Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Coordenação de Pós-Graduação em Ciência da Computação

Transporte de Datagramas Controlados e Não Confiáveis para Distribuição de Conteúdos Multimídia entre Pares na Internet

Leandro Melo de Sales

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

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Angelo Perkusich e Hyggo Almeida

(Orientadores)

Campina Grande, Paraíba, Brasil

Resumo

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

Abstract

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

Conteúdo

Lista de Símbolos

3WHS - *Three Way Hand Shake*

CCID - *Congestion Control IDentifier*

CPM - *Cooperative Peer Assists and Multicast*

DCCP - *Datagram Congestion Control Protocol*

ECN - *Explicit Congestion Notification*

GMTP - *Global Media Transport Protocol*

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

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

PDTP - *Peer Distributed Transfer Protocol*

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

PPSP - *P2P Streaming Protocol*

RTT - *Round Trip Time*

SCTP - *Stream Control Transmission Protocol*

Swift - *The Generic Multiparty Transport Protocol*

TCP - *Transport Control Protocol*

TTL - *Time-To-Live*

UDP - *User Datagram Protocol*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A transmissão de conteúdos multimídia em tempo real através da Internet tornou-se uma necessidade em aplicações como voz sobre IP (VoIP), videoconferência, jogos e WebTV. Aplicações deste tipo implementam mecanismos sofisticados para melhorar a qualidade dos fluxos de dados e utilizar de forma eficiente os recursos disponíveis da rede. Na prática, tais mecanismos são implementados em forma de protocolos de rede com o intuito de: (i) reduzir altos níveis de congestionamento na rede; (ii) manter a equidade entre diferentes fluxos transmitidos pelos sistemas finais; e (iii) garantir níveis mínimos de qualidade do conteúdo multimídia sendo transmitido.

Aplicações como as supracitadas estão ganhando cada vez mais espaço, principalmente na Internet. Para se ter uma idéia deste crescimento, segundo um relatório publicado pela empresa Cisco [62], em 2014 o tráfego de video será maior do que o tráfego de redes entre pares (P2P) para compartilhamento de arquivos em 2009, correspondendo a 39 % do tráfego de dados total na Internet. No relatório também é mencionado que em 2014 o tráfego de VoIP, vídeo e jogos na Internet alcançará a marca de 40 Exabytes/-mês, quase 50 % do tráfego de dados total na Internet previsto para 2014. Embora já se saiba o potencial do modelo de serviço P2P, esta previsão demonstra o nível de aceitação dos usuários em compartilhar seus arquivos e obter mais dados disponibilizados por outros usuários. Recentemente, a *Paramount Pictures* publicou uma nota [36] informando que passará a transmitir filmes através de grandes redes P2P, tais como a BitTorrent [107; 102]. Em nota, representantes da empresa BitTorrent anunciaram que está trabalhando em uma aplicação P2P para transmissão de conteúdos multimídia em tempo real. Já a Ama-

zon anunciou que também entrará na disputa por uma fatia de mercado para esse serviço [5], disponibilizando um reprodutor de conteúdo multimídia completamente *online*, onde as pessoas poderão comprar músicas e vídeos e em seguida reproduzi-los em tempo real, com os dados armazenados na sua infra-estrutura de computação nas nuvens [13; 49].

Seguindo a tendência das previsões e dos exemplos citados anteriormente, é notório o crescimento de serviços na Internet para distribuição de conteúdos multimídia, sejam serviços onde as empresas disponibilizam os conteúdos totalmente online, como a Netflix [9], famosa por transmitir canais de TV e filmes *online*, sejam os casos em que os próprios usuários disponibilizam tais conteúdos a partir de suas próprias residências. Para este último caso, sites *online*, como o `livestream.tv`, estão se popularizando cada vez mais ao permitirem que os usuários ou empresas transmitam conteúdos multimídia a partir de seus computadores para milhares de usuários conectados à Internet, seguindo o modelo de serviço cliente-servidor com suporte a uma grande rede de distribuição de conteúdos (CDNs) [98].

No contexto desse tipo de aplicação, o que preocupa é o crescimento acentuado do consumo de recursos computacionais de rede resultante principalmente das estratégias e protocolos de rede adotados para distribuir os conteúdos multimídia produzidos por grandes empresas, como Netflix e Google. No final do primeiro semestre de 2011, um artigo publicado no *Financial Post* chamou atenção ao *anunciar momentos de grandes congestionamentos na Internet nos próximos anos* [10]. Destacou-se a empresa Netflix como sendo a maior consumidora de banda de rede da América do norte, respondendo por 24,71 % de todos os dados transferidos durante o horário de pico de uso da Internet por norte-americanos em março de 2011, ultrapassando até mesmo a rede BitTorrent de compartilhamento de arquivos (Figura 1.1(a) [10]). "*A questão é que se não fosse a Netflix, teria sido a Amazon, se não fosse a Amazon teria sido o Google, mas o verdadeiro e iminente problema de congestionamento da rede começará quando todos passarem a fazer isto.*", afirmou Colin Gillis, analista sênior de tecnologia da BGC Partners[1]. Serviços como os da Netflix estão se mostrando extremamente populares entre os consumidores: em menos de 8 meses de funcionamento da Netflix no Canadá a empresa já possuía mais de 800.000 clientes, representando cerca de 10 % das famílias canadenses com conexões de banda larga.

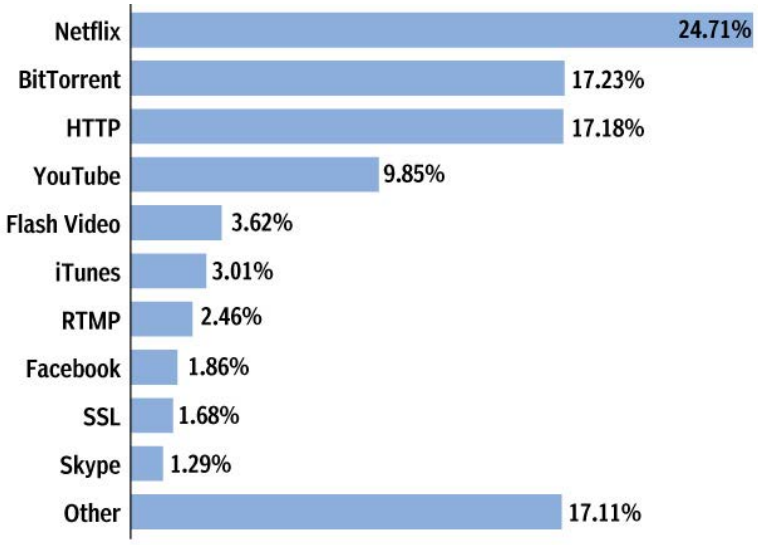
[1]BGC Partners: http://www.bgcpartners.com/

Essa discussão se generaliza em um relatório publicado pela empresa de consultoria Sandvine, onde menciona-se que "*com o rápido sucesso da Netflix, os provedores de Internet em todo o mundo devem se preparar para um futuro em que vídeo sob demanda e em tempo real estará sendo disponibilizado em grandes proporções, podendo ser responsável pela maior fatia do tráfego de dados na Internet*" [18]. De acordo com o que foi publicado nesse relatório, serviços em tempo real de entretenimento, que incluem Netflix e Youtube, foram responsáveis por quase metade (49,2 %) de todo o tráfego de Internet na América do Norte no primeiro trimestre de 2011 (Figura 1.1(b) [10]). Estima-se que esse número alcance 60 % até o final de 2011 e "*o fato é que o volume de tráfego na Internet cresce exponencialmente e o congestionamento da rede vai só piorar. As pessoas sabem o que está acontecendo, mas eu acho que está acontecendo mais rápido do que elas esperavam.*", comentou Tom Donnelly, vice-presidente da Sandvine.

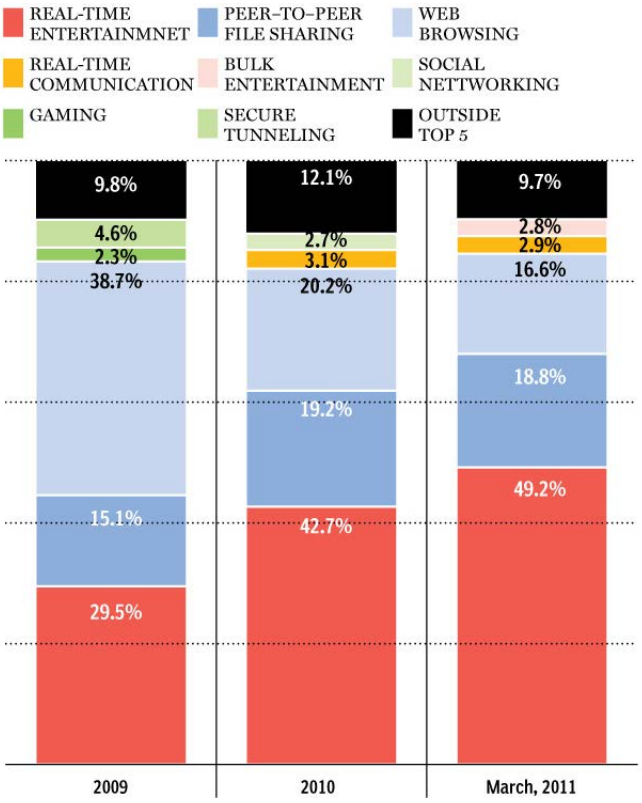
Diante deste cenário, desenvolver protocolos de rede para transportar dados desse tipo de aplicação e dar suporte a estas novas formas de fornecer serviços multimídia através da Internet tem se tornado uma tarefa ainda mais complexa. Exige-se um comprometimento em reduzir o congestionamento nas redes, mantendo a equidade entre diferentes fluxos transmitidos com garantia mínima da qualidade do conteúdo multimídia sendo transmitido, em meio ao aumento da demanda por conteúdos multimídia *online*, mudanças nas estratégias adotadas pelos fornecedores de conteúdos para distribuí-los aos seus clientes e mudança na disponibilidade de recursos de rede, que nem sempre são utilizados de forma eficiente.

Os cenários de aplicação mais críticos do ponto de vista da ineficiência de utilização de recursos de rede e no contexto deste trabalho são aqueles que apresentam um nó de rede gerador de conteúdo e milhares de nós receptores, estabelecendo portanto uma relação $1 \rightarrow n$. Tais aplicações compartilham uma característica comum: a existência de muitos usuários interessados por um mesmo conteúdo e pouco ou nenhum dado individualizado, ou seja, que precisa ser transmitido apenas para um usuário ou um grupo restrito deles. Existem diversos exemplos de aplicações que podem ser citados, principalmente quando se utiliza o modelo de serviço P2P:

- aplicações para transmissão de conteúdo multimídia em tempo real de um nó da rede para um outro, ou para um conjunto de nós. Por exemplo, sites *online* como o `livestream.tv`, `ustream.tv`, `twitcam.com` e `streamtheworld.com` permi-



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



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

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

tem que um usuário transmita conteúdos multimídia do seu computador para milhares de outros usuários conectados à Internet. Soluções de distribuição de conteúdos multimídia em tempo real providas por empresas como a Netflix e Youtube também se enquadram nesta categoria;

- aplicações de telefonia IP, tais como o Skype, principalmente considerando o modo de conversa em grupo;
- TV *Online*, por exemplo, transmissões através da Internet de jogos da copa do mundo ou do campeonato brasileiro de futebol. A rede Globo de televisão e o Youtube já possui aplicações experimentais para transmissão da copa do mundo de 2014, com experiências já realizadas na copa américa 2011, respectivamente;
- jogos e rádios *online* e videoconferência 1-para-muitos, serviços típicos da Internet.

Ao longo dos anos, diversos esforços acadêmicos e da indústria foram feitos para disponibilizar protocolos e sistemas de transmissão para distribuição de conteúdo ao vivo na Internet [102; 86; 2; 75; 70; 99; 64; 48; 47; 111; 110; 80; 19]. As aplicações mais promissoras para este fim são as baseadas no modelo de serviço P2P. Porém, a disseminação de diferentes sistemas e protocolos de rede com este propósito tem levado a uma pulverização de soluções para transporte de dados multimídia na Internet, gerando uma falta de padronização na forma como essas aplicações são projetadas e implementadas, principalmente por serem implementadas na camada de aplicação [67; 104; 77].

Um ponto chave é que as soluções disponibilizadas tentam suprir as limitações existentes dos protocolos da camada de transporte, tratando-se de soluções intermediárias e não fundamentais para uma forma efetiva e padronizada para transportar dados multimídia nas redes de computadores [58; 34], salvas suas devidas contribuições científicas e reais. Desta forma, é premissa no contexto deste trabalho que soluções mais efetivas no uso de recursos da rede e principalmente em sua adoção sejam realizadas na camada de transporte e não na camada de aplicação.

Do ponto de vista da camada de transporte da pilha TCP/IP, existem basicamente dois protocolos tradicionais, o UDP (*User Datagram Protocol*) e o TCP (*Transmission Control*

Protocol), que não foram projetados para transporte de dados multimídia na Internet. Ao utilizar o UDP, largamente adotado na Internet em aplicações multimídia, os desenvolvedores das aplicações devem implementar seus próprios mecanismos para controle de congestionamento de datagramas sem garantia de entrega, o que raramente é feito. Nos casos em que se tenta utilizar TCP em aplicações multimídia em tempo real, o mesmo não apresenta um desempenho satisfatório porque implementa garantia de entrega com retransmissão de dados perdidos, uma forma não adequada para aplicações multimídia com transporte de dados em tempo real [23], resultando em uma má utilização dos canais de transmissão de rede quando se considera aplicações para distribuição de multimídia $1 \rightarrow n$.

Do ponto de vista de pesquisa acadêmica, existem alguns desafios para o desenvolvimento de aplicações de rede nesses cenários, tais como: (i) permitir que fluxos multimídia convivam com fluxos de dados de aplicações elásticas [61] sem que estes últimos sejam degradados pelos primeiros – vasta utilização do protocolo TCP; e (ii) evitar perdas excessivas de dados por parte das aplicações multimídia em questão, pois, neste caso, não faz sentido retransmitir dados quando estes são perdidos, uma vez que o conteúdo transmitido por essas aplicações é transiente.

Para viabilizar a utilização de cenários como os apresentados anteriormente, deve-se abordar problemas relacionados ao gerenciamento de conexão multi-ponto, incluindo suporte a conexões partindo de diferentes redes e entre uma fonte transmissora e múltiplos receptores interessados no fluxo multimídia ($1 \rightarrow n$); controle de congestionamento na rede, incluindo aspectos de compartilhamento equânime do canal entre os diversos fluxos de dados e de forma padronizada; adaptação de fluxo multimídia e qualidade de serviço; segurança; dentre outros.

Nesse contexto, para contemplar requisitos como os mencionados anteriormente, considera-se que o modelo de serviço de rede mais adequado para abordar soluções de distribuição de conteúdos multimídia multi-ponto seja o modelo de serviço entre pares (P2P). Esta escolha é de fundamental importância neste trabalho porque as tecnologias baseadas em P2P aliviam a carga imposta aos servidores e aos canais de transmissão das redes. Na arquitetura P2P, cada participante do sistema pode obter o serviço de visualização da mídia e também contribuir com outros participantes, fornecendo parte do conteúdo da mídia. Assim, a banda de rede necessária em um único ponto no modelo cliente-servidor é compartilhada

entre os diversos participantes do sistema [100].

É no contexto de utilização de conceitos de P2P com o desenvolvimento de protocolos de redes de computadores para viabilizar transmissões de conteúdos multimídia da classe de aplicações de rede $1 \rightarrow n$ que se insere esse trabalho, motivado pelo grande interesse de pesquisa, indústria e mercado em evoluir o estado da arte das soluções para transporte de conteúdos multimídia em larga escala com suporte a controle de congestionamento, especialmente na Internet. A fim de completar as discussões iniciais nesse contexto, a seguir, discutem-se as opções existentes de protocolos de transporte de dados mais adequados para este fim, uma escolha considerada fundamental neste trabalho.

Panorama Atual: Protocolos de Transporte da Internet

Atualmente, as principais propostas de protocolos de transporte de dados para Internet são o TCP, o UDP e o recém padronizado DCCP (*Datagram Congestion Control Protocol*) [57; 59]. Quando se projeta um protocolo de rede para transporte de dados nos cenários discutidos anteriormente, deve-se levar em consideração as características da aplicação para que se possa obter resultados significativos na qualidade do fluxo de dados multimídia sendo transmitido e na eficiente utilização dos recursos da rede. Porém, nota-se que os protocolos da camada de transporte existentes são limitados com vista aos cenários de distribuição de conteúdo $1 \rightarrow n$ e envolvendo suporte a controle de congestionamento.

O protocolo UDP tem sido largamente utilizado em aplicações multimídia em tempo real por ser um protocolo simplificado, fazendo uso apenas do serviço de melhor esforço do IP para transmitir dados na Internet. Com o passar dos anos e antes do DCCP, o UDP se tornou a primeira e única opção para transmissão de dados multimídia em tempo real, porém gerando diversos efeitos colaterais nas grandes redes, os quais são discutidos a seguir e com vastas referências na literatura [34; 25; 24; 23; 45; 59].

Para se ter uma idéia dos efeitos colaterais gerados na rede com o uso do UDP, observe o gráfico *vazão* $\times$ *tempo* ilustrado na Figura 1.2. Este gráfico corresponde a um ensaio realizado com a transmissão de 1 fluxo TCP competindo com 3 fluxos de áudio UDP em uma rede *Ethernet* de $100\,Mbps$. Observe que o UDP sempre ocupa o máximo da largura de banda disponível na rede ao passo que não oferece chances para outros fluxos utilizarem o canal, como é o caso do TCP. Por este motivo, o UDP sempre se apresenta com altas taxas

de perda de pacotes, sobretudo quando há congestionamento na rede. No caso deste ensaio, nos primeiros 50 *s*, quando não disputava com nenhum outro fluxo, o fluxo TCP utilizou a rede de forma satisfatória, alcançando uma vazão em torno de 20 *Mbps*. Entretanto, após esta fase, quando os três fluxos UDP foram transmitidos na rede, a vazão do fluxo TCP reduziu praticamente para 0 (zero), permanecendo assim até o final do ensaio.

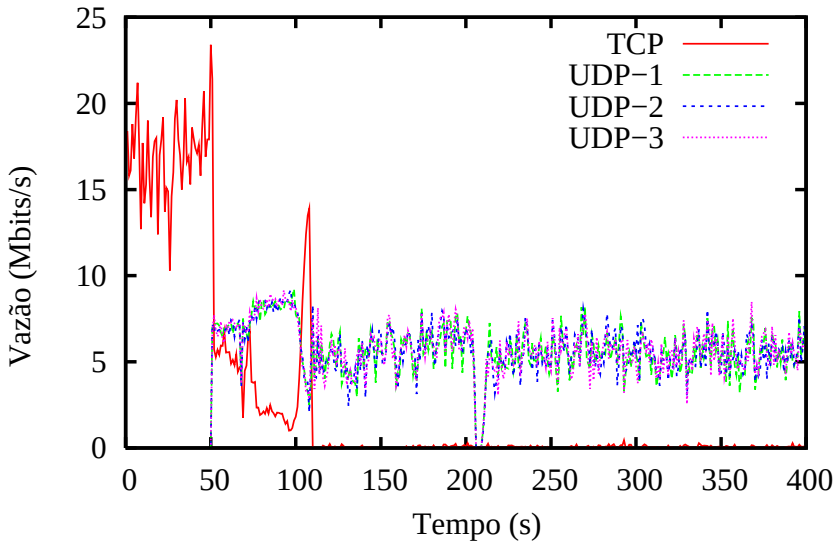


Figura 1.2: TCP × UDP. Após 50 *s* do início do experimento, o fluxo UDP ocupa toda a largura de banda disponível na rede. Fonte: [23].

O protocolo TCP, por sua vez, atende de forma satisfatória às aplicações que toleram atrasos na entrega de dados e que exigem que estes sejam todos entregues corretamente e em ordem (aplicações elásticas). Porém, em se tratando de transporte de dados multimídia em tempo real, o TCP se torna o protocolo menos apropriado para este fim, pelo menos comparando-o com o UDP e o DCCP. Nas aplicações de fluxo multimídia em tempo real, é preferível manter o fluxo de dados e reproduzir o conteúdo que chega a esperar que a informação perdida seja retransmitida, mesmo diante do fato de que parte dos dados da aplicação tenha sido perdida. Ao utilizar o TCP, isto não é possível. O principal motivo é que o TCP implementa entrega confiável de dados adotando a abordagem de retransmitir qualquer dado perdido. Esta estratégia resulta em atrasos indesejáveis quando se trata de transmissão de dados multimídia em tempo real, fazendo com que o usuário perceba interrupções na reprodução do conteúdo.

Em condições de congestionamento na rede, o atraso fim-a-fim aumenta e conseqüen-

temente degrada a qualidade do conteúdo multimídia sendo transmitido. Esta situação se agrava com a retransmissão de pacotes perdidos e que podem não fazer mais sentido para a aplicação receptora devido ao comportamento transiente dos fluxos multimídia, em particular os transmitidos em tempo real. Neste caso, se os pacotes de dados retransmitidos não alcançarem o receptor até um determinado instante, estes serão descartados ao preço do desperdício no uso dos recursos da rede, pois *buffers* dos roteadores são alocados para processar e repassar pacotes que terminam sendo inúteis às aplicações.

O comportamento do TCP para situações como a mencionada anteriormente pode ser observado no gráfico ilustrado na Figura 1.3. No ensaio realizado, transmitiu-se um áudio com duração de 100 s, sendo armazenado no destino e em seguida comparado com o original. Neste caso, constatou-se que apenas 32 % do áudio alcançou o destino, fato ocorrido devido ao excesso de retransmissões de pacotes que foram perdidos na rede quando considerados fluxos TCP disputando com fluxos UDP e a rede apresentando altos níveis de congestionamento.

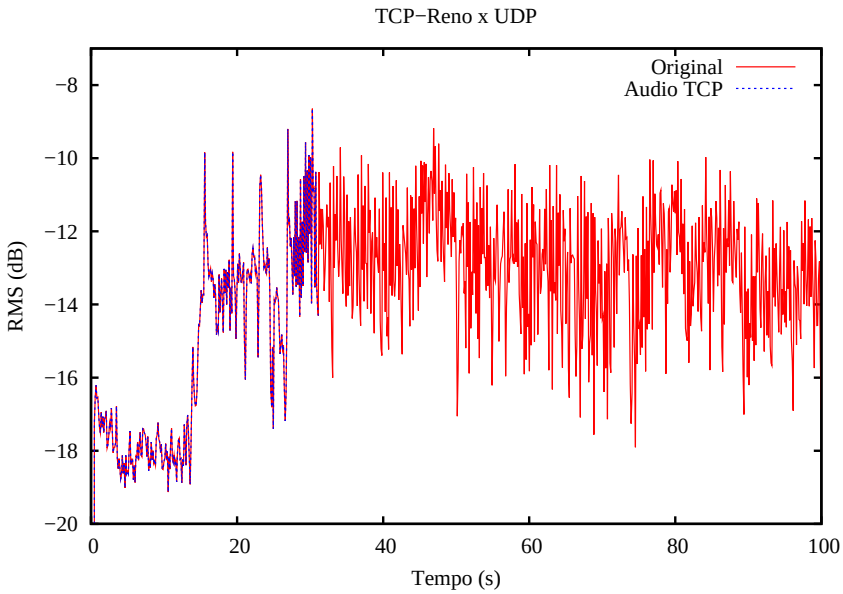


Figura 1.3: TCP Reno × UDP, sendo o TCP enviando um arquivo de áudio. Fonte: [23].

Com apenas essas duas opções para transporte de dados na Internet e objetivando promover melhorias nos serviços oferecidos pelas aplicações multimídia, a IETF (*Internet En-*

gineering Task Force) aprovou a especificação do protocolo DCCP para transporte de dados multimídia para Internet. É um protocolo orientado à conexão, não garante entrega e nem ordenação dos dados transmitidos, todavia, o DCCP implementa controle de congestionamento para transmissão não-confiável de fluxo de dados [25; 57].

O DCCP herda do TCP as características de ser orientado à conexão e fornecer controle de congestionamento. Do UDP, o DCCP herda as características de não garantir entrega e nem ordenação dos dados transmitidos. Além destas características, no DCCP foram adicionados dois conceitos novos: a escolha tardia de dados [54] e um arcabouço para gerenciamento dos algoritmos de controle de congestionamento de forma modular. A escolha tardia de dados permite a mudança de dados de um pacote mesmo depois que estes dados já tenham sido enviados para a camada de transporte, mas ainda não tenham sido enviados através da rede – isto é um alternativa ao mecanismo de retransmissão do TCP. Já o arcabouço de gerenciamento de algoritmos de controle de congestionamento permite adicionar novos algoritmos de controle de congestionamento à aplicação e substituí-los mesmo que uma conexão DCCP já tenha sido estabelecida.

Para entender as melhorias providas pelo protocolo DCCP, considere o gráfico *vazão* $\times$ *tempo* apresentado na Figura 1.4 [23]. Neste gráfico, ilustram-se os comportamentos dos protocolos TCP e DCCP quando utilizados para transmissão de um arquivo e de um conteúdo multimídia, respectivamente. A partir do gráfico, é possível constatar que os protocolos TCP e DCCP compartilham entre si a largura de banda disponível, onde cada fluxo consegue transmitir dados na rede. Note que o comportamento do protocolo TCP para os $50\,s$ iniciais foi similar ao confronto TCP $\times$ UDP (Figura 1.2). Porém, diferentemente do que ocorreu naquele caso, após os primeiros $50\,s$ dos confrontos TCP $\times$ DCCP, a vazão do protocolo TCP continuou sendo satisfatória, assim como a do DCCP.

