers* de um repassador GMTP.



3.4.4.2 Transmissão *unicast*

Por se tratar de um protocolo para envio de mídia ao vivo, o servidor do GMTP envia o mesmo pacote de dados a todos os repassadores, através de um *pool* de conexões, no modo *unicast*. O servidor envia os dados para cada repassador de acordo com a taxa de transmissão definida através do GMTP-UCC.

A especificação original do GMTP-UCC preconiza que, assim como no RCP, o procedimento de cálculo e envio da nova taxa de transmissão ao servidor deve ocorrer, no máximo, a cada RTT. Na prática, isso implicaria que o procedimento de renovação do registro de participação fosse realizado a cada RTT, o que seria inviável, conforme discutido na Seção 3.4.3.1.

No contexto deste trabalho, implementou-se o mecanismo de envio da nova taxa de transmissão ao servidor a cada RTT, porém utilizando apenas um pacote do tipo *GMTP-Ack*, evitando assim o consumo desnecessário de largura de banda.

No caso dos clientes que se autopromoveram à condição de repassadores, não é adequada a utilização do GMTP-UCC para controle de congestionamento *unicast*, visto que o referido algoritmo foi projetado para utilização em roteadores. Nesses casos, o controle de congestionamento é feito através de uma versão do GMTP-MCC adaptada para fins de controle de congestionamento *unicast*. As principais diferenças entre a versão padrão e a adaptada do GMTP-MCC são:

- Não há eleição de relatores. Todos os clientes autopromovidos a repassador enviam relatórios ao servidor, como se fossem relatores.
- O servidor deve aceitar os pacotes do tipo *GMTP-Feedback* para fins de controle de congestionamento. Nesse caso, o servidor deve ignorar o campo de 8 *bits* referente ao

número de clientes que estão ativos e registrados em um relator, porque esse campo não é utilizável no contexto do controle de congestionamento *unicast*.

- Opcionalmente, os clientes autopromovidos podem enviar pacotes do tipo *GMTP-Ack* para fins de controle de congestionamento. Isto é possível porque o tipo de pacote *GMTP-Feedback* nada mais é que um pacote *GMTP-Ack* estendido para uso dos relatores no GMTP-MCC.

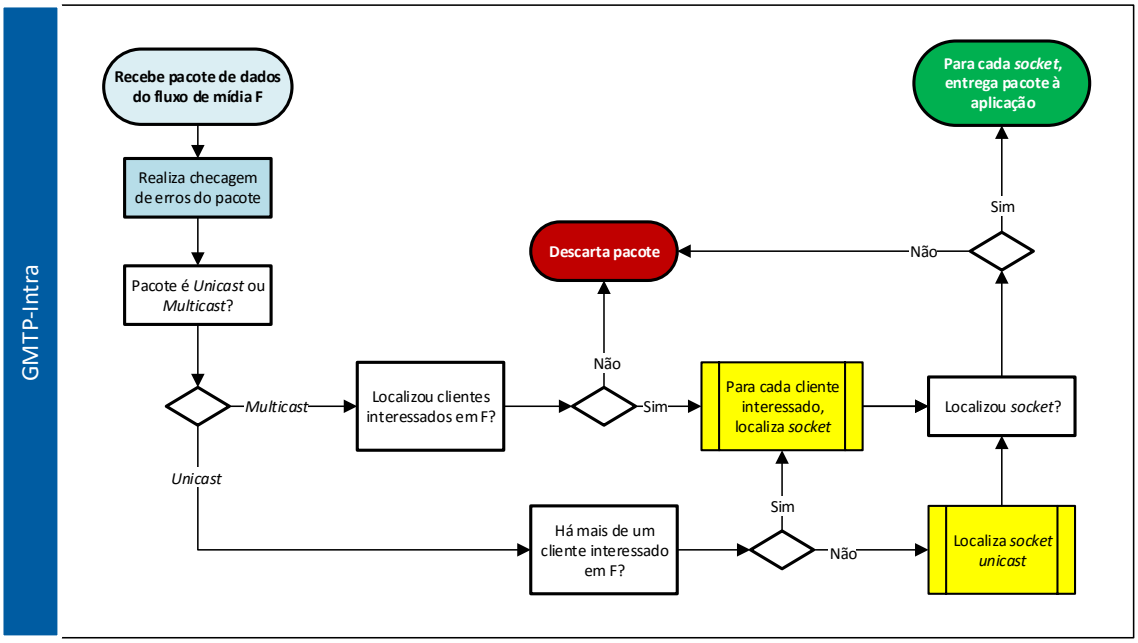
3.4.4.3 Transmissão *multicast*

No Linux, todas as informações sobre a conexão do cliente são armazenadas em uma tabela *hash*. Ao receber um pacote de rede, o protocolo de transporte faz uma busca pelo *socket* correspondente ao pacote, através da tabela *hash* supracitada. Essa busca é necessária porque o protocolo precisa das informações do *socket* a fim de tratar o pacote recebido apropriadamente. Caso o protocolo de transporte não encontre o *socket*, o pacote é descartado. Dentre os parâmetros necessários para buscar o *socket*, estão os endereços IP e portas do remetente e do receptor do pacote.

Conforme descreve-se na Seção 3.4.1, os clientes GMTP sempre se conectam ao servidor através do seu endereço IP (*unicast*) e porta. Assim, registra-se o *socket* do cliente GMTP no *kernel* como se fosse um *socket unicast* comum. Ao receber um pacote de dados referente ao fluxo de dados F, o cliente GMTP precisa localizar o *socket* correspondente ao pacote, a fim de tratá-lo apropriadamente. Geralmente, o pacote de dados será destinado ao endereço do canal *multicast* definido pelo repassador. Nesses casos, o GMTP não pode realizar a busca pelo *socket* correspondente utilizando a forma tradicional, porque para o Linux, o cliente GMTP está conectado ao servidor de forma *unicast*. Mesmo que o pacote recebido seja destinado ao endereço *unicast* do cliente (porque este se autopromoveu a repassador, por exemplo), o GMTP não pode realizar a busca pelo *socket* de forma semelhante ao TCP, pois o GMTP-Intra compartilha a mesma conexão para todos os clientes locais que estejam recebendo determinada mídia, conforme descrito na Seção 3.4.1.

Conforme ilustra-se na Figura 27, ao receber um pacote de dados, o cliente GMTP faz a busca na tabela de clientes do GMTP-Intra utilizando o identificador da mídia F como chave. Caso não se localize uma entrada na tabela e o referido pacote seja do tipo *multicast*, o pacote será descartado. Conforme descrito na Seção 3.3.2.1, cada entrada da tabela contém o endereço IP e porta do canal *multicast*, além da lista dos clientes daquele *host* que requisitaram a mídia F. Assim, o GMTP pode realizar a busca pelo *socket* do cliente que requisitou a mídia F utilizando como parâmetro os endereço IP e porta *unicast* do cliente, mesmo que o pacote recebido seja do tipo *multicast*. Se no mesmo *host* mais de um cliente solicitou a mídia F, o GMTP faz a busca pelo *socket* correspondente para cada solicitante. Após obter o *socket* correspondente, o GMTP pode enviar o pacote de dados às aplicações.

Figura 27 – Fluxograma do processo de recepção de mídia em um nó cliente GMTP.



Esse procedimento para recepção de dados adotado no GMTP foi concebido de forma a favorecer o compartilhamento da conexão local e a implementação do *multicast* transparente à aplicação.

Através do GMTP-MCC, realiza-se o controle de congestionamento *multicast*, conforme descrito nas Seções 2.2.6 e 3.4.2.2.

3.4.5 Formação da Rede de Favores

No GMTP, um nó repassador pode formar parcerias com outros repassadores basicamente de três formas: (a) no ato do registro de participação, quando um repassador pode interceptar o registro de participação de outros repassadores durante seu trajeto até o servidor; (b) através de requisição explícita do repassador para o servidor, que indica as parcerias; e (c) através da participação ativa do servidor, que indica a parceria aos repassadores, mesmo sem ser solicitado a fazê-lo.

Nesse contexto, deve-se levar em conta os avanços de novas arquiteturas de rede, como as ICNs, baseadas em um modelo de serviço do tipo *pull* e no provimento de *cache* dos dados mais acessados nos roteadores. Desta forma, o papel do repassador GMTP, assim como nos roteadores das ICNs, supera as funções de um roteador de datagramas encontrado na Internet atual. Portanto, deve-se considerar um cenário onde os roteadores serão projetados com maior poder de processamento, visando a sua atuação em redes ICN e participação ativa na distribuição de conteúdo.

Conforme discutido na Seção 3.4.3, um repassador pode interceptar um datagrama

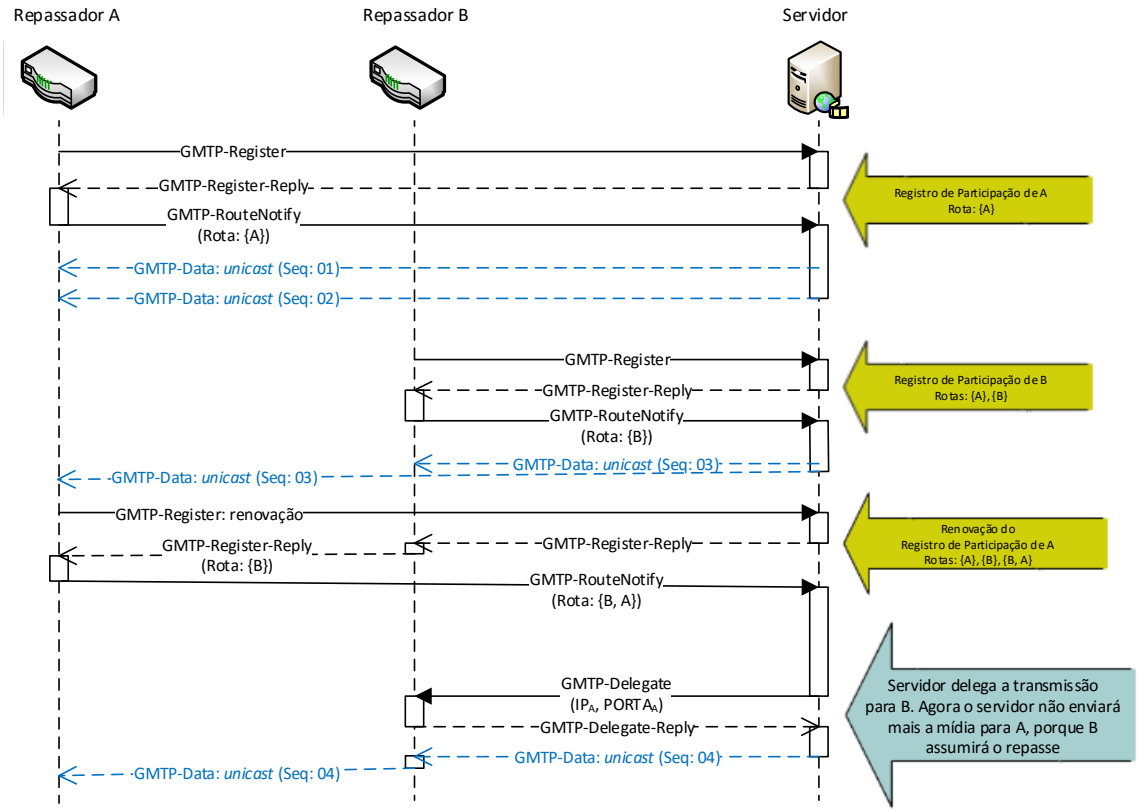
referente ao registro de participação de outros repassadores durante seu trajeto até o servidor. Conforme ilustra-se na Figura 13c, através desse mecanismo evita-se o envio de mais de um fluxo de dados através da mesma rota e pertencente à mesma mídia ao vivo. Consequentemente, evita-se o consumo desnecessário de recursos de rede. O repassador que já está recebendo a mídia, ao interceptar o registro do seu futuro parceiro, impede que múltiplas requisições alcancem o servidor, evitando (a) o consumo excessivo de recursos computacionais do servidor e (b) a duplicação de fluxos de dados na rota. Caso a requisição alcançasse o servidor, o repassador receberia dois fluxos de dados idênticos: um destinado a si próprio e outro destinado ao novo repassador. Com a interceptação, o repassador continua recebendo apenas um fluxo de dados, o qual repassa ao novo parceiro. No contexto deste trabalho, não se limitou a quantidade de interceptações que um repassador pode realizar.

Outra forma prevista na especificação original do GMTP para formalização de parcerias é através de requisição explícita do repassador para o servidor. Esse mecanismo de ampliação de parcerias propõe que o repassador tome a iniciativa de procurar novas parcerias, consultando o servidor através de um pacote do tipo *GMTP-RelayQuery*. Nesse contexto, o nó repassador pode enviar periodicamente um pacote do tipo *GMTP-RelayQuery* para o nó servidor a fim de descobrir melhores parceiros e aumentar o número de parcerias. Em seguida, o nó servidor constrói uma lista de nós parceiros e envia como resposta ao nó repassador. No entanto, propôs-se um mecanismo um pouco diferente dessa proposta prevista no GMTP. Neste caso, o servidor ativamente indica uma nova parceria a um repassador, sem que tenha sido solicitado a fazê-lo. Assim, apesar da não implementação do recurso *GMTP-RelayQuery*, novas parcerias podem ser formadas periodicamente, a partir de sugestões do servidor, através do novo recurso *GMTP-Delegate*.

Nesse contexto, outra forma de realizar novas parcerias entre os repassadores é através da intervenção ativa do servidor. Conforme ilustra-se na Figura 28, o servidor pode identificar a possibilidade de novas parcerias após atualizar a sua tabela de rotas ao receber uma renovação de participação. Ao constatar uma potencial nova parceria, o servidor deve enviar um pacote do tipo *GMTP-Delegate* para informar a um repassador que este deve formalizar parceria com outro repassador. Ao receber o referido pacote, o repassador deve armazenar os dados do novo parceiro e aguardar que este envie a sua renovação de registro. Ao receber a renovação de registro, o repassador encaminha um pacote do tipo *GMTP-Delegate-Reply* ao servidor, indicando que formalizou a parceria e que assumiu o repasse dos pacotes de dados.

Ao receber o pacote *GMTP-Delegate-Reply*, o servidor interrompe a transmissão de dados ao repassador que foi delegado, mantendo o envio de dados apenas ao repassador que recebeu o *GMTP-Delegate*. O servidor deve considerar que o repassador não aceitou o pedido de delegação, caso não receba a resposta do repassador após três tentativas. No

Figura 28 – Funcionamento do *GMTP-Delegate*.



contexto deste trabalho, não se limitou a quantidade de parcerias que um repassador pode realizar através do mecanismo de delegação da transmissão.

É importante salientar que o procedimento de delegação descrito acima é transparente aos repassadores delegados. Nem o repassador delegado nem os clientes tomam conhecimento de que pararam de receber os dados diretamente do servidor para receber os dados de outro repassador.

Em uma transmissão ao vivo pode haver milhares de nós requisitando a mídia a um servidor e, portanto, ao interceptar uma requisição ou formar parcerias, os repassadores contribuem para a redução do consumo de recursos computacionais do servidor e da rede. Ao delegar a transmissão, o servidor reduz sua sobrecarga de processamento e a quantidade de dados que precisa transmitir na rede.

3.4.6 Encerramento da transmissão

No GMTP, o procedimento de desconexão de um nó varia de acordo com o papel exercido.

3.4.6.1 Clientes e relatores

Na especificação original do GMTP, preconiza-se que para sinalizar uma desconexão, um nó cliente transmite um pacote do tipo *GMTP-Close* pelo canal de controle, contendo o identificador do fluxo que deseja se desconectar. Ao receber este tipo de pacote, o nó repassador transmite ao nó cliente um pacote do tipo *GMTP-Reset*, sinalizando que está ciente do fechamento da conexão. Nesse ínterim, os nós desalocam recursos relacionados à respectiva conexão.

Porém, no contexto deste trabalho, não foi possível implementar o procedimento de desconexão através do canal de controle, devido a uma limitação temporária relacionada à criação de *socket* de controle no repassador. Trata-se de uma dificuldade técnica que não foi possível superar até a data de publicação deste trabalho. Como clientes GMTP sempre se conectam ao servidor através do seu endereço IP (*unicast*) e porta, criando-se um *socket unicast*, no contexto deste trabalho, adotou-se provisoriamente o procedimento de desconexão a seguir.

Para se desconectar, o nó cliente envia um pacote do tipo *GMTP-Close* através do *socket unicast*, ou seja, endereçado ao servidor. O repassador de origem do cliente intercepta o pacote e procede conforme definido na especificação original do GMTP: transmite ao nó cliente um pacote do tipo *GMTP-Reset*, sinalizando que está ciente do fechamento da conexão.

O processo de desconexão de um nó cliente que se autopromoveu a repassador é semelhante ao procedimento de desconexão de um nó cliente: envia-se um pacote do tipo *GMTP-Close* endereçado ao servidor, que responde com um pacote do tipo *GMTP-Reset*, encerrando a conexão.