Para transmissões de dados multimídia em redes de computadores, onde satisfazer os requisitos de tempo pode definir o nível de qualidade da transmissão multimídia, o DCCP pode melhorar a qualidade do fluxo multimídia e ainda resolver diversos problemas de congestionamento da rede, como os causados por retransmissões desnecessárias de pacotes feitas pelo protocolo TCP ou por problemas de excessiva perda de pacotes e altos níveis de congestionamento da rede quando se utiliza o protocolo UDP. Estudos anteriores realizados no contexto deste trabalho [28; 23; 25; 89; 24; 27] e outros publicados por terceiros [105; 12; 7; 78; 65;

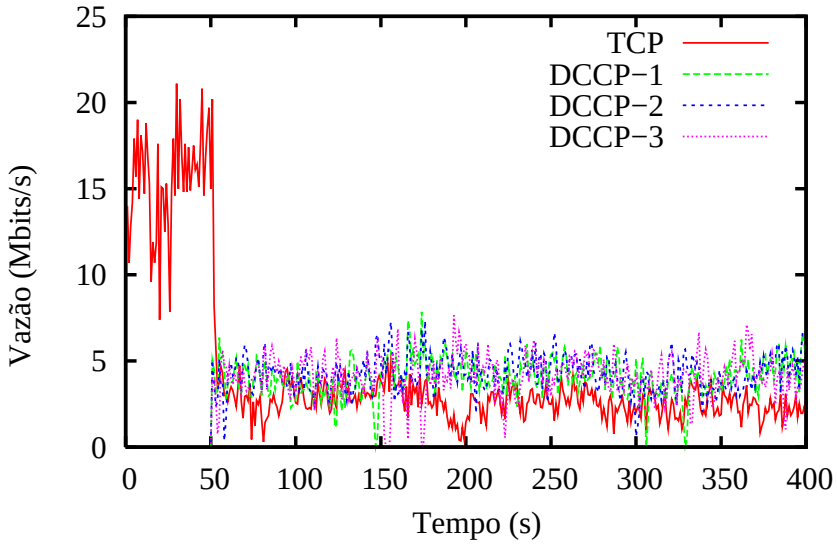


Figura 1.4: TCP × DCCP. Ambos os protocolos conseguem transmitir dados na rede. Fonte: [23].

95; 79] constatam que a utilização do protocolo DCCP tem trazido diversas vantagens na transmissão de fluxos multimídia, apresentando-se como uma opção efetiva a ser adotada para transmissão de dados multimídia, principalmente por ter um comportamento similar ao protocolo TCP e padronizado pela IETF.

Sendo assim, a motivação para a definição do DCCP está relacionada com as características intrínsecas das aplicações com restrição de tempo de resposta e no fato de que grande parte desse tipo de aplicação utiliza o UDP. Considerando os problemas e limitações dos protocolos TCP e UDP discutidos até aqui e os cenários de aplicações considerados neste trabalho, o DCCP foi projetado para atender as necessidades das aplicações multimídia com suporte a controle de congestionamento, evitando assim um colapso de congestionamento na Internet.

Pelo exposto, o protocolo DCCP aparece como o protocolo de transporte mais adequado a ser utilizado em uma solução para transmissão padronizada de conteúdos multimídia na Internet. Com isto, elimina-se o uso do protocolo UDP em aplicações multimídia e consequentemente diminui-se o tráfego de dados na rede sem controle de congestionamento, ao passo que compartilha-se efetivamente os canais de transmissão com fluxos de dados TCP, cujas aplicações (*Web*, E-mail etc.) correm o risco de tornarem-se inutilizáveis devido ao iminente e inevitável crescimento do uso de aplicações multimídia baseadas em UDP.

1.1 Descrição do Problema

Apesar da eficiência do DCCP em alguns cenários de transmissão de dados multimídia na Internet, o mesmo possui falhas críticas quando utilizado em larga escala em cenários $1 \rightarrow n$. Além disso, ao longo dos anos o uso de outros protocolos de transporte para a Internet tem se mostrado pouco efetivo na prática para a distribuição de conteúdos multimídia em tempo real, tal como o uso do UDP ou diversas outras propostas não padronizadas, acentuando a motivação em se desenvolver pesquisas nesse contexto. Isto tem ocorrido porque as soluções existentes sempre se apresentam de forma independente e sem qualquer preocupação com sua facilidade de implantação em larga escala.

O que ocorre é que o DCCP é um protocolo orientado à conexão e portanto para cada novo usuário interessado em receber um fluxo multimídia transmitido, uma nova conexão se faz necessária. As conseqüências desta limitação do DCCP são desastrosas, tornando-o um protocolo paradoxal para o que foi proposto. Em outras palavras, propõe-se com o uso do DCCP resolver os problemas de congestionamento de rede gerados pelo protocolo UDP em cenários de aplicações multimídia, porém, quando utilizado em cenários $1 \rightarrow n$, o protocolo não funciona devido aos seguintes motivos:

1. *excessivo consumo de recurso computacional*: para cada nova conexão, o nó transmissor deve alocar recursos computacionais (memória e processamento) para tratar cada nova conexão. Em cenários de transmissão multimídia $1 \rightarrow n$, se muitos nós estão conectados em um único servidor, então isto elevará sobremaneira o consumo de recurso computacional do nó transmissor proporcionalmente à quantidade de nós receptores interessados pelo fluxo multimídia transmitido. Além disso, embora o conteúdo transmitido por um nó seja de interesse de muitos outros nós, os fluxos são enviados independentemente uns dos outros, o que gera duplicações desnecessárias e conseqüentemente desperdício de recursos de rede. O trabalho mais notável que discute as consequências deste problema em aplicações de rede é o encontrado na referência [38].

2. *a taxa de transmissão de fluxos DCCP individualmente tenderá a 0 (zero)*: o protocolo DCCP realiza controle de congestionamento utilizando uma equação matemática para definir a taxa de transmissão de uma conexão. À medida que mais nós se conectam a

um nó transmissor, menor será a taxa de transmissão do nó transmissor para cada um dos nós receptores conectados a ele. Para a rede, esta estratégia é equânime e evita que a mesma entre em colapso de congestionamento, mas para cada fluxo de dados isto é ruim. Este problema tem uma relação estreita com o dilema observado por Garrett Hardin em 1968 e denominado de *Tragédia dos Comuns* [41], apresentando-se em diferentes áreas do conhecimento. No caso de protocolos como o DCCP, a tragédia dos comuns ocorre porque à medida que novos fluxos são transmitidos na rede, menor será a taxa de transmissão individual de cada fluxo, a qual pode se tornar insuficiente para a recepção de um fluxo multimídia e, por consequência, nenhum nó receptor reproduzirá o fluxo transmitido pelo nó transmissor, embora todos os fluxos terão possibilidades semelhantes sobre o uso do canal.

Sendo assim, apesar dos algoritmos de controle de congestionamento serem corretos visando o caso do melhor global (equidade para com todos os fluxos e assim evitar congestionamento da rede), isto provoca o efeito do caso do pior local (redução da taxa de recepção de cada nó da rede).

Este fato pode ser explicado analiticamente utilizando como base a Equação 1.1, que define cada taxa de transmissão X_i calculada pelo DCCP[2] durante a transmissão de dados para realizar o controle de congestionamento em cada conexão. Nesta equação, X_i é a taxa de transmissão em bytes/segundo, *s* é o tamanho do pacote em bytes, *R* é o RTT (*Round Trip Time*) em segundos, *p* é a taxa de ocorrência de perdas, entre 0 e 1, RTO (*Retransmission TimeOut*) é o valor do temporizador de retransmissão do TCP em segundos e *b* é igual a 1 e representa o número máximo de pacotes confirmados por um único ACK.

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

[2]Essa equação é utilizada no DCCP pelo algoritmo para controle de congestionamento chamado de CCID-3 (*Congestion Control IDentifier 3*). Para efeito de estudo, esta equação foi a escolhida, porém qualquer equação adotada em outros algoritmos para controle de congestionamento poderia ter sido utilizada.

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

Embora esta seja uma discussão teórica sobre o principal problema tratado neste trabalho, também foram realizados e publicados experimentos em busca de evidências mais contundentes de que este fato ocorre na prática [28; 26]. Foram executadas simulações de rede no NS-2[3] [22] cuja topologia da rede foi definida como uma árvore binária completa (Figura 1.5). Cada nó da árvore representou um roteador e cada roteador tinha 10 nós DCCP receptores conectados a ele. Cada tratamento foi definido como sendo um nível da árvore binária. Por exemplo, o primeiro tratamento tinha 10 nós receptores e 1 roteador, pois o nível da árvore *L* foi igual a 0 (zero); no tratamento seguinte utilizou-se 30 nós receptores e 3 roteadores, pois L=1; no tratamento seguinte utilizou-se 70 nós receptores e 7 roteadores, pois L=2; e assim por diante até L=9, quando utilizou-se 10.230 nós receptores e 1.023 roteadores (utiliza-se $n = 2^{L+1} - 1$ para se obter a quantidade *n* de roteadores dado um nível *L* da topologia de rede utilizada). A transmissão ocorreu da seguinte forma: um nó localizado na raiz da árvore transmitiu o mesmo conteúdo multimídia para todos os outros nós conectados à rede, simulando uma típica transmissão multimídia $1 \rightarrow n$ e um tráfego de comportamento equivalente a um vídeo MPEG-2. Em cada tratamento foram estudadas duas variáveis: a perda de dados e a taxa de transmissão de cada conexão DCCP partindo do nó raiz até cada nó receptor. As simulações de cada tratamento foram repetidas a quantidade de vezes necessária até se alcançar uma média com nível de confiança de 95% para a métrica vazão, de acordo com o método estatístico descrito no Capítulo 5 deste documento.

Os resultados obtidos com as simulações dos tratamentos descritos anteriormente são apresentados no gráfico da Figura 1.6. Nas abscissas do gráfico representa-se o número de nós receptores para cada tratamento, ao passo que nas ordenadas representa-se a taxa de transmissão média conseguida por cada conexão DCCP, com a porcentagem de perda de dados em cada conexão DCCP em cada ponto marcado no gráfico.

É possível observar no gráfico apresentado na Figura 1.6 que a vazão média de cada fluxo DCCP transmitido aos receptores tende a 0 (zero) à medida que o número de receptores aumenta, sendo possível concluir que o protocolo DCCP não escala quando utilizado

[3]Network Simulator 2: `http://nsnam.isi.edu/nsnam/index.php/Main_Page`

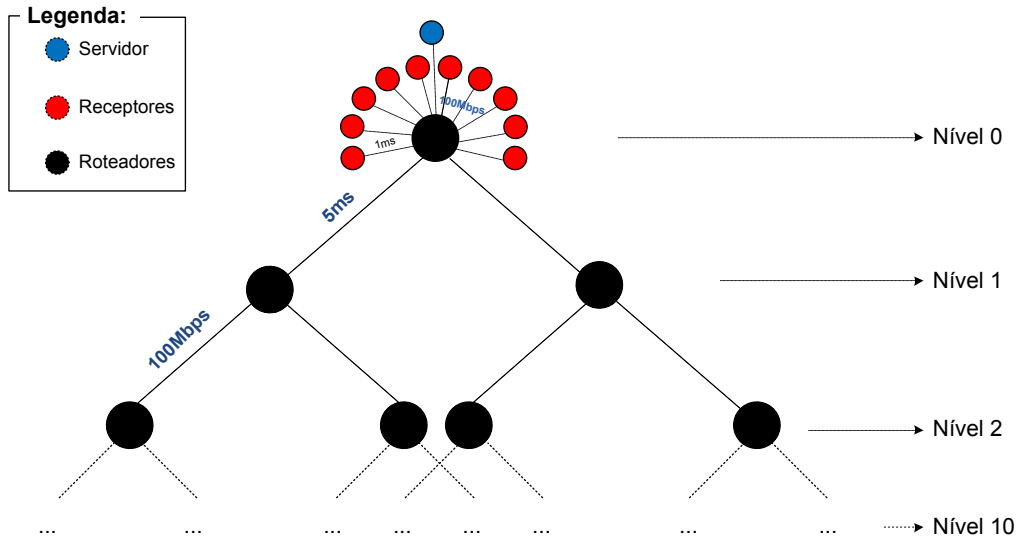


Figura 1.5: Topologia da rede definida para as simulações realizadas. Cada rede é representada por um roteador e com 10 nós em cada rede.

para transmissão de dados multimídia em cenários de aplicações com um transmissor transmitindo para vários receptores ($1 \rightarrow n$).

Uma questão intrigante neste aspecto é que o protocolo DCCP funciona perfeitamente em cenários simplórios, mas sofre claramente de um problema de escalabilidade, o que é crítico para aplicações consideradas neste trabalho, as quais não podem continuar utilizando protocolos como o UDP pelos efeitos colaterais causados por este. Isto torna o protocolo DCCP pouco eficaz para cenários de distribuição de conteúdo multimídia, fazendo com que os desenvolvedores continuem sem motivações para efetivamente utilizar o protocolo DCCP em suas aplicações.

Note que apenas fluxos DCCP foram transmitidos neste tratamento, os quais já foram necessários para causar o problema em questão. Em situações mais realistas, o problema se torna ainda mais grave, pois o protocolo DCCP disputará o canal não apenas com outros fluxos DCCP, mas também com fluxos de protocolos tradicionais, como o TCP e UDP, além de outros protocolos modernos, como o SCTP [50; 46].

Apesar do protocolo DCCP ter sido utilizado para evidenciar o problema trazido à tona neste trabalho e assim apresentar resultados concretos, esta discussão pode ser generalizada em direção a qualquer outro protocolo de rede que seja orientado à conexão e que suporte

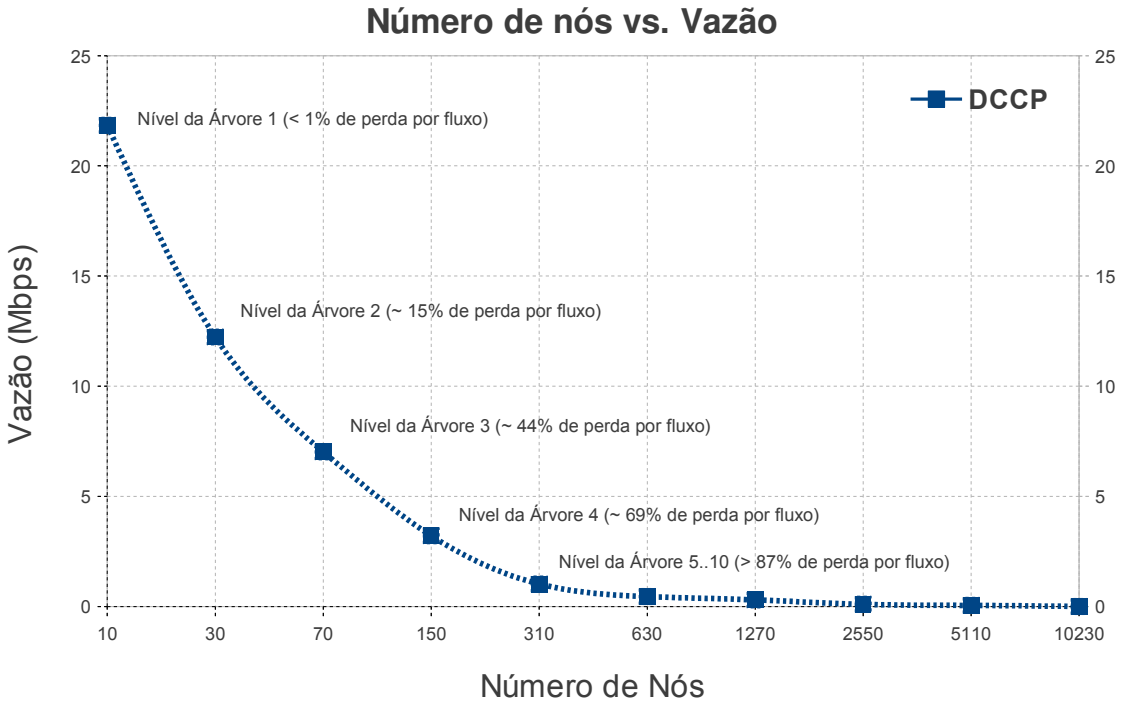


Figura 1.6: Gráfico para uma transmissão DCCP com um transmissor enviando dados de áudio VoIP utilizando o protocolo DCCP. Nota-se que a vazão média de cada fluxo tende a 0 (zero) à medida que o número de nós receptores aumenta.

mecanismos para controle de congestionamento de fluxos não confiáveis de dados. A fim de aumentar a representatividade de protocolos nesse contexto, o leitor pode consultar mais trabalhos nos documentos referenciados em [20; 17; 72; 82; 30; 94; 73; 68; 16; 8] e no Capítulo 7 deste documento.

Assim, até o presente momento, os desenvolvedores de aplicações multimídia não têm outra opção a não ser continuar utilizando o protocolo UDP, porém ao custo do que já foi discutido anteriormente, ou seja, gerando congestionamentos na rede e degradando outros fluxos de dados controlados, como os transmitidos pelo TCP, além de causar um impacto no desempenho das aplicações orientadas a dados.

Diante do exposto e buscando melhorar a qualidade dos fluxos de dados multimídia transmitidos nas redes de computadores, em particular em larga escala na Internet, pretende-se responder à seguinte questão de pesquisa: *como transportar dados multimídia para distribuição de conteúdos em topologias de rede 1 → n com suporte a controle de congestionamento*

evitando o fenômeno da tragédia dos comuns e de forma padronizada?

Observando esta questão que deu origem às pesquisas desenvolvidas neste trabalho, propõe-se a seguinte tese de doutorado:

O controle de congestionamento baseado na combinação do modo de transmissão multicast com o compartilhamento de múltiplos fluxos unicasts entre pares (P2P) realizado na camada de transporte da pilha TCP/IP possibilita a distribuição padronizada e eficiente de conteúdos multimídia na Internet.

1.2 Objetivos da Tese

1.2.1 Objetivo Principal

Neste trabalho, tem-se como objetivo principal a concepção e o desenvolvimento de um protocolo para transporte de fluxos de dados controlados e não confiáveis para a distribuição de conteúdos multimídia na Internet, considerando-se os cenários de aplicações $1 \rightarrow n$. Mais especificamente, propõe-se a especificação e o desenvolvimento de um protocolo para a classe de aplicações apresentadas, validando-o através de experimentações e simulações com um conjunto de ferramentas para dar suporte ao desenvolvimento de aplicações nesse contexto, assim como promovendo ferramentas para futuras investigações em pesquisa sobre o protocolo estudado, considerando os seguintes requisitos de sucesso ao fim deste trabalho.

- *Transparência para o desenvolvedor* – deve-se prover um protocolo de rede que permita ao desenvolvedor, apenas através da utilização deste protocolo, desenvolver aplicações para envio de dados multimídia em cenários $1 \rightarrow n$ com suporte a controle de congestionamento de fluxo de dados não confiável. Em momento algum, o desenvolvedor deverá implementar mecanismos para controle de congestionamento no desenvolvimento de sua aplicação, bem como outros mecanismos comuns nesse tipo de sistema. Os mecanismos para controle de congestionamento, assim como os de distribuição de conteúdo, devem ser transparentes para o desenvolvedor.

- *Independência de linguagem e plataforma* – o protocolo proposto deve ser independente de linguagem de programação ou plataforma específica para a sua execução.

Sendo assim, espera-se que seja possível utilizar tal protocolo em diferentes contextos de aplicações, considerando diferentes linguagens de programação, incluindo linguagens populares como C, C++, Java e Python, bem como sua implementação em diferentes sistemas operacionais.

- *Comprometimento com práticas já utilizadas em aplicações multimídia* – é imprescindível que o protocolo proposto seja capaz de transmitir dados em modo *multicast* e que suporte algoritmos para controle de congestionamento neste modo, considerando a restrição de que este serviço seja implementado na camada de transporte, e não na camada de aplicação. Esta restrição se estende a funcionalidades como estabelecimento de conexão, compartilhamento de conteúdos, descoberta e seleção de nós parceiros e tolerância a falhas.

- *Monitoramento e notificação de requisitos mínimos para transmissão* – o protocolo deve permitir que as aplicações definam requisitos mínimos de funcionamento para o envio e o recebimento de dados, notificando-as caso os limiares definidos não sejam cumpridos. Parâmetros como taxa mínima para transmissão e recepção, além do atraso máximo, figuram como exemplos desses requisitos. Isto permitirá que as aplicações multimídia construam soluções para adaptação e transcodificação em tempo real de fluxo de dados multimídia baseando-se em métricas obtidas de forma padronizada e sem a necessidade de se implementar na camada de aplicação.

- *Especificação do protocolo para uso em larga escala* – o protocolo deve ser descrito e publicado sob domínio público seguindo o formato determinado pela IETF (*Internet Engineering Task Force*) e, desta forma, facilitar a utilização do protocolo em escala global. Isto inclui a especificação formal do protocolo e sua efetiva validação.

1.2.2 Objetivos Específicos

Considerando os requisitos descritos anteriormente, pode-se dividir o objetivo principal deste trabalho nos objetivos específicos descritos a seguir:

1. compreender os sistemas e protocolos para distribuição de conteúdos multimídia com suporte a controle de congestionamento e multicast, em especial soluções que forne-

cem tais funcionalidades na camada de aplicação. Além disso, entender quais outras funções estão presentes nesse tipo de sistema.

2. definir um protocolo na camada de transporte da pilha TCP/IP para distribuição de conteúdos multimídia capaz de transmitir dados através do modo multicast e do modo multi-unicast entre pares, quando o modo multicast não estiver disponível. Neste contexto, deve-se definir também um algoritmo para controle de congestionamento em transmissões multicast a ser acoplado ao protocolo proposto;

3. definir um modelo que permita formalizar as características do protocolo e assim avaliar os aspectos comportamentais dos nós quando o protocolo estiver em uso, bem como os aspectos da rede, como o consumo de recursos computacionais;

4. implementar o protocolo e o algoritmo para controle de congestionamento em um simulador, bem como em um sistema operacional, levando-se em consideração seus aspectos de extensão de novos algoritmos para as funcionalidades suportadas pelo protocolo. Isto permitirá validar o seu uso em cenários reais;

5. avaliar a relevância do uso do modo multicast e o compartilhamento de fluxos unicast entre pares com relação à melhoria na qualidade do conteúdo multimídia recebido pelos nós, assim como no consumo eficiente do canal de transmissão da rede;

6. avaliar a capacidade do protocolo em lidar com situações de falhas (desconexão, por exemplo) de um nó responsável pelo repassar de conteúdos multimídia. Além disso, avaliar o impacto dessas falhas sob outros nós na rede, como por exemplo, qual é o impacto sob a qualidade do conteúdo multimídia recebido por um nó quando há uma falha em um nó de repasse, ou ainda, qual é o máximo atraso aceitável sem que o usuário perceba interrupções na reprodução do conteúdo multimídia;

7. propor e redigir uma RFC (*Request For Comments*) e submetê-la para aprovação junto à IETF. Deve-se descrever tecnicamente e em detalhes o protocolo e os algoritmo envolvidos para a sua devida implementação. Neste ponto, deve-se também estudar e descrever o grau de compatibilidade do protocolo proposto com outros protocolos e recomendações anteriormente aprovadas pela IETF no contexto de transporte de dados multimídia.

1.3 Relevância do Tema e da Tese

Transporte de fluxos de dados multimídia com suporte a controle de congestionamento para distribuição de conteúdos multimídia, em particular, na Internet, é um tema relevante no contexto de redes de computadores porque ao utilizar de forma eficiente e equânime os canais de transmissão, obtêm-se melhoras da qualidade do conteúdo multimídia percebido pelo usuário. Além disso, permite-se que aplicações elásticas, como as executadas na *Web*, funcionem de forma satisfatória. Esta foi a motivação para a concepção do protocolo de rede proposto no contexto deste trabalho, assim como o algoritmo para controle de congestionamento a ser utilizado em transmissões de conteúdos multimídia multicast, principalmente com vista a sua ampla utilização na Internet.

No caso do tema específico tratado neste trabalho, a distribuição de conteúdos multimídia em larga escala é cada vez mais relevante devido às características inerentes à classe de aplicações e cenários que têm sido disponibilizados e adotados na Internet, tais como controle de congestionamento, sensibilidade a atrasos e transporte de dados de forma não confiável. Estas características, aliadas à padronização e à transparência da solução no ponto de vista do desenvolvedor da aplicação, tornam o tema ainda mais relevante para o contexto de boas práticas para a distribuição de conteúdos multimídia, especialmente em cenários $1 \rightarrow n$.

Como a demanda por serviços multimídia em redes de computadores tem aumentado dia após dia, o estudo sendo desenvolvido e os artefatos de software produzidos no contexto desta proposta de tese podem contribuir para o desenvolvimento de aplicações multimídia mais eficientes e de forma padronizada, além de facilitar as tomadas de decisões sobre futuros desenvolvimentos desse tipo de aplicação. Isto é possível porque neste trabalho são apresentados um protocolo para distribuição de conteúdo multimídia com suporte a controle de congestionamento na camada de transporte da pilha TCP/IP; resultados e discussões sobre este protocolo e também sobre o desempenho acerca dos três principais protocolos de transporte disponíveis para uso na Internet, com destaque para o protocolo DCCP. Os resultados obtidos nesta última parte foram um dos primeiros a serem publicados na literatura. Além disso, o autor destaca a importância do presente trabalho por ser o primeiro a trazer à tona um problema e uma proposta de solução do uso de um protocolo de transporte exclusivamente projetado para a distribuição de conteúdos multimídia para a Internet, antes realizada

apenas com o protocolo UDP ou com protocolos independentes implementados na camada de aplicação.

No que diz respeito à relevância do trabalho, em se tratando de uma proposta de tese em engenharia de software para redes de computadores, o autor considera indispensáveis três principais requisitos que estão sendo contemplados e que reforçam a relevância da tese. Estes requisitos servem como motivação para a realização das atividades desenvolvidas até o presente momento.

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

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

O terceiro requisito é o potencial prático. A implementação do protocolo de rede, assim como o conjunto de ferramentas que tem sido desenvolvida no contexto deste trabalho, tem como objetivo demonstrar que a abordagem é viável e praticável. Um protocolo de rede simplesmente especificado sem nenhuma implementação real tornaria as reais contribuições deste trabalho apenas suposições. O compromisso com a utilização dos conceitos para construir mecanismos que possam ser aplicados na indústria tornam o trabalho relevante em termos práticos, sobretudo em escala global na Internet.

1.4 Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

- No Capítulo 2 são apresentados os principais conceitos relacionados aos sistemas de distribução de mídias ao vivo em arquiteturas P2P;
- No Capítulo 3 é apresentada uma visão geral do *Global Media Transmission Protocol* (GMTP), um protocolo de transporte baseado em uma arquitetura P2P para distribuição de fluxos de dados multimídia de aplicações com um nó transmissor e muitos nós receptores ($1 \rightarrow n$), desenvolvido para operar principalmente na Internet.
- No Capítulo 4 continua-se com as discussões sobre o protocolo GMTP, porém fornecendo-se detalhes acerca do funcionamento do protocolo no tocante a três principais aspectos, o cabeçalho de pacotes, o processo de conexão e o mecanismo para controle de congestionamento.
- No Capítulo 5 são apresentados os métodos e experimentos adotados neste trabalho, destacando-se o mecanismo estatístico utilizado para obtenção de métricas específicas a fim de analizar o desempenho do protocolo GMTP em transmissões de fluxos de dados multimídia.
- No Capítulo 6 são apresentados os resultados preliminares obtidos no contexto deste trabalho, destacando-se os resultados obtidos em simulações de rede com o uso do protocolo GMTP, apresentado no Capítulo 3.
- No Capítulo 7 são apresentados os trabalhos relacionados a esta pesquisa, destacando-se os principais avanços científicos no que diz respeito a protocolos de transporte com ênfase na distribuição de conteúdos multimídia utilizando arquiteturas P2P.
- Por fim, no Capítulo 8 são apresentadas as considerações finais, discutindo-se os principais tópicos elencados neste trabalho e o planejamento para a conclusão deste trabalho.

Capítulo 2

Fundamentação

A concepção de protocolos de rede para sistemas de distribuição de conteúdos multimídia em tempo real trás à tona conceitos necessários para o entendimento da solução proposta neste trabalho.

Neste capítulo apresenta-se apenas o funcionamento dos sistemas para distribuição de mídia ao vivo em arquiteturas P2P, ao passo que conceitos sobre o funcionamento de protocolos de transporte e de rede e outros temas como controle de congestionamento e modos de transmissão são assuntos consolidados e vastamente disponíveis na literatura. Apesar da omissão de discussões acerca desses e outros tópicos relacionados a esta pesquisa, recomenda-se a leitura de referências clássicas, como as encontradas em [83; 61; 93]. Com relação aos conceitos e funcionamento do protocolo DCCP, recomenda-se a leitura do Capítulo 2 da dissertação de mestrado do autor deste trabalho, disponível através da referência [23].

Os conceitos e discussões apresentados neste capítulo foram obtidos e adaptados do documento disponível através da referência [100].

2.1 Distribuição de Mídia ao Vivo em P2P

Nesta seção, apresenta-se uma revisão dos esforços prévios no sentido de estruturar e organizar os sistemas de transmissão ao vivo em arquiteturas P2P. São discutidas as duas principais vertentes utilizadas para enviar um fluxo de mídia contínua ao vivo em P2P. Uma abordagem é baseada em estruturas de árvores e com uma forte estruturação. A outra abordagem é

baseada em malha e não apresenta uma estrutura rígida entre seus participantes.

As aplicações de vídeo na Internet têm atraído um grande número de usuários recentemente. Somente o Youtube, um dos hospedeiros mais populares de conteúdo de vídeo, hospedava em agosto de 2010, mais de 400 terabytes de vídeos. Além disso, o Youtube atraiu mais de 3.73 bilhões de visualizações até esta época. Alguns relatórios recentes apontam que esse hospedeiro de vídeos é responsável por mais de 40 % de todo o tráfego de Internet da América do Norte e o site de disponibilização de conteúdo multimídia mais acessado no mundo.

A transmissão de vídeos na Internet pode se classificar em duas grandes categorias: vídeos pré-armazenados, enviados sob demanda (*on-demand*) e vídeos ao vivo (*live*). Os usuários de vídeos assistidos sob demanda têm a flexibilidade de assistir um conteúdo previamente armazenado, da maneira que eles querem e no momento desejado. De forma contrária, um conteúdo ao vivo é transmitido no mesmo momento em que o fluxo é gerado. Logo, todos os usuários devem estar sincronizados e devem assistir o fluxo de vídeo ao mesmo tempo. Essa é a classe de transmissão de conteúdo tratado neste trabalho.

A solução básica para o envio do fluxo de vídeo na Internet é a utilização do modelo cliente-servidor. Nesse modelo, um cliente cria uma conexão com um servidor de vídeo e o conteúdo é enviado para o cliente diretamente do servidor. Existem algumas variantes deste modelo, mas as soluções baseadas em cliente-servidor demandam uma larga banda no servidor, o que gera um alto custo operacional [38].

Recentemente, vários sistemas P2P foram desenvolvidos para prover conteúdo de vídeo ao vivo e sob-demanda na Internet, com baixo custo operacional [101; 84; 19; 43; 14; 103; 31; 108; 110]. As redes entre pares (P2P) emergiram como um novo paradigma para construir aplicações distribuídas [38]. Neste tipo de aplicação, os usuários são encorajados para atuarem como clientes e servidores. Em uma rede P2P, os participantes, além de obterem serviços da rede, também os provêem. Assim, a banda de rede dos usuários finais é utilizada para reduzir a grande demanda por banda de rede, outrora necessária aos servidores.

Os sistemas de envio de vídeo que utilizam arquitetura P2P podem ser classificados em duas categorias quanto a sua estrutura: podem ser baseados em uma estrutura de árvore ou em malha. As seções a seguir descrevem e discutem o funcionamento de cada uma destas estruturas.

2.1.1 Estrutura Baseada em Árvore

Sistemas baseados em árvore têm uma estrutura sobreposta bem organizada e, tipicamente, distribuem o fluxo de vídeo enviando dos nós para seus filhos. Um dos maiores problemas desta abordagem é que são vulneráveis à entrada e abandono dos participantes da rede (*churns*) [21; 74]. Assim, quando um participante deixa a rede, a estrutura de árvore se rompe, e parte do sistema sofre, temporariamente, uma ruptura no fluxo do vídeo.

Uma maneira eficiente de se estruturar e enviar um fluxo de vídeo a um grupo de usuários na Internet seria a utilização de multicast no nível de IP [38]. Em uma sessão de multicast IP uma estrutura de árvore é formada. A fonte de vídeo se torna a raiz desta árvore multicast, e os clientes recebem o fluxo de vídeo através dos vários nós desta árvore, formado pelos roteadores que suportam o multicast em nível de IP.

Para contornar a falta de suporte de multicast em nível de IP, a função equivalente tem sido implementada no nível da camada de aplicação. Os servidores de vídeo e os usuários formam uma rede sobreposta à rede real e, assim, organizam-se para distribuir o fluxo de vídeo. De maneira similar ao multicast IP, formado por uma árvore de roteadores no nível de rede, os participantes da sessão de vídeo formam uma árvore na camada de aplicação, cuja origem é o servidor de vídeo.

Cada usuário do sistema se conecta à árvore em um certo nível. Ele recebe o vídeo de seus pais, no nível superior, e reenvia o conteúdo aos seus filhos, no nível mais baixo. Algumas aplicações, como Overcast [52], utilizam esta abordagem. Na Figura 2.1 ilustra-se um sistema com quinze nós participantes.

Existem várias maneiras possíveis de se construir a árvore para o envio de fluxo de vídeo. Deve-se considerar a altura da árvore e a quantidade de filhos de cada nó da árvore. Nós em níveis inferiores da árvore recebem o fluxo de vídeo após ele percorrer vários outros nós, e isto pode induzir a grandes latências. Para reduzir esse problema, deve-se preferir uma árvore com o mínimo de níveis possível, o que pode requerer usuários com grande largura de banda, retransmitindo para vários filhos.

Tão importante quanto a construção da árvore é a manutenção da sua estrutura. Os usuários de uma aplicação de vídeo em sistemas P2P podem ser muito dinâmicos, entrando e deixando a rede de forma muito imprevisível. Quando um nó abandona a aplicação de transmissão de fluxo contínuo em P2P, ele interrompe a transmissão e todos os seus descendentes

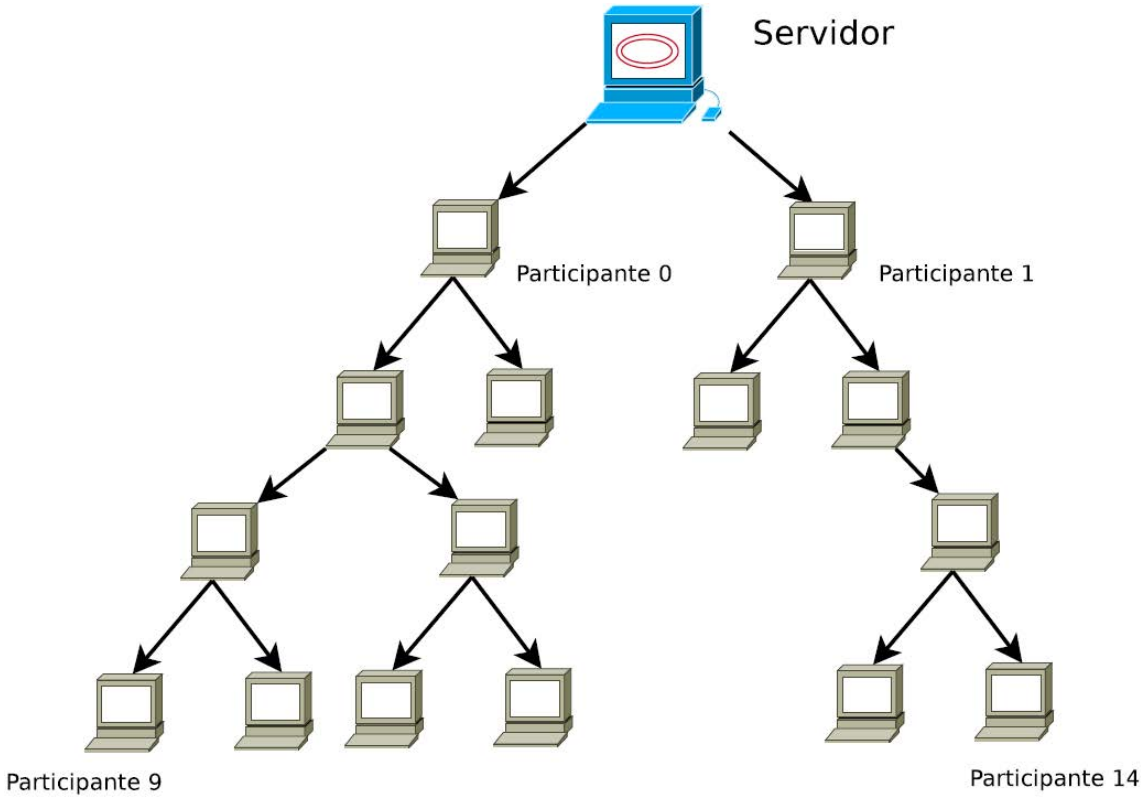


Figura 2.1: Árvore de multicast em nível da camada de aplicação.

ficam sem uma fonte do fluxo de vídeo. Para reduzir essas interrupções, a árvore de envio de fluxo de vídeo deve ser reconstruída o mais rapidamente possível. Na Figura 2.2, ilustra-se um cenário em que um nó deixa o sistema de vídeo e a árvore de multicast ao nível de aplicação que deve ser reconstruída.

A construção e manutenção da árvore de envio de fluxo P2P pode ser realizada de maneira centralizada ou descentralizada. Em uma abordagem centralizada, um servidor controla a construção da árvore e sua recuperação. Para grandes sistemas de envio de vídeo, uma abordagem centralizada pode se tornar um gargalo e um ponto de falha [38]. Vários algoritmos distribuídos abordam e tratam o problema de manutenção e construção da árvore de maneira distribuída [97]. Mesmo assim, uma abordagem baseada em árvore não consegue se recuperar de maneira rápida o suficiente para lidar com a dinâmica dos participantes, pois a constante interrupção do fluxo e a reconstrução da árvore de envio de fluxo contínuo podem causar uma sensação de baixa qualidade no serviço oferecido [21; 38; 74].

Outro problema encontrado ao se usar uma árvore simples é que os nós, que estão na

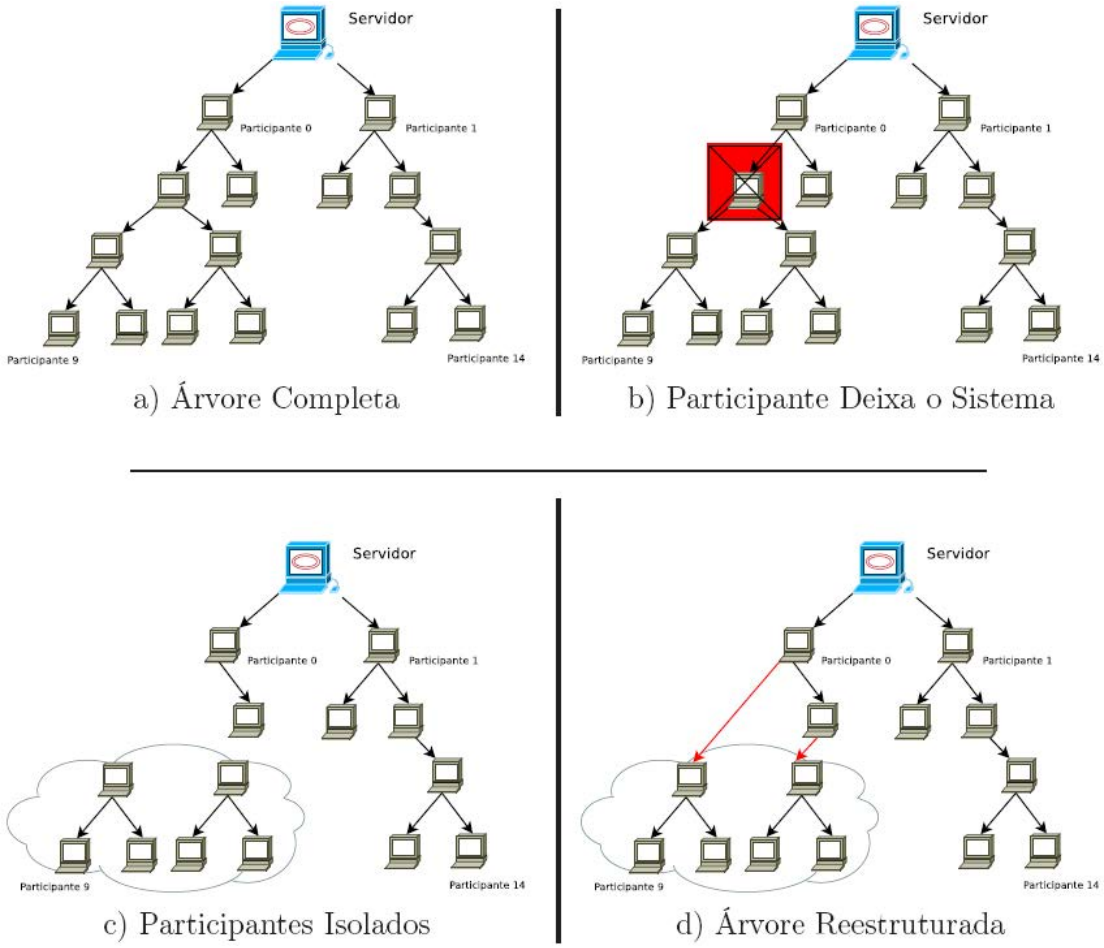


Figura 2.2: Manutenção da árvore de multicast em nível da camada de aplicação.

folha da árvore, acabam por não contribuir com o sistema. Assim a utilização de banda não é totalmente aproveitada. Uma vez que existe um grande número de nós folhas, a capacidade da árvore se torna subestimada. Para lidar com esse problema, foram propostas abordagens baseadas em múltiplas árvores como em [19]. Nesta abordagem, um servidor divide o fluxo de vídeo em vários subfluxos e para cada um destes, uma árvore multicast ao nível de aplicação é construída. Cada participante deve se conectar a todas as árvores criadas, para obter um fluxo de vídeo completo. Preferivelmente, os participantes se conectam em lugares diferentes nos vários níveis existentes. Assim, os nós folhas de uma árvore podem se tornar nós internos em outra, fazendo melhor uso da capacidade disponível. A Figura 2.3 ilustra uma aplicação de envio de fluxo de vídeo com duas árvores.

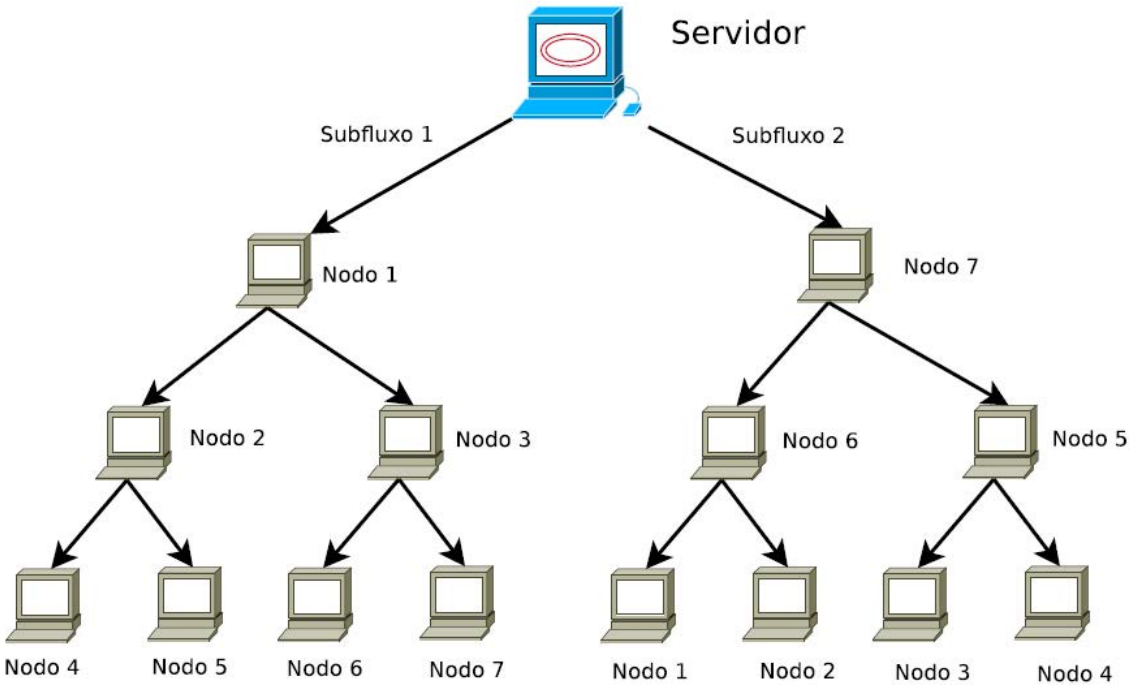


Figura 2.3: Sistema baseado em múltiplas árvores com dois subfluxos.

2.1.2 Estrutura Baseada em Malha

Em uma estrutura baseada em malha (*mesh-based*), os participantes não se organizam em uma topologia estática. As relações são estabelecidas baseando-se nos recursos disponíveis momentaneamente. Um participante se conecta a um subconjunto de outros participantes do sistema e, periodicamente, eles trocam informações. Os dados são buscados nos participantes que já os têm. Como um participante tem múltiplos vizinhos ao mesmo tempo, a organização em malha é robusta à dinâmica dos nós. Entretanto, essa relação dinâmica faz com que a distribuição de vídeo se torne imprevisível.

Diversos trabalhos recentes na área de fluxo contínuo P2P adotam uma estrutura baseada em malha [44; 103; 108; 111]. Em um sistema desse tipo não existe uma topologia fixa da rede P2P. Os nós estabelecem suas conexões dinamicamente, de acordo com seus interesses. Os participantes sempre mantêm parcerias com vários outros vizinhos. Eles podem fazer envio ou recepção de dados de múltiplos parceiros e, se um participante deixa o sistema, seus vizinhos continuam recebendo o conteúdo desejado dos demais nós, com os quais eles mantêm contato. Caso seja do interesse de um participante, ele poderá encontrar novos parceiros para manter um nível de conectividade alto. Um alto grau de conectividade faz com

que a estrutura em malha torne-se robusta à dinâmica dos participantes do sistema. Trabalhos recentes, como o disponível através da referência [74], mostram que uma estrutura baseada em malha tem um desempenho superior que uma estrutura baseada em árvores.

De maneira similar ao que acontece a um dos sistemas de compartilhamento de arquivos mais populares, o Bittorrent, uma estrutura em malha, tem um servidor centralizado. Esse servidor mantém uma lista dos participantes ativos na sessão de vídeo. Quando um usuário junta-se à aplicação de distribuição de mídia contínua ao vivo, ele contata este servidor e se cadastra. O servidor de *bootstrap*, *rendevouz* ou *tracker*, como costuma ser chamado, retorna ao novo participante uma lista com informação de um subconjunto aleatório de participantes da sessão de vídeo.

Após receber a lista com os possíveis parceiros, o novo participante tenta realizar as parcerias. Se a parceria é aceita pelo nó contatado, o novo participante irá adicioná-lo a sua lista de vizinhos. Depois de obter alguns vizinhos, o novo participante começa a trocar pedaços de vídeo com seus parceiros. A Figura 2.4 mostra o processo inicial de cadastro no sistema e realização das parcerias iniciais.

Os participantes do sistema trocam regularmente mensagens de informação de vida (*keep-live messages* ou *ping*). Caso um vizinho não responda às mensagens de vida, um participante o remove da lista e, possivelmente, tenta obter novos parceiros para manter sua conectividade [108]. Uma parceria é estabelecida por um acordo mútuo entre os participantes. Os diferentes sistemas existentes possuem estratégias variadas para estabelecimento destes acordos. Por exemplo, o número de vizinhos que os participantes possuem, a banda de rede disponível, a dinâmica dos seus vizinhos e a qualidade percebida do fluxo de vídeo [38]. Com base nesses critérios, um participante se conecta a um novo vizinho e também procura por novas parcerias.

Em uma estrutura baseada em árvore, o fluxo de vídeo é transmitido a partir de uma fonte geradora para todos os participantes do sistema, seguindo a estrutura lógica da árvore formada. Em uma estrutura baseada em malha, não existe um fluxo contínuo transmitido nestes mesmos moldes. Nesses sistemas, a fonte do vídeo (servidor) faz a codificação e a divisão do vídeo, criando os pequenos pedaços chamados *chunks*. Cada *chunk* contém dado para um pequeno intervalo de tempo de visualização. Por exemplo, as aplicações atuais transmitem dados a uma taxa aproximada de 6 *chunks* por segundo de vídeo [38]. Esses *chunks* são nu-

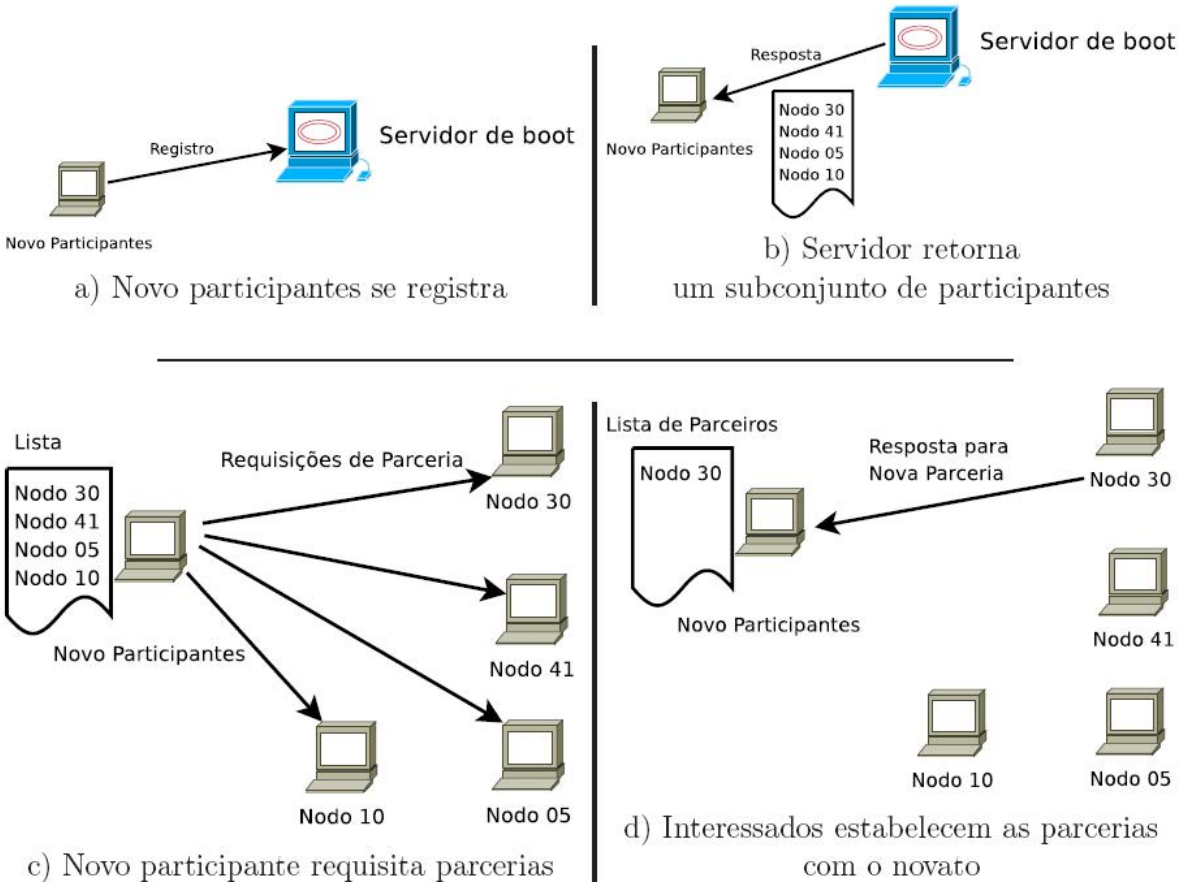


Figura 2.4: Atividade inicial de um novato - rede P2P baseada em malha.

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

Os dados são trocados principalmente através de duas estratégias: requisitando ou enviando (*pull* e *push*). Em um sistema do tipo *mesh-push* (malha e requisição), um usuário envia os dados que recebe aos seus vizinhos que provavelmente ainda não os obtiveram. Não

há uma relação clara de pai-filho neste esquema e o envio dos dados é estabelecido por interações passadas entre os participantes, onde indicam quais são os dados desejados. Um participante pode estabelecer parcerias com diversos outros e anunciar a necessidade por dados a todos estes. Por consequência, pode existir envio de dados redundantes na rede, pois mais de um dos parceiros pode responder por um pedido. Para tratar esse problema deve existir um planejamento entre os participantes do sistema, com escalonamento das transferências dos dados [108].

Caso seja usado um sistema *mesh-pull*, os participantes, periodicamente, trocam entre si um mapa de *chunks*. Este mapa tem informações dos *chunks* disponíveis localmente por um participante. Contém também informações sobre os dados faltantes. Ao obter os mapas de seus vizinhos, um participante decide como escalonar o pedido de *chunks* (e a qual vizinho enviar o pedido). As transmissões redundantes são evitadas, uma vez que os participantes solicitam *chunks* a um único parceiro. Porém, as frequentes trocas de mapas de *chunks* e mensagens por pedidos aumentam a sobrecarga do protocolo e podem introduzir novos atrasos ao sistema.