Seguem abaixo algumas vantagens de não utilizar o canal de controle para finalizar a conexão dos clientes GMTP:

- Ao utilizar o canal de conexão padrão para enviar a mensagem de saída da rede, a forma do cliente sair da rede se assemelha à forma padrão de protocolos conhecidos, como o TCP e o DCCP.
- Torna-se desnecessário adaptar a forma de sair da rede para os clientes que se autopromoveram a repassador. Os clientes que se autopromoveram a repassador não podem utilizar o canal de controle para sair da rede.

Por outro lado, a desvantagem de não se utilizar o canal de controle para finalizar a conexão dos clientes GMTP é que quando um cliente encerrar a conexão, o seu relator GMTP não tomará conhecimento do ocorrido. Para que o relator tome conhecimento da saída do cliente, sem utilização do canal de controle, há duas soluções possíveis: (i) ou o cliente deve enviar o pacote *GMTP-Close* também ao relator, ou o relator deve aguardar

um período de *timeout* sem receber *acks* do cliente para então considerar que o cliente está inativo. Caso o canal de controle fosse utilizado para finalizar a conexão do cliente, o seu relator automaticamente seria notificado da saída através do canal de controle.

Futuramente, pretende-se avaliar a possibilidade de implementar esta função da forma como definida na especificação original do GMTP, analisando as vantagens e desvantagens.

O processo de desconexão de um nó relator é semelhante ao procedimento de desconexão de um nó cliente, exceto que deve-se realizar um procedimento para eleger um novo nó relator quando um nó com tal responsabilidade solicitar desconexão. Na Seção 3.4.2.3, descreve-se o procedimento adotado para eleição de novos nós relatores.

3.4.6.2 Repassadores

Um nó repassador realiza o procedimento de desconexão quando não há mais clientes recebendo o fluxo de dados F ou quando o servidor explicitamente encerra a conexão. Para sinalizar o encerramento de sua participação da rede, o repassador deve enviar ao servidor um pacote do tipo *GMTP-Close*. Ao receber um pacote do tipo *GMTP-Close* oriundo de um repassador, o servidor deve finalizar o procedimento de desconexão enviando um pacote do tipo *GMTP-Reset*.

No GMTP, define-se o chamado *período de carência de novas parcerias*, um parâmetro que determina o tempo em que um nó repassador, em processo de desconexão, continuará repassando o fluxo de dados F para seus parceiros repassadores. No contexto deste trabalho, definiu-se o *período de carência de novas parcerias* como **indeterminado**. Na prática, isto significa que mesmo quando determinado repassador não tem mais nenhum cliente recebendo um fluxo F, ele continua a repassar os dados a outros repassadores parceiros até que (i) não haja mais repassadores parceiros recebendo o fluxo ou (ii) o servidor explicitamente encerre a conexão.

O mecanismo descrito a seguir não foi implementado, mas propõe-se que seja levado em consideração para implementação futura. Opcionalmente, um repassador pode continuar registrado ao servidor mesmo sem que haja clientes ou repassadores parceiros recebendo um fluxo de mídia. Para isto, basta que o repassador pare de enviar pacotes do tipo *GMTP-Ack* ao servidor, mas continue enviando a renovação do registro de participação. Ao parar de receber *acks*, o servidor marcará o repassador como *inativo*, porém *disponível*. Neste caso, o servidor interrompe o envio de dados ao repassador, mas não finaliza a conexão, visto que o repassador continuará renovando o seu registro de participação. Desta forma, o repassador continuará disponível para participar da rede de favores. Quando formalizar uma nova parceria, o repassador voltará a enviar os *acks* periódicos e o servidor voltará a considerá-lo *ativo*, enviando novamente o fluxo de mídia.

3.4.6.3 Servidores

Ao final da transmissão, o servidor deve enviar aos repassadores um pacote do tipo *GMTP-Close*. Cada repassador, ao receber o pacote *GMTP-Close*, deve responder ao servidor com um pacote do tipo *GMTP-Reset*.

Para sinalizar aos seus clientes e repassadores parceiros que o servidor encerrou a transmissão, o repassador deve inserir o pacote *GMTP-Close* recebido no *buffer* de dados, para ser enviado logo após o último pacote de dados. A utilização de um canal *multicast* para o envio do pacote *GMTP-Close* aos clientes locais é importante porque assim evita-se que o repassador tenha que manter o registro de todos os clientes conectados na rede local. Ao enviar a notificação por *multicast*, todos os clientes receberão a notificação.

A especificação original do GMTP não deixa explícito o canal pelo qual o repassador deve enviar o pacote *GMTP-Close* aos clientes, embora deixe implícito que deve-se utilizar o *Canal de Controle* para o envio desse tipo de pacote. Porém, utilizar um canal paralelo ao de dados para transmitir o pacote de encerramento de conexão, aumentaria de forma desnecessária a complexidade de implementação da chamada *close* do *socket* GMTP. Se o repassador utilizasse o *Canal de Controle* para enviar o pacote *GMTP-Close*, o cliente, ao receber o *close*, teria que manter seu *socket* aberto por um período de *timeout* para que houvesse tempo hábil de receber os últimos pacotes de dados que ainda estavam no *buffer* do repassador. Como o repassador adiciona o pacote *GMTP-Close* na mesma fila dos pacotes de dados, o cliente GMTP tem a garantia de que ao receber o pacote *GMTP-Close* não há mais dados a serem recebidos. Com essa garantia, o cliente pode imediatamente desalocar recursos relacionados à respectiva conexão.

4 ANÁLISE DO DESEMPENHO DO GLOBAL MEDIA TRANSMISSION PROTOCOL (GMTP)

Nesta seção, apresenta-se a análise do desempenho do GMTP. Os experimentos foram divididos em dois tipos, visando avaliar diferentes métricas.

- **Experimentos gerais**: executados em máquinas virtuais, com o objetivo avaliar a implementação e sua consonância com a proposta original. Nos casos onde foram propostas mudanças no protocolo, avaliar o funcionamento com as modificações.
- **Simulações de desempenho**: tiveram como objetivo avaliar o desempenho da implementação do GMTP no núcleo do Linux, com suas respectivas adaptações, e o possível impacto que as mudanças propostas possam ter no desempenho do GMTP. Para isso, acoplou-se a implementação do GMTP/Linux ao simulador de redes NS-3 (HENDERSON; LACAGE; RILEY, 2008). Além disso, comparou-se os resultados obtidos com os resultados disponíveis em (SALES, 2014).

4.1 Experimentos em máquinas virtuais

Durante a implementação, realizaram-se alguns experimentos utilizando máquinas virtuais com o objetivo de avaliar a corretude da implementação proposta e sua consonância com a especificação original do GMTP.

4.1.1 Metodologia

Visando avaliar os resultados dos experimentos, definiu-se que a métrica a ser estudada seria a taxa de recepção total percebida nos clientes, medida em *bytes/segundo* (B/s). A **taxa de recepção** é a razão entre a quantidade total de dados recebidos, medida em *bytes*, pelo tempo total do experimento, medido em *segundos*.

Escolheu-se a métrica taxa de recepção porque esta métrica reflete tanto o desempenho do protocolo como o estado da rede. Por exemplo, conforme discute-se nas Seções 2.2.5 e 2.2.6, os algoritmos de controle de congestionamento do GMTP reduzem a taxa de transmissão caso haja aumento do valor do RTT, do tamanho da fila de repasse ou caso haja perdas de pacotes. Logo, a redução da taxa de recepção pode indicar um desempenho do protocolo abaixo do esperado na topologia definida no experimento.

Os tipos de nós utilizados no experimento foram definidos da seguinte forma:

1. **Servidor**: aplicação executada em uma máquina *servidor de mídia* capaz de transmitir dados através de um dos seguintes protocolos de transporte: TCP, UDP e GMTP. Os dados foram transmitidos a uma taxa máxima de 50.000 B/s.
2. **Clientes**: aplicação executada em uma máquina cliente capaz de receber dados através de um dos seguintes protocolos de transporte: TCP, UDP e GMTP. Cada cliente registra a sua taxa de recepção de dados em arquivos de registro (*log*).
3. **Roteador**: máquina virtual que realiza o papel de roteador, permitindo a interconexão entre a sub-rede do servidor e a sub-rede dos clientes. A máquina funciona como um roteador comum quando o módulo GMTP-Inter está desativado. Quando ativa-se o módulo GMTP-Inter, a máquina passa a funcionar como um repassador GMTP.

No experimento, quatro máquinas virtuais foram utilizadas para realizar a simulação, cada uma com seu papel definido:

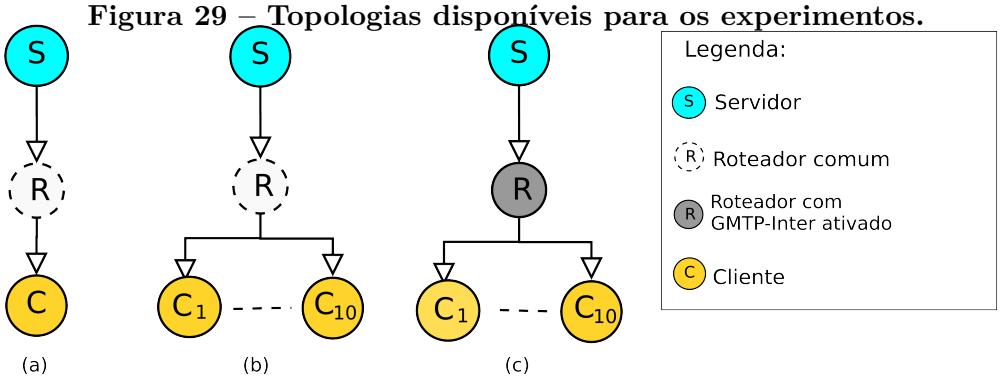
- Em uma máquina, hospedou-se o servidor, sendo alocada em determinada sub-rede.
- Em outras duas máquinas hospedaram-se os clientes. Estas máquinas estavam alocadas em uma sub-rede diferente da sub-rede do servidor.
- Outra máquina realizou o papel de roteador. A comunicação entre as máquinas do servidor e as máquinas dos clientes ocorreu através da máquina roteadora.

Todas as máquinas virtuais estavam hospedadas na mesma máquina real. Logo, todas as transmissões de dados foram locais, sem interferências externas. Todos os clientes requisitaram o mesmo fluxo de dados. Em cada experimento, os clientes receberam 6500 pacotes de dados.

Com relação a execução de cada experimento, executaram-se 10 ensaios iniciais de cada sistema estudado, obtendo-se assim 10 amostras para a variável estudada. Em seguida, calculou-se a média dessas amostras e, para realizar comparações com 95 % de nível de confiança, calculou-se a quantidade mínima de ensaios (n_t) de cada experimento a fim de atingir este nível de confiança, utilizando-se a Equação C.2. Após o cálculo de n_t, definiu-se a quantidade de ensaios como $n_t + 1$. No Apêndice C.1, apresentam-se mais detalhes sobre o cálculo de n_t. Por fim, cada experimento foi repetido 10 vezes ($n_t = 9$) e os dados foram agregados utilizando-se a média aritmética dos resultados obtidos.

Conforme ilustra-se na Figura 29, organizou-se três topologias básicas, com base no ambiente descrito acima, executando-se experimentos para cada uma das topologias definidas.

Figura 29 – Topologias disponíveis para os experimentos.



Os experimentos realizados foram os seguintes:

1. **Experimento A**: realizado utilizando a topologia ilustrada na Figura 29a. Apenas um Cliente GMTP requisitando o fluxo de dados do Servidor GMTP.

2. **Experimento B-TCP**: realizado utilizando a topologia ilustrada na Figura 29b. Dez Clientes TCP, sendo cinco em cada máquina cliente, requisitando o fluxo de dados do Servidor TCP.

3. **Experimento B-UDP**: realizado utilizando a topologia ilustrada na Figura 29b. Dez Clientes UDP, sendo cinco em cada máquina cliente, requisitando o fluxo de dados do Servidor UDP.

4. **Experimento B-GMTP**: realizado utilizando a topologia ilustrada na Figura 29b. Dez Clientes GMTP, sendo cinco em cada máquina cliente, requisitando o fluxo de dados do Servidor GMTP.

5. **Experimento C**: realizado utilizando a topologia ilustrada na Figura 29c. Dez Clientes GMTP, sendo cinco em cada máquina cliente, requisitando o fluxo de dados Servidor. Neste experimento, habilitou-se o módulo GMTP-Inter no roteador.

Como os experimentos tinham como objetivo avaliar o funcionamento básico do GMTP em cenários simples, optou-se pela comparação com o TCP e o UDP. A comparação com o UDP, que não provê garantia de entrega ou controle de congestionamento, serviu como base para a avaliação do funcionamento básico do GMTP em um cenários desfavorável, sem a participação do módulo GMTP-Inter. Esperava-se que o desempenho do GMTP em um cenário desfavorável fosse, pelo menos, semelhante ao desempenho do UDP. Caso o desempenho das aplicações GMTP fosse inferior ao observado nas aplicações que utilizaram o UDP ou o TCP, haveria a possibilidade de erro de implementação no GMTP.

4.1.2 Resultados

Primeiramente, realizaram-se experimentos com o TCP e UDP visando determinar uma referência para o GMTP. Em seguida, analisou-se o desempenho do GMTP em

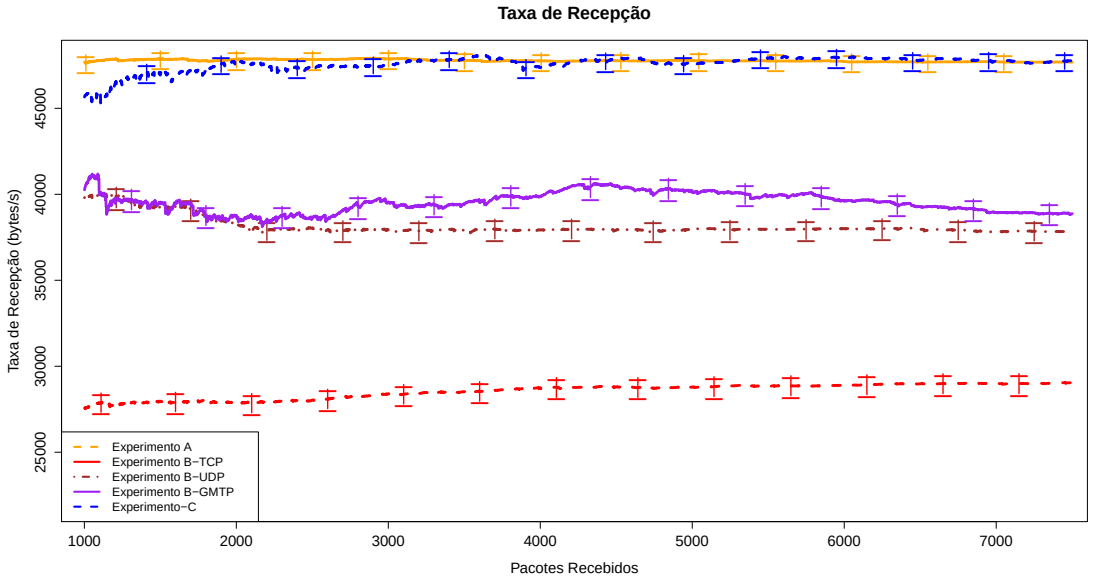
comparação com os demais protocolos. Na Tabela 1, apresenta-se um resumo dos resultados dos experimentos.

Tabela 1 – Taxa de recepção medida nos experimentos (B/s)

Simulação	Mínima	Média	Máxima	Coef. de Variação
Experimento A	47.600	47.770	47.920	0.0012
Experimento B-TCP	27.520	28.550	29.050	0.0146
Experimento B-UDP	37.710	38.140	40.010	0.0136
Experimento B-GMTP	38.110	39.530	41.180	0.0146
Experimento C	45.240	47.580	48.120	0.0091

Na Figura 30, ilustra-se a evolução da métrica *taxa de recepção* ao longo da transmissão multimídia nos experimentos realizados.

Figura 30 – Evolução da taxa de recepção em relação ao tempo.



Conforme ilustra-se na Figura 30, no Experimento A, a taxa de recepção de dados permaneceu muito próxima à taxa de envio de 50.000 B/s fornecida pelo servidor. Conclui-se que em interações simples cliente-servidor em um sistema operacional real, o GMTP apresenta desempenho satisfatório.

Nos experimentos B (B-TCP, B-UDP e B-GMTP), a taxa de recepção de dados dos clientes foi sempre inferior à taxa de envio fornecida pelo servidor. Esse resultado já era esperado, considerando alguns fatores como:

- a sobrecarga de processamento no servidor, que precisa tratar múltiplas requisições de diversos clientes e;
- a sobrecarga na rede, devido à quantidade de fluxos de dados trafegando.

Nesses experimentos, para cada requisição, o servidor gerou um fluxo de dados independente, o que ocasionou o crescimento na fila de pacotes do roteador e o aumento do atraso na recepção dos pacotes percebido pelos clientes.