Na Figura 2.5 ilustra-se a troca de *chunks* em uma aplicação com estrutura baseada em malha. Por esta figura, o nó 2 gera seu mapa, indicando quais *chunks* ele tem disponível em seu armazenamento temporário. Ele troca este mapa com os participantes 1 e, como resposta, o nó 1 envia o seu mapa. Observe que o nó 1 possui uma lista com os diversos mapas de seus parceiros. Os pedaços de vídeo faltantes no nó 2 serão requisitados ao nó 1. Finalmente, o nó 1 responde às requisições pelo nó 2.

2.1.3 Estrutura Híbrida

Uma estrutura híbrida para transmissões ao vivo em P2P pode ser caracterizada de duas formas. Na primeira, a arquitetura da rede é um misto entre uma arquitetura baseada em árvores e uma arquitetura baseada em malhas. Na segunda, o método de transmissão de dados entre os participantes é um misto entre um sistema P2P, orientado por pedidos explícitos por dados, e um encaminhamento automático dos dados da mídia. Em ambos os casos, há uma tentativa de se obter os benefícios de cada uma das propostas e isolar os pontos fracos das mesmas.

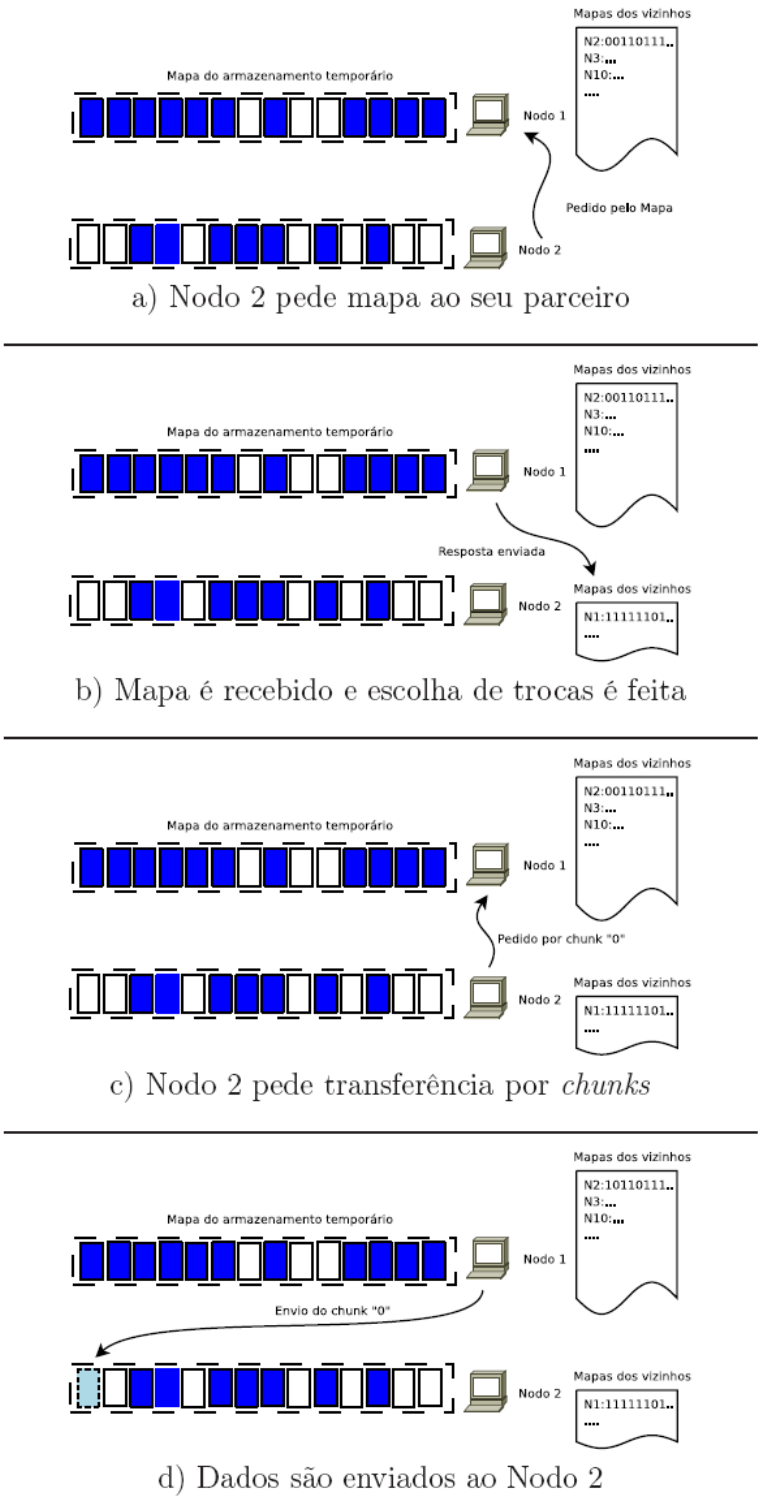


Figura 2.5: Troca de dados na aplicação baseada em malha.

Híbrido de Árvore-Malha

Em uma rede sobreposta P2P baseada em árvore, os participantes da rede são organizados de forma hierárquica. Assim, a transmissão ao vivo flui dos níveis mais altos na hierarquia (do

participante que está codificando o vídeo) para os níveis mais baixos. Os participantes mais próximos à fonte apresentam menores latências no vídeo assistido e menos problemas com relação a rupturas na hierarquia da árvore. Além disso, com o uso de uma única árvore, os participantes no nível mais baixo (folhas) não contribuem com o sistema. Isso pode diminuir a escalabilidade do sistema de transmissão ao vivo.

As estruturas baseadas em malha contornam o problema de rupturas na árvore. Nesse modelo, os participantes do sistema realizam parcerias e trocam dados entre si. Não há hierarquias e, assim, todos os participantes colaboram com o sistema o que torna a utilização dos recursos do sistema mais eficiente. Entretanto, como não há uma estrutura bem definida, a recepção dos dados da transmissão ao vivo está sujeita a atrasos e imprevisibilidade [47].

Uma abordagem de construção híbrida da rede sobreposta adota partes da rede como uma árvore, e outras partes como uma rede em malha. Os participantes do sistema podem participar de ambas as estruturas. Sistemas como [48] adotam estratégias de alocação dos participantes na árvore e, na rede em malha formada, adotam estratégias para otimizar o agendamento de entrega de dados. Alguns dos critérios utilizados para alocar os participantes na árvore são estabilidade do participante na rede e proximidade entre os participantes na rede física.

Mais precisamente, o Anysee2 [48] estrutura seus participantes em uma rede de controle e em uma rede de troca de dados. A rede de controles é baseada em uma árvore, enquanto a rede de troca de dados é baseado em uma malha.

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

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

Existem alguns mecanismos propostos com a combinação do "*push-pull*" [70; 109]. Es-

ses mecanismos usam o "*push*" para espalhar os dados rapidamente e o "*pull*" para preencher as lacunas dos dados recebidos. Nesses dois trabalhos supracitados, ambos os mecanismos coexistem, não havendo uma alternância entre eles.

O protocolo proposto em [69] alterna as operações de "*push*" e "*pull*". Cada participante é autônomo e independente, sem a necessidade de sincronia com outros participantes. Durante a operação de "*push*", o participante envia dados para algum ou alguns de seus parceiros. Durante a operação de "*pull*", o participante busca por dados que ele necessita localmente.

A utilização do mecanismo de "*push-pull*", como discutido no trabalho disponível através da referência [63], pode levar a uma redução da sobrecarga do tráfico da rede. Os resultados nesse trabalho mostram que, em comparação com um sistema do tipo "*mesh-pull*" e com o GridMedia [113], houve uma redução da sobrecarga de rede de 33 % e 37 % respectivamente. Além disso, o sistema com a abordagem híbrida alcançou resultados com latência e taxa de execução do vídeo melhores que os sistemas comparados.

2.2 Sistemas de Transmissão ao Vivo em P2P

Nesta seção apresentam-se a descrição e o funcionamento de uma aplicação de envio de mídia ao vivo em P2P. A descrição apresentada utiliza como base as principais aplicações existentes atualmente, como a Sopcast [31], o PPLive [42] e o GridMedia [113; 108]. Na Figura 2.6 ilustra-se um exemplo do sistema de transmissão ao vivo em P2P que será detalhado nessa seção.

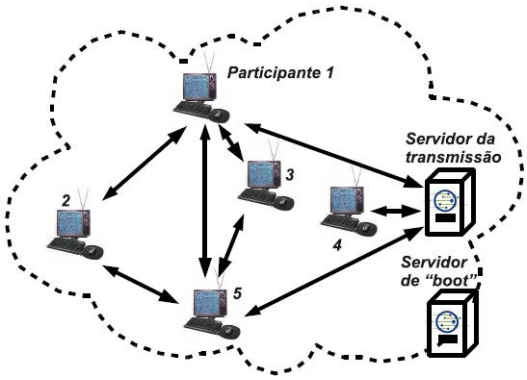


Figura 2.6: Modelo de sistema utilizado.

As principais entidades envolvidas nesses sistemas são as seguintes:

- **servidor da transmissão ao vivo:** o servidor de transmissão é um nó especial do sistema P2P. Ele captura e codifica o vídeo que será transmitido pela rede. O servidor é a fonte inicial dos dados de vídeo da rede;

- **servidor de *boot*:** o servidor de *boot* (ou *bootstrap*) é uma entidade centralizadora por meio do qual os demais nós do sistema encontram seus parceiros iniciais para entrar na rede P2P. Todo nó se registra nesse servidor para fazer parte da lista dos nós do sistema. Quando um novo nó se registra, o servidor de *boot* envia a ele uma lista com alguns nós parceiros candidatos. Quando um nó antigo deseja realizar mais parcerias, este pede ao servidor de *boot* uma lista com alguns nós para tentar novos contatos.

- **participantes (clientes/nós/*peers*)**: são os usuários do sistema P2P de transmissão ao vivo. Cada participante está em contato com um subconjunto de todos os participantes do sistema. Não há hierarquia entre esses participantes, e qualquer um pode servir dados de vídeo e também responder a pedidos por dados oriundos de seus parceiros.

Os sistemas de envio de mídia contínua ao vivo em P2P são sistemas compostos por nós que colaboram entre si para a disseminação do conteúdo gerado por um servidor. Esses nós se organizam em uma rede virtual, sobreposta à rede de computadores real. A organização dessa rede baseia-se, geralmente, em duas estruturas, a de árvore e a de malha, como discutido na Seção 2.1.

Nesses sistemas de transmissão ao vivo um servidor S gera todo o conteúdo a ser disseminado pela rede e os demais nós do sistema recebem a mídia gerada em S, dividida em pedaços conhecidos por *chunks*, reproduzindo-as e repassando-as para seus nós parceiros.

Mais detalhadamente, os sistemas P2P para envio de mídia contínua ao vivo usam recursos do conjunto $P = p_1, p_2, ..., p_n$ de seus participantes para repassar o conteúdo que é transmitido pelo servidor B. Cada participante p_i é livre para entrar e sair do sistema a qualquer momento. Tal comportamento diferencia a aplicação de transmissão ao vivo em P2P de aplicações baseadas em IP multicast, pois, em IP multicast a estrutura formada é pouco dinâmica.

Os sistemas mais populares de envio de mídia ao vivo em P2P utilizam uma rede sobreposta baseada em malha. Uma rede sobreposta é uma organização virtual dos participantes,

que pode ser completamente diferente da organização física existente. Mais ainda, no modelo de malha, a rede não é estruturada de forma rígida e as parcerias no sistema são formadas aleatoriamente. As interações e trocas de dados entre os participantes p_i e p_j, com $i \neq j$, são normalmente orientadas pelos pedidos de dados e informações entre p_i e p_j. Assim, esse tipo de rede sobreposta do tipo malha é utilizada para aliviar os efeitos de entrada e saída dos participantes na rede [113; 112].

O funcionamento desse tipo de sistema acontece da seguinte forma: inicialmente, um nó p_i conecta-se a um servidor centralizado de inicialização, denominado de *bootstrap* B ou rastreador. Na inicialização de p_i, o servidor B envia um subconjunto dos nós do sistema para o nó p_i. O subconjunto é a lista inicial de parceiros candidatos do nó p_i é definido por LPC_i ($LPC_i \subseteq P$ e $LPC_i \neq \emptyset$). Além de se registrar e pegar uma lista de candidatos a parceiros, o novo nó sincroniza a posição atual da mídia ao vivo com a posição informada pelo servidor B [112]. Assim, p_i tem uma referência do ponto da mídia ao vivo que está sendo gerada pelo servidor S e saberá a partir de qual ponto deverá solicitar os dados para reproduzir a mídia ao vivo.

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

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

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

Inicialmente, cada posição do mapa é marcada como "desejada", ou seja, $cm_i[x] = desejada$, onde $x = [0..m]$. Periodicamente, p_i requisita cm_j a cada um de seus parceiros p_j, com ($p_j \in LPi$). Dessa forma, p_i verifica quais parceiros podem satisfazer a sua

necessidade por determinado *chunk* c_t. Quando p_i recebe um *chunk* x qualquer, p_i marca $cm_i[x] = disponivel$.

Periodicamente, p_i verifica quais *chunks* c_t ele necessita ($cm_i[c_t] = desejada$) e verifica entre seus parceiros p_j, com $p_j \in LP_i$, quais possuem o *chunk* c_t com $cm_j[c_t] = disponivel$. O parceiro p_j que contém o *chunk* c_t e que possui maior disponibilidade de recursos é escolhido por p_i para a realização do pedido de *chunk*. Quando p_i recebe o *chunk* c_t de p_j, p_i marca $cm_i[c_t] = disponivel$. Os processos de escolha do *chunk* a ser requisitado e a escolha do parceiro serão detalhados nas seções seguintes. Na Tabela 2.1 apresenta-se um resumo dos parâmetros utilizados para descrever um sistema P2P de transmissão ao vivo.

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

Parâmetro	Descrição
S	Servidor de mídia contínua.
$P = p_1, p_2, ..., p_n$	Conjunto dos participantes do sistema.
B	*Bootstrap* ou rastreador do sistema.
p_i	Participante i do sistema
LP_i	Conjunto de parceiros de i.
LPC_i	Conjunto de parceiros candidatos de i.
L_i	Número de candidatos mínimo de p_i.

2.2.1 Geração do Conteúdo da Transmissão

Na aplicação de envio de mídia contínua ao vivo em P2P, o servidor S é responsável pela aquisição e codificação da mídia em um formato apropriado para a transmissão. O servidor S gera o conteúdo da transmissão e o divide em *chunks*. Cada novo *chunk* c_i é armazenado na área de memória apropriada de S (*buffer*). Assim, S marca $cm_s[c_i]$ como disponível no seu mapa de *chunks*, ou seja, $cm_s[c_t] = disponivel$.

Os parceiros do servidor S atualizarão as informações sobre ele a partir da troca dos mapas de *chunks* e perceberão a existência de novos dados. Os nós fazem requisições ao servidor S por *chunks* produzidos por ele e então S começará a disseminar os dados da

mídia. Outros nós do sistema irão encontrar os novos dados produzidos por S quando algum de seus parceiros os receberem, seja por um pedido direto a S ou por outro nó do sistema.

Nas aplicações mais populares, o servidor S gera os dados da mídia contínua ao vivo a uma taxa aproximada de 6 *chunks* por segundo. Normalmente, um vídeo é codificado a aproximadamente 300 *kbps* e assim, cada *chunk* tem cerca de 6 *KB* de dados.

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

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

Parâmetro	Descrição
S	Servidor que gera o conteúdo da transmissão.
cm_i	Mapa de *chunks* de p_i.
$cm_i[x] = desejada$, onde $x = [0..m]$	Conteúdo inicial do mapa de *chunks* de p_i.

2.2.2 Realização de Parcerias

Um nó p_i realiza parcerias logo após seu primeiro contato com o servidor de *boostrap* B. A partir do estabelecimento das parcerias iniciais, o nó efetivamente começa a participar do sistema de envio de mídia contínua ao vivo em P2P. Além deste momento inicial de estabelecimento de parcerias, um nó pode ser contatado por um outro participante p_j, requisitando sua parceria, ou p_i pode tentar novas parcerias para aumentar sua conectividade.

Tanto no estabelecimento inicial de parcerias, quanto na descoberta de novos parceiros para aumento da conectividade, o nó p_i recorre à lista de parceiros candidatos LPC_i para selecionar um nó, ao qual envia uma requisição de parcerias. Vários critérios podem ser utilizados para a escolha do candidato p_k, como uma escolha aleatória entre os nós pertencentes a LPC_i ou pelos recursos disponíveis em p_k, informados por B.

Uma vez selecionado o candidato, p_i envia uma mensagem de pedido de parceria ao candidato p_k selecionado de LPC_i. Caso p_k tenha recursos disponíveis (por exemplo, banda de rede, conexões disponíveis na aplicação etc.), adiciona p_i à LP_k e responde a solicitação de

p_i. Quando p_i recebe a resposta de p_k, ele adiciona p_k à LPC_i e a nova parceria é formalmente estabelecida. Caso não seja de interesse de p_k o estabelecimento da nova parceria (por exemplo, por falta de recursos), p_k ignora o pedido de nova parceria. O nó p_i espera por um período de tempo pela resposta de p_k e, caso não a receba, p_i retira o candidato p_k de sua LPC e repete o processo com um novo candidato selecionado.

No momento que um nó p_i adiciona um novo parceiro p_k à LPC, p_i cria um temporizador T_{ik}, o qual é acionado em intervalos de tempo tp_i. A cada acionamento do temporizador T_{ik}, o nó p_i envia uma mensagem ao parceiro p_k para verificar seu estado na aplicação. O objetivo desta mensagem é verificar se a parceria está ativa, além de haver troca de informações, como os mapas de *chunks* de ambos os nós.

O nó p_i espera a resposta de p_k acerca do pedido de parceria por um intervalo de tempo tr_i. Caso p_k não responda, p_i remove p_k de sua LP. Caso p_i receba a resposta de p_k, p_i atualiza os dados relativos a p_k, como por exemplo, o mapa de *chunks* cm_k.

Na Tabela 2.3 apresenta-se um resumo dos parâmetros utilizados para descrever a realização de parcerias no sistema de transmissão ao vivo em P2P.

Tabela 2.3: Resumo dos parâmetros utilizados na realização de parcerias.

Parâmetro	Descrição
T_{ik}	Temporizador de *Ping* entre p_i e p_k.
tp_i	Intervalo de tempo para acionar T_{ik}.
tp_k	Tempo máximo de espera de resposta de *Ping*.

2.2.3 Armazenamento e Consumo de Dados

Os nós do sistema P2P de transmissão ao vivo devem conseguir obter a mídia da rede a uma taxa apropriada. Caso não o consigam, a execução da mídia terá falhas e a experiência do usuário com relação a qualidade do conteúdo multimídia recebido poderá ser ruim. Assim, os nós devem obter os dados da mídia o mais rápido possível, uma vez que a aplicação de transmissão ao vivo exige uma baixa diferença de tempo entre a criação do trecho de mídia e a sua exibição nos nós do sistema P2P.

Caso a latência seja alta, os nós irão experimentar um atraso indesejável e, no pior dos casos, a informação será exibida muito tempo depois do fato ocorrido. Por exemplo, um gol em uma partida de futebol poderá ser comemorado pelo vizinho e muito tempo depois o usuário do nó com recepção em atraso o verá em sua aplicação responsável por reproduzir o conteúdo. Por esse motivo, os nós devem obter a mídia a uma taxa de *c chunks* por segundo. Essa é a mesma taxa de produção *chunks* pelo servidor S.

Cada nó do sistema apresenta uma área de armazenamento temporário B_i (*Buffer*), onde são armazenados os dados da mídia recebidos da rede P2P. Esse *buffer* pode guardar uma quantidade pré-determinada de *chunks*. Inicialmente, cada posição *j* de B_i ($B_i[j]$) é inicializada com conteúdo vazio e, à medida que p_i recebe os *chunks* de seus parceiros, cada posição vai sendo preenchida. Nesse esquema de armazenamento temporário, o nó p_i tem por objetivo manter o *buffer* B_i preenchido de forma que se garanta uma contínua exibição da mídia ao vivo, mesmo se ele perder conexão temporariamente com seus parceiros.

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

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

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

Na Figura 2.7 exemplifica-se o mecanismo de consumo da mídia por um nó do sistema. Nessa figura, considera-se que a taxa de criação de *chunks* é de uma unidade por intervalo de tempo. Desta forma, a cada unidade de tempo, o nó deve tentar obter o próximo *chunk*

criado. Assim, a cada intervalo de tempo, o nó desloca a sua janela de interesse para o próximo *chunk* indicado em seu mapa.

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

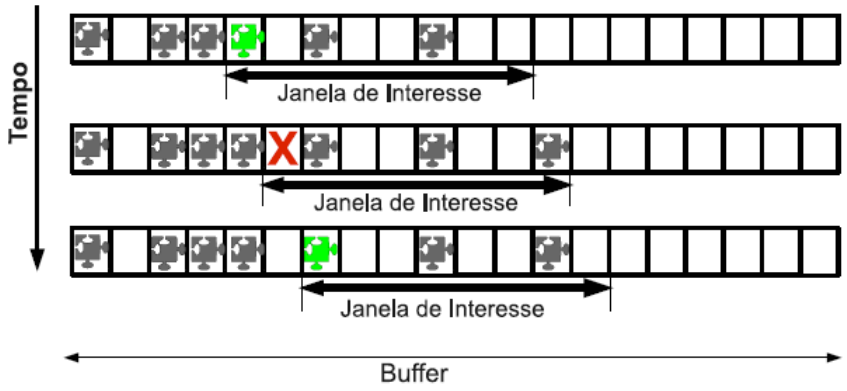


Figura 2.7: Mecanismo de consumo da mídia ao vivo.

2.2.4 Estratégia de Seleção de Chunks

A estratégia de seleção do *chunk* pode influenciar no desempenho da aplicação que reproduz o conteúdo multimídia. As estratégia existentes determinam qual dos *chunks*, entre os vários necessários, deve ser requisitado em um determinado momento. Essas estratégias de seleção tentam manter a continuidade da exibição da mídia ao vivo para um determinado nó do sistema, difundindo os *chunks* que acabaram de ser gerados o mais rápido possível para os demais nós do sistema.

Existem duas estratégias de seleção de chunks comumente utilizadas em aplicações P2P. A primeira é chamada de “Mais Raro Primeiro”, que é adotada em protocolos de aplicações de compartilhamento de arquivos em P2P, como o BitTorrent, e em envio de mídia ao vivo P2P, como o CoolStreaming [103]. A segunda estratégia é denominada “Gulosa”, onde os participantes privilegiam a escolha de *chunks* que estão próximos ao fim de suas janelas de visualização.

Estratégia "Mais Raro Primeiro"

Na estratégia "Mais Raro Primeiro", um nó p_i irá requisitar o *chunk* que está menos replicado pelo sistema de transmissão.

Para exemplificar essa estratégia, considere o *buffer* B_i do participante p_i. A posição $B_i[0]$ será a mais rara e está vazia, pois acabara de ser criada pelo servidor S. A probabilidade de encontrar um parceiro que tenha esse dado disponível cresce com o tempo. Assim, no próximo intervalo de tempo, o *chunk* da posição $B_i[0]$ irá para a posição $B_i[1]$ que, no intervalo consecutivo, será movido para a posição $B_i[2]$ e assim por diante. Dessa forma, percebe-se claramente que o *chunk* mais raro a ser buscado é o que acabara de ser criado, ou seja, $B_i[0]$. Portanto, a estratégia "Mais Raro Primeiro" seleciona os *chunks* em ordem crescente.

Estratégia "Gulosa"

Por outro lado, a estratégia "Gulosa" tem como objetivo preencher os espaços do *buffer* que estão próximos de seu prazo final de visualização. Assim, um nó p_i selecionará o *chunk* mais próximo à posição de visualização no seu *buffer* B_i. Por motivos de simplificação, pode-se considerar o *chunk* final do *buffer* ($B_i[b]$) como sendo o elemento de próximo prazo final para visualização. Sendo assim, o nó p_i selecionará o *chunk* $B_i[b]$ caso este não esteja preenchido em seu *buffer*, depois o *chunk* $B_i[n-1]$ e assim por diante. Desse modo, os nós do sistema tendem a ter armazenado em seus *buffers* os dados mais antigos produzidos pelo servidor S.

2.2.5 Seleção de Parceiros para Troca de Dados

Um participante p_i do sistema de transmissão ao vivo em P2P, além de selecionar quais *chunks* ele deve buscar em um determinado momento, deve escolher de qual parceiro realizar o pedido. Vários critérios podem ser utilizados para definir essa escolha, entre eles, critérios baseados em disponibilidade de recursos, proximidade temporal e até mesmo proximidade geográfica.

O modelo apresentado realiza uma seleção de parceiros em duas etapas. Por simplificação, adota-se a disponibilidade de recursos como critério de escolha de um parceiro para realização do pedido por um *chunk* específico. Essa mesma maneira de seleção pode ser

realizada com quaisquer outros critérios de seleção.

Na primeira etapa, um nó p_i seleciona o parceiro p_k entre todos os seus parceiros, que apresenta maior disponibilidade de recursos em um determinado instante. Os recursos de um participante podem ser conhecidos por duas maneiras: por informação direta entre os parceiros, ou por requisição e envio de informações no servidor B. No caso apresentado a seguir, os nós parceiros trocam essas informações entre si e o recurso monitorado é a banda de rede disponível para o compartilhamento.

A partir da seleção de p_k, o nó p_i envia uma requisição pelo *chunk* de interesse c_j. O participante p_i cria um temporizador para esse pedido. Quando p_k recebe um pedido por dados vindo de um parceiro ativo p_i ($p_i \in LP_k$), p_k cria uma mensagem de resposta contendo o *chunk* c_j requisitado e o envia ao nó p_i. Caso o pedido seja enviado por um parceiro não parceiro, ou seja, $pi \notin LPk$, p_k ignora o pedido por c_j, mas adiciona p_i à lista de parceiros candidatos LPC_k.

Se o pedido pelo chunk c_j for respondido durante o intervalo de tempo esperado, p_i coloca o novo dado em seu *buffer* de reprodução e o assinala como disponível em seu mapa de *chunks*, ou seja, $cm_i[c_j] = disponivel$. Fazendo-se dessa forma, p_i passa a ter a capacidade de compartilhar o *chunck* que acabara de ser recebido com seus nós parceiros.