Nesse caso, o comportamento do GMTP, similar ao UDP, era previsível, uma vez que nessa rodada dos experimentos os principais recursos do GMTP-Inter não foram habilitados. Observa-se que sem os recursos do GMTP-Inter, o desempenho do GMTP equipara-se ao desempenho do UDP. Outro aspecto importante observado nesse resultado é que mesmo com o uso de algoritmos de controle de congestionamento no GMTP, este obteve um desempenho similar ao do UDP. Isto significa que o algoritmo de controle de congestionamento do GMTP não adiciona sobrecarga significativa durante uma transmissão em cenários sem congestionamento na rede.

Conforme ilustra-se na Figura 30, no experimento onde GMTP-Inter estava habilitado, observa-se que os clientes apresentaram taxas mínimas e máximas de recepção próximas à taxa de transmissão fornecida pelo servidor. De fato, no Experimento C, a taxa de recepção de dados dos Clientes GMTP se manteve próxima à taxa obtida no Experimento A, mesmo havendo vários clientes requisitando o fluxo de dados.

Já no cenário em que se utilizou o GMTP-Inter, constataram-se melhorias em relação à utilização do UDP, pois o GMTP provê o suporte transparente ao uso de *multicast* por parte da aplicação, compartilhando-se os pacotes de dados entre múltiplos sistemas interessados por um mesmo conteúdo, o que reduz o consumo de recursos de rede e melhora a qualidade de serviço oferecida às aplicações. Através do GMTP-Inter, os clientes não recebem os pacotes de dados diretamente dos servidores, mas sim de roteadores já envolvidos na transmissão da mídia, evitando-se múltiplas conexões ao servidor. Dentre os benefícios dessa abordagem, estão:

- Como o servidor recebe apenas uma requisição ao servidor, evita-se o consumo excessivo de recursos computacionais do servidor.
- O servidor envia apenas um fluxo de dados ao roteador, o que evita o crescimento desnecessário da fila de pacotes do roteador.
- O roteador transmite os dados recebidos aos clientes utilizando *multicast*, de forma transparente à aplicação.
- Ao reduzir o consumo de recursos de rede, ocorre melhora na qualidade de serviço oferecida às aplicações.

Através destes experimentos, foi possível avaliar o funcionamento básico do GMTP em cenários simples, comparando-se com outros dois protocolos de transporte (TCP e UDP). Considerando que o UDP é um protocolo de transporte simples, que não provê

garantia de entrega ou controle de congestionamento, utilizou-se o desempenho do UDP como base para a avaliação do funcionamento básico do GMTP. Constatou-se que o desempenho do GMTP em um cenário simples foi, no pior caso, semelhante ao desempenho do UDP.

4.2 Simulações em maior escala

As simulações do GMTP via NS-3 tiveram como objetivo avaliar o desempenho da implementação do GMTP no núcleo do Linux em cenários com uma quantidade maior de nós clientes. Nessas simulações, considerou-se o impacto que as adaptações e propostas na Seção 3 pudessem ter no desempenho do GMTP, comparando-se, quando possível, os resultados obtidos com os resultados disponíveis em (SALES, 2014).

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

4.2.1 Metodologia

Visando avaliar os resultados dos experimentos, definiu-se a metodologia utilizada nas simulações através da definição de uma topologia de rede, as variáveis dependentes e independentes, população e amostras, tratamentos, instrumentações e o tipo de mídia a ser transmitido. Nesta seção, discute-se cada um desses aspectos.

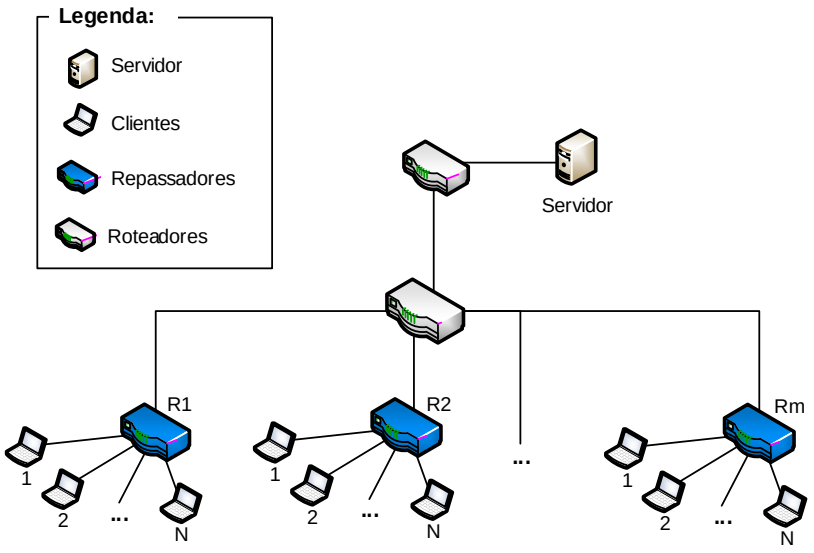
4.2.1.1 Topologia da rede

Primeiramente, definiram-se as topologias de rede, como ilustra-se na Figura 31. Em seguida, conforme ilustra-se na Figura 31a, utilizou-se a Topologia A para simular o desempenho do GMTP em um cenário desfavorável, definindo-se configurações dos nós de tal modo que a cooperação entre os repassadores fosse limitada. Conforme discute-se na Seção 3.4.5, no contexto deste trabalho, não se implementou a busca por mais parceiros através do *GMTP-RelayQuery*. Nesse contexto, a Topologia A é desfavorável à formação de parcerias entre roteadores, visto que em tal topologia não há possibilidade de interseção das rotas do servidor até os repassadores. Definiu-se tal topologia visando avaliar o

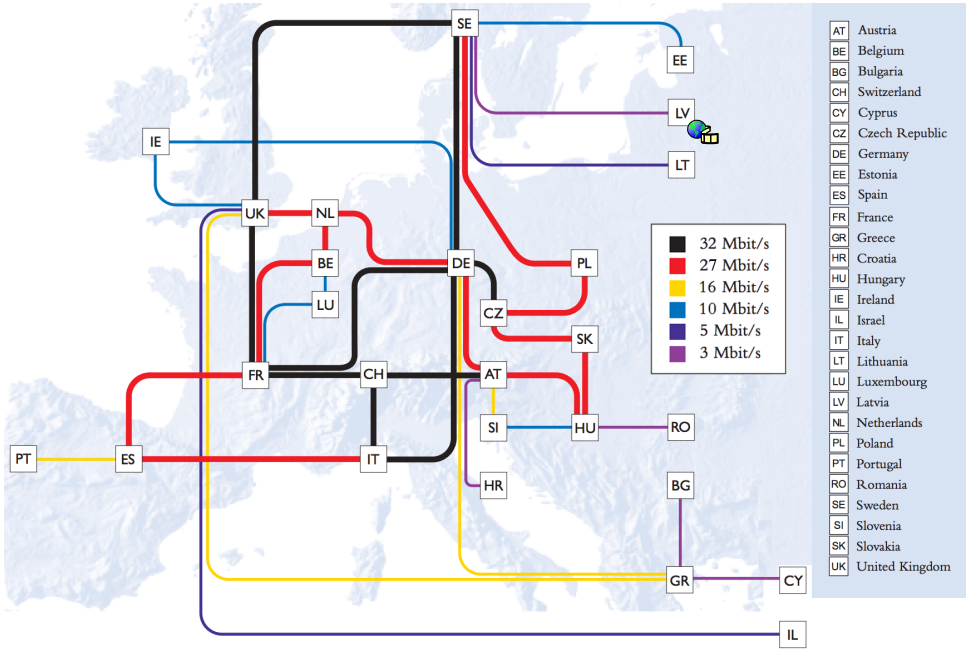
desempenho do GMTP no início de sua implantação, quando poucos roteadores darão suporte ao protocolo.

Figura 31 – Topologias utilizadas na simulações em maior escala.

(a) Topologia A, onde apenas o roteador local tem o GMTP ativado.



(b) Topologia B, versão adaptada do *backbone* da rede utilizada em (SALES, 2014).



Em seguida, conforme ilustra-se na Figura 31b, utilizou-se a Topologia B para simular o desempenho do GMTP em uma rede semelhante à rede simulada em (SALES, 2014), ou seja, uma versão adaptada da atual rede GÉANT[1], composta por 27 roteadores.

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

Na Topologia B, é possível a formação de parcerias entre roteadores, através de (a) interceptações de requisições dos repassadores, ou (b) através do novo mecanismo *GMTP-Delegate*, onde o servidor explicitamente indica uma nova parceria a um repassador, sem que tenha sido solicitado a fazê-lo, com base nas intercessões das rotas dos repassadores.

Com relação à conectividade dos nós clientes à rede, simularam-se redes locais através das quais os nós clientes requisitaram um fluxo multimídia de teste. Para isto, foram gerados 27 sub-grafos aleatórios de um grafo completo, com 12 vértices, representando os roteadores da rede local.

4.2.1.2 Variáveis independentes

Na Tabela 2, apresentam-se as variáveis independentes utilizadas no experimento, com base na topologia da rede apresentada anteriormente.

Tabela 2 – Variáveis independentes utilizadas no experimento.

Parâmetros	**Valores**
Largura de banda entre LANs	10 *Mbps*
Atraso de propagação das redes locais	1 *ms*
Tempo de simulação de cada ensaio	900 *s*
Tamanho do *buffer* (roteadores)	100 pacotes
Tamanho máximo do pacote de dados	1500 *Bytes*

4.2.1.3 Variáveis dependentes

As principais métricas para medir um sistema de distribuição de mídias ao vivo podem ser organizadas em duas categorias, apresentadas a seguir.

1. **Qualidade de serviço à aplicação:** avaliam-se a taxa de recepção da mídia (RX); a taxa de perdas de pacotes (LX) e o índice de continuidade (IC), sendo:

   a) *Taxa de recepção da mídia (RX)*: calculada pela divisão da quantidade total de dados de mídia recebidos pela duração total da transmissão multimídia. Calcula-se a taxa de recepção instantânea dividindo-se a quantidade de dados recebidos em determinado instante pela duração desse instante. Em um cenário ideal, a taxa de recepção dos clientes seria igual à taxa de transmissão do servidor.

   b) *Taxa de perdas de pacotes (LX)*: calculada pela divisão da quantidade total de dados de mídia recebidos pelo cliente pela quantidade total de dados de mídia enviados pelo servidor. Em um cenário ideal, o total de dados recebidos pelos clientes seria igual ao total de dados transmitidos pelo servidor.

c) *Índice de Continuidade (IC)*: calculada pela divisão entre o número de pacotes de dados da mídia entregues ao nó cliente antes do momento de reproduzi-los e o número total de pacotes de dados transmitidos. Assim, contabiliza-se o tempo em que os nós clientes não conseguiram sequer receber pacotes de dados da mídia. Em um cenário ideal, todos os dados de mídia seriam entregues aos clientes antes do momento de reproduzi-los.

2. **Sobrecarga de controle:** avalia-se a quantidade de pacotes de controle (PC) transmitidos por um protocolo durante o tempo de simulação (contagem dos pacotes que não transportam dados da mídia).

Na Tabela 3, apresentam-se as variáveis dependentes, definidas com base nas métricas supracitadas.

Tabela 3 – Variáveis dependentes consideradas no experimento.

Variáveis dependentes	**Símbolo**
Taxa de recepção da mídia	RX
Taxa de perdas de pacotes	LX
Índice de continuidade (%)	IC
Número de pacotes de controle	PC

4.2.1.4 População e amostras

Constituiu-se a população por dados coletados durante a execução dos ensaios de acordo com às variáveis dependentes apresentadas na Tabela 3, com amostras coletadas a cada pacote recebido. Em cada ensaio, o nó servidor enviou 10000 pacotes. O valor final de cada variável dependente em cada ensaio foi determinado pela média aritmética das respectivas amostras.

4.2.1.5 Tratamentos

Na Tabela 4, apresentam-se os tratamentos considerados no experimento. Nessas simulações, utilizou-se apenas o protocolo GMTP. Na coluna n_t, apresenta-se a quantidade de repetições de cada tratamento.

Com relação a execução de cada tratamento, executaram-se 10 ensaios iniciais, obtendo-se assim 10 amostras para cada variável dependente. Em seguida, calculou-se a média dessas amostras e, para realizar medições com 95 % de nível de confiança, calculou-se a quantidade total de ensaios (n_t) de cada tratamento a fim de atingir este nível de confiança. Para isto, calculou-se a quantidade de ensaios necessários para obter 95 % de nível de confiança com base na média (das amostras iniciais) μ de cada variável dependente, utilizando-se a Equação C.2. Sendo assim, a quantidade total de ensaios de

Tabela 4 – Tratamentos executados no experimento.

Trat. #	Número de repassadores	Número de clientes	n_t
1		1	16
2	1	10	18
3		20	34
4		3	18
5	3	30	25
6		60	50
7		30	54
8	30	300	63
9		600	104
13		500	58
14	39	1.500	61
15		15.000	56

cada tratamento foi determinado por $n_t = max(n_{RX}, n_{LX}, n_{IC}, n_{PC}) + 1$. Por exemplo, se para determinada variável dependente obteve-se $\{12, 18, \mathbf{23}, 20\}$, assumiu-se $n_t = 24$. Ou seja, repetiu-se 24 vezes o mesmo tratamento. No Apêndice C.1, apresentam-se mais detalhes sobre o cálculo de n_t.

Após a execução dos experimentos, para determinar o intervalo de confiança das amostras obtidas, aplicou-se o teorema central do limite para construir um intervalo de confiança aproximado, conforme detalha-se no Apêndice C.2.

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

1. Configurou-se todos os clientes para enviar a requisição da mídia ao servidor.
2. Assegurou-se que os nós clientes fossem submetidos as mesmas capacidades de transmissão, independente do sistema avaliado.
3. Todos os nós clientes requisitaram a mídia durante o primeiro terço da transmissão. Dessa forma, garantiu-se que os sistemas foram submetidos às mesmas condições iniciais de conexão.
4. Todos os nós clientes se conectaram ao servidor por intermédio de um repassador. Assegurou-se que nenhum cliente se autopromovesse a repassador, conforme mecanismo descrito na Seção 3.4.1.

4.2.1.6 Instrumentação

Com relação à instrumentação, utilizou-se o simulador de redes NS-3 (HENDERSON; LACAGE; RILEY, 2008) integrado com o *Direct Code Execution (DCE) Cradle* (TA-

ZAKI; URBANI; TURLETTI, 2013), um arcabouço que possibilita a integração entre a pilha de protocolos do núcleo do Linux com o NS-3. Através do uso do *DCE Cradle* integrado ao NS-3, é possível criar simulações de rede que utilizam a implementação real, sem modificações, de protocolos de rede do Linux, facilitando a criação de experimentos reproduzíveis. Camara et al. (2014) demonstraram que as simulações realizadas através do DCE/NS-3 apresentam resultados semelhantes aos esperados em um cenário real.

4.2.1.7 Formato da mídia

Na Tabela 5, apresentam-se as propriedades da mídia utilizada no experimento.

Tabela 5 – Propriedades da mídia transmitida.

Propriedades	**Valores**
Mídia sintetizada	*Áudio*
Taxa de transmissão da mídia	2 *Mbps*
Tamanho médio de cada pacote de dados	744 *B*
Quantidade total de dados transmitidos	7.440.000 *B* (10.000 pacotes)

Com base na metodologia apresentada nesta seção, coletaram-se as amostras para as variáveis dependentes e realizou-se uma análise dos dados.

4.2.2 Resultados e Discussões

Nesta seção, apresentam-se os resultados das simulações realizadas. A discussão foi organizada de acordo com as métricas apresentadas na Seção 4.2.1.3: *Taxa de recepção da mídia (Total e Instantânea)*, *Taxa de perdas de pacotes*, *Índice de Continuidade* e *Sobrecarga de controle*.

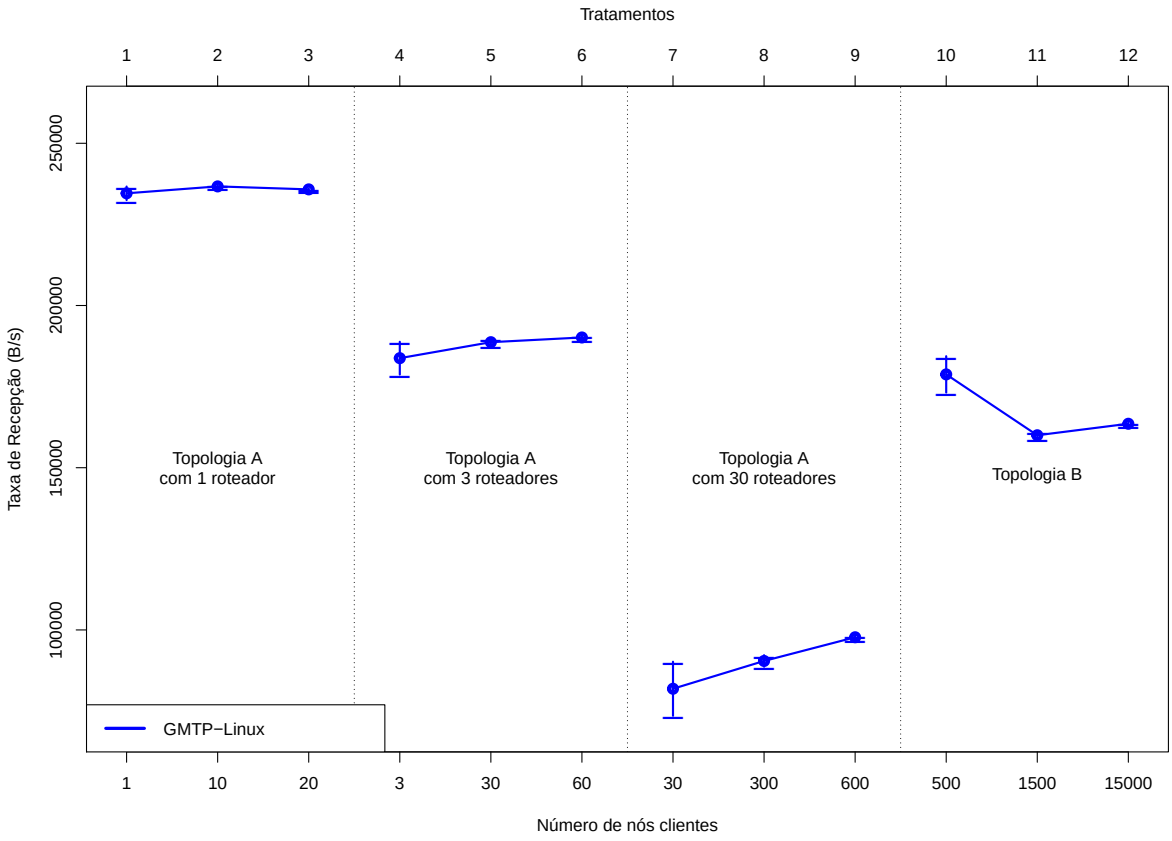
4.2.2.1 Taxa de recepção da mídia

A métrica *taxa de recepção da mídia* é importante, pois mapeia o desempenho do sistema durante todo o ensaio, a partir do instante em que os nós clientes começam a receber o primeiro pacote de dados. Através da taxa e recepção total percebe-se o desempenho total da rede durante a transmissão. De forma semelhante, através da taxa de recepção instantânea, tem-se o conhecimento do desempenho da rede em determinado instante, além da variação de desempenho da rede durante a transmissão.

Na Figura 32, apresentam-se as *taxas de recepção totais* medidas em cada tratamento. Observa-se que o valor da taxa de recepção total não depende necessariamente do números de clientes requisitando a mídia, mas da quantidade de repassadores recebendo a mídia diretamente do servidor. Para determinar o intervalo de confiança, aplicou-se o teorema

central do limite para construir um intervalo de confiança aproximado, conforme detalha-se no Apêndice C.2.

Figura 32 – Resultado dos tratamentos (1-12) para a métrica *taxa de recepção*.



Nos tratamentos 1, 2 e 3, onde havia apenas um repassador conectado ao servidor, a taxa de recepção total permaneceu muito próxima à taxa de envio máxima fornecida pelo servidor (250.000 B/s). Esse resultado é compatível com os resultados obtidos na Seção 4.1.2. Nestes tratamentos, estabeleceu-se uma conexão *unicast* entre o repassador e o servidor, e uma conexão *multicast* entre o repassador e os clientes. Assim, a variação do número de clientes conectados ao repassador não acarretou mudanças significativas na taxa de recepção dos clientes GMTP.

Nos tratamentos 4, 5 e 6 haviam três repassadores recebendo a mídia diretamente do servidor, enquanto nos tratamentos 7, 8 e 9 haviam trinta repassadores recebendo a mídia diretamente do servidor. Devido à ausência da função do *GMTP-RelayQuery*, na topologia dos tratamentos, não houve cooperação entre os repassadores. Em cada repassador havia o mesmo número de clientes conectados em modo *multicast*. Nesses tratamentos, houve uma redução da taxa de recepção dos clientes quando comparada com a taxa obtida nos tratamentos 1, 2 e 3. Essa redução ocorreu porque, para cada registro de participação, o servidor gerou um fluxo de dados independente, o que ocasionou o aumento da carga de processamento do servidor e o crescimento na fila de pacotes do

roteador. Esses dois fatores tendem a aumentar o RTT entre o servidor e os repassadores, o que impacta o controle de congestionamento *unicast* do GMTP, que reduziu a taxa de transmissão de dados do servidor para evitar o congestionamento da rede. Nos tratamentos 7, 8 e 9, a redução da taxa de recepção foi ainda maior, não somente pela limitação de envio de dados imposta ao servidor pelo controle de congestionamento, como também pela perda de pacotes ocasionada pelo congestionamento gerado por trinta fluxos de dados. Na Seção 4.2.2.2, discute-se sobre as perdas de pacotes detectadas durante as simulações. Esses resultados estão dentro do que se esperava em um cenário onde não há cooperação entre os repassadores e um alto número de repassadores recebendo a mídia diretamente do servidor.

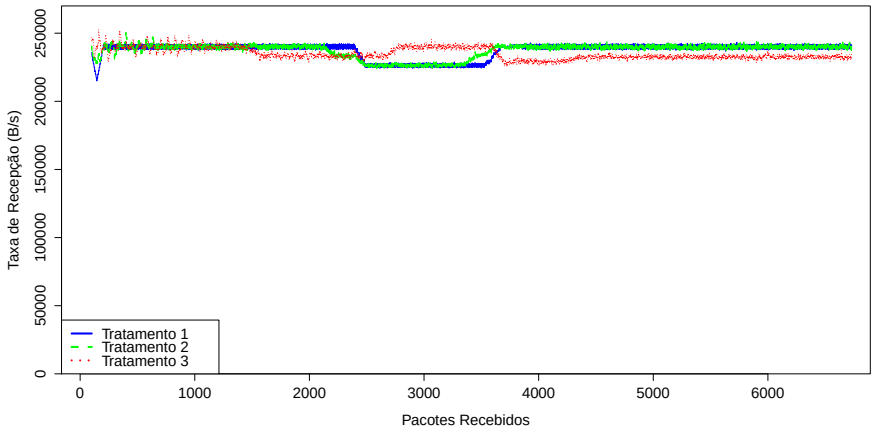
Já nos tratamentos 10, 11 e 12, onde utilizou-se a Topologia B, ilustrada na Figura 31b, a taxa de recepção foi melhor que a taxa observada nos tratamentos 7, 8 e 9 que, embora tenham menos repassadores registrados na rede, dificultava a cooperação entre os nós, devido à sua topologia. Observa-se que numa topologia mais semelhante a topologias de redes reais, como por exemplo a Internet, a cooperação entre os repassadores incrementa o desempenho das transmissões de mídia ao vivo através do GMTP. Esses resultados estão dentro do que se esperava em um cenário onde há possibilidade de cooperação entre os repassadores, reduzindo o número de repassadores recebendo a mídia diretamente do servidor.

Na Figura 33, apresenta-se a evolução da *taxa de recepção instantânea*, a cada pacote recebido. Assim como na taxa de recepção total, observa-se que o valor da taxa de recepção instantânea não depende necessariamente do números de clientes requisitando a mídia, mas da quantidade de repassadores que recebem a mídia *diretamente* do servidor, em vez de recebê-la de um repassador parceiro. Nos tratamentos onde há menos repassadores recebendo a mídia diretamente do servidor, a taxa de recepção instantânea é maior e mais estável ao longo da transmissão, enquanto nos tratamentos onde há mais repassadores recebendo a mídia diretamente do servidor, a taxa de recepção tende a diminuir, com maior variação. Nesse contexto, a formação de parcerias entre repassadores favorece a melhora da *taxa de recepção*, pois as parcerias evitam que esses repassadores recebam a mídia diretamente do servidor, mas a recebam de repassadores parceiros, contribuindo para a redução do consumo de recursos computacionais do servidor e da rede. Mesmo nos tratamentos cuja topologia era desfavorável à formação de parcerias, a utilização do GMTP favoreceu o desempenho da transmissão multimídia. Caso as aplicações estivessem usando outro protocolo, diferente do GMTP, cada cliente solicitaria o conteúdo multimídia de forma independente, gerando fluxos independentes da mesma mídia. Isso aumentaria sobremaneira o congestionamento da rede se comparado ao GMTP, onde se utilizam os repassadores.

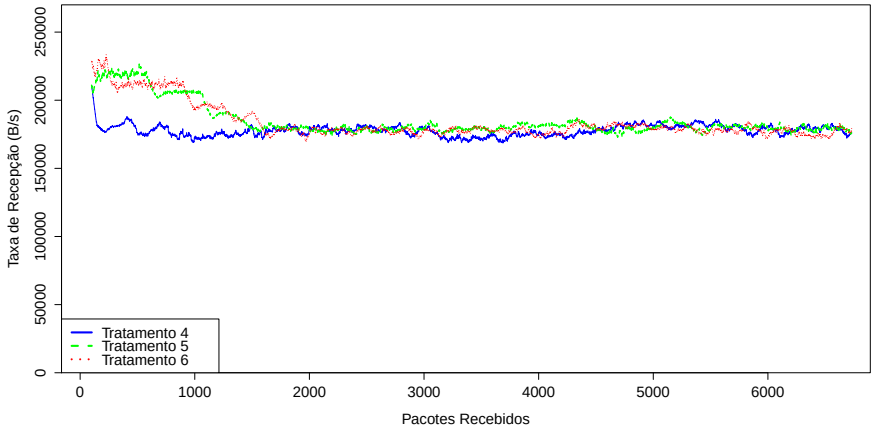
Conforme ilustra-se na Figura 33a, nos tratamentos 1, 2 e 3, onde havia apenas

Figura 33 – Taxa de recepção instantânea agrupada por número de clientes e repassadores envolvidos na simulação - Topologia A.

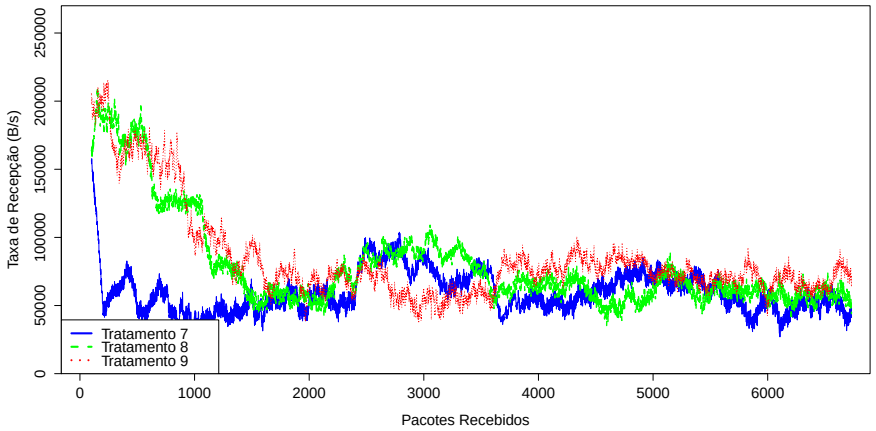
(a) Taxa de recepção instantânea dos Tratamentos 1, 2 e 3 (Topologia A).



(b) Taxa de recepção instantânea dos Tratamentos 4, 5 e 6 (Topologia A).



(c) Taxa de recepção instantânea dos Tratamentos 7, 8 e 9 (Topologia A).



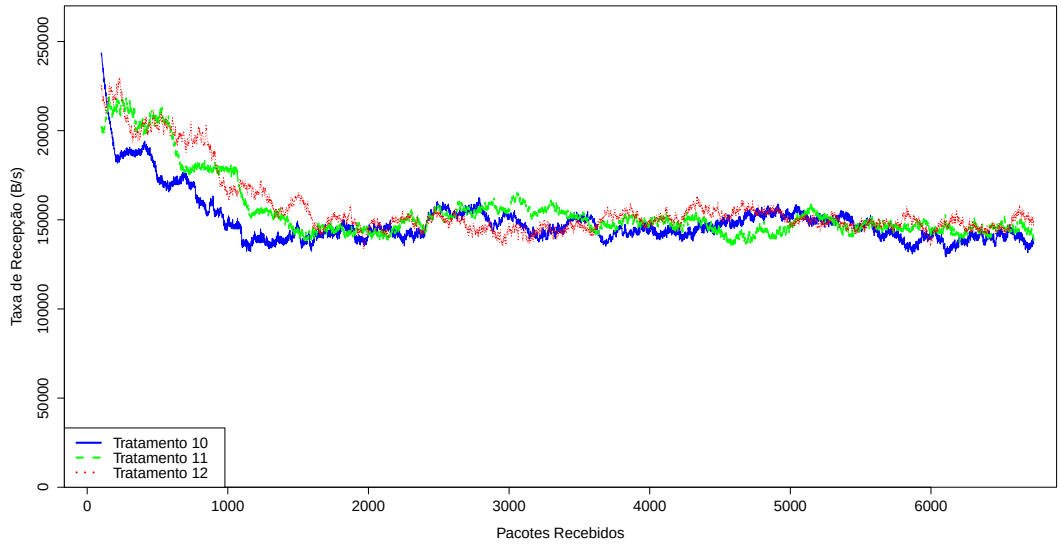
um repassador recebendo a mídia diretamente do servidor, a taxa de recepção instantânea permaneceu durante todo a simulação muito próxima à taxa de envio máxima fornecida pelo servidor. Esse resultado é compatível com os resultados obtidos na Seção 4.1.2. Conforme discute-se na Seção 4.2.2.2, nesses tratamentos não houve perda de pacotes, o que contribui para a manutenção da taxa de transmissão pelo controle de congestionamento *multicast* do GMTP. Nesses tratamentos, as variações na taxa de recepção instantânea se mostraram pequenas em relação aos demais tratamentos, ocasionadas por eventuais variações do RTT durante a transmissão.

Conforme ilustra-se nas Figuras 33b e 33c, nos tratamentos 4 ao 9, onde vários repassadores estavam recebendo a mídia diretamente do servidor, em modo *unicast*, houve uma redução da taxa de recepção instantânea dos clientes quando comparada com a taxa obtida nos tratamentos 1 a 3, durante toda a transmissão. A redução da taxa de recepção instantânea ocorre pelos mesmos motivos da redução da taxa de recepção total: múltiplos fluxos de dados partindo do servidor, aumento da carga de processamento do servidor, e o crescimento na fila de pacotes dos roteadores. Esses fatores influenciam o crescimento do RTT, o que leva o algoritmo de controle de congestionamento a reduzir da taxa de transmissão. Esses resultados estão dentro do que se esperava em um cenário onde não há cooperação entre os repassadores e um alto número de repassadores recebendo a mídia diretamente do servidor.

Conforme ilustra-se na Figura 34, observa-se que nos tratamentos 10, 11 e 12, onde utilizou-se a Topologia B, favorável à formação de parcerias entre os repassadores, a taxa de recepção instantânea foi melhor e mais estável que a taxa observada nos tratamentos 7, 8, 9. Mais uma vez, esses resultados estão dentro do que se esperava em um cenário onde há possibilidade de cooperação entre os repassadores, reduzindo o número de repassadores recebendo a mídia diretamente do servidor.

Além do menor valor, a taxa de recepção instantânea sofreu maior variação da taxa ao logo do tempo nos tratamentos 4 ao 12. Nesses tratamentos há uma maior incidência de perdas de pacotes, conforme relata-se na Seção 4.2.2.2. Os eventos de perdas de pacotes fazem com que o algoritmo de controle de congestionamento *multicast* do GMTP reduza a taxa de envio entre repassadores e clientes, resultando em crescimento na fila de pacotes do repassador. Como consequência, o algoritmo de controle de congestionamento *unicast* do GMTP reduz a taxa de envio do servidor. A redução da taxa de envio ocasiona o esvaziamento dos *buffers* dos roteadores e a redução da perda de pacotes. Com isso, os algoritmos de controle de congestionamento tendem a aumentar a taxa de envio do servidor para os repassadores, e dos repassadores para os clientes. Essa alternância entre aumento e redução da taxa de envio é mais frequente quanto mais fluxos de dados estiverem trafegando na rede e ocasiona o crescimento da variação da taxa de recepção nos clientes. Salienta-se que a variação da taxa instantânea de recepção é ainda maior nos tratamentos

Figura 34 – Taxa de recepção instantânea dos Tratamentos 10, 11 e 12 (Topologia B)



4 ao 9, onde não há cooperação entre os roteadores, e menor nos tratamentos 10 ao 12, onde há cooperação entre os repassadores.