A segunda etapa ocorre caso o pedido por uma parceria não seja respondido até a expiração de um temporizador marcado em p_i. Nesse caso, p_i refaz o processo de seleção de parceiros e escolhe um novo parceiro p_l, onde $l \neq k$. Esse processo pode ser repetido até que o *chunk* c_i seja recebido corretamente, ou que não faça mais sentido a obtenção daquele determinado *chunk*, por exemplo, por já ter passado o seu tempo de visualização.

2.3 Sumário do Capítulo

Neste capítulo, apresentou-se os principais conceitos de funcionamento dos sistemas de transmissão ao vivo P2P. Inicialmente, descreveu-se as estruturas de uma rede P2P constituídas por esses sistemas. Em seguida, apresentou-se o funcionamento básico de um sistema de transmissão e suas principais estratégias para geração e disseminação de *chunks*, bem como a seleção de nós parceiros.

Os cenários de transmissão de mídia ao vivo considerados são baseados em uma arqui-

tetura em malha e sem organização rígida dos nós do sistema. Os nós podem entrar e sair a qualquer instante e realizam parcerias com um subconjunto de outros nós. Os parceiros trocam informações entre si para colaborar uns com os outros na obtenção de uma mídia transmitida ao vivo.

Para participar de um sistema P2P de transmissão ao vivo, a aplicação do usuário se registra em um servidor chamado de *boostrap*. Esse servidor armazena as informações de todos os nós ativos do sistema. O novo nó recebe uma lista com outros nós do sistema e essa lista é utilizada para a tentativa inicial de estabelecimento de parcerias.

Entre os nós do sistema há um especial: o servidor de mídia ao vivo. Este servidor captura o vídeo a ser transmitido, codifica-o em um formato apropriado e disponibiliza-o para toda a rede P2P. Esse servidor atua da mesma forma que todos os nós do sistema, mas não requisita dados de seus parceiros.

Os nós têm um armazenamento local, onde guardam os dados do vídeo para uma execução contínua. Eles devem verificar, periodicamente, quais os pedaços de dados de que eles necessitam. Há maneiras apropriadas de selecionar e de se fazer uma requisição por dados específicos. Neste capítulo, foram apresentadas duas abordagens denominadas "Gulosa" e "Mais Raro Primeiro". Essas estratégias têm como objetivo, respectivamente, manter o fluxo da execução sem interrupções, e disseminar o conteúdo rapidamente pela rede P2P.

Caso mais de um parceiro possa contribuir com o dado necessário, um nó deve escolher a qual fará a solicitação. O mecanismo adotado baseia-se na disponibilidade de recursos de cada parceiro, que será escolhido de acordo com a maior quantidade de recursos disponíveis, como por exemplo, banda de rede, processamento, memória etc.

Capítulo 3

GMTP: Transporte de Datagramas Controlados e Não Confiáveis para Distribuição de Conteúdos Multimídia entre Pares na Internet

Neste capítulo apresenta-se o protocolo proposto neste trabalho, denominado *Global Media Transmission Protocol* (GMTP). O GMTP é um protocolo de transporte baseado em uma arquitetura P2P para distribuição de fluxos de dados multimídia de aplicações com um nó transmissor e muitos nós receptores ($1 \rightarrow n$), desenvolvido para operar principalmente na Internet. O GMTP permite a transmissão de pacotes de dados com suporte a controle de congestionamento de fluxos não confiáveis, operando em modo de transmissão multicast ou múltiplos fluxos unicast compartilhados entre os nós participantes da transmissão, através de uma rede de favores constituída dinamicamente a fim de evitar a relação de uma conexão por cliente ao servidor. Na Figura 3.1, apresenta-se um cenário geral de atuação do protocolo GMTP, onde observa-se um nó servidor e diversos nós clientes interessados por um fluxo multimídia transmitido pelo servidor. Alguns nós clientes são especiais porque repassam o conteúdo multimídia para os demais clientes e este conteúdo é recebido e retransmitido ao longo da rede por outros nós clientes e assim por diante.

O GMTP é um protocolo distribuído que funciona com a cooperação de dois conjuntos de nós especiais chamados de GMTP Relays e GMTP Reporters, eleitos automaticamente

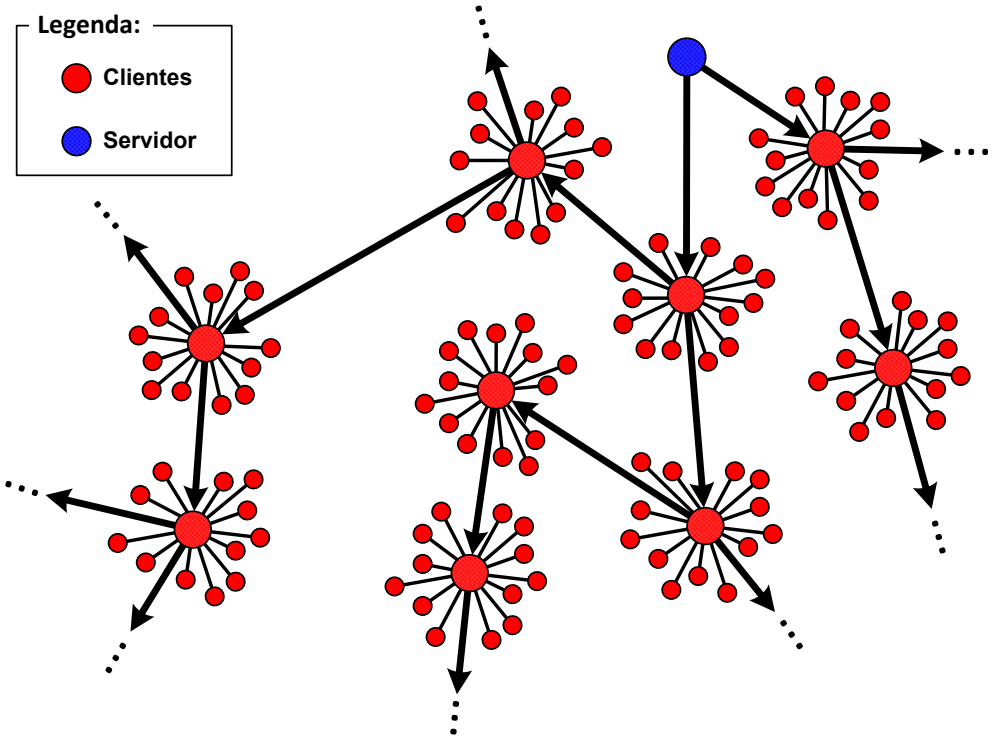


Figura 3.1: Cenário Geral de Atuação do GMTP.

em uma rede de sobreposição (*overlay network*) formada por todos os clientes interessados por um mesmo fluxo multimídia, sendo esta transparente para a aplicação que o utiliza (Figura 3.2). Os nós GMTP Relays compartilham seus fluxos de dados com um subconjunto de nós interessados pelo conteúdo multimídia, ao passo que os nós GMTP Reporters são responsáveis por enviar relatórios periódicos sobre o estado da transmissão ao nó transmissor da mídia ao vivo ou a um GMTP Relay, que os utilizam para regular a taxa de transmissão e assim impedir que a rede entre em colapso de congestionamento. Além disso, a rede de sobreposição construída pelo GMTP é organizada em níveis de acordo com a posição em que um nó GMTP Relay se encontra. Por exemplo, observa-se na Figura 3.2 que existem 4 níveis na estrutura da rede de sobreposição apresentada.

O GMTP não necessita explicitamente da instalação de um nó na rede para encaminhar o conteúdo de uma rede externa para uma rede interna (*proxy*). Além disso, o GMTP mantém a *interface* de programação com a camada de aplicação inalterada, apenas adicionando uma extensão na API padrão de *socket* BSD para preservar a compatibilidade com as aplicações multimídia existentes e, ao mesmo tempo, permitir que as aplicações façam uso dos novos recursos do GMTP. Esta decisão pode ajudar em uma rápida adoção do GMTP nas aplicações

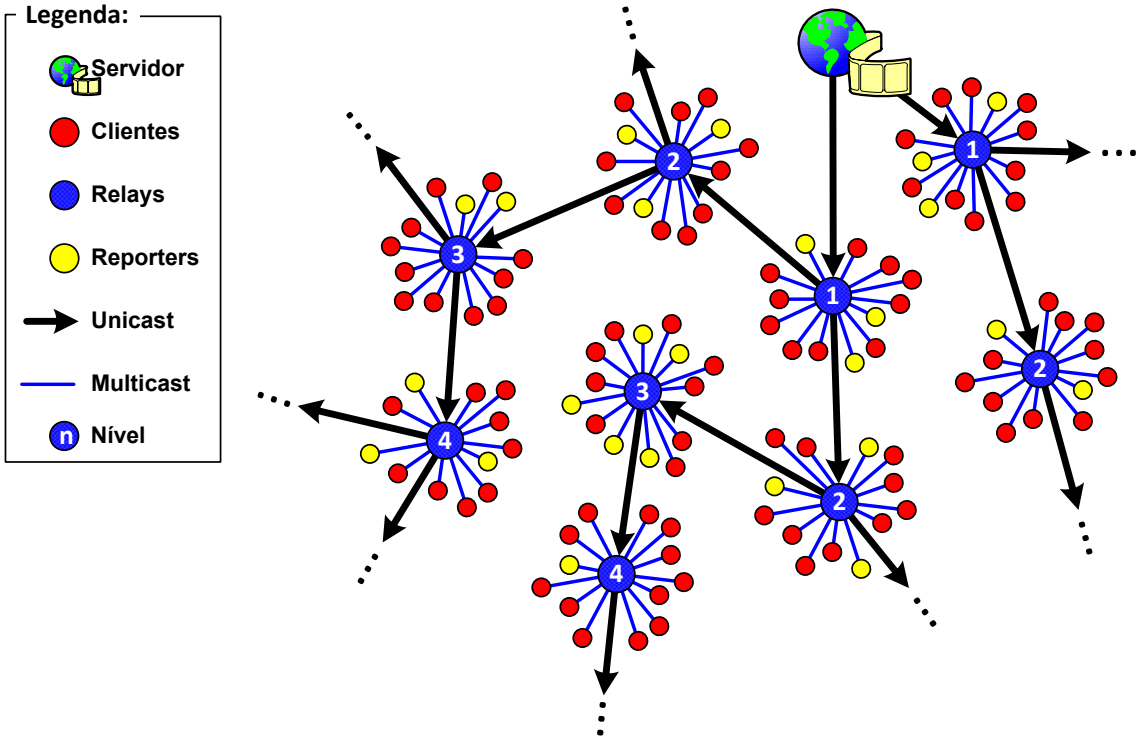


Figura 3.2: Rede de Sobreposição Construída Dinamicamente pelo GMTP com a Presença de Nós *Relays* e *Reporters*.

multimídia, permitindo-se simples alterações das aplicações existentes e, ao mesmo tempo, a efetiva padronização da forma como algumas funcionalidades hoje em dia são implementadas.

Nas próximas seções serão apresentadas as funcionalidades do GMTP, sem qualquer formalismo e com o principal propósito de permitir que o leitor obtenha uma visão conceitual acerca de tal protocolo. No Capítulo 4, o leitor poderá entender detalhes acerca do funcionamento do GMTP, inclusive considerando-se aspectos técnicos sobre sua implementação.

3.1 Metodologia e Justificativas

A definição do protocolo GMTP baseou-se nas características das aplicações que transmitem e recebem conteúdos multimídia ao vivo, considerando suas funcionalidades e arquiteturas discutidas no Capítulo 2, levando-se em conta os seguintes questionamentos:

- quais as funcionalidades das aplicações de transmissão de mídia ao vivo que tiveram que ser implementadas na camada de aplicação por falta de um protocolo de transporte

de dados ideal para esse tipo de aplicação?

- quais dessas funcionalidades podem ser implementadas na camada de transporte a fim de torná-las padronizadas para todas as aplicações existentes e assim evitar o retrabalho de desenvolvimento (implementação de controle de congestionamento, descoberta de nós, tolerância a falhas etc.)?

- como se pode, de forma eficiente e padronizada, distribuir um mesmo conteúdo multimídia de uma transmissão partindo de um servidor para múltiplos nós clientes conectados a uma rede, considerando o fato de que diferentes aplicações clientes possam ser utilizadas?

Diante dessas questões, realizou-se um estudo sobre as principais estratégias adotadas pelos desenvolvedores de sistemas de transmissão de conteúdos multimídia em tempo real. O resultado foi o seguinte:

1. constatou-se que os desenvolvedores de sistemas para transmissão de conteúdo multimídia ao vivo na Internet preferem o modelo de serviço P2P ao modelo de serviço cliente-servidor. Existem pelo menos 40 sistemas que adotam essa abordagem, entre soluções gratuitas, incluindo as de código aberto, e as proprietárias;

2. observou-se que nos sistemas desse tipo não implementam-se mecanismos para controle de congestionamento e, quando o fazem, os desenvolvedores são forçados a realizá-lo na camada de aplicação, sem qualquer padronização na forma como os fluxos de dados (conexões) são controlados, com diferentes equipes de desenvolvimento implementando, das mais variadas formas, a mesma funcionalidade presente nesse tipo de aplicação, sem qualquer compartilhamento desse esforço;

3. percebeu-se que não há uma forma efetiva de centralizar e/ou disponibilizar as boas soluções e práticas (algoritmos) para os diferentes mecanismos empregados nesse tipo de sistema, dentre eles os mais importantes são: algoritmos para descoberta e seleção de nós parceiros, controle de congestionamento, tolerância à desconexão, adaptação de fluxos de dados multimídia e compartilhamento de conexão e conteúdos.

Ao tentar resolver problemas relacionados a cada um desses pontos, os desenvolvedores de sistemas para transmissão de conteúdos multimídia enfrentam situações reincidentes já experimentadas por outras equipes de desenvolvimento. Uma analogia servirá para explicar o que acontece.

Considere um sistema de TV tradicional (teledifusão). Quando uma pessoa está assistindo um canal e troca para outro canal, o aparelho de TV consegue interpretar o novo conteúdo mesmo este sendo transmitido por outra emissora de TV. Para que isto funcione, existem padrões que descrevem o formato do vídeo/áudio (NTSC, PAL-M etc.) e a forma como os sinais devem ser decodificados, transmitidos em diferentes frequências de acordo com cada emissora de TV. Todas as TVs funcionam simplesmente porque seguem esses padrões, independente da sua marca e modelo, pois caso contrário só conseguiriam interpretar o sinal transmitido apenas de um conjunto restrito de emissora de TV.

Apesar de existirem protocolos para Internet para descrever o conteúdo da mídia sendo transmitida (RTP e RTSP, equivalentes aos padrões NTSC e PAL-M), atualmente na Internet não existe um mecanismo que transporte o conteúdo de mídia de forma transparente para as aplicações, considerando-se toda a complexidade de se trabalhar em redes orientadas a datagrama. Ao invés disso, cada aplicação implementa suas próprias formas de transporte de dados, com predominância do uso dos protocolos RTP/RTSP e UDP, e com algoritmos para controle de congestionamento implementados na camada de aplicação.

Durante as pesquisas realizadas no contexto deste trabalho, encontrou-se pelo menos 20 sistemas de transmissão de mídia ao vivo baseados em P2P, sem qualquer compatibilidade entre eles. Através dos nós servidores desses sistemas, transmite-se pacotes de dados que são recebidos por aplicações clientes para reproduzir o conteúdo recebido. Considerando a analogia anterior, os nós servidores desses sistemas são as emissoras de TV tradicionais, ao passo que as aplicações clientes, os aparelhos de TV. A diferença gritante é que as aplicações não seguem um padrão para transmitir e receber os dados e consequentemente uma aplicação cliente de um sistema não consegue reproduzir o conteúdo multimídia transmitido por outro sistema. É como se cada TV funcionasse apenas para um determinado canal, não servindo para reproduzir o sinal de vídeo oriundo de outra emissora de TV.

No GMTP propõe-se evitar estes impasses concentrando os principais mecanismos dessas aplicação em um único protocolo de transporte, tornando-os disponíveis para diferentes

aplicações, todas elas compartilhando não só as boas práticas de desenvolvimento, mas também fazendo melhor uso dos recursos computacionais e de rede. Isto permite soluções mais estáveis e eficientes, pois tais soluções serão cada vez mais testadas por um grupo maior de desenvolvedores, aprimorando-as com o passar do tempo.

3.2 Definições, Terminologias e Convenções

Para facilitar o entendimento das funcionalidades do GMTP, nesta seção, apresentam-se algumas definições, terminologias e convenções utilizadas no restante deste documento.

3.2.1 Tipos de Nós

Na Figura 3.3, apresenta-se a organização geral dos nós clientes GMTP e na Figura 3.4 apresenta-se as interações entre os tipos de nós do GMTP.

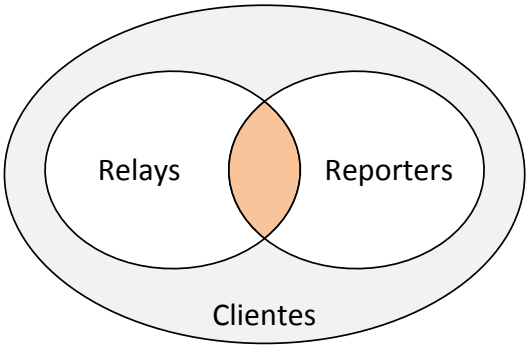


Figura 3.3: Conjunto do clientes do GMTP.

Nó GMTP: um sistema final capaz de executar o protocolo GMTP, ou seja, um nó de rede capaz de interpretar os cabeçalhos dos pacotes definidos pelo GMTP e realizar ações pré-definidas;

Servidor de mídia GMTP ou Servidor GMTP: um programa de computador em execução em um nó GMTP que captura e transmite mídias ao vivo na rede. O nó servidor mantém conexões de nós relays;

Cliente GMTP: um programa de computador em execução em um nó GMTP interessado em receber mídias ao vivo enviadas por um servidor GMTP. Um nó cliente GMTP funciona apenas de forma passiva, recebendo o fluxo de dados transmitido na rede e reproduzindo o conteúdo multimídia através da aplicação em execução;

GMTP Relay: cliente GMTP especial responsável por repassar os fluxos de dados vindos dos nós servidores ou de outros nós GMTP Relay para os clientes conectados a ele;

GMTP Reporter: cliente GMTP especial responsável por enviar relatórios periódicos sobre o estado da transmissão ao servidor GMTP ou a um GMTP Relay. O nó relay utiliza esse relatório para definir suas próximas taxas de transmissão.

Deste ponto em diante, considere os termos *cliente GMTP*, *servidor GMTP*, *GMTP Relay* e *GMTP Reporter* em sua forma simplificada, ou seja, *cliente*, *servidor*, *relay* e *reporters*, respectivamente, exceto quando explicitamente mencionado de outra forma. Estes termos não serão mais formatados em itálico, como ~~como~~ os termos *unicast*, *multi-unicast* e *multi-cast*.

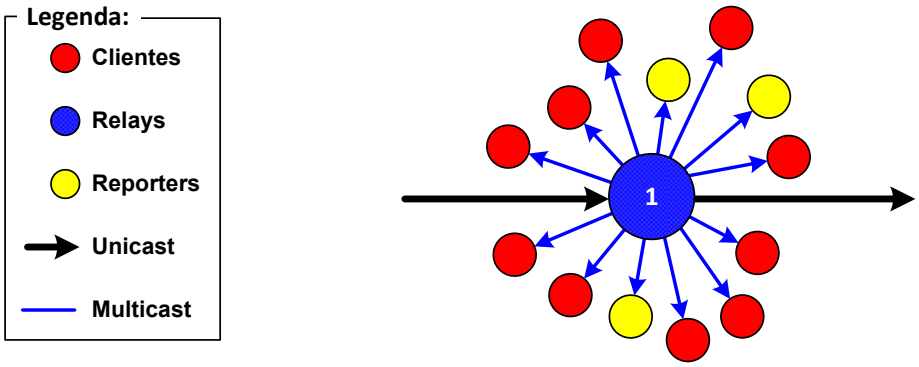


Figura 3.4: Tipos de Nós do GMTP.

3.2.2 Tipos de Pacotes

No cabeçalho dos pacotes GMTP existe um campo denominado *tipo do pacote*. Este campo determina que informação está contida em um determinado pacote GMTP e, ao processá-los, um nó GMTP executa uma determinada ação. Na Tabela 3.1 são apresentados os possíveis valores desse campo, nome do pacote e sua função.

No Capítulo 4 serão apresentados detalhes técnicos acerca do uso desses pacotes.

Tabela 3.1: Tipos de Pacotes do protocolo GMTP.

#	Tipo	Descrição
0	Request	Pedido de estabelecimento de conexão multicast
1	Response	Resposta ao pedido de estabelecimento de conexão multicast
2	Data	Contém dados da aplicação
3	Ack	Confirmação de recebimento de pacote
4	DataAck	Dados da aplicação e confirmação de recepção
5	Elect	Inicia o processo de eleição de um nó em relay ou reporter
6	ElectReply	Sinaliza o interesse de um nó em se transformar em relay ou reporter
7	ElectAck	Confirmação do nó eleito para relay ou reporter
8	RelayQuery	Transmitido por um nó para consultar a lista de relays
9	RelayReply	Resposta ao pedido de consulta da lista de relays
10	AdvConn	Utilizado por um nó relay ou reporter para anunciar que está ativo na rede
11	Reservado	Uso futuro e ignorado pelo receptor
12	Reservado	Uso futuro e ignorado pelo receptor
13	CloseReq	Servidor ou Relay solicita término de conexão sem TIMEWAIT
14	Close	Servidor/Cliente/Relay solicita término da conexão
15	Reset	Determina, incondicionalmente, o final da conexão

3.3 Visão Geral do GMTP

No GMTP emprega-se o Princípio da Cooperação de Brigadas (*Bucket Brigade Principle*), onde cada nó repassa o conteúdo recebido para outro usuário e assim por diante (Figura 3.5), sem a influência da aplicação sobre este processo.

Ao considerar essa abordagem, diversos aspectos precisam ser previstos no protocolo para permitir que as aplicações funcionem corretamente. O primeiro aspecto é a descoberta e seleção de usuários parceiros (Figura 3.6). O segundo aspecto é o tratamento quanto a capacidades de transmissão dos diferentes usuários (Figura 3.7). O terceiro aspecto é a capacidade de tratar desconexões repentinas dos usuários parceiros (Figura 3.8). E, por fim, o quarto aspecto é evitar que um usuário mal-intencionado polua o sistema com conteúdos

inapropriados (Figura 3.9).

Figura 3.5: Princípio da Cooperação de Brigadas utilizado no GMTP.

Figura 3.6: Um usuário precisa descobrir e selecionar seus parceiros.

Figura 3.7: Uma aplicação pode não ter recurso suficiente, adaptações devem ser realizadas.

3.3.1 Arquitetura

Na Figura 3.10, apresenta-se a arquitetura do protocolo GMTP e em seguida uma breve descrição das suas funcionalidades. Note que o GMTP fornece serviços à camada de aplicação através de uma API de sockets, permitindo-se às aplicações utilizarem funcionalidades frequentemente implementadas na camada de aplicação.

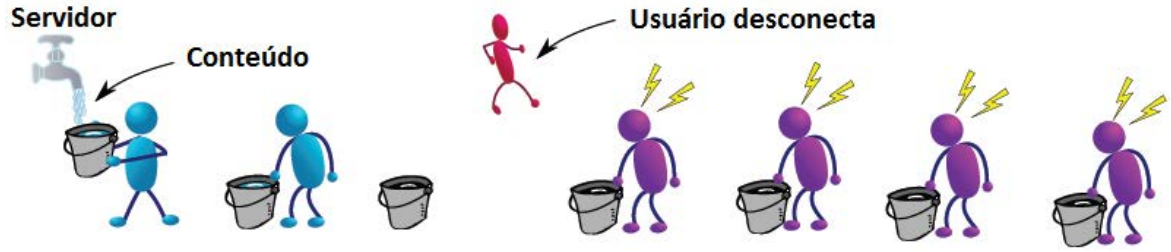


Figura 3.8: Um usuário pode desconectar e é preciso um mecanismo de tolerância a falhas.

Figura 3.9: Usuários mal-intencionados podem poluir o sistema com conteúdos alterados.

- O processo de conexão acontece em três-vias (*3WHS*), porém transmite-se o pedido de conexão em modo multicast a fim de descobrir a existência de nós relays conectados ao mesmo destino desejado.
- Detecção automática do modo de transmissão multicast e chaveamento automático entre esse modo e o modo de múltiplos fluxos unicast.
- Compartilhamento de fluxos de dados entre nós da mesma rede e com suporte a controle de congestionamento.
- Eleição automática de nós relays e reporters.
- Opções de negociação de parâmetros da conexão com confirmação de recebimento, incluindo negociação do mecanismo de controle de congestionamento a ser utilizado e notificação sobre as eleições de nós relays e reporters.
- Controle de congestionamento com suporte a ECN (*Explicit Congestion Notification*), inclusive em modo multicast.
- Arcabouço para descoberta de nós, controle de congestionamento, tolerância a falhas e adaptação de fluxo, permitindo assim que novos algoritmos sejam adicionados ao

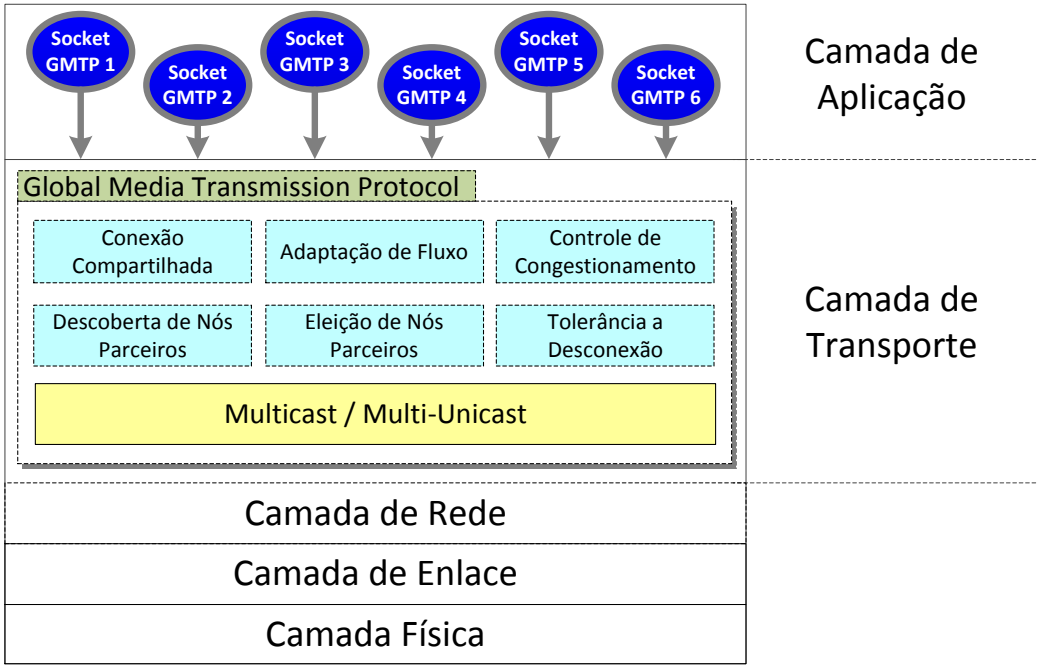


Figura 3.10: Arquitetura do Protocolo GMTP.

protocolo como módulos do sistema operacional.

- Tolerância a desconexões de relays e reporters, com reeleição automática desses conjuntos de nós.

- Mecanismo para reduzir o atraso fim-a-fim causado por muitos níveis de repasses entre os relays. O protocolo é capaz de decidir se deve obter a mídia ao vivo de um relay ou se deve conectar diretamente ao nó servidor, baseando-se em critérios como o RTT e número de saltos entre o cliente e o servidor GMTP .

3.3.2 Canais de Comunicação

No protocolo GMTP utilizam-se três canais de comunicação para implementar suas funcionalidades (Figura 3.11). Esses três canais são, na prática, *sockets* criados pelo GMTP na camada de transporte, sem que a aplicação tenha controle direto sobre eles.

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

que uma mesma interface de rede seja utilizada como entrada e saída ao mesmo tempo.

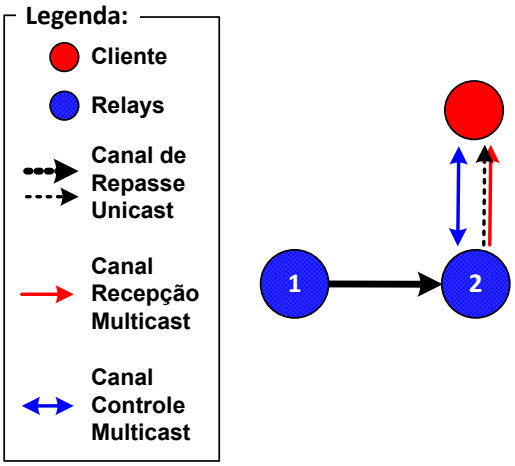


Figura 3.11: Canais de Comunicação do GMTP.

Canal de Controle

É obrigatório que todo nó GMTP ao iniciar uma instância do protocolo GMTP deve criar um *socket* multicast no endereço IP 238.255.255.250 e na porta 1900. Através desse *socket*, um nó GMTP é capaz de enviar e receber pacotes de controle utilizados para implementar as funcionalidades do protocolo. Este *socket* multicast é de suma importância no processo de conexão do GMTP e na execução das suas demais funcionalidades. Desse ponto em diante, este *socket* multicast será chamado de *canal de controle de uma conexão GMTP* ou simplesmente *canal de controle*.

O canal de controle é utilizado para criar uma rede de sobreposição e assim permitir que o GMTP execute procedimentos como estabelecimento de conexão entre os nós participantes da rede, descoberta de nós e notificações de desconexões dos mesmos, eleição de nós relays e nós reporters, envio e recebimento de relatórios para controle de congestionamento, entre outros. Tais recursos são fundamentais para o funcionamento do GMTP, fazendo-o um protocolo original e diferente dos demais encontrados na literatura.

Na prática, como o GMTP é um protocolo de transporte e portanto implementado no sistema operacional, o ciclo de vida do *socket* do canal de controle se inicia quando a primeira aplicação cria um socket GMTP e termina quando nenhuma aplicação estiver utilizando um *socket* GMTP.

A decisão do uso do endereço IP multicast 238.255.255.250 foi baseada na RFCs 2365 [76], que define o escopo administrativo do uso dos endereços IPv4 multicast entre 239.0.0.0 e 239.255.255.255. O endereço 238.255.255.255 é definido no escopo de uso global e por este motivo esse endereço foi o escolhido. Com a padronização do protocolo GMTP e a publicação da sua RFC, será necessário solicitar registro do uso desse endereço IP multicast por parte do protocolo GMTP através da *Internet Assigned Numbers Authority* - IANA[1].

Canal de Repasse

Além do canal de controle, define-se no protocolo GMTP um canal de repasse utilizado por um nó relay para repassar os dados vindos de um servidor ou de outro relay para a rede local. Esse canal de repasse, na prática, é um *socket* multicast criado pelo relay para transmitir dados para os clientes localizados em sua rede local e que tem interesse em reproduzir o mesmo fluxo de dados recebido pelo nó relay.

O *socket de repasse dos dados* deve ser criado quando um cliente se promove a relay. Na prática, quando isto acontece, o relay deve criar um *socket* multicast em um endereço IP e número de porta escolhida aleatoriamente para repassar os dados vindos do servidor ou de outro relay para dentro da rede local do relay. Isto permitirá que outros clientes recebam os dados vindo do servidor através do seu relay local. A faixa de endereços IP multicast que um cliente deve utilizar para criar seu *socket* de repasse é a de escopo local 239.192.0.0/14, definida também na RFC 2365 [76].

No caso de ocorrer a padronização do protocolo GMTP e a publicação da sua RFC, não será necessário solicitar registro do uso dessa faixa de endereços IP multicast por se tratar de uma faixa de domínio local.

Um outro canal de repasse pode ser criado por um relay caso um cliente interessado em seu conteúdo esteja localizado em outra rede que não seja a mesma do relay. Nesse caso, o relay deve criar um canal de repasse unicast e informar ao cliente o endereço IP e porta que este deve se conectar para obter os dados.

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

Canal de Recepção de Dados

O canal de recepção de dados é um *socket* multicast criado por um cliente quando o mesmo encontra um relay transmitindo o fluxo de dados de interesse dele em algum endereço IP da faixa 239.192.0.0/14. Nesse caso, os dados transmitidos por um relay devem ser recebidos por um cliente através desse relay.

Alternativamente, um canal de recepção de dados será um *socket* unicast quando um cliente não encontra nenhum relay em sua rede local e terá que se conectar a outro relay localizado fora da sua rede. O endereço IP e o número de porta que o cliente deve se conectar é determinado pelo relay no momento da conexão através do pacote do tipo GMTP-Response, como será discutido mais adiante.

Uma vez discutido sobre o que é e para que servem os canais de comunicação definidos no protocolo GMTP, a seguir discute-se o processo de estabelecimento de conexão e de que forma os canais de comunicação são utilizados nesse processo.

3.4 Processo de Conexão do GMTP

O processo de conexão do protocolo GMTP é separado em duas fases. A primeira fase acontece quando um cliente deseja obter um conteúdo multimídia transmitido por um servidor e não existe nenhum outro cliente em sua rede local conectado a tal servidor (passos 1 e 2 da Figura 3.12). Já a segunda fase acontece quando um cliente B qualquer inicia uma conexão a um servidor e um outro cliente A, alcançavel por B, já está recebendo o fluxo de dados desejado (passos 3 e 4 da Figura 3.12). Neste caso, o cliente A é um relay do servidor, cuja principal responsabilidade é repassar para a rede local os dados vindos do servidor em modo multicast para o cliente B e/ou para quaisquer outros novos clientes interessados pelo referido fluxo de dados. Considerando isto, só pode existir 1 relay por rede para uma dada conexão a um servidor remoto, sendo este eleito na primeira fase de uma conexão GMTP.

Os passos do processo de conexão do GMTP executados na primeira e na segunda fase são apresentados a seguir.

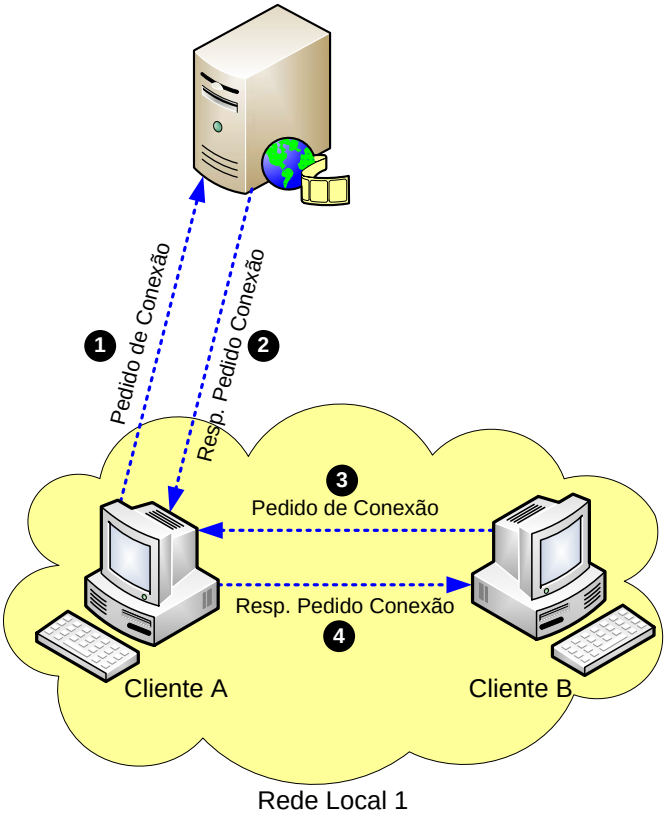


Figura 3.12: Processo Básico de Estabelecimento de Conexão do GMTP.

3.4.1 Fase 1: conexão sem relay na rede local

Para estabelecer uma conexão GMTP, o cliente inicia o *socket* do canal de controle e envia um pacote do tipo GMTP-Request, onde são especificadas duas informações no cabeçalho de transporte, o endereço IP e o número da porta do servidor. Estas informações devem ser preenchidas nos campos endereço IP e porta do servidor de mídia disponíveis no cabeçalho do pacote GMTP-Request. O pacote GMTP-Request deve ser transmitido com o campo endereço IP de destino do cabeçalho IP correspondendo ao endereço IP e número de porta do canal de controle. Nessa fase, o valor para o campo TTL do cabeçalho IP deve ser igual a 1, objetivando encontrar algum relay na rede local recebendo o fluxo de dados desejado pelo cliente que está solicitando a conexão.

Como na fase 1 assume-se que não existe nenhum relay disponível na rede local do cliente, o envio do pacote do tipo GMTP-Request não surtirá efeito, ou seja, nenhum nó GMTP enviará o pacote do tipo GMTP-Response para confirmar o pedido de conexão. Neste

caso, o cliente GMTP deve iniciar uma conexão unicast com o servidor desejado utilizando o processo de conexão tradicional em três vias (3WHS), como se faz em protocolos como o TCP e o DCCP. Ao estabelecer a conexão, o cliente GMTP se promove a relay e passa a aguardar solicitações de conexão vindas de outros clientes, preferencialmente localizados na sua rede local, utilizando o pacote do tipo GMTP-Request, o que habilitará a fase 2 do processo de conexão do protocolo GMTP.

3.4.2 Fase 2: conexão através de um relay

A segunda fase de uma conexão inicia quando já existe um cliente A na rede recebendo um fluxo de dados de interesse de um conjunto de clientes, supondo-se um cliente B desse conjunto. Neste caso, o cliente B envia um pacote do tipo GMTP-Request para o canal de controle e um relay deve responder com um pacote do tipo GMTP-Response. O relay pode estar localizado na rede local do cliente A ou em outra rede externa. O melhor caso é quando já existe um relay na rede local, pois evitam-se esforços para descobrir nós relays, uma vez que todo o trabalho para isto já foi realizado pelo atual relay, outrora apenas um cliente A qualquer, interessado pelo mesmo fluxo de dados que o cliente B tem interesse.

Quando o cliente B envia um pedido de conexão utilizando o pacote do tipo GMTP-Request, deve-se preencher os campos endereço IP e o número da porta do servidor da mesma forma que na fase 1. Ao receber o pacote GMTP-Request, o relay examina os valores desses dois campos e compara-os com todas as conexões que este tem com os servidores externos, mesmo que seja com outros relays. Se o relay possuir alguma conexão estabelecida com o servidor informado pelo cliente B, o mesmo cria um pacote do tipo GMTP-Response e preenche os campos endereço IP e número de porta do relay de mídia com os valores do seu endereço IP e do número de porta correspondente ao socket de repasse de dados do servidor para a rede local do cliente B, em modo multicast. Nesse momento, o relay aguarda do cliente B um pacote do tipo GMTP-Ack de confirmação, informando que recebeu o pacote GMTP-Response, o que caracteriza um processo de conexão em 3 vias. Após o estabelecimento de conexão, os relay utiliza o pacote do tipo GMTP-Data e GMTP-DataAck para enviar dados em modo multicast para a sua rede local.

O processo de estabelecimento de conexão GMTP permite que as aplicações compartilhem fluxos de dados entre si mesmo que elas não tenham sido desenvolvidas pela mesma

equipe. Assim, caso existam duas aplicações clientes que utilizam o protocolo GMTP, uma pode detectar a outra transparentemente e compartilhar o fluxo de dados de interesse comum de ambas as aplicações.

Um ponto específico no processo de conexão do GMTP é que este funciona com busca em profundidade e este procedimento pode ser bastante oneroso. Está em estudo uma estratégia alternativa caso nenhum relay seja encontrado utilizando o procedimento descrito anteriormente. Sendo assim, quando um cliente A não conseguir encontrar nenhum relay, o mesmo pode enviar uma requisição da lista de relays de nível 1 ao servidor ao qual está conectado em modo unicast, ou seja, o cliente solicita a lista de todos os clientes conectados diretamente ao servidor, pois estes podem ser potenciais relays e estarem localizados mais próximos a ele. Na Seção 3.7.4, discute-se como este processo funciona no protocolo GMTP.

Com a padronização do GMTP e o consequente registro de endereço multicast do canal de controle do GMTP, espera-se que os roteadores passem a rotear pacotes de controle do GMTP para as suas redes adjacentes, passando a ser possível encontrar nós relays mais facilmente em redes de salto 2 em diante, considerando a rede local do cliente como ponto de partida. Embora este tipo de funcionalidade possa demorar a ser adotado pelas empresas, espera-se que isto aconteça quando a RFC do protocolo GMTP seja aprovada. Ao longo dos anos, ocorreram casos parecidos com este, como é o caso do padrão UPnP[2] (*Universal Plug and Play*), que hoje em dia é implementado na maioria dos roteadores de rede, inclusive roteadores de pequeno porte, de uso residencial.

Note que o cliente GMTP pode decidir sobrepor o uso de um nó relay e escolher se conectar diretamente no servidor. À medida que nós relays ficam mais distantes, em termos de saltos, do servidor, pode-se observar um aumento significativo no atraso de recepção de pacotes de dados. Considerando-se o uso de um limiar de atraso aceitável e configurável pela aplicação, um nó deve estabelecer uma conexão direta com o servidor e ignorar o uso do nó relay. Este procedimento reduzirá o atraso fim-a-fim causado por muitos níveis de repasses entre os GMTP Relays.

No Capítulo 4, discute-se o funcionamento do mecanismo de conexão do GMTP em mais detalhes.

[2]UPnP: http://www.upnp.org/

3.5 Troca de Dados no GMTP

Após o processo de estabelecimento de conexão, o GMTP entra no estado de transmissão de dados. Se o GMTP estiver em funcionamento em um servidor ou em um relay, o estado é o de *transmitindo dados*, ao passo que quando executado em um cliente o estado é o de *recepção de dados*. Nesta seção, discute-se o funcionamento do mecanismo de transmissão e recepção de dados no GMTP.

Para o transporte de dados da aplicação, um servidor ou um relay deve criar pacotes do tipo GMTP-Data ou o GMTP-DataAck e enviá-los ao cliente através do *socket* informado no pacote do tipo GMTP-Response. Embora o protocolo GMTP transmite dados de forma não confiável, em alguns casos, dados de controle podem ser transmitidos de forma confiável. Durante a transmissão de dados, um nó GMTP utiliza-se do pacote do tipo GMTP-Data para enviar dados, ao passo que utiliza-se do pacote GMTP-Ack para confirmar a recepção de pacotes e o GMTP-DataAck para enviar pacotes de dados e ao mesmo tempo confirmar a recepção de pacotes de dados vindos da direção oposta (*piggyback*).

Um relay cria um *socket* de repasse a partir do momento que há a confirmação da primeira conexão por parte de um cliente. Dependendo da sua posição na rede com relação ao cliente, o relay cria um *socket* multicast ou unicast. Neste caso, o termo "*posição*" empregado aqui está relacionado com a posição lógica do relay na perspectiva da rede, ou seja, se o relay está na mesma rede do cliente ou não. Caso o relay esteja na mesma rede do cliente, o *socket* deve ser do tipo multicast, caso contrário o *socket* deve ser unicast. Isso quer dizer que, caso o relay precise repassar dados GMTP-Data para um cliente localizado em sua rede ao mesmo tempo que precise repassar dados para outro cliente localizado em uma rede externa qualquer, dois *sockets* de repasse devem ser criados, um multicast e outro unicast.

3.5.1 Modos de Transmissão

O GMTP pode operar em dois modos de transmissão: (i) multicast; e (ii) multi-unicast. O modo multicast sempre é utilizado, porém quando este modo não é suportado pela rede, o modo multi-unicast do protocolo é executado. É requerido que o modo multicast seja utilizado para transmissões de um salto, ou seja, em redes locais. O modo unicast é utilizado para que um cliente estabeleça uma conexão com um servidor ou um relay e passe a distribuir

o conteúdo de dados multimídia em sua rede local.

Como mencionado anteriormente, a mensagem de resposta de conexão enviada por um servidor ou relay é essencial para que o cliente entenda qual modo de transmissão está sendo utilizado. Três campos são utilizados para isto, são eles: (i) o bit de *multicast*, que indicará se a transmissão será em modo multicast (bit ativado) ou unicast (bit desativado); (ii) o campo endereço IP, que especificará qual endereço IP o relay passará a transmitir os dados (canal de repasse); e (iii) o campo número de porta (16 bits), que especifica a porta correspondente ao canal de repasse.

3.6 Controle de Congestionamento do GMTP

No GMTP permite-se a adição de novos algoritmos de controle de congestionamento e atualmente estão em estudo dois algoritmos para este fim, um voltado para transmissões em modo unicast e outro voltado para transmissões em modo multicast.

Na prática, definiu-se um algoritmo para controle de congestionamento híbrido, cujo comportamento dependerá se o nó que o executa está transmitindo em modo unicast ou em multicast. Em transmissões unicast, para definir a taxa de transmissão da conexão um nó GMTP executa um algoritmo de janela deslizante baseado em uma equação cúbica, ao passo que em modo multicast, executa-se um algoritmo baseado em relatórios transmitidos pelos nós receptores. Como ilustrado na Figura 3.13, para a parte do algoritmo que funciona em modo unicast, dá-se o nome de *GMTP Unicast Congestion Control* (GMTP-UCC), ao passo que para a parte do algoritmo que funciona em modo multicast, dá-se o nome de *GMTP Multicast Congestion Control* (GMTP-MCC).

3.6.1 Controle de Congestionamento Unicast

O GMTP-UCC funciona da mesma forma que o algoritmo de controle de congestionamento TCP Cubic [39], com uma diferença primordial se comparado ao algoritmo executado no protocolo TCP. Uma das funcionalidades do TCP é computar os pacotes que alcançam o sistema de destino, uma vez que se trata de um protocolo com suporte a garantia de entrega cujo mecanismo funciona com base na retransmissão de cada pacote perdido. No caso de protocolos como o GMTP, a computação dos pacotes recebidos e perdidos pelo nó recep-

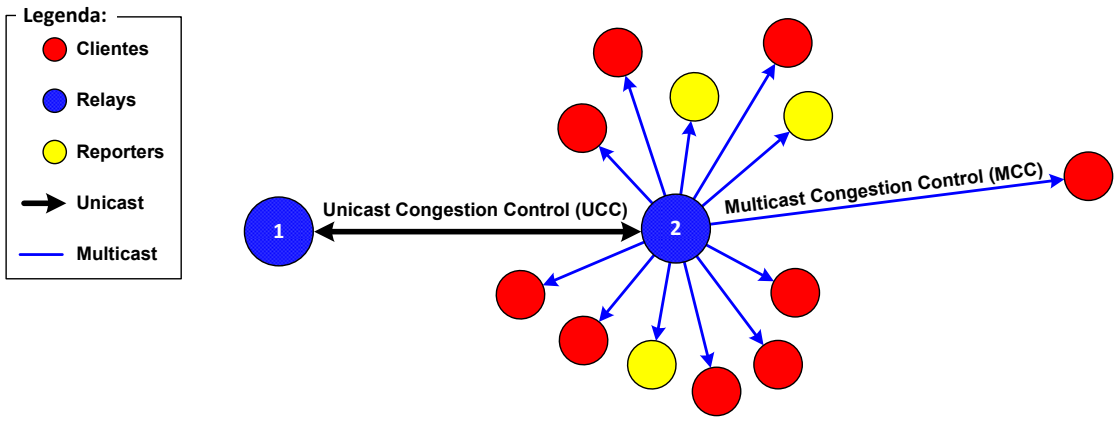


Figura 3.13: Organização do algoritmo de controle de congestionamento no GMTP.

tor é realizada pelo algoritmo de controle de congestionamento. Isto também acontece no protocolo DCCP. Considerando isto, no GMTP-UCC utiliza-se um mecanismo de vetores de ACKs, computado pelo nó receptor e transmitido para o nó transmissor, que então são repassados para o algoritmo Cubic a fim de definir a próxima taxa de transmissão. Os vetores de ACKs contém informações sobre pacotes perdidos ou pacotes marcados com ECN. Para maiores detalhes de como funciona o mecanismo de vetores de ACKs o leitor pode consultar a RFC4341 [56].

Por se tratar de um algoritmo já consolidado e utilizado como padrão no núcleo do sistema operacional Linux, decidiu-se omitir explicações detalhadas do funcionamento do algoritmo Cubic no GMTP. Embora não será apresentada uma explicação detalhada do algoritmo TCP Cubic, considera-se de suma importância justificar os motivos que levaram a escolha do TCP Cubic para transmissões unicast no GMTP.

O primeiro motivo está relacionado com os diversos resultados de pesquisas anteriores, incluindo uma série de resultados obtidos no contexto deste trabalho. Nos últimos anos diversas pesquisas científicas constataram a eficácia do TCP Cubic em termos da sua equidade para com outros fluxos TCP e, ao mesmo tempo, para com fluxos de dados TCP transmitidos utilizando outras variantes do TCP, como o TCP Vegas [71], TCP HSTCP [32] e o recém lançado TCP Compound [96; 1], utilizado na versão do sistema operacional Windows Vista em diante. O TCP Cubic não degrada os fluxos de dados transmitidos utilizando estas variantes do TCP e também não é degradado quando em disputa com fluxos de dados não-controlados, como os transmitidos utilizando o protocolo UDP.

O segundo motivo é que o TCP Cubic tem sido utilizado pela maioria dos sistemas em

execução na Internet da atualidade, uma vez que este é o algoritmo para controle de congestionamento utilizado por padrão para o sistema operacional Linux. Diante disto, desenvolver um protocolo cujo mecanismo para controle de congestionamento seja compatível com a maioria dos fluxos de dados é uma decisão primordial para o correto funcionamento e aproveitamento dos recursos de rede, em especial na Internet.

3.6.2 Controle de Congestionamento Multicast

Da mesma forma que no GMTP-UCC, o objetivo principal do GMTP-MCC é manter a equidade entre os fluxos de dados transmitidos pelo GMTP e os fluxos transmitidos pelo TCP, porém considerando o modo de transmissão multicast do GMTP.

O GMTP-MCC foi inspirado em um protocolo publicado pela IETF e conhecido pelo nome de *TCP-friendly Rate Control protocol (TFRC)* (RFC 3448 [40]). O TFRC é um mecanismo para controle de congestionamento de fluxos unicast que tenta prevê a taxa de transmissão de um fluxo TCP e utilizá-la em protocolos diferentes do TCP [35]. Trata-se de uma abordagem diferente da utilizada em algoritmos baseados em janela deslizante e que utilizam pacotes de confirmação para determinar a taxa de transmissão de uma conexão, como acontece no TCP. No TFRC, o receptor envia para o transmissor relatórios sobre as perdas observadas por ele, o que ficou conhecido como algoritmos de controle de congestionamento baseados em equação (*Equation Based Congestion Control*). Algoritmos desse tipo são adotados em diversos protocolos, como é o caso dos CCIDs 3 e 4 do DCCP. Em linhas gerais, o algoritmo TFRC funciona da seguinte forma:

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

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

3° o transmissor utiliza a Equação 3.1 para determinar qual será a sua próxima taxa de transmissão em função do relatório de perdas e o RTT obtidos anteriormente;

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

$$T = \frac{s}{t_{rtt} \times (\sqrt{\frac{2 \times p}{3}} + (12 \times \sqrt{\frac{3 \times p}{8}}) \times p \times (1 + 32 \times p^2))} \tag{3.1}$$

Na Equação 3.1 [81], T é a taxa de transmissão medida em bytes/segundo definida em função de s, que é o tamanho do pacote medido em bytes; t_{rtt}, que é o RTT entre o nó transmissor e o receptor, medido em segundos e p, que é a taxa de perda de pacotes observado pelo nó receptor.

Apesar de ser uma estratégia interessante e funcionar em conexões unicast, em transmissões multicast o algoritmo descrito anteriormente não é eficiente. O algoritmo é limitado devido a um problema conhecido por *explosão de retorno* (*feedback implosion*). O problema da *explosão de retorno* acontece devido a muitos receptores enviarem para o transmissor seus relatórios de perdas, o que resulta em uma inundação de relatórios os quais o transmissor é incapaz de processar em tempo hábil, além de onerar recursos de rede com dados de controle.

Embora existe essa limitação do TFRC, as idéias utilizadas no algoritmo inspiraram a criação do GMTP-MCC. Como o GMTP foi desenvolvido para operar em cenários onde existe apenas um nó transmissor (servidor ou relay) e diversos nós clientes, receber relatórios sobre a taxa de recepção de todos os nós receptores pode sobrecarregar o nó transmissor GMTP com demasiados pacotes de relatórios. Por este motivo, determinou-se que apenas alguns nós especiais chamados de GMTP Reporters são obrigados a enviar tais relatórios ao nó transmissor. Estes nós prestam uma papel fundamental para o GMTP-MCC, determinando algumas diferenças entre o GMTP-MCC e o TFRC, descritas a seguir.