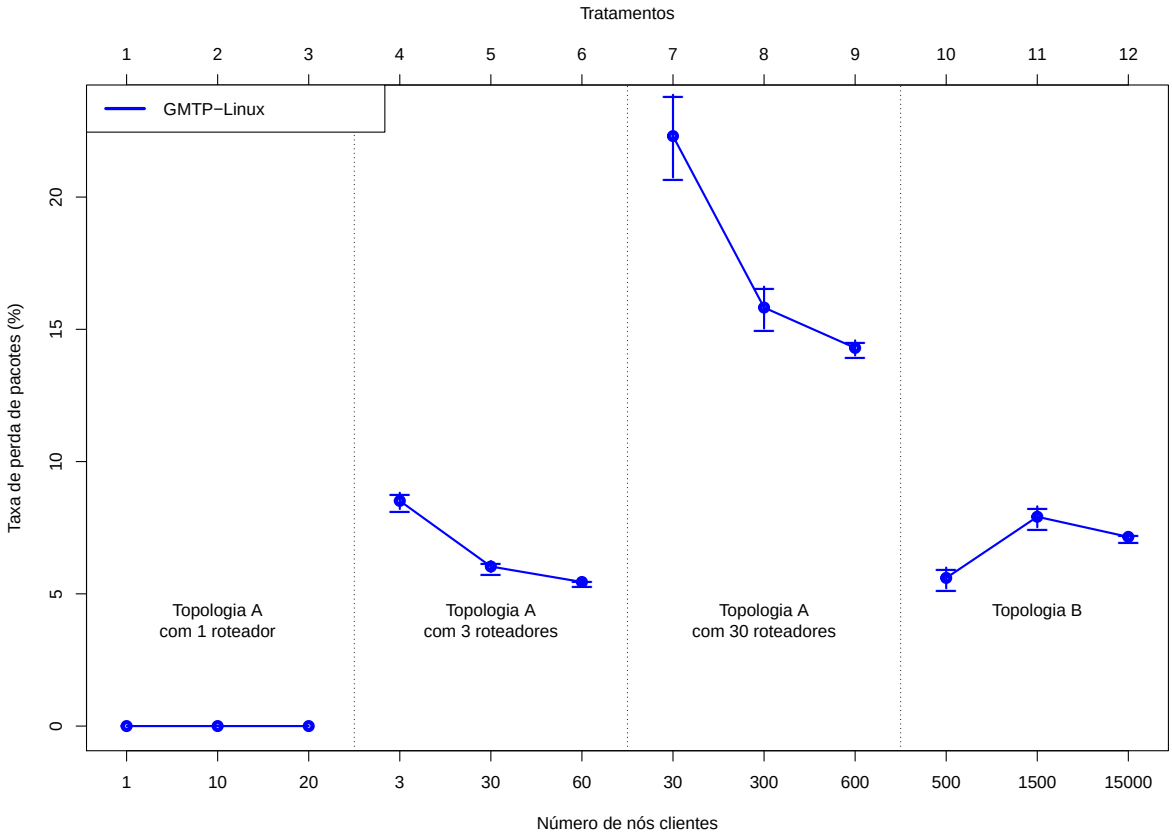
4.2.2.2 Taxa de perdas de pacotes

Diversos algoritmos de controle de congestionamento utilizam a *taxa de perdas de pacotes* para inferir sobre as condições da rede e realizar o controle de congestionamento de forma eficiente. O próprio GMTP-MCC utiliza esta métrica para calcular a taxa de transmissão ideal entre os repassadores e os seus clientes.

Na Figura 35, apresentam-se as taxas de perdas de pacotes medidas em cada tratamento. Para determinar o intervalo de confiança, aplicou-se o teorema central do limite para construir um intervalo de confiança aproximado, conforme detalha-se no Apêndice C.2.1. Observa-se que nos tratamentos onde há poucos repassadores recebendo a mídia diretamente do servidor, a taxa de perdas é menor que nos tratamentos onde há vários repassadores recebendo a mídia diretamente do servidor.

Observam-se taxas de perdas maiores nos tratamentos 7, 8 e 9, que correspondem a um maior número de repassadores recebendo a mídia diretamente do servidor. Deve-se ressaltar que nos tratamentos 4 ao 9, os repassadores não cooperam entre si. Já nos tratamentos 10 ao 12, a taxa de perdas é muito inferior às taxas dos tratamentos 7 ao 9, mesmo esses últimos tendo menos clientes requisitando a mídia. Isso ocorre devido aos benefícios da cooperação entre os repassadores que existe nos tratamentos 10 ao 12. Os eventos de perdas de pacotes implicam na redução da taxa de envio do servidor através dos algoritmos de controle de congestionamento. Portanto, esses resultados são compatíveis

Figura 35 – Resultado dos tratamentos (1-12) para a métrica *taxa de perda de pacotes*.



com os observados na Seção 4.2.2.1, onde observa-se uma menor taxa de recepção de dados para os tratamentos onde a taxa de recepção é menor e mais variável.

Observa-se também que, ao contrário do que se espera nos protocolos de transporte tradicionais, o aumento da razão entre número de clientes por repassador ocasionou uma redução na taxa de perdas de pacotes. Em sistemas de transmissão de conteúdo multimídia ao vivo, onde potencialmente milhares de usuários podem requisitar o conteúdo, uma preocupação é o crescimento acentuado do consumo de recursos computacionais e de rede. Utilizando-se os protocolos de transporte tradicionais, observa-se que o desempenho tende a degradar à medida que se aumenta o número de clientes interessados pela mesma mídia (SALES, 2014). No entanto, observa-se que no GMTP, o desempenho da transmissão multimídia ao vivo não tende a degradar à medida que se aumenta o número de clientes interessados pela mesma mídia. No GMTP, o principal fator que pode ocasionar a degradação da transmissão multimídia é a quantidade de repassadores que estão recebendo a mídia diretamente do servidor, sem formação de parcerias. O número de usuários que requisitam a mesma mídia não é fator determinante para influenciar a qualidade da transmissão de mídia ao vivo no GMTP. Logo, em um cenário onde a maior parte dos roteadores executem o GMTP, torna-se possível transmitir o conteúdo multimídia ao

vivo de alta resolução com qualidade, de forma praticamente independente do número de usuários requisitantes de tal conteúdo.

Nesse contexto, observa-se que nos tratamentos com maior proporção de clientes por repassador, obteve-se um índice menor de perda de pacotes que nos tratamentos com menos clientes por repassador. Isso ocorre devido ao modo de funcionamento do algoritmo de controle de congestionamento *multicast* do GMTP. No GMTP-MCC, quanto mais clientes estiverem conectados aos repassadores, mais nós relatores estarão atuando na rede. O número maior de nós relatores permite que o repassador receba um número maior de relatórios periódicos sobre o estado da transmissão, resultando em informações mais precisas sobre o estado da transmissão, o que ocasiona uma resposta mais apropriada a eventos de congestionamento de rede e, consequentemente, redução no índice de perdas de pacotes.

4.2.2.3 Índice de Continuidade

A métrica *índice de continuidade* é importante, pois mapeia o desempenho do sistema durante todo o ensaio, a partir do instante em que os nós clientes começam a receber o primeiro pacote de dados.

Na Figura 36, apresentam-se os índices de continuidade medidos em cada tratamento. Para determinar o intervalo de confiança, aplicou-se o teorema central do limite para construir um intervalo de confiança aproximado, conforme detalha-se no Apêndice C.2.1. Observa-se que nos tratamentos onde há poucos repassadores recebendo a mídia diretamente do servidor, o índice de continuidade é maior que nos tratamentos onde há vários repassadores recebendo a mídia diretamente do servidor.

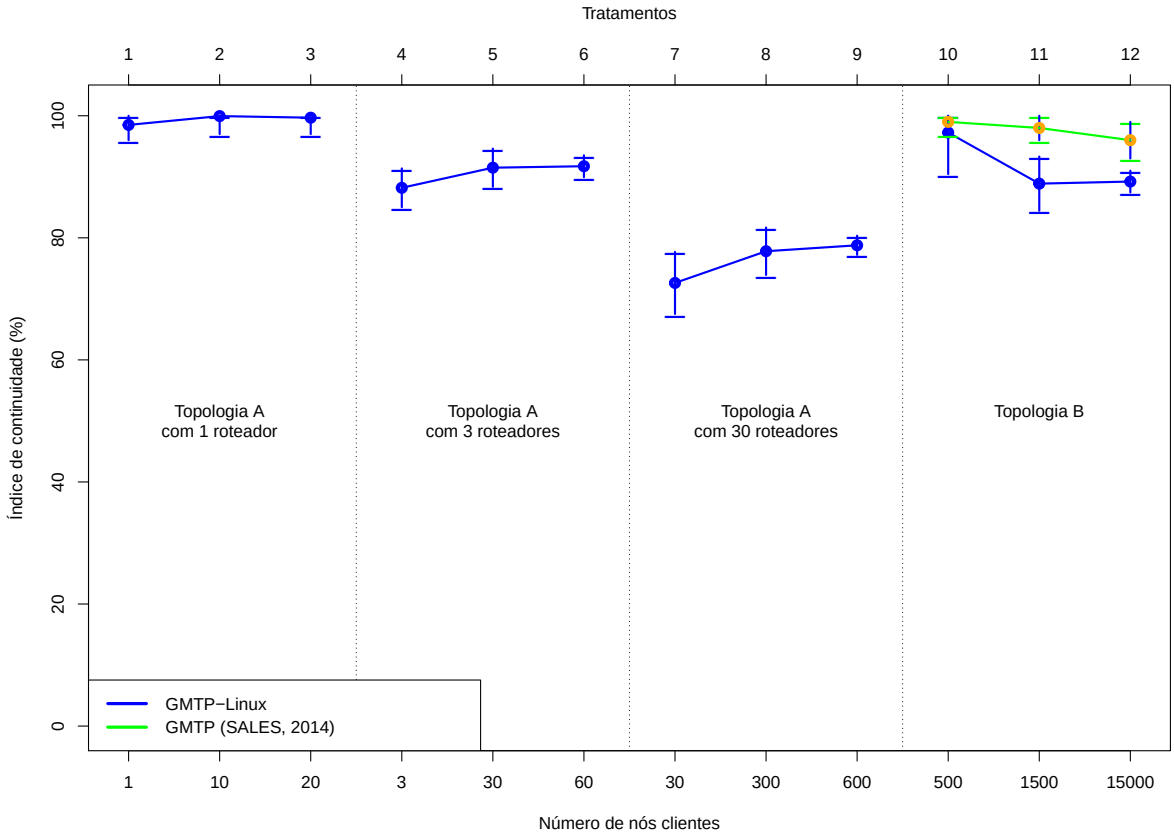
O índice de continuidade está diretamente ligado à taxa de perdas de pacotes e à variação da taxa de recepção instantânea dos clientes. Contabiliza-se o tempo em que os nós clientes não conseguiram sequer receber pacotes de dados da mídia, o que ocorre quando há perdas de pacotes ou quando o intervalo de chegada entre dois pacotes consecutivos é maior que o normal.

Portanto, os resultados obtidos são compatíveis com os observados na Seção 4.2.2.2. Nos tratamentos onde houve as maiores taxas de recepção e os menores índices de perda, o índice de continuidade foi de 100%, enquanto nos demais tratamentos, o índice de continuidade reduziu com a redução da taxa de recepção e o aumento do índice de perdas.

Conforme observa-se na Figura 36, o índice de continuidade observado nos tratamentos 10, 11 e 12 da simulação é um pouco inferior ao índice de continuidade observado nos resultados de simulação apresentados em (SALES, 2014). Esse resultado já era esperado, considerando alguns fatores como:

1. O arcabouço de simulação utilizado em (SALES, 2014) considera a perda de pacotes

Figura 36 – Resultado dos tratamentos (1-12) para a métrica *índice de continuidade*.



como variável independente, onde definiu-se uma função de probabilidade para modelar a perda de pacotes nos roteadores. Neste trabalho, os eventos de perda de pacotes são simulados pelo arcabouço de simulação, sem intervenção do usuário, conforme o tráfego de rede e a capacidade dos canais de transmissão.

2. O mecanismo de *busca por mais parceiros*, onde um nó repassador pode enviar periodicamente um pacote do tipo *GMTP-RelayQuery* para o nó servidor a fim de descobrir melhores parceiros e aumentar o número de parcerias, foi implementado na versão do GMTP-Omnet++ (SALES, 2014), mas não foi implementado no contexto deste trabalho. Assim, a diferença de desempenho pode ser uma consequência da ausência dessa função no GMTP-Linux.

4.2.2.4 Sobrecarga de controle

Diferentemente das métricas anteriores, que são relacionadas à qualidade de serviço oferecida à aplicação, a sobrecarga de controle é uma relação de proporção entre a quantidade de pacotes de controle (sem dados de aplicação) e a quantidade de dados da aplicação que precisam ser transmitidos. Para obter o valor correspondente, mede-se a quantidade de pacotes de controle transmitido e contabiliza-se a quantidade de dados de

controle transmitidos através da rede, com base no tamanho do cabeçalho de cada tipo de pacote.

Em geral, os pacotes de controle que o GMTP transmite se dividem em dois tipos: locais e inter-redes. Os pacotes de controle locais circulam na rede local do repassador de origem, e não atingem o servidor. Já os pacotes de controle inter-redes são pacotes de controle que os repassadores trocam entre si ou com o servidor.

Neste contexto, na Figura 37, apresentam-se os resultados relativos à quantidade de dados de controle medidos em cada tratamento. Observa-se que nos tratamentos onde há poucos nós na rede, a quantidade total de dados de controle é menor que nos tratamentos onde há vários nós.

Entre os pacotes de controle locais estão os pacotes do 3WH (*three way handshake*) entre cliente e repassador, os pacotes do controle de congestionamento *multicast*, e os pacotes do mecanismo de *keep alive* entre clientes e relatores. Observa-se que a quantidade total de pacotes de controle locais está relacionada ao número de clientes por repassador. Esse resultado já era esperado, visto que quanto mais clientes estiverem conectados a um repassador, mais vezes ocorrerá o 3WH, mais relatores estarão na rede, enviando relatórios periódicos aos repassadores, e mais pacotes de *keep-alive* serão trocados entre clientes e relatores.

No entanto, observa-se que quanto mais clientes estiverem conectados à rede, menor será a quantidade de pacotes de controle por cliente. A quantidade de pacotes de controle por cliente diminui à medida que o número de clientes aumenta. Esse resultado era esperado, porque:

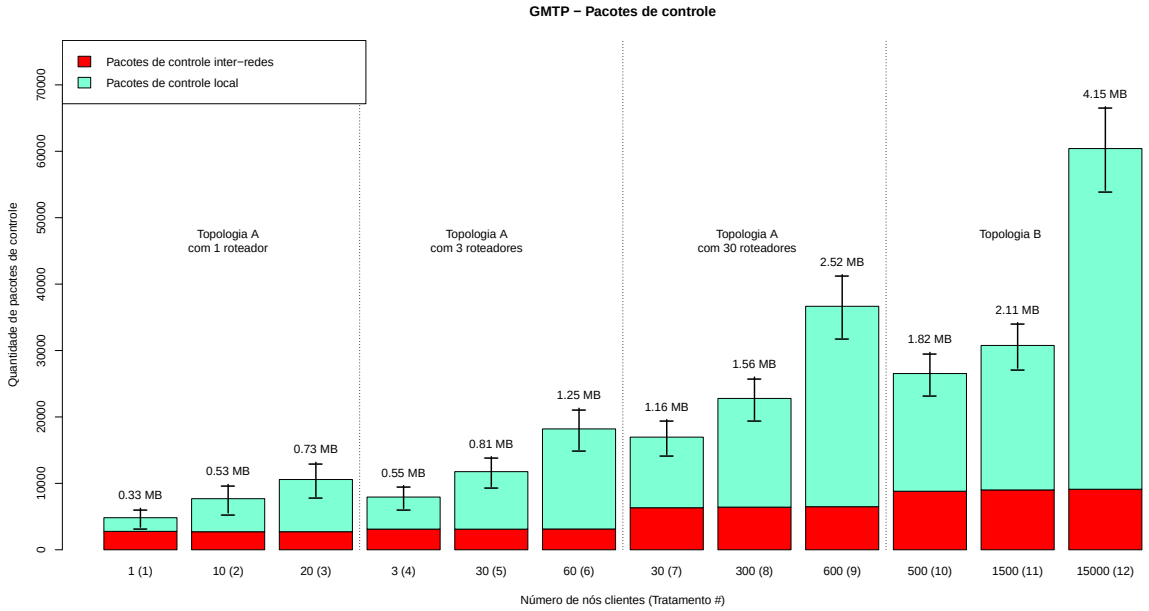
- o mecanismo de eleição de relatores possibilita que apenas uma fração dos nós clientes enviem pacotes de controle periodicamente aos seus repassadores. Logo, o aumento da quantidade de pacotes de controle trocados entre os relatores e os repassadores não ocorre na mesma medida que o aumento no número de clientes na rede.

- a quantidade de pacotes de controle trocados entre clientes e relatores para fins de manutenção da conexão (*keep-alive*) é menor que a quantidade de dados de controle trocados entre relatores e repassadores.