1° O GMTP Reporters calculam a taxa de transmissão utilizando a Equação 3.1, ao invés do transmissor realizar este cálculo;

2° O GMTP Reporters determinam os eventos de perda, e não mais todos os receptores do grupo multicast;

3° Cada GMTP Reporter calculam o RTT entre ele e o nó transmissor, ao invés do transmissor realizar este cálculo;

4° A taxa de transmissão a ser utilizada pelo nó transmissor é a média aritmética de todas as taxas enviadas pelos GMTP Reporters.

Note que é de fundamental importância eleger os nós GMTP Reporters para que o algoritmo funcione corretamente. O processo de eleição dos nós GMTP Reporters é descrito na Seção 3.7.2. No Capítulo 4, discute-se o funcionamento do mecanismo de controle de congestionamento em mais detalhes.

3.7 Outros Aspectos do GMTP

Além do processo de estabelecimento de conexão, do processo de troca de dados e do mecanismo de controle de congestionamento, existem outros aspectos do protocolo GMTP que devem ser mencionados. Estes aspectos são discutidos a seguir.

3.7.1 Finalização da Conexão

O processo de finalização de uma conexão GMTP ocorre com algumas diferenças se comparado com outros protocolos orientados à conexão. Para sinalizar o pedido de desconexão, um cliente GMTP transmite no canal de controle um pacote do tipo GMTP-Close, contendo as informações de qual servidor e porta deseja se desconectar. Ao receber este tipo de pacote, o nó relay transmite ao cliente um pacote do tipo GMTP-Reset para sinalizar que está ciente do fechamento da conexão. Este procedimento é suficiente para o pedido de finalização de uma conexão de um cliente GMTP, porém para finalizar uma conexão de um nó cliente relay ou reporter, outros procedimentos devem ser realizados.

A desconexão de um nó relay gera uma situação crítica para todos os nós clientes que recebem os dados de uma transmissão através dele. Por isso no GMTP utiliza-se um mecanismo que promova um dos clientes conectados para se tornar relay em substituição aquele que está em processo de desconexão. No GMTP, são clientes candidatos a relay todos os nós reporters conectados ao atual nó relay. Em linhas gerais, quando um nó relay deseja se desconectar, o mesmo transmite para o canal de controle um pacote do tipo GMTP-Elect e aguarda por um GMTP-ElectReply. Ao receber um ou mais GMTP-ElectReply, o nó relay seleciona um dos nós que transmitiram o pacote GMTP-ElectReply e envia um pacote do

tipo GMTP-ElectAck. O pacote do tipo GMTP-ElectAck deve conter o endereço IP do cliente GMTP eleito para se tornar um nó relay. Como este tipo de pacote é transmitido através do canal de controle, todos os outros clientes também o receberão e este passo é essencial no processo de substituição de um nó relay que solicita desconexão. Isto porque, a partir do momento que um cliente GMTP é promovido a um nó GMTP relay, os clientes outrora conectados ao nó relay em processo de desconexão devem estabelecer uma conexão com o novo nó relay.

Uma estratégia complementar a anterior é o nó relay em processo de desconexão transmitir ao novo nó relay a lista de todas as conexões ativas. Desta forma evita-se que todos os clientes solicitem uma nova conexão ao novo nó relay. Apesar dessa estratégia reduzir a quantidade de dados de controle transmitidos na rede, na versão atual do GMTP esta funcionalidade não foi incorporada por ser mais complexa e necessitar de mais estudos quanto a viabilidade desta solução. Isto porque se a quantidade de clientes conectados ao nó relay em processo de desconexão for muito grande, pode não dá tempo do relay em desconexão transmitir toda a lista de conexões. Além disso, para que este procedimento funcione, deve-se atualizar as informações de *sockets* de todos os clientes para o endereço IP e porta do novo nó relay.

A desconexão de um nó reporter não gera uma situação muito crítica ao protocolo GMTP caso a quantidade de nós reporters seja maior do que 1. Se existir apenas um nó reporter e mais outros nós clientes conectados a um relay e um nó reporter solicitar desconexão, o nó GMTP deverá transmitir um pacote do tipo GMTP-Elect através do canal de controle solicitando que algum cliente se promova a um nó reporter. Quando um cliente receber um pacote do tipo GMTP-Elect, este deverá transmitir um pacote do tipo GMTP-ElectReply através do canal de controle, da mesma forma que no processo de desconexão do relay, porém um indicador especial contido no pacote GMTP-Elect sinalizará que trata-se de um processo de eleição de um nó reporter e não de um nó relay.

3.7.2 Eleição, Monitoramento e Tolerância a Desconexão

Os nós relays são selecionados de duas formas: (i) serão nós relays aqueles que iniciarem a primeira conexão unicast com algum outro nó relay ou com o nó servidor, ou seja, o nó transmissor original; (ii) serão relays aqueles que negociarem com algum outro nó relay sua

promoção para tal. Note que para este segundo caso, o nó que conceder a promoção de um nó relay para outro nó, ele deverá se rebaixar para um cliente GMTP ou estar em processo de desconexão, como discutido anteriormente. Além disso, quando um relay conceder este status a outro cliente, o mesmo poderá se desconectar do nó gerador (relay ou servidor) dos dados enviando um GMTP-Close, que conterá o endereço do novo nó relay. É possível também que um nó relay eleja outros nós relays secundários, localizados na sua própria rede local. Esta funcionalidade é importante porque caso o atual relay perca sua conexão ou desconecte do nó transmissor gerador dos dados, qualquer relay secundário poderá assumir o papel de relay primário. Neste caso, o nó que passar a assumir este papel deverá enviar um pacote do tipo GMTP-Elect informando que assumirá a transmissão de dados outrora provida pelo nó relay antigo.

Ao GMTP foi incorporado um mecanismo de tolerância a desconexão que funciona de modo a evitar que os nós clientes deixem de receber dados da transmissão em questão, caso um nó relay desconecte repentinamente sem conseguir transmitir um pacote do tipo GMTP-ElectAck, tal como explicado na Seção 3.7.1. Considere T uma variável corresponde a 4 vezes o valor do tempo do RTT. Um nó relay deve transmitir no canal de controle um pacote do tipo GMTP-AdvConn a cada instante de T, anunciando aos demais nós da rede que está ativo e operando corretamente. Caso um nó relay secundário não receba o pacote do tipo GMTP-AdvConn durante o período de tempo T, assume-se que o relay atual foi desconectado por algum motivo desconhecido e o relay secundário que não recebeu o pacote do tipo GMTP-AdvConn deverá transmitir um pacote do tipo GMTP-ElectAck. Na prática, o nó relay secundário torna-se um nó relay primário do o grupo de clientes, incluindo os nós reporters, conectados ao relay que foi desconectado. Neste caso, o novo nó relay deve iniciar um novo processo de estabelecimento de conexão. Após o estabelecimento dessa conexão, como descritos na Seção 3.4, o novo nó relay deve criar o canal de repasse e começa a repassar os dados da transmissão multimídia.

Com relação aos nós reporters, o processo de eleição funciona de forma similar e da seguinte forma: à medida que um relay recebe pacotes do tipo GMTP-Request, no pacote GMTP-Response o nó relay ativa um indicador sinalizando que o cliente deverá se comportar como um nó reporter, passando a enviar relatórios da taxa de transmissão observada por ele. Note que este modo de transmissão deve ser implementado com garantia de entrega, ou

seja, com a confirmação de recepção de pacotes e retransmissão caso este tipo de pacote seja perdido. Assim, um nó relay poderá ter controle sobre a quantidade de nós reporters e receber relatórios apenas de um sub-conjunto de nós da rede.

3.7.3 Adaptação de Fluxo de Dados Multimídia

Uma funcionalidade peculiar do GMTP é sua capacidade de permitir a realização de adaptação de fluxos multimídia de forma distribuída. A maioria das soluções para transmissão de dados multimídia, além de realizar controle de congestionamento no nível de aplicação, realizam adaptação de fluxo multimídia na fonte geradora dos dados. Em diversas soluções existentes, os autores consideram a transmissão de fluxos de dados multimídia adaptados e transmitidos em diferentes canais, sendo que em cada canal transmite-se os fluxos multimídia em uma determinada qualidade. Dependendo da qualidade desejada pelo nó receptor, o sistema cliente solicita a transmissão em um determinado canal. O problema dessa abordagem é que o nó transmissor, necessariamente deve transmitir os dados em múltiplos canais, o que aumenta a complexidade da aplicação e a quantidade de fluxos de dados sendo transmitidos a partir do servidor.

No GMTP, é possível realizar a adaptação de fluxo de dados de forma distribuída, na prática, em cada relay. Por exemplo, considere duas redes adjacentes, rede 1 e rede 2. Considere que existe um nó relay na rede 1 e entre a rede 1 e o nó transmissor a largura de banda de transmissão disponível seja de 100 *Mbps*. Caso a largura de banda disponível na rede 2 seja de no máximo 10 *Mbps*, um nó receptor na rede 2 teria que solicitar um fluxo multimídia em um canal diferente, considerando as soluções que adotam a estratégia de adaptação de fluxo com o uso de múltiplos canais de transmissão. No caso do GMTP é possível que um nó na rede 2 obtenha o fluxo multimídia através do relay presente na rede 1, com o relay da rede 1 adaptando o fluxo multimídia de acordo com a capacidade do canal de transmissão disponível para a rede 2. Desta forma, pode-se diminuir o tráfego na rede do nó transmissor e ainda permitir que nós em redes com largura de banda limitada consigam obter o fluxo multimídia adaptado (caso mais comum para clientes residenciais).

3.7.4 Outra Estratégia para Descoberta de Nós Relays

Um aspecto primordial do GMTP é a capacidade de obter fluxos de dados multimídia através de nós relays, os quais repassam esses dados vindo de uma fonte geradora. No processo de conexão, esses nós relays são encontrados, aceitam conexões de clientes e repassam dados da aplicação como se fossem o nó servidor. Um gargalo no procedimento padrão adotado no GMTP é que pode-se demorar até que um cliente GMTP encontre um nó relay e comece a receber o fluxo de dados desejado devido ao mecanismo de busca por profundidade por nós relays utilizando transmissões multicast, utilizando-se valores incrementais para o campo de TTL presente no cabeçalho IP.

Diante disso, está em estudo no contexto desse trabalho um mecanismo alternativo para permitir que um nó cliente encontre um nó relay mais rapidamente. Este mecanismo consiste em permitir que um nó cliente solicite diretamente ao nó servidor a lista de nós relays conectados a ele, ou seja, a lista dos nós relays de primeiro nível (Figura 3.2).

O mecanismo de busca por nós relays permitirá que o cliente consulte, ao longo dos níveis dos nós relays, aquele nó relay que mais se adequa aos requisitos da aplicação, principalmente com relação ao atraso observado desde do servidor até um determinado relay. Um nó cliente que desejar solicitar esse tipo de requisição, utiliza o pacote do tipo GMTP-RelayQuery e transmite o pedido de consulta ao nó servidor, o qual responde ao cliente com a lista dos nós relays de primeiro nível utilizando o pacote do tipo GMTP-RelayReply. Com isto, é possível encontrar um melhor relay cujo atraso não ultrapasse um determinado limiar de tempo definido pela aplicação, o que não necessariamente será o nó relay mais próximo geograficamente do nó cliente.

3.7.5 Benefícios e Aplicabilidade

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

Considerando-se as discussões realizadas nas Seção 1.1 e os requisitos dos sistemas de transmissão de mídia em tempo real, o GMTPatende-os na medida em que:

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

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

- permite que as aplicações façam uso de soluções estáveis e largamente testadas, uma vez adicionadas ao protocolo em questão. Neste caso, grupos diferentes de desenvolvimento podem adicionar e testar suas propostas em um protocolo de rede que, uma vez consideradas estáveis podem ser compartilhadas entre os diferentes sistemas. Até mesmo soluções já implementadas nos sistemas existentes podem ser portadas para o GMTP;

- padroniza a forma como os fluxos de dados multimídia gerados pelos sistemas considerados são transportados na Internet, incluindo aspectos de controle de congestionamento e compartilhamento desses fluxos de dados entre os nós participantes de uma transmissão, além de um arcabouço que permite estender o protocolo por meio da adição de novos algoritmos;

- indiretamente diminui os fluxos de dados na rede sem qualquer controle de congestionamento, pois aplicações para distribuição de mídia ao vivo atualmente fazem uso do protocolo UDP ou variantes;

- flexibilidade no acesso a rede por parte dos nós participantes, pois eles podem entrar e sair da rede a qualquer momento, realizando parcerias com um subconjunto de nós participantes a fim de receber o conteúdo multimídia interessado, caso esteja-se utilizando o GMTP;

- disponibiliza uma solução unificada que permite o uso da API padrão de *sockets* BSD, o que facilita a migração das aplicações existentes para utilizam este novo protocolo. Além disso, permite-se que as aplicações continuem utilizando outros padrões de redes definidos pela IETF, tais como o RTP e o RSTP;

- elimina ou pelo menos inibe a presença de nós *free-riders* na rede, uma vez que todo nó GMTP é obrigado a compartilhar conteúdo sem a influência da aplicação;

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

Dentre os diversos sistemas P2P para transmissão de mídia ao vivo que podem se beneficiar com o uso do GMTP, destacam-se os baseados em uma arquitetura em malha e sem organização rígida dos nós participantes do sistema. Isto também se aplica a todas as variantes dessa arquitetura como, por exemplo, híbrido por encaminhamento automático e pedido explícito (*Push-Pull*) e híbrido árvore-malha (vide Capítulo 2). Especificamente, os sistemas de distribuição de mídia ao vivo mais conhecidos e que podem se beneficiar diretamente com o uso do GMTP são o Sopcast [31], o PPLive [42] e o GridMedia [113; 108].

3.8 Sumário do Capítulo

Neste capítulo, apresentou-se uma visão geral do *Global Media Transmission Protocol* (GMTP), um protocolo de transporte baseado em uma arquitetura P2P para distribuição de fluxos de dados multimídia de aplicações com um nó transmissor e muitos nós receptores (1 $\rightarrow n$), desenvolvido para operar principalmente na Internet. O GMTP permite a transmissão de pacotes de dados com suporte a controle de congestionamento de fluxos não confiáveis, operando em modo de transmissão multicast ou múltiplos fluxos unicast compartilhados entre os nós participantes da transmissão, através de uma rede de favores constituída dinamicamente a fim de evitar a relação de uma conexão por cliente ao servidor, como acontece em protocolos unicast de transporte de dados multimídia disponíveis na literatura.

O GMTP possui um mecanismo de conexão separado em duas fases, onde a primeira fase acontece quando o primeiro nó em uma rede local deseja estabelecer uma conexão com um

servidor que está transmitindo um determinado fluxo multimídia. Ao perceber que nenhum outro nó em sua rede local está recebendo o fluxo de dados desejado, o cliente estabelece uma conexão unicast com o servidor e se auto promove a um nó especial chamado de relay. A segunda fase do processo de conexão do GMTP acontece quando um segundo nó cliente deseja obter o mesmo fluxo de dados multimídia que o primeiro nó cliente, considerado o nó relay daquela rede. No momento em que isto acontece, o nó cliente é capaz de perceber a presença de um relay e passa a receber o fluxo de dados através do nó relay em modo multicast, evitando assim um novo pedido de conexão ao nó servidor de dados.

Um aspecto importante do GMTP é seu mecanismo de controle de congestionamento de fluxos não confiáveis. O controle de congestionamento empregado no GMTP funciona de forma híbrida, a depender do modo de conexão utilizado por um determinado nó. Quando o protocolo GMTP está operando em modo unicast, utiliza-se o GMTP-UCC, um algoritmo de controle de congestionamento baseado no TCP Cubic e escolhido para operar no GMTP porque tem alta capacidade de convergência no compartilhamento do canal de transmissão entre os diferentes fluxos de dados. Existe também o GMTP-MCC, que é o algoritmo para controle de congestionamento utilizado quando um nó GMTP opera em transmissões multicast. Tal algoritmo é baseado em uma equação TFRC (*TCP Friend Rate Control*) que faz uso de nós especiais para determinar a próxima taxa de transmissão que um nó transmissor GMTP deverá utilizar. Esses nós especiais são chamados de reporters.

Em seguida, discutiu-se sobre outras funcionalidades do protocolo GMTP, tais como seu mecanismo para finalização de conexão, eleição, monitoramento e desconexão de nós relays e reporters, assim como possíveis mecanismos para adaptação de fluxos de dados multimídia de acordo com a capacidade do canal, ainda em definição e estudo no contexto deste trabalho. Por fim, apresentou-se os benefícios trazidos pelo GMTP às aplicações que o utiliza.

No próximo capítulo, continua-se com as discussões sobre o protocolo GMTP, porém apresentando-o de forma mais técnica e com discussões sobre a sua implementação.

Capítulo 4

GMTP: Detalhes de Funcionamento e Estado Atual de Desenvolvimento

Neste capítulo, apresenta-se detalhes de funcionamento do protocolo GMTP no tocante a três principais aspectos, o cabeçalho de pacotes, o processo de conexão e o mecanismo para controle de congestionamento. Neste capítulo, apresentar-se detalhes técnicos de funcionamento do GMTP, sendo este mais dedicado aos leitores interessados em sua implementação. Na versão final deste trabalho, este capítulo se tornará um *draft* de RFC a ser submetida para a IETF, justificando assim o seu teor mais técnico.

As palavras “deve”, “não deve”, “requerido”, “pode”, “não pode”, “recomendado” e “opcional”, incluindo suas variações morfológicas, devem ser interpretadas como descrito na RFC 2119 [15], em inglês.

Todos os *bytes* no GMTP, tais como números de portas, números de seqüência e valores para opções são transmitidos em *network byte order* (primeiro os bytes mais significativos).

Os números aleatórios no GMTP são utilizados por razões de segurança e podem ser escolhidos de acordo com a RFC 4086 [29].

4.1 Cabeçalhos e Tipos de Pacotes do GMTP

Na Figura 4.1, ilustra-se o cabeçalho genérico do GMTP. O nome genérico é justificado porque o cabeçalho assume um formato diferente dependendo do tipo de pacote transmitido. De acordo com o tipo de pacote transmitido, o GMTP poderá utilizar até 48 bits para diferentes

finalidades e, nestes casos, o tamanho total do cabeçalho passa a ser de 20 bytes. A descrição dos campos do cabeçalho genérico é apresentada a seguir.

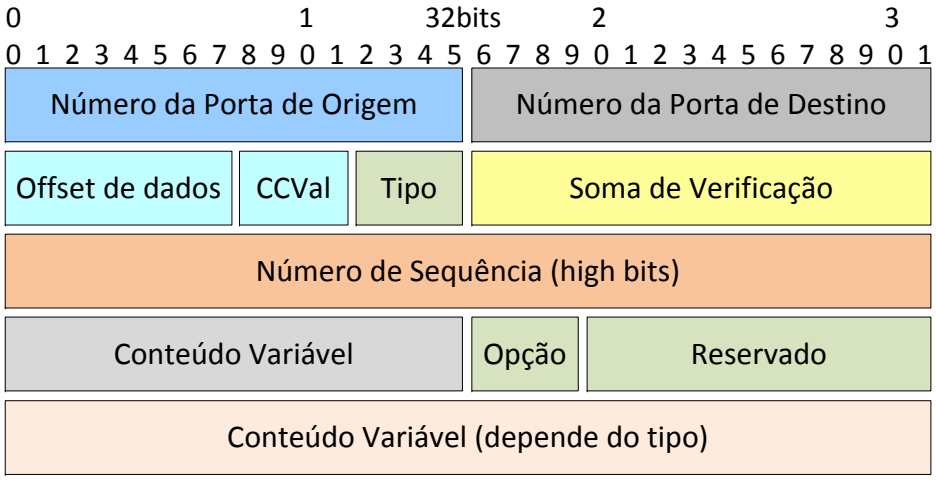


Figura 4.1: Cabeçalho Genérico do protocolo GMTP.

Porta de origem e destino: cada porta possui um tamanho de 16 bits. Estes campos identificam a conexão, como acontece com os protocolos TCP, UDP e o DCCP;

***Offset* de dados:** ou simplesmente *offset*, determina o tamanho do cabeçalho GMTP, contando do início do cabeçalho até o início de onde estão os dados da aplicação. Este campo tem o tamanho de 8 bits;

CCVal: é utilizado pelo controle de congestionamento do sistema transmissor. O tamanho desse campo é de 4 bits. Em uma transmissão GMTP entre um cliente e um servidor GMTP ou de um GMTP Relay, o algoritmo para controle de congestionamento de cada lado pode enviar 4 bits de informação para o lado oposto utilizando este campo para tal;

Tipo do pacote: tamanho de 4 bits. Este campo determina o tipo de pacote que está sendo transmitido/recebido. Os possíveis valores desse campo serão apresentados na Seção 4.1.1;

***Checksum*:** tamanho de 16 bits. Este campo é utilizado para checagem de erro, tradicionalmente como acontece em outros protocolosde transporte;

Número de seqüência: número de seqüência com 32 bits utilizado para transmitir requisições, podendo ser estendido para 48 bits ao utilizar-se dos próximos 16 bits de conteúdo variável, o que dependerá do tipo de pacote a ser transmitido. Como em outros protocolos, este campo identifica unicamente um pacote transmitido na rede por um sistema final. O valor deste campo aumenta-se em 1 a cada pacote transmitido;

Opção: tamanho de 4 bits. Este campo é utilizado para sinalizar a ativação ou não de alguma opção do GMTP, por exemplo, para sinalizar se a conexão entre um cliente e um relay GMTP deve ser unicast ou multicast;

Reservado: tamanho de 12 bits. Campo reservado para utilizações futuras;

Conteúdo variável: tamanho de 48 bits. Campo reservado para uso em mecanismos específicos do GMTP como, por exemplo, especificar o endereço IP e número da porta do servidor GMTP no momento de uma conexão multicast.

4.1.1 Tipos de Pacotes

No Capítulo 3, apresentou-se a Tabela 3.1, quando descreveu-se brevemente os tipos de pacotes utilizados no GMTP. No campo *tipo do pacote* desse cabeçalho genérico do GMTP, determina-se que tipo de informação está contida no pacote transmitido por um nó GMTP. Isto permite que um nó execute uma determinada ação ao recebe um pacote de um outro nó GMTP e possivelmente gerando-se outros pacotes como resposta. Nesta seção, apresenta-se detalhes do uso de cada um dos tipos de pacotes, discutindo-se através de exemplos o preenchimento dos campos do cabeçalho genérico apresentado anteriormente.

GMTP-Request

O pacote do tipo GMTP-Request, número 0 (0000_2), é utilizado pelo cliente GMTP para enviar um pedido de estabelecimento de conexão em modo multicast. Quando transmitido na rede, um nó GMTP Relay captura esse tipo de pacote e responde ao cliente GMTP, notificando-o a respeito do fluxo de interesse e que este é um dos GMTP Relays do servidor de mídia GMTP. Considerando o cabeçalho genérico do GMTP ilustrado na Figura 4.1, os dois campos variáveis desse cabeçalho são utilizados. Como pode-se observar na Figura 4.2,

o campo variável de 16 bits é utilizado para armezenar o número da porta do servidor de mídia GMTP e o segundo campo de 32 bits é utilizado para armazenar o endereço IP desse servidor. No processo de conexão, esses dois campos variáveis são lidos por um GMTP Relay a fim de identificar o fluxo de mídia desejado pelo usuário e, caso exista algum GMTP Relay recebendo o fluxo de mídia de interesse, o mesmo responde pelo pedido de conexão como se fosse o servidor de mídia GMTP Relay original, utilizando-se do pacote GMTP-Response, descrito a seguir. Na Seção 3.4, discutem-se detalhes do processo de estabelecimento de conexão do GMTP.

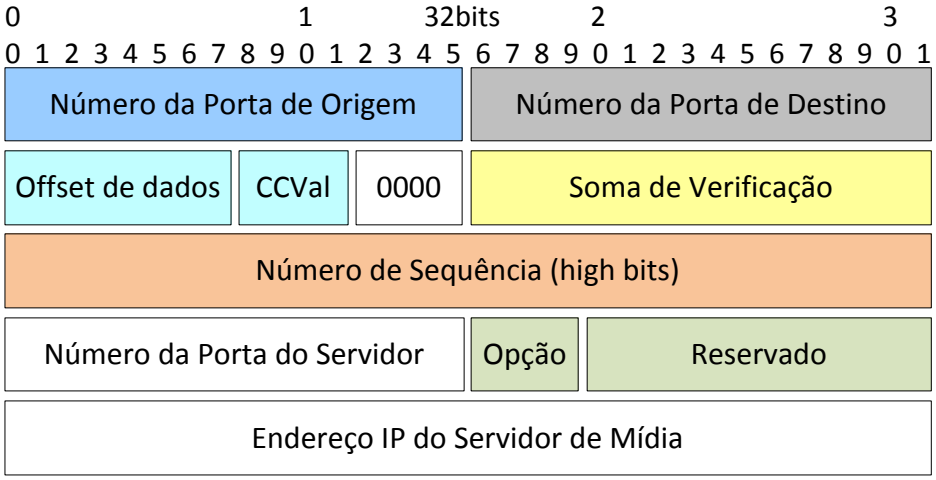


Figura 4.2: Cabeçalho do pacote GMTP-Request.

GMTP-Response

O pacote do tipo GMTP-Response, número 1 (0001_2), é utilizado pelo GMTP Relay para enviar uma resposta a um pedido de estabelecimento de conexão enviado por um cliente GMTP em modo multicast. Quando um nó GMTP Relay recebe um pacote GMTP-Request, este cria um pacote do tipo GMTP-Response para informar ao cliente GMTP sobre o estabelecimento de conexão. Neste caso e considerando o cabeçalho genérico do GMTP ilustrado na Figura 4.1, os dois campos variáveis desse cabeçalho são utilizados. Como pode-se observar na Figura 4.3, o campo variável de 16 bits é utilizado para armezenar o número da porta do GMTP Relay e o segundo campo de 32 bits é utilizado para armazenar o endereço IP desse Relay. Desta forma, um cliente GMTP é capaz de ler pacotes do tipo GMTP-Data transmitidos por um GMTP Relay via multicast na rede e reproduzir a mídia de interesse. O

tipo de pacote GMTP-Data e GMTP-DataAck são descritos a seguir.

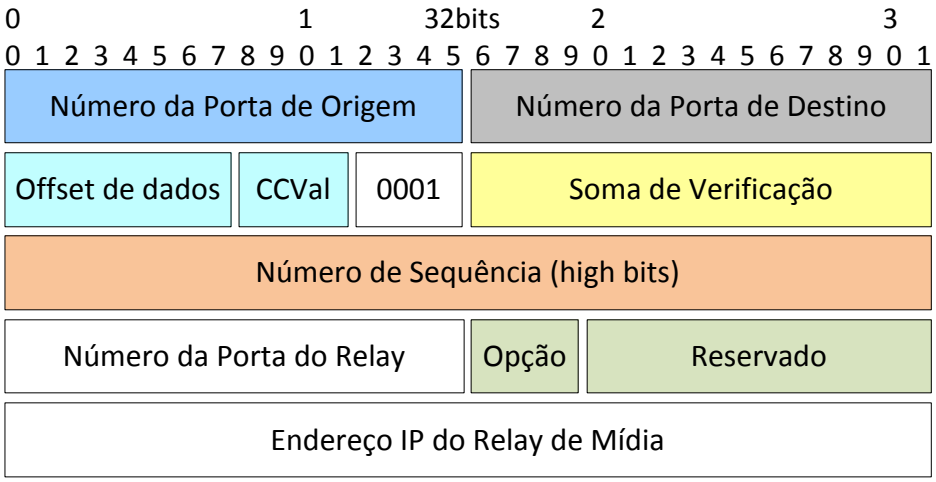


Figura 4.3: Cabeçalho do pacote GMTP-Response.

GMTP-Data e GMTP-DataAck

Os pacotes do tipo GMTP-Data e GMTP-DataAck, números 2 e 4 (0010_2 e 0100_2), respectivamente, são utilizados por um GMTP Relay para enviar dados em modo multicast a todos os clientes GMTP interessados pelo fluxo por ele transmitido. A partir do momento que um nó GMTP se torna um nó GMTP Relay, através do processo de eleição de nós GMTP Relays, descrito na Seção 3.7.2, este começa a retransmitir, em modo multicast, os dados vindos do servidor de mídia GMTP ou de outro GMTP Relay, utilizando pacotes dos tipos GMTP-Data ou GMTP-DataAck para este fim. Neste caso, o cabeçalho genérico do GMTP ilustrado na Figura 4.1, passa a ter a forma dos cabeçalhos ilustrados nas Figuras 4.4 e 4.5, respectivamente. Note que ambos pacotes não possuem os campos endereço IP e porta relacionados ao servidor de mídia GMTP. Esta decisão foi intencional para forçar que um cliente GMTP realize o pedido de conexão enviando o pacote GMTP-Request, caso contrário um cliente GMTP poderia capturar um pacote GMTP-Data sem que um GMTP Relay soubesse de sua existência.

GMTP-Ack

O pacote do tipo GMTP-Ack, número 3 (0011_2), é utilizado por um nó GMTP para enviar confirmações de recepção de pacotes contendo dados enviados com garantia de entrega. Por

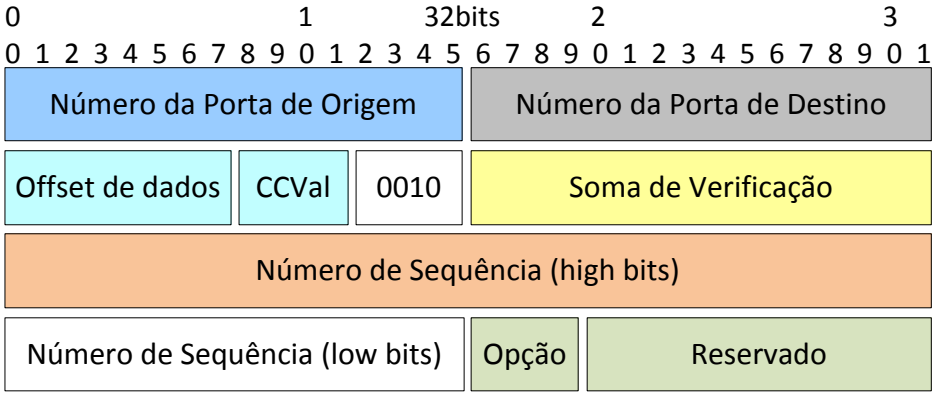


Figura 4.4: Cabeçalho do pacote GMTP-Data.

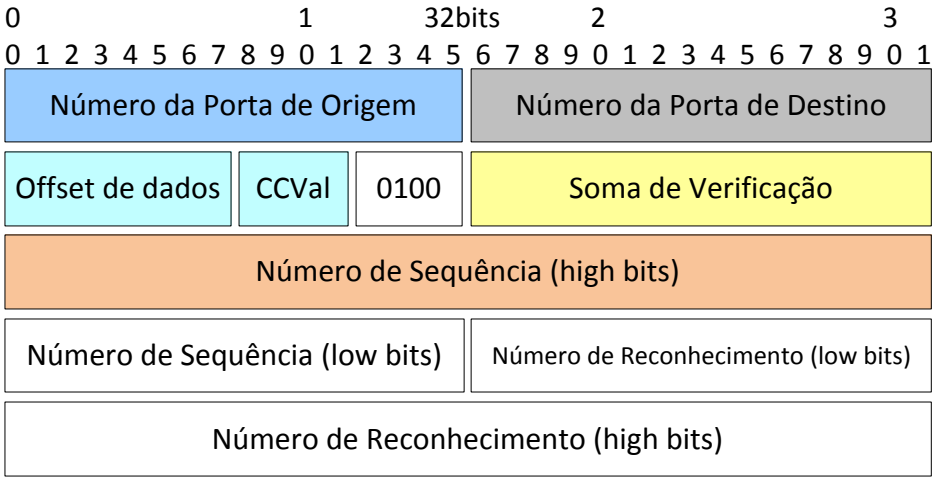


Figura 4.5: Cabeçalho do pacote GMTP-DataAck.

exemplo, um pacote GMTP-Ack pode ser enviado por um GMTP Relay para confirmar pacotes de definições de opções de uma conexão GMTP ou por um nó GMTP ao aceitar ser eleito para ser um GMTP Reporter.

GMTP-Elect, GMTP-ElectReply e GMTP-ElectAck

Os pacotes do tipo GMTP-Elect, GMTP-ElectReply e GMTP-ElectAck, números 5, 6 e 7 (0101_2, 0110_2 e 0111_2), respectivamente, são utilizados por um GMTP Relay ou por um cliente GMTP para tratar do processo de eleição de nós GMTP Reporters ou de promoções de clientes GMTP para se tornarem GMTP Relay. Quando um GMTP Relay assume seu papel de repassar o fluxo de dados em modo multicast para os clientes GMTP interessados, o mesmo precisa obter informações sobre o estado da rede. Para isto, um GMTP Relay cria

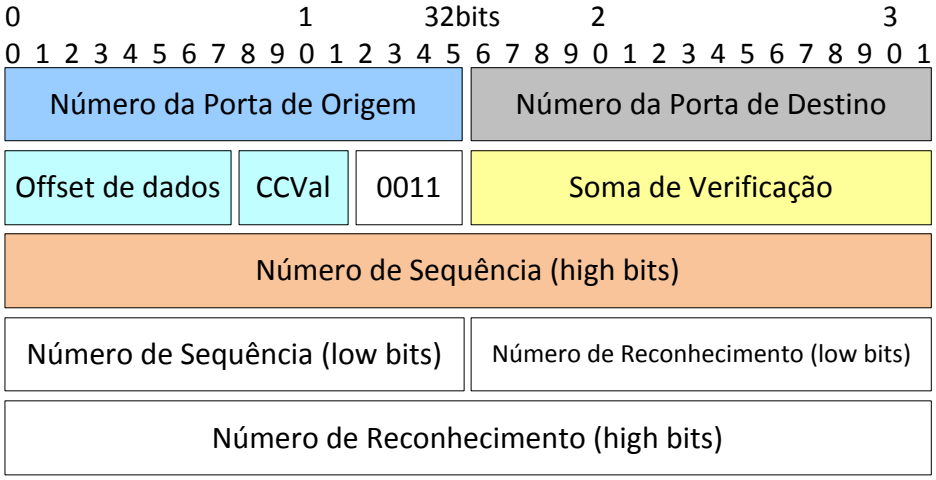


Figura 4.6: Cabeçalho do pacote GMTP-Ack.

um pacote do tipo GMTP-Elect e transmite no canal multicast. Quando um cliente GMTP recebe um pacote deste tipo, o mesmo pode se candidatar a um GMTP Reporter, enviando um pacote do tipo GMTP-ElectReply para o nó GMTP Relay que enviou o pacote GMTP-Elect. Como muitos nós GMTP podem receber um pacote GMTP-Elect, o nó GMTP Relay utiliza o pacote do tipo GMTP-ElectAck para confirmar a eleição apenas de um subconjunto de clientes GMTP.

Neste caso, para os pacotes GMTP-Elect, GMTP-ElectReply e GMTP-ElectAck, o cabeçalho genérico do GMTP ilustrado na Figura 4.1 passa a ter a forma dos cabeçalhos ilustrados nas Figuras 4.7, 4.8 e 4.9, respectivamente. Note que no pacote GMTP-Elect, os campos variáveis de 16 e 32 bits são utilizados para o GMTP Relay especificar um número de porta e um endereço IP para o qual um cliente GMTP enviará um pacote do tipo GMTP-ElectReply.

Outros pacotes: GMTP-RelayQuery, GMTP-RelayReply, GMTP-AdvConn, GMTP-CloseReq, GMTP-Close e GMTP-Reset

Os pacotes GMTP-RelayQuery, GMTP-RelayReply, GMTP-AdvConn, GMTP-CloseReq, GMTP-Close e GMTP-Reset tem funções e formatos similares aos outros pacotes anteriormente discutidos.

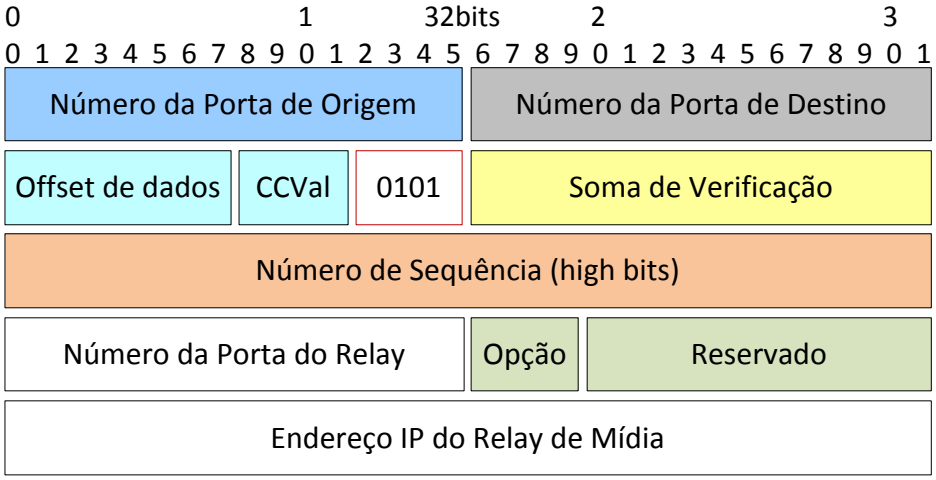


Figura 4.7: Cabeçalho do pacote GMTP-Elect.

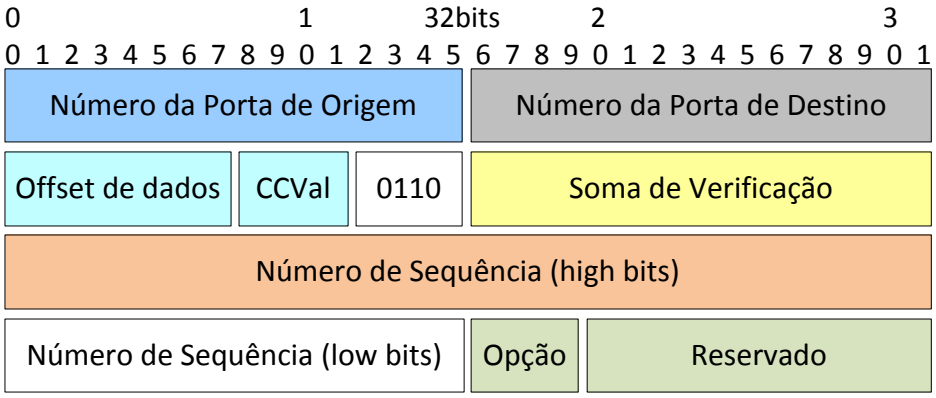


Figura 4.8: Cabeçalho do pacote GMTP-ElectReply.

4.2 Detalhamento do Processo de Conexão do GMTP

Como discutido na Seção 3.4, o processo de conexão do protocolo GMTP acontece em duas fases. A fase 1 ocorre quando não existe nenhum nó recebendo os dados desejados por um outro nó GMTP. Já a fase 2 ocorre quando existe um nó na rede local recebendo um fluxo de dados de interesse de um segundo nó interessado em também recebê-lo.

4.2.1 Fase 1

O primeiro pacote a ser utilizado neste processo é o GMTP-Request, apresentado na Figura 4.2, seguindo-se da forma como discutido na Seção 3.4.1. Um aspecto importante ainda não discutido é o tempo que um cliente deve esperar para receber um pacote do tipo GMTP-Response. Baseando-se em simulações de rede realizadas no contexto desse trabalho e consi-

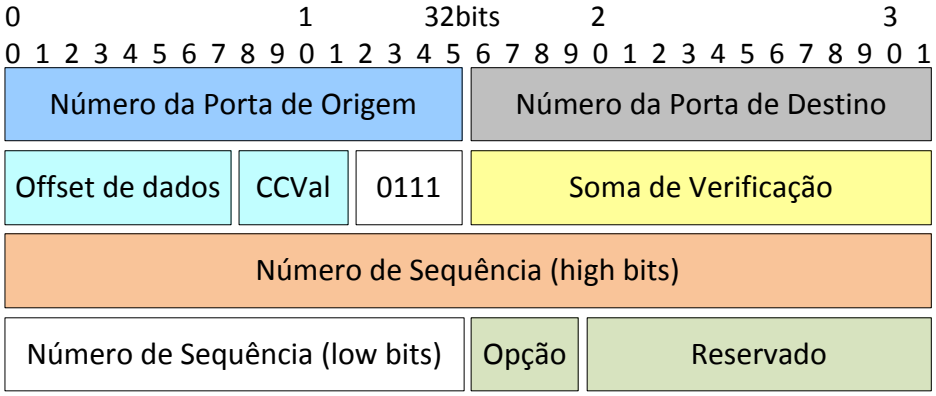


Figura 4.9: Cabeçalho do pacote GMTP-ElectAck.

derando os padrões de tecnologia de rede atualmente difundidas, constatou-se ser suficiente que um cliente espere por um GMTP-Response até no máximo 300 *ms*. Considerando-se uma rede local, este tempo é suficiente para que um relay receba um pacote GMTP-Request, processe-o e em seguida crie e envie um GMTP-Response de volta para o cliente.

Mesmo com este procedimento, é possível que no final do processo existam dois nós relays presentes na mesma rede local, o que significa duas conexões na mesma rede local recebendo o mesmo conteúdo de dados. Como discutido na Seção 1.1, este tipo de situação deve ser evitada ao utilizar protocolos orientados a conexão a fim de evitar o problema da tragédia dos comuns, principalmente considerando os cenários de aplicações estudados neste trabalho. Dito isto, deve-se garantir que existirá apenas um relay na rede local e, para garantir esta premissa, outra decisão foi tomada. Caso um cliente inicie uma conexão unicast com o servidor, mas receba um pacote GMTP-Response durante este tempo ou após o estabelecimento da conexão com o servidor, o mesmo deve encerrá-la e obter os dados da conexão multicast transmitidos pelo relay que o enviou o pacote GMTP-Response. Antes de iniciar o processo de encerramento de conexão, um cliente nessa situação deverá contactar primeiro o nó relay e começar a receber o fluxo de dados de interesse e em seguida este deverá parar de agir como nó relay.

No contexto deste trabalho, atualmente estão sendo avaliadas outras propostas para definição do tempo que um cliente deve esperar por uma resposta ao pedido de conexão enviado por ele. Porém, de acordo com uma série de simulações realizadas até o momento, o uso de um tempo fixo de 300 *ms* é suficiente e ao mesmo tempo simples de se implementar, pois não requer quaisquer cálculos extras, como por exemplo, cálculos baseados no valor do RTT

ou na quantidade de saltos entre o cliente e o servidor, práticas bastante adotadas por outros protocolos de transporte, como o TCP. Independente disto, no protocolo GMTP permite-se que uma aplicação cliente altere o tempo de espera padrão por um GMTP-Response através da própria API de *sockets* padrão BSD. Para isto, o desenvolvedor da aplicação deve utilizar a função *setsockopt* e alterar o valor da opção SO_SNDTIMEO, aumentando-se ou diminuindo-se o tempo padrão de espera como desejado.

Como o pacote GMTP-Request é transmitido na rede local com TTL igual a 1, o pacote GMTP-Request inicial não será roteado para a rede externa e apenas os nós da rede local o receberá. Note que se houvesse um GMTP Relay na rede local, este responderia com um pacote do tipo GMTP-Response, notificando o cliente de que o mesmo passará a transmitir dados multimídia em modo multicast relacionado à conexão desejada. Este procedimento está relacionado com a fase 2 do processo de estabelecimento de conexão do protocolo GMTP, a seguir discutida mais adiante.

Como exemplo de uma conexão GMTP na fase 1, suponha que um servidor de mídias ao vivo esteja respondendo por conexões *sockets* através do endereço IP 200.200.211.5 e porta 8900. Suponha também que um cliente com endereço IP 200.200.200.1 e número de porta de origem 53900 esteja interessado pelo fluxo de dados enviado por este servidor. Neste caso, o cliente deve enviar um pacote do tipo GMTP-Request para o canal de controle do GMTP com os campos endereço IP e número de porta do servidor de mídia preenchidos com os dados do *socket* do servidor em questão, ou seja, endereço IP 200.200.211.5 e número de porta 8900.

Na Figura 4.10, ilustra-se como os campos mais relevantes do cabeçalho do pacote GMTP-Request devem ser preenchidos para o caso do exemplo supracitado. Note que os valores estão preenchidos em decimal para facilitar o entendimento, mas na prática esses valores devem estar representados em binário. Note também que o campo IPPROTO do cabeçalho do IP deve ser preenchido com o valor 253. O valor para este campo é também regulado pela IANA, que definiu o valor 253 para protocolos experimentais, como é o caso do GMTP [3].

Quando um nó cliente GMTP se promove a relay, um outro procedimento deve ser executado por ele. Após estabelecer a conexão, o relay já começa a receber o fluxo de dados, mas em segundo plano, o relay deve continuar em busca de outro nó relay mais próximo a

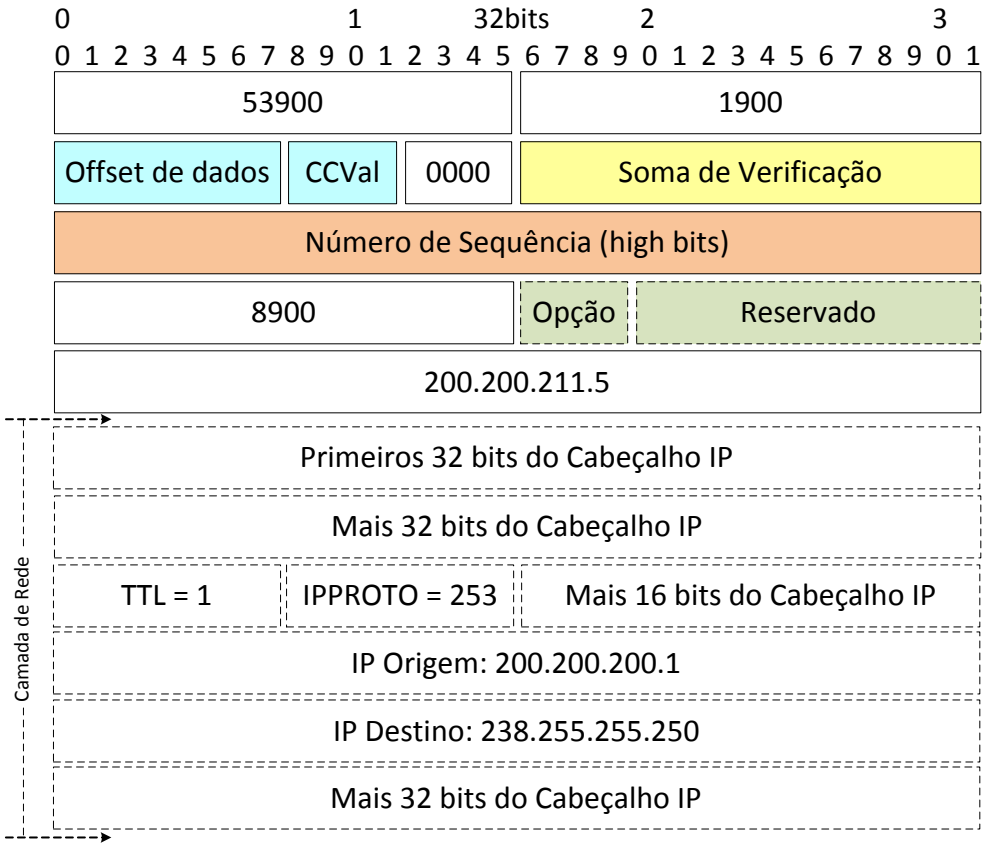


Figura 4.10: Exemplo do Cabeçalho do GMTP-Request e do IP.

ele. O objetivo desse procedimento é sempre evitar a sobrecarga de acessos simultâneos no servidor partindo de uma mesma rede.

Para encontrar um relay em outras redes, um cliente deve enviar o pedido de conexão utilizando o pacote do tipo GMTP-Request, da mesma forma que da primeira tentativa, porém com o valor de TTL igual a 2 em diante. Neste caso, se o roteador da rede do cliente estiver participando do grupo multicast do canal de controle do GMTP, o mesmo deverá repassar o pacote GMTP-Request para suas interfaces de rede de saída. Se houver algum relay correspondente ao pedido de conexão do cliente, este deverá responder ao cliente em modo unicast, através do endereço IP e porta do *socket* do cliente.

Note que o procedimento para descoberta de novos relays discutido anteriormente não funcionará em redes que utilizam NAT [85] e, para a atual versão do GMTP, este assunto está fora do escopo deste trabalho. Como o procedimento de encontrar um relay é baseado em busca por profundidade, não se pode conhecer facilmente o limite de saltos até encontrar um relay correspondente a conexão desejada pelo cliente. Desta forma, este procedimento

de busca deve ser limitado a no máximo 5 saltos, ou seja TTL igual a 5.

4.2.2 Fase 2

A fase 2 inicia quando um relay cria um socket de repasse multicast. Por exemplo, suponha um socket de repasse multicast no endereço IP 239.255.255.252 e número de porta 23456. Como ilustrado na Figura 4.11, o relay deve preencher os campos endereço IP e o número de porta do pacote do tipo GMTP-Response com os valores 239.255.255.252 e 23456, respectivamente. Note que o pacote do tipo GMTP-Response a ser transmitido pelo nó relay ao cliente em resposta ao pacote GMTP-Request deve ser transmitido em modo unicast, neste caso para o endereço IP 200.200.200.1 e na porta 53900. Note que neste exemplo o endereço IP do relay é o 200.200.200.2.

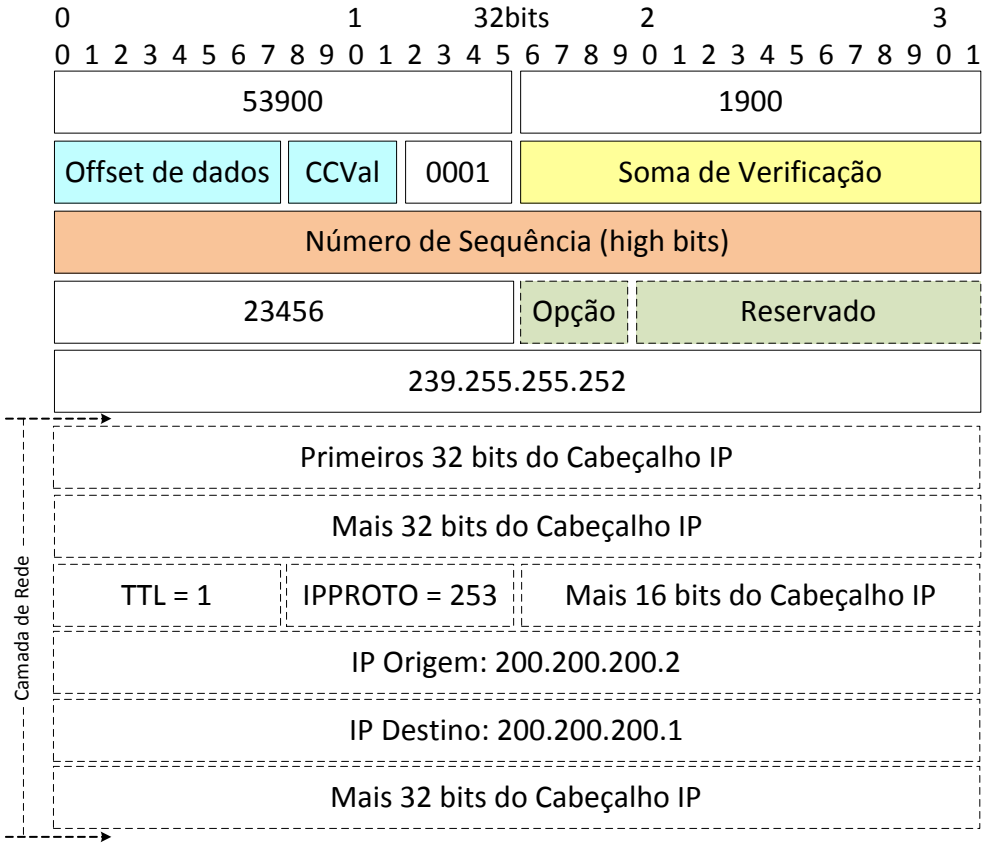


Figura 4.11: Exemplo do Cabeçalho do GMTP-Response.

Um aspecto importante nesse processo é que todos os pacotes GMTP-Response e GMTP-Ack utilizados no processo de conexão do GMTP, transmitidos através do canal de controle,

devem ser transportados de forma confiável, ou seja, com o uso de confirmação de recebimento utilizando o pacote do tipo GMTP-Ack e retransmissão caso pacotes desse tipo sejam perdidos.

Para os casos em que um cliente encontre um relay localizado fora da sua rede local, o relay deve iniciar um *socket* unicast e repassar os dados recebidos do servidor para o cliente em questão, criando-se portanto um *socket* para o canal de repasse. Para que o cliente saiba dessa decisão, o relay