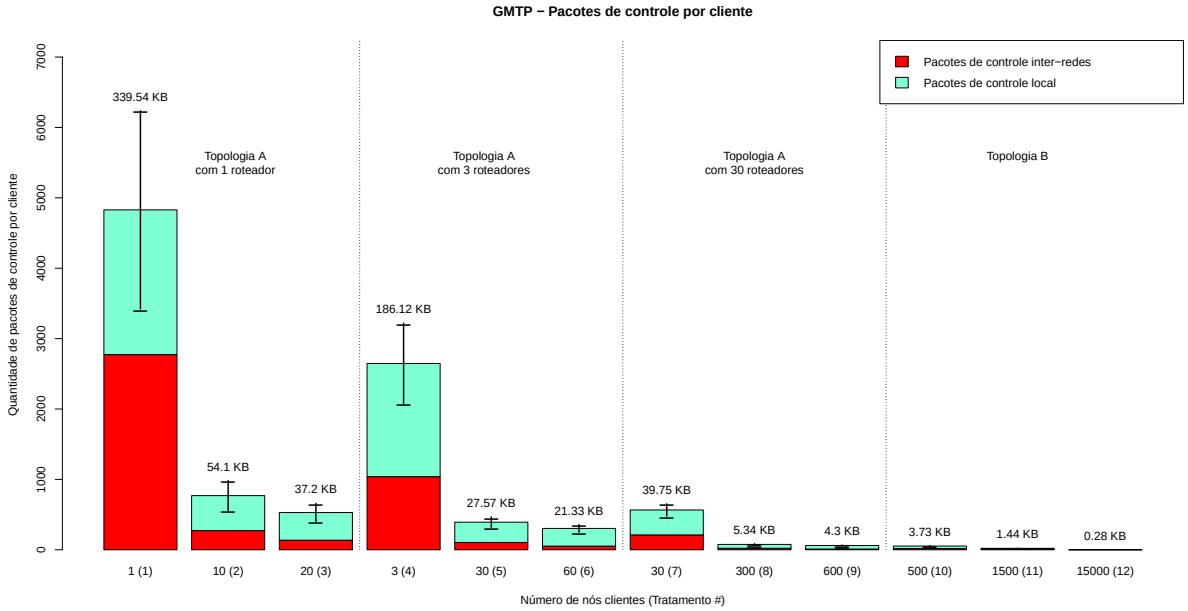
Conforme discute-se na Seção 4.2.2.2, o aumento no número de pacotes de controle locais não necessariamente é prejudicial ao desempenho do GMTP. Nos tratamentos 4 ao 9, o aumento na quantidade de pacotes de controle acompanhou uma redução da taxa de perdas de pacotes. Nesses tratamentos, o aumento da quantidade de relatórios sobre o estado da transmissão, resultou em informações mais precisas sobre o estado da transmissão para o repassador, que agiu apropriadamente para reduzir o índice de perdas de pacotes.

Figura 37 – Resultado dos tratamentos (1-12) para a métrica *sobrecarga de controle*.

(a) Resultado dos tratamentos (1-12) para a métrica *sobrecarga de controle* (total).



(b) Resultado dos tratamentos (1-12) para a métrica *sobrecarga de controle* (por cliente).



Entre os pacotes de controle inter-redes, estão pacotes do registro e da renovação do registro de participação entre repassadores e servidor, os relacionados ao estabelecimento de parcerias entre repassadores e pacotes do controle de congestionamento *unicast*. Nesse contexto, deve-se considerar que os pacotes do tipo *GMTP-Register-Reply* aumentam de tamanho gradativamente, conforme seguem seu caminho do servidor até o repassador. Além disso, no processo de registro de participação, alguns pacotes transmitidos podem ser interceptados por nós repassadores, reduzindo-se antecipadamente a quantidade de tráfego

de controle. Nesse contexto, observa-se que a quantidade de pacotes de controle inter-redes é proporcional ao número de repassadores na rede. Esse resultado já era esperado, visto que quanto mais repassadores estiverem na rede, mais pacotes de controle de congestionamento e de renovação de registro serão trocados entre repassadores e servidor.

Observa-se ainda que a quantidade de pacotes de controle inter-redes não cresce na mesma proporção do número de repassadores na rede. Esse resultado era esperado, porque:

- nos tratamentos onde não há parcerias entre os roteadores, observou-se a diminuição da taxa de recepção e do índice de continuidade. A redução dos valores observados para essas métricas sugere que, nesses tratamentos, houve aumento do valor do RTT. De fato, a análise dos arquivos de registro do GMTP confirma o aumento do valor do RTT. Salienta-se que no GMTP, define-se que o pacote de controle (*GMTP-Ack*) deverá ser enviado ao servidor a cada intervalo de RTT. Logo, o crescimento do valor do RTT ocasiona a redução da quantidade de pacotes de controle inter-redes trafegados.

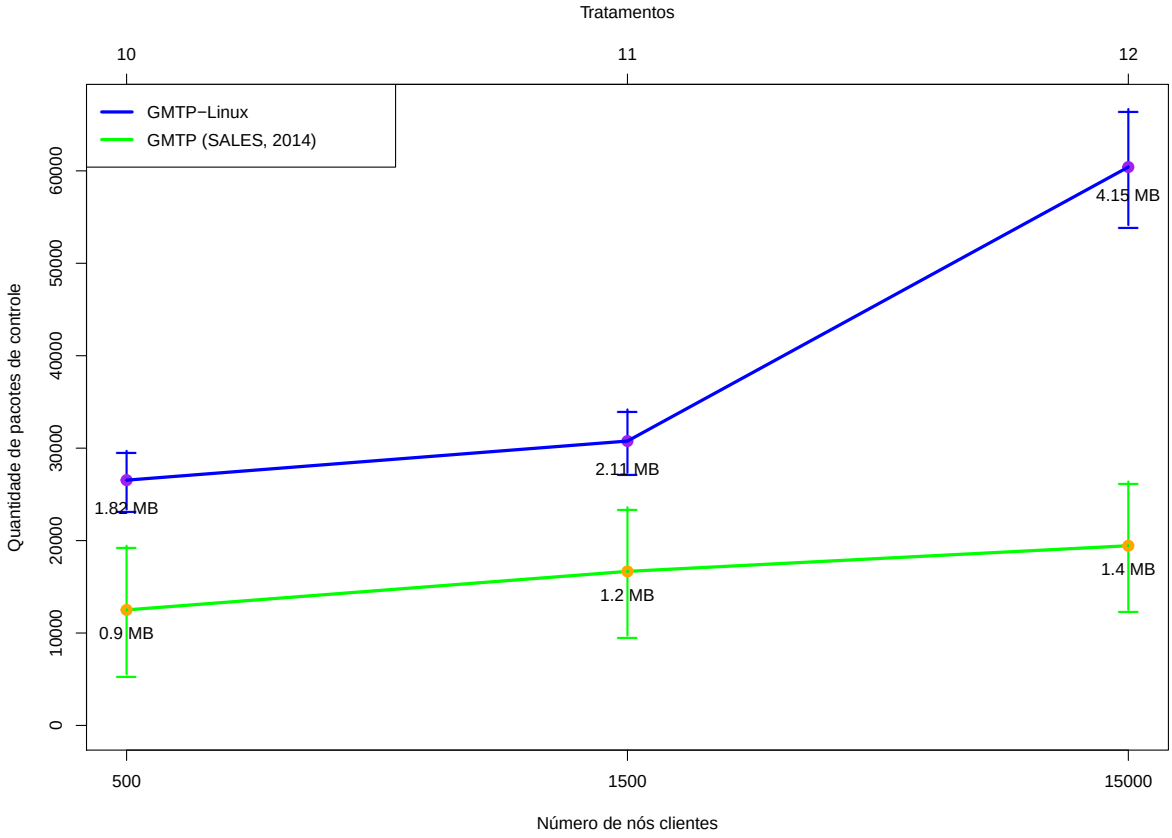
- nos tratamentos onde há parcerias entre os roteadores (Tratamentos 10 ao 12), nem todos os pacotes de controle inter-redes alcançam o servidor, pois muitos desses pacotes trafegam apenas entre dois repassadores parceiros. No contexto deste trabalho, a medição da quantidade de pacotes de controle inter-redes foi realizada apenas no nó servidor. Assim, não se fez a contagem dos pacotes de controle que trafegam apenas entre dois repassadores parceiros, sem alcançar o servidor.

Observa-se também que a quantidade de pacotes de controle locais foi superior à quantidade de pacotes de controle inter-redes porque, nas simulações, a quantidade de nós clientes foi maior ou igual à quantidade de repassadores. Salienta-se que, como a maior parte dos pacotes do GMTP de controle é local, não observa-se a degradação da transmissão como consequência da geração de pacotes de controle.

Na Figura 38, apresentam-se os resultados relacionados à quantidade total de dados de controle medidos em cada tratamento, em comparação com o GMTP-Omnet++ (SALES, 2014).

Observa-se que a quantidade de dados de controle do GMTP-Linux apresenta maior variação em relação ao número de nós clientes que o GMTP-Omnet++. Em geral, a versão do GMTP-Linux gera mais pacotes de controle que a versão do GMTP-Omnet++. Esse resultado já era esperado, considerando que definiu-se e implementou-se o mecanismo de *keep-alive* na rede local. Como consequência, o GMTP-Linux gera pacotes adicionais de controle que são trocados na rede local, entre os clientes e os relatores.

Figura 38 – Detalhe sobre o resultado dos tratamentos (10-12) para a métrica *sobrecarga de controle*.



5 CONSIDERAÇÕES FINAIS

Neste trabalho, apresentou-se um estudo da aplicabilidade do GMTP através da sua implementação no sistema operacional Linux. Dessa forma, validou-se a especificação do GMTP no que diz respeito a sua implementação. Para isto, implementou-se o GMTP sem utilizar como base nenhuma outra implementação existente, visando validar a especificação do protocolo no que diz respeito a sua implementação. Avaliou-se a implementação do GMTP-Linux em termos de desempenho e, como resultado deste trabalho, identificaram-se limitações e propuseram-se melhorias no GMTP.

Na perspectiva de implementação, efetuou-se um estudo da API de desenvolvimento de novos protocolos de transporte na pilha de rede do sistema operacional Linux. Estudou-se detalhadamente a especificação original do protocolo GMTP e suas funções, registrando-se quais funções poderiam ser implementadas tal como especificadas e quais deveriam ser adaptadas de acordo com as limitações do sistema operacional Linux ou das aplicações. Nesse contexto, definiram-se funções e mecanismos que foram apenas citados, sem definição detalhada, ou parcialmente definidos na especificação original do protocolo. Além disso, definiram-se funções e mecanismos não previstos na especificação original do GMTP, quando necessário.

Assim, implementou-se e configurou-se o GMTP no núcleo do Linux. A implementação provê uma abstração para a camada de aplicação de modo que os processos em execução utilizem o GMTP através de uma API compatível com as especificações de *socket* BSD e POSIX, facilitando o uso do GMTP nos atuais e futuros sistemas. Neste sentido, este trabalho contribui diretamente com o avanço no desenvolvimento do GMTP em direção a chancelá-lo como um padrão definido IETF, em busca de melhorar o desempenho das aplicações multimídia na Internet e utilizar, de forma mais eficiente, os recursos de rede, especialmente quando se trata de fluxos multimídia ao vivo.

Em seguida, realizaram-se testes básicos do acesso aos recursos do GMTP pela API de *socket* BSD e POSIX, através de aplicações de rede, utilizando máquinas virtuais. Além dos testes básicos, realizaram-se testes de desempenho sobre a referida implementação através de simulações de rede, acoplando a implementação do GMTP no *Kernel* do Linux em um simulador de rede. Nesse sentido, este trabalho contribui com a avaliação o GMTP, a proposta de melhorias e uma comparação entre os resultados da versão do GMTP para simulador com o os resultados obtidos através da implementação do GMTP em um sistema operacional.

Além disso, este trabalho traz como contribuição a primeira versão de uma implementação do GMTP em um sistema operacional equivalente à implementação feita para

simulador. Ressalta-se que, conforme discute-se na Seção 1.1, a implementação do GMTP em um sistema operacional apresenta desafios específicos, que podem se agravar quando se trata da possibilidade de se ter o GMTP funcionando em múltiplos sistemas operacionais, ou mesmo em diferentes versões do mesmo sistema. Nesse sentido, este trabalho contribui ao apresentar propostas de soluções para diversos problemas, tais como:

- Adaptação do protocolo à API de rede do Linux;
- Soluções para utilização de temporizadores e *threads* quando necessário;
- Definição das estruturas de dados necessárias para representar entidades ou armazenar estados e informações previstos no protocolo;
- Elaboração de soluções alternativas para contornar a ausência de operações de ponto flutuante no núcleo do Linux.

É importante salientar que o *feedback* de implementações reais de protocolos é de fundamental importância para avaliar as soluções teóricas na Internet. Nesse contexto, este trabalho contribui com a implementação do GMTP e o conjunto de ferramentas associadas, a fim de demonstrar se a abordagem do protocolo GMTP é viável e praticável para uso das aplicações de rede.

5.1 Conclusões

A proposta de uma solução para transporte de dados multimídia através do GMTP é viável e aplicável, sob o ponto de vista da implementação de um protocolo de transporte na pilha de rede do sistema operacional Linux. Nesse contexto, implementou-se o GMTP no núcleo do Linux sem utilizar como base nenhuma outra implementação existente. Através de experimentos e simulações, demonstrou-se qual é o desempenho de uma implementação do GMTP em um sistema operacional real considerando algumas topologias de rede proeminentes e que simulam parte da Internet.

As funções definidas originalmente no GMTP foram, em sua maioria, completamente compatíveis com a API do Linux. Algumas funções precisaram de adaptações para funcionar corretamente no Linux. Nesse contexto, necessitou-se realizar adaptações e melhorias em relação à especificação original do GMTP a fim de aplicá-lo em um sistema operacional. Foi necessário definir, criar e implementar diversas mecanismos e estruturas para alcançar o pleno funcionamento do protocolo. Além disso, criou-se um novo mecanismo de formação de parcerias, além do já especificado originalmente, para aprimorar o desempenho do protocolo.

No testes básicos, demonstrou-se que o acesso aos recursos do GMTP pela API de *socket* BSD e POSIX, através de aplicações de rede, é viável e aplicável. Nos testes de

desempenho, através de simulações em maior escala, acoplou-se a implementação do GMTP no *Kernel* do Linux em um simulador de rede. Em seguida, com base em simulações, observou-se que o os resultados relativos ao desempenho do GMTP implementado no Linux são semelhantes aos resultados disponíveis em (SALES, 2014). Salienta-se que as funções definidas originalmente no GMTP poderiam não ser completamente compatíveis com o Linux, necessitando ser adaptadas de acordo com a API disponível. Esta foi uma das principais motivações para investigar as possíveis limitações que poderiam restringir o pleno funcionamento do protocolo e avaliar a aplicabilidade do GMTP.

Como resultado deste trabalho, é possível utilizar o GMTP a fim de transmitir ou receber dados multimídia na Internet, desde que os sistemas finais utilizem o Linux. Nesse contexto, é necessário integrar o GMTP-Inter em uma distribuição Linux direcionada a roteadores como, por exemplo, o OpenWrt, para que os roteadores participem da distribuição de mídia ativamente, através do GMTP. O conceito de *socket P2P* empregado no GMTP permite uma abstração de distribuição P2P com suporte transparente ao uso de *multicast* por parte da aplicação, tornando o desenvolvimento de sistemas mais rápido e com menor chances de erros devido ao emprego de funções de *software* relacionadas à distribuição de mídias ao vivo desacopladas da aplicação.

Desta forma, implementou-se e configurou-se o GMTP no núcleo do Linux. Realizaram-se testes básicos do acesso aos recursos do GMTP, utilizando-se máquinas virtuais, e testes de desempenho em diferentes cenários de rede. Nesses testes, observou-se que o os resultados relativos ao desempenho do GMTP implementado no Linux são semelhantes aos resultados disponíveis em (SALES, 2014). Deste modo, conclui-se que a abordagem do protocolo GMTP é viável e praticável para uso das aplicações em um sistema operacional.

5.2 Trabalhos Futuros

Nessa seção, apresentam-se as propostas de trabalhos futuros relacionados ao GMTP. Alguns recursos definidos na especificação original do GMTP não foram implementados, devendo ser desenvolvidos futuramente. Segue abaixo a lista dos recursos não implementados.

1. *Busca por mais parceiros*: na especificação original do GMTP, preconiza-se que o nó repassador pode enviar periodicamente um pacote do tipo *GMTP-RelayQuery* para o nó servidor a fim de descobrir melhores parceiros e aumentar o número de parcerias. O nó servidor deve então construir uma lista de possíveis nós parceiros e enviá-la de volta ao repassador. No contexto deste trabalho, esse procedimento de requisição explícita de parcerias do repassador para o servidor não foi implementado. Em compensação, propôs-se um novo mecanismo através do qual o servidor ativamente indica uma nova parceria a um repassador, sem que tenha sido solicitado a fazê-lo.

2. *Chave de autorização entre servidores e repassadores*: na especificação original do GMTP, preconiza-se que a cada registro de participação que um repassador r qualquer envia para um servidor, este deve gerar uma chave de autorização única para r e transmiti-la como resposta no pacote do tipo *GMTP-Register-Reply* (Seção 2.2.2.3). No contexto deste trabalho, esse procedimento de criação da chave de autorização entre servidores e repassadores não foi implementado, pois o objetivo do referido procedimento é viabilizar o mecanismo de busca por mais parceiros através dos pacotes do tipo *GMTP-RelayQuery*.

3. *Mapa de Buffer*: na especificação original do GMTP, descrevem-se dois tipos de pacote denominados *GMTP-DataPull-Request* e *GMTP-DataPull-Response*. O pacote do tipo *GMTP-DataPull-Request* tem como objetivo possibilitar que um repassador envie um pedido para obter o mapa de *buffer* atual de um repassador parceiro, enquanto o pacote do tipo *GMTP-DataPull-Response* deve carregar a resposta ao pedido de obtenção do mapa de *buffer*. Nesse contexto, quando um nó repassador r_i transmite um mapa de *buffer* para um outro nó qualquer r_j, caracteriza-se automaticamente o uso do método *pull*, em vez do método *push*, que é o modo padrão do GMTP. Salienta-se ainda que na proposta original do GMTP, orienta-se que o método *pull* deve ser evitado devido à transitoriedade dos pacotes de dados. Assim, um nó r_i apenas realiza tal procedimento após completar a fase de busca por mais parceiros (utilizando *GMTP-RelayQuery*) do processo de estabelecimento de conexão (SALES, 2014). No contexto deste trabalho, as funções relacionadas ao mapa de *buffer* não foram implementadas, pois dependem das funções relacionadas aos pacotes do tipo *GMTP-RelayQuery*, que também não foram implementadas.

4. *Descrição da mídia*: na especificação original do GMTP, descreve-se um tipo de pacote denominado *GMTP-MediaDesc*. A função desse tipo de pacote é descrever informações sobre a mídia sendo transmitida em um determinado fluxo de dados. Esse pacote é gerado pelo servidor e pode ser processo e/ou distribuído pelos repassadores. No contexto deste trabalho, as funções relacionadas à descrição da mídia este tipo de pacote não foram implementadas.

5. *Verificação de autenticidade*: não implementou-se a função de verificação de autenticidade do fluxo de dados do GMTP, cuja descrição encontra-se na Seção 2.2.7. Salienta-se que a função de verificação de autenticidade no GMTP é opcional e desabilitada por padrão, porque um sistema de transmissão, em execução na camada de aplicação, pode ou não desejar tal função.

6. *Uso do canal de controle para enviar o GMTP-Close do cliente*: no contexto deste trabalho, não foi possível implementar o procedimento de desconexão através do canal de controle, porque as tentativas de implementação dessa funcionalidade

não foram executadas com sucesso. Como clientes GMTP sempre se conectam ao servidor através do seu endereço IP (*unicast*) e porta, criando-se um *socket unicast*, no contexto deste trabalho, adotou-se provisoriamente o procedimento de desconexão onde o nó cliente envia um pacote do tipo *GMTP-Close* através do *socket unicast*, endereçado ao servidor. Em seguida, o repassador de origem do cliente intercepta o pacote e responde ao nó cliente com um pacote do tipo *GMTP-Reset*, sinalizando que está ciente do fechamento da conexão. Na Seção 3.4.6, discutiu-se com mais detalhes sobre o procedimento para encerramento da transmissão por parte dos clientes.

7. *Eleição de novo relator*: No contexto deste trabalho, não implementou-se o procedimento de eleição de um novo relator quando se desconecta da rede. No entanto, implementou-se o mecanismo que possibilita que um cliente se autopromova a relator após determinado período, caso seu relator saia da rede.

A seguir, enumeram-se propostas de temas que podem ser objeto de estudo futuro com vistas a aprimorar o GMTP.

1. É necessário investigar o consumo de recursos computacionais que a interceptação de requisições e formação de parcerias pode ocasionar ao repassador. Assim, deve-se encontrar uma forma de calcular o número sub-ótimo de parceiros para cada repassador.

2. Investigar uma forma de encontrar a proporção sub-ótima de clientes por relator para uma transmissão multimídia, de acordo com o estado da rede. Conforme discute-se na 4.2.2.2, observou-se que o aumento do número de clientes por repassador ocasionou uma redução na taxa de perdas de pacotes. Isso ocorreu porque quanto mais clientes estiverem conectados aos repassadores, mais nós relatores estarão atuando na rede. Isso permitiu que o repassador tivesse mais precisão sobre as informações relacionadas ao estado da transmissão e respondesse apropriadamente aos eventos de congestionamento de rede. Logo, deve-se investigar uma forma de calcular o número sub-ótimo de relatores dado determinada quantidade de clientes.

3. Estudar a viabilidade de um mecanismo que possibilite a um repassador continuar registrado no servidor, mesmo sem clientes ativos ou repassadores parceiros. Esse mecanismo amenizaria o possível impacto do *churn*. Por exemplo, pode haver situações onde o repassador tem apenas um cliente conectado, que saiu da rede durante a transmissão. Pode-se considerar o caso onde executa-se a aplicação cliente em um dispositivo móvel, que saiu momentaneamente da zona de cobertura da rede sem fio local. Caso o repassador continue registrado, é possível que o cliente volte a receber a mídia assim que voltar à zona de cobertura, sem a necessidade de um novo registro do repassador no servidor. Mesmo que o primeiro cliente não

retorne, é possível que outro cliente local receba o mesmo evento ao vivo sem que o repassador necessite se registrar novamente. Além disso, com a manutenção do registro, é possível que o repassador inativo contribua em algum momento para a rede como parceiro de outro repassador.

4. Pesquisar formas de otimizar o valor do *timeout* para renovação do registro de participação. A renovação do registro de participação é importante para atualizar a tabela de caminhos no servidor. Caso o valor do *timeout* da renovação seja superior ao valor ideal, o servidor manterá a tabela de caminhos desatualizada por mais tempo que o ideal, e ocasionalmente deixará de propor as melhores parcerias aos repassadores. Por outro lado, se o valor do *timeout* da renovação for inferior ao valor ideal, poderá haver sobrecarga de pacotes de controle, além de sobrecarga de processamento no servidor, que consumirá seus recursos computacionais consultando a tabela de caminhos com frequência desnecessária. Esse aspecto do GMTP pode ser revisto e aprimorado, tanto no que se refere ao intervalo ideal para a renovação do registro de participação, quanto aos algoritmos de cálculo e rotas e parcerias.

5. Adaptar uma aplicação de referência para utilização do protocolo GMTP. Nesse caso, sugere-se também avaliar o grau de complexidade de adaptar um sistema existente para utilizar o GMTP.

6. Integrar o GMTP-Inter em uma distribuição Linux direcionada a roteadores como, por exemplo, o OpenWrt.

7. Efetuar testes do GMTP em uma rede real, concorrendo com tráfego comum de uma rede local e de Internet. Nesse sentido, deve-se investigar o impacto do tráfego de dados gerados por outros protocolos sobre o GMTP.

8. Estudar o desempenho do GMTP em diversas topologias de rede, incluindo redes com diferentes padrões de mobilidade, como redes de celular e redes veiculares.

5.3 Resumo das Contribuições

Dentre as principais contribuições deste trabalho, estão:

1. Propostas de melhorias e adaptações ao protocolo GMTP.

   a) Mudanças na organização do cabeçalho fixo;

   b) Adição de novos tipos de pacote;

   c) Alterações na estrutura da tabela de repasse;

   d) Alteração no mecanismo de *keep-alive* dos repassadores;

e) Simplificação no mecanismo do GMTP-UCC, para evitar sobrecarga de pacotes de controle;

f) Adição de um novo mecanismo de ampliação de parcerias entre os roteadores, baseado na iniciativa do servidor;

g) Alteração dos mecanismos de finalização de conexão.

2. Definição e mecanismos e estruturas internas do protocolo, algumas delas específicas para a API do Linux

   a) Definição da estrutura das tabelas dos clientes;

   b) Mecanismo de compartilhamento do *socket* dos clientes GMTP no *host* local.

   c) Mecanismo de entrega dos pacotes *multicast* de forma transparente às aplicações (que instanciam um *socket unicast*).

   d) Definição da estrutura das tabelas dos servidores;

   e) Alterações na estrutura da tabela de repasse;

3. Criação de uma API geral para construção de tabelas *hash* que, no contexto deste trabalho, foi utilizada para construir as tabelas do GMTP-Intra. Potencialmente, a API pode ser melhorada e utilizada para propósito geral, servindo como contribuição para desenvolvedores do Linux.

4. Identificação de limitações na implementação do GMTP.

5. Avaliação da implementação do GMTP em termos de funcionalidade e desempenho.

   a) Experimentos executados em máquinas virtuais, com o objetivo avaliar a correção da implementação e sua consonância com a especificação original.

   b) Simulações de desempenho com o objetivo de avaliar o desempenho da implementação do GMTP no núcleo do Linux, com suas respectivas adaptações, e o possível impacto que as mudanças propostas possam ter no desempenho do GMTP.

   c) Comparação dos resultados obtidos com os resultados disponíveis em (SALES, 2014).

Além das contribuições supracitadas, destacam-se outras contribuições importantes:

1. Estudo da aplicabilidade do GMTP, através da sua implementação no sistema operacional Linux.

   a) Validação da especificação do GMTP no que diz respeito a sua implementação. Para isto, implementou-se o GMTP sem utilizar como base nenhuma outra implementação existente.

   b) Implementação e configuração do GMTP no núcleo do Linux. Essa implementação provê uma abstração para a camada de aplicação de modo que os processos em execução utilizem o GMTP através de uma API compatível com as especificações de *socket* BSD e POSIX.

   c) Exemplos de aplicações para utilização do GMTP na Internet.

2. Definição detalhada de funções e mecanismos parcialmente ou implicitamente definidos na especificação original do GMTP.

   a) Especificação da organização dos cabeçalhos variáveis;

   b) Especificação de códigos para o pacote do tipo *RequestNotify*;

   c) Definição do mecanismo de *keep-alive* dos clientes através dos relatores.

   d) Especificação das opções de *socket* (*sockopt*) do GMTP.

   e) Definição dos estados do *socket* GMTP.

   f) Definição dos campos disponíveis na estrutura do *socket* GMTP.

3. Estudo e documentação da API de desenvolvimento de novos protocolos de transporte na pilha de rede do sistema operacional Linux.

4. Registro das funções do GMTP implementadas tal como especificadas e adaptadas de acordo com as limitações do sistema operacional Linux ou das aplicações.

REFERÊNCIAS

BENVENUTI, C. *Understanding Linux Network Internals*. [S.l.]: O'Reilly Media, Inc., 2005. ISBN 0596002556.

BRADEN, R. *Requirements for Internet Hosts – Communication Layers*. 1989. Último acesso: 9 de junho de 2016. Disponível em: <http://www.ietf.org/rfc/rfc1122.txt>.

BRADNER, S. *The Internet Standards Process – Revision 3*. 1996. Último acesso: 9 de junho de 2016. Disponível em: <https://tools.ietf.org/html/rfc2026>.

CAMARA, D. et al. Dce: Test the real code of your protocols and applications over simulated networks. *Communications Magazine, IEEE*, v. 52, n. 3, p. 104–110, March 2014. ISSN 0163-6804.

CARPENTER, B. E. *Architectural principles of the Internet*. 1996. Último acesso: 9 de junho de 2016. Disponível em: <https://www.ietf.org/rfc/rfc1958.txt>.

CIANCAGLINI, V. et al. CCN-TV: A Data-centric Approach to Real-Time Video Services. In: *Advanced Information Networking and Applications Workshops (WAINA), 27th International Conference on*. [S.l.: s.n.], 2013. p. 982–989.

DUKKIPATI, N. *Rate Control Protocol (RCP): Congestion control to make flows complete quickly*. Tese (Doutorado) — Stanford University, Stanford, CA, USA, 2007.

DUKKIPATI, N. et al. Processor sharing flows in the internet. In: *Quality of Service–IWQoS 2005*. [S.l.]: Springer, 2005. p. 271–285.

HANDLEY, M. et al. *TCP Friendly Rate Control (TFRC): Protocol Specification*. 2003. Último acesso: 9 de junho de 2016. Disponível em: <http://www.ietf.org/rfc/rfc3448.txt>.

HENDERSON, T. R.; LACAGE, M.; RILEY, G. F. Network simulations with the ns-3 simulator. In: CITESEER. *In Sigcomm (Demo)*. 2008. Último acesso: 9 de junho de 2016. Disponível em: <http://conferences.sigcomm.org/sigcomm/current/papers/p527-hendersonA.pdf>.

JAHROMI, A. F. Temporal Scalable Live Video Streaming over Hybrid CDN-P2P Architecture. *International Journal of Computer Applications*, v. 46, n. 17, p. 14–20, 5 2012. Published by Foundation of Computer Science, New York, USA.

KUROSE, K. W. R. J. F. *Redes de computadores e a Internet: uma abordagem top-down*. 5. ed. São Paulo: Addison Wesley, 2010. 428-483 p.

LEACH, P.; MEALLING, M.; SALZ, R. *A Universally Unique IDentifier (UUID) URN Namespace*. 2005. Último acesso: 9 de junho de 2016. Disponível em: <http://www.ietf.org/rfc/rfc4122.txt>.

LIU, Y.; GUO, Y.; LIANG, C. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications*, Springer, v. 1, n. 1, p. 18–28, 2008.

LIU, Y. et al. Friendly P2P: Application-level congestion control for peer-to-peer applications. In: IEEE. *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. [S.l.], 2008. p. 1–5.

MESKOVIC, M.; BAJRIC, H.; KOS, M. Content delivery architectures for live video streaming: Hybrid cdn-p2p as the best option. In: *CTRQ 2012, The Fifth International Conference on Communication Theory, Reliability, and Quality of Service*. [S.l.: s.n.], 2012. p. 26–32.

MOGUL, J.; DECWRL; DEERING, S. *Path MTU Discovery*. 1990. Último acesso: 9 de junho de 2016. Disponível em: <http://www.ietf.org/rfc/rfc1191.txt>.

PADHYE, J. et al. 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*. New York, NY, USA: ACM Press, 1998. p. 303–314.

ROSEN, R. *Linux Kernel Networking: Implementation and Theory*. 1st. ed. Berkely, CA, USA: Apress, 2013. ISBN 143026196X, 9781430261964.

SALES, L. M. de. *GMTP: Distribuição de Mídias Ao Vivo através de uma Rede de Favores Constituída entre Roteadores*. 2014. Tese (Doutorado) — Centro de Engenharia Elétrica e Informática - Universidade Federal de Campina Grande, Campina Grande, PB, Brasil, 2014.

SEVERANCE, C. Van Jacobson: Content-Centric Networking. *Computer*, v. 46, n. 1, p. 11–13, 2013. ISSN 0018-9162. Último acesso: 9 de junho de 2016. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6419703>.

SEYYEDI, S. M. Y.; AKBARI, B. Hybrid CDN-P2P Architectures for Live Video streaming: Comparative Study of Connected and Unconnected Meshes. In: *Computer Networks and Distributed Systems (CNDS), International Symposium on*. [S.l.: s.n.], 2011. p. 175–180.

SODAGAR, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *MultiMedia, IEEE*, v. 18, n. 4, p. 62–67, 4 2011. ISSN 1070-986X.

SRIVASTAVA, V.; MOTANI, M. Cross-layer design: a survey and the road ahead. *Communications Magazine, IEEE*, v. 43, n. 12, p. 112–119, Dec 2005. ISSN 0163-6804.

TAZAKI, H.; URBANI, F.; TURLETTI, T. DCE Cradle: Simulate Network Protocols with Real Stacks for Better Realism. In: *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013. (SimuTools '13), p. 153–158. ISBN 978-1-4503-2464-9. Último acesso: 9 de junho de 2016. Disponível em: <http://dl.acm.org/citation.cfm?id=2512734.2512755>.

WELTE, H.; AYUSO, P. N. *The netfilter.org project*. 2016. Último acesso: 9 de junho de 2016. Disponível em: <http://www.netfilter.org/>.

WIDMER, J.; HANDLEY, M. TCP-friendly multicast congestion control (TFMCC): Protocol specification. *IETF Online RFC*, v. 4654, 2006. Último acesso: 9 de junho de 2016. Disponível em: <https://tools.ietf.org/html/rfc4654>.

XYLOMENOS, G. et al. A survey of information-centric networking research. *Communications Surveys Tutorials, IEEE*, v. 16, n. 2, p. 1024–1049, Second 2014. ISSN 1553-877X.

ZHANG, X. et al. CoolStreaming/DONet: a Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In: *IEEE INFOCOM*. [S.l.: s.n.], 2005. v. 3, p. 2102–2111. ISBN 0-7803-8968-9.

Apêndices

APÊNDICE A – EXEMPLOS DE APLICAÇÕES GMTP

Neste apêndice, apresentam-se alguns exemplos de códigos fonte de aplicações para utilização do GMTP como protocolo de transporte.

A.1 Constantes para uso do GMTP

Na Listagem A.1, apresenta-se o conteúdo de um arquivo de código fonte escrito na linguagem C, denominado *gmtp.h*, onde definem-se as principais constantes necessárias para se instanciar um *socket* GMTP.

Listagem A.1 – Constantes e definições para uso das aplicações GMTP

```
#ifndef GMTP_H_
#define GMTP_H_

#define SOCK_GMTP      7
#define IPPROTO_GMTP   254
#define SOL_GMTP       281

enum gmtp_sockopt_codes {
    GMTP_SOCKOPT_FLOWNAME = 1,
    GMTP_SOCKOPT_MEDIA_RATE,
    GMTP_SOCKOPT_MAX_TX_RATE,
    GMTP_SOCKOPT_UCC_TX_RATE,
    GMTP_SOCKOPT_GET_CUR_MSS,
    GMTP_SOCKOPT_SERVER_RTT,
    GMTP_SOCKOPT_SERVER_TIMEWAIT,
    GMTP_SOCKOPT_PULL,
    GMTP_SOCKOPT_ROLE_RELAY,
    GMTP_SOCKOPT_RELAY_ENABLED,
    GMTP_SOCKOPT_UCC_TYPE
};

#endif /* GMTP_H_ */
```

No referido arquivo de código fonte, utilizaram-se os valores 7, 254 e 281 para definir, respectivamente, o valor do identificador do tipo de *socket* do GMTP, para definir

o número do protocolo de transporte[1] relativo ao GMTP e, para o identificador utilizado pelas funções de leitura e escrita das opções do *socket* GMTP. Além disso, a enumeração *gmtp_sockopt_codes* serve como identificador das opções de *socket* do GMTP, na chamada das funções *setsockopt* e *getsockopt*.

A.2 Cliente GMTP

Na Listagem A.2, encontra-se disponível o código fonte de um cliente GMTP escrito na linguagem C. No referido código, faz-se uso do arquivo *gmtp.h*, definido no Apêndice A.1, através da Listagem A.1.

Listagem A.2 – Trecho de código de um cliente GMTP escrito na linguagem C.

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include "gmtp.h"

#define SERVER_PORT 2000

int main(int argc, char **argv) {
    char *server_addr = argv[1];
    struct sockaddr_in addr;
    int sockfd = socket(AF_INET, SOCK_GMTP, IPPROTO_GMTP);

    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = inet_addr(server_addr);
    addr.sin_port = htons (SERVER_PORT);
    int ret = connect(sockfd, (struct sockaddr *)&addr,
                      sizeof(addr));

    /* A partir deste ponto, a aplicacao pode receber dados
     * atraves de funcoes como recv() ou recvmsg().
     */

    return 0;
}
```

No exemplo, a porta do servidor é definida como 2000 e o endereço IP do servidor deve ser passado como parâmetro de linha de comando.

[1] A IANA definiu o valor 254 para testes e experimentos. Todos os códigos de protocolos constam no documento *Protocol Numbers*, disponível em <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

A.3 Servidor GMTP

Na Listagem A.3, está disponível o código fonte de um servidor GMTP escrito na linguagem C. No referido código, faz-se uso do arquivo *gmtp.h*, definido no Apêndice A.1, através da Listagem A.1.

Listagem A.3 – Trecho de código de um servidor GMTP escrito na linguagem C.

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include "gmtp.h"

#define SERVER_PORT 2000

int main(int argc, char *argv[])
{
    int sockfd, newSocket;
    struct sockaddr_in addr;
    struct sockaddr_storage addr_s;
    int bitrate = 250000; /* 250000 B/s, ou 2Mbps */

    sockfd = socket(PF_INET, SOCK_GMTP, IPPROTO_GMTP);
    setsockopt(sockfd, SOL_GMTP, GMTP_SOCKOPT_MEDIA_RATE, &media_rate,
       sizeof(bitrate));

    addr.sin_family = AF_INET;
    addr.sin_port = htons(SERVER_PORT);
    addr.sin_addr.s_addr = INADDR_ANY;
    memset(addr.sin_zero, '\0', sizeof(addr.sin_zero));

    bind(sockfd, (struct sockaddr *)&addr, sizeof(addr));

    int ret = listen(sockfd, 5);

    socklen_t addr_size = sizeof(addr_s);
    newSocket = accept(sockfd, (struct sockaddr *)&addr_s, &addr_size);

    /* A partir deste ponto, a aplicacao pode enviar dados
     * atraves de funcoes como send() ou sendmsg().
     */

     return 0;
}
```

No referido código fonte, a porta do servidor é definida como 2000 e utiliza-se o código de opção de *socket* *GMTP_SOCKOPT_MEDIA_RATE* para informar ao GMTP que o *bitrate* da mídia transmitida é equivalente a 2 *Mbps*.

APÊNDICE B – OPÇÕES DE *SOCKET* DO GMTP

Neste apêndice, apresentam-se todas as opções de *socket* disponíveis para utilização no GMTP, através da chamada das funções *getsockopt* e *setsockopt*. Salienta-se que os protótipos das referidas funções são definidos conforme se segue:

Listagem B.1 – Protótipos das funções *getsockopt* e *getsockopt*

```
int getsockopt(int s, int level, int optname, void *optval, socklen_t *
    optlen);
int setsockopt(int s, int level, int optname, const void *optval,
    socklen_t optlen);
```

Nas referidas funções, o parâmetro *s* refere-se ao descritor do *socket*, *level* aponta se as configurações serão realizadas ao protocolo do próprio *socket* ou ao seu protocolo subjacente, *optname* refere-se ao nome da opção de *socket* a ser configurada, *optval* é um ponteiro para o valor da opção que deseja-se configurar/ler, e *optlen* refere-se ao tamanho em *bytes* de *optval*.

Para configurar ou consultar as opções de *socket* do GMTP, deve-se configurar o parâmetro *level* como o valor 281 (*SOL_GMTP*), conforme ilustra-se no Apêndice A.3. As opções de *socket* disponíveis no GMTP, ordenadas pelo valor numérico de seus respectivos *optnames* são:

1. *GMTP_SOCKOPT_FLOWNAME*: Configura ou consulta o nome do fluxo de mídia transmitido/recebido. Seu *optval* é uma *string* de 16 caracteres, correspondente a um código de *hash* MD5 no formato UUID de 128 bits, conforme descrito na Seção 2.2.1.1. Seu *optlen* será sempre igual a 16.

2. *GMTP_SOCKOPT_MEDIA_RATE*: Configura ou consulta o *Bitrate* da mídia transmitida/recebida. Seu *optval* é um valor inteiro não negativo de 32 *bits*.

3. *GMTP_SOCKOPT_MAX_TX_RATE*: Configura ou consulta a taxa de transmissão máxima configurada manualmente no servidor. Seu *optval* é um valor inteiro não negativo de 32 *bits*.

4. *GMTP_SOCKOPT_UCC_TX_RATE*: Consulta a taxa de transmissão máxima recebida via GMTP-UCC. Essa opção está disponível apenas para leitura. Seu *optval* é um valor inteiro não negativo de 32 *bits*.

5. *GMTP_SOCKOPT_GET_CUR_MSS*: Retorna o MSS ou *maximum segment size*. MSS é o tamanho máximo de dados que pode ser enviado, em bytes, em um único pacote. No MSS não são computados os bits pertencentes ao cabeçalho de dados. Essa opção está disponível apenas para leitura. Seu *optval* é um valor inteiro não negativo de 32 *bits*.

6. *GMTP_SOCKOPT_SERVER_RTT*: RTT (round-trip time) medido no servidor, para determinado fluxo de dados. Essa opção está disponível apenas para leitura. Seu *optval* é um valor inteiro não negativo de 32 *bits*.

7. *GMTP_SOCKOPT_SERVER_TIMEWAIT*: *Timewait* no servidor. Atualmente não é utilizado.

8. *GMTP_SOCKOPT_PULL*: Habilita o envio de pacotes do tipo *Data-Pull-Request*. Atualmente não é utilizado.

9. *GMTP_SOCKOPT_ROLE_RELAY*: Muda ou consulta o tipo de *socket*, que pode ser *comum* ou *repassador*. Só pode ser configurado caso o *socket* esteja inativo, ou seja, não esteja no estado *listening*, enviando ou recebendo dados. Seu *optval* é um valor inteiro não negativo de 32 *bits*, sendo igual a 1, caso o *socket* seja configurado como *Relay* (repassador), e 0, caso seja um *socket* comum (cliente, servidor ou relator).

10. *GMTP_SOCKOPT_RELAY_ENABLED*: Habilita ou desabilita o GMTP-Inter em determinado *host*. Só é válido caso o *socket* seja do tipo *Relay* (repassador), ou seja, se a opção *GMTP_SOCKOPT_ROLE_RELAY* foi configurada com o valor 1. Seu *optval* é um valor inteiro não negativo de 32 *bits*, sendo igual a 1 para habilitar o GMTP-Inter e igual a 0 para desabilitar o GMTP-Inter.

11. *GMTP_SOCKOPT_UCC_TYPE*: Configura ou consulta o modo de controle de congestionamento do GMTP-UCC. Atualmente não é utilizado.

APÊNDICE C – DETALHES DOS EXPERIMENTOS

Neste apêndice, apresentam-se detalhes a respeitos dos experimentos realizados no contexto deste trabalho.

C.1 Quantidade de Ensaios

Nos experimentos realizados para analisar o desempenho do GMTP, apresentados na Seção 4, fez-se necessário determinar a quantidade de repetições dos tratamentos para que fosse possível analisar, com 95% de certeza, os valores obtidos para as variáveis dependentes nos experimentos estabelecidos. O valor de n foi calculado através da Inequação C.1, onde ρ^2 e E_0 correspondem à variância da amostra e ao erro amostral máximo tolerado, respectivamente.

$$n \geq \frac{z_\gamma^2 \times \rho^2}{E_0^2} \tag{C.1}$$

Ao considerar o grau de confiança como $0,95$, tem-se que E_0 é igual a 0,05 (obtido através da subtração $1 - 0,95$) e z_γ é igual a $1,96$, obtendo-se a Inequação C.2.

$$n \geq \frac{1.96^2 \times \rho^2}{0.05^2} \tag{C.2}$$

C.2 Intervalo de confiança

Em uma amostra aleatória simples $X_1, \ldots, X_n$ obtida de uma população com distribuição normal, com média μ e variância σ^2 conhecida, a distribuição amostral da média também é Normal com média μ e variância σ^2/n, ou seja, $\overline{X} \sim N\left(\mu, \frac{\sigma^2}{n}\right)$.

Deste modo, obtêm-se a $Z = \frac{\overline{X}-\mu}{\frac{\sigma}{\sqrt{n}}} \sim N(0,1)$, ou seja, a variável Z tem distribuição normal padronizada.

Nesse contexto, considerando um intervalo de confiança de $100(1-\alpha)\%$, obtêm-se a Equação C.3, onde:

$$\mathbb{P}[-Z_{\alpha/2} \leq Z \leq Z_{\alpha/2}] = 1 - \alpha \tag{C.3}$$

$Z_{\alpha/2}$ corresponde ao valor de fronteira da área de $\alpha/2$ na cauda direita da distribuição normal padronizada, obtida pela obtidos na tabela da distribuição normal.

Isso implica na Equação C.4,

$$\mathbb{P}\left[-Z_{\alpha/2} \leq \frac{\overline{X}-\mu}{\frac{\sigma}{\sqrt{n}}} \leq Z_{\alpha/2}\right] = (1-\alpha) \tag{C.4}$$

o que implica na Equação C.5:

$$\mathbb{P}\left[\overline{X} - Z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \overline{X} + Z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right] = 1-\alpha \tag{C.5}$$

Nesse contexto, o intervalo de confiança da média é dado pela Equação C.6:

$$IC(\mu, 1-\alpha) = \left(\overline{X} - Z_{\alpha/2}\frac{\sigma}{\sqrt{n}}; \overline{X} + Z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right) \tag{C.6}$$

Caso os dados não tenham distribuição normal, pode-se aplicar o teorema central do limite e construir um intervalo de confiança aproximado.

C.2.1 Intervalo de confiança para a distribuição de Poisson

Em uma amostra aleatória $X_1, \ldots, X_n$ de uma população com distribuição de Poisson com parâmetro λ, isto é, $X_1, X_2, \ldots, X_n \sim$ Poisson(λ), sabe-se que $\hat{\lambda} = \sum_{i=1}^{n} \frac{X_i}{n}$ é um estimador de máxima verossimilhança para λ. Utilizando o teorema central do limite, obtém-se a Equação C.7,

$$\hat{\lambda} = \sum_{i=1}^{n} \frac{X_i}{n} \sim N\left(\lambda, \frac{\lambda}{n}\right) \tag{C.7}$$

o que implica na Equação C.8, onde:

$$Z = \frac{\hat{\lambda}-\lambda}{\sqrt{\frac{\hat{\lambda}}{n}}} \sim N(0,1) \tag{C.8}$$

Nesse contexto, obtêm-se um intervalo com $100(1-\alpha)\%$ de confiança para a taxa, utilizando-se a Equação C.9:

$$IC(\lambda, 1-\alpha) = \left(\hat{\lambda} - Z_{\alpha/2}\sqrt{\frac{\hat{\lambda}}{n}}; \hat{\lambda} + Z_{\alpha/2}\sqrt{\frac{\hat{\lambda}}{n}}\right) \tag{C.9}$$

Anexos

ANEXO A – ESTRUTURAS QUE REPRESENTAM *SOCKETS* NO LINUX

Neste anexo, apresentam-se detalhes das estruturas que representam *sockets* no núcleo do Linux.

A.1 Interface para o espaço de usuário

Na Listagem A.1, apresenta-se parcialmente a estrutura `struct sock`, que representa os *sockets* no espaço de usuário, para a camada de aplicação do Linux .

Listagem A.1 – Estrutura que representa um *socket* no espaço de usuário do Linux

```
struct socket {
    socket_state            state;

    kmemcheck_bitfield_begin(type);
    short                   type;
    kmemcheck_bitfield_end(type);
    ...

    struct file             *file;
    struct sock             *sk;
    const struct proto_ops  *ops;
};

/* (include/linux/net.h) */
```

A seguir, a descrição dos membros da estrutura:

- *state*: Um *socket* pode encontrar-se em vários estados, como `SS_UNCONNECTED` ou `SS_CONNECTED`, conectado e desconectado, respectivamente.
- *type*: O tipo do *socket*, por exemplo, `SOCK_STREAM` para TCP, `SOCK_DGRAM` para UDP.
- *file*: O arquivo associado ao *socket*.
- *sk*: O objeto `sock` associado ao *socket*. Representa a interface para a camada de rede.

- *ops*: Um apontador para a estrutura `struct proto_ops`. Consiste basicamente de *callbacks* para o *socket*, conforme discutido na Seção 3.1.1.

A.2 Interface para a camada de rede

Na Listagem A.2, apresenta-se parcialmente a estrutura `struct sock`, que representa os *sockets* na camada de rede do Linux.

Listagem A.2 – Estrutura que representa um *socket* na camada de transporte do Linux

```
struct sock {
    ...

    struct sk_buff_head sk_receive_queue;
    ...

    int                 sk_rcvbuf;
    ...

    int                 sk_sndbuf;
    struct sk_buff_head sk_write_queue;
    ...
    unsigned int        sk_shutdown  : 2,
                        sk_no_check_tx : 1,
                        sk_no_check_rx : 1,
                        ...
                        sk_protocol  : 8,
                        sk_type      : 16;
    ...

    void                (*sk_data_ready)(struct sock *sk);
    void                (*sk_write_space)(struct sock *sk);
    ...

};

/* (include/net/sock.h) */
```

A seguir, a descrição dos membros da estrutura:

- *sk_receive_queue*: Uma fila para pacotes recebidos.
- *sk_rcvbuf*: O tamanho do *buffer* de recepção em *bytes*.
- *sk_sndbuf*: O tamanho do *buffer* de envio em *bytes*.

- *sk_write_queue*: Uma fila de pacote para enviar.
- *sk_no_check_tx* e *sk_no_check_rx*: Um marcador para desabilitar o *checksum* na transmissão e na recepção, respectivamente.
- *sk_protocol*: Um número de protocolo, conforme discutido na Seção 3.1.1.
- *sk_type*: O tipo do *socket*, por exemplo, `SOCK_STREAM` para TCP, `SOCK_DGRAM` para UDP, conforme discutido na Seção 3.1.1.
- *sk_data_ready*: Um *callback* para notificar o *socket* sobre a chegada de dados.
- *sk_write_space*: Um *callback* para indicar que existe memória livre disponível para transmissão de dados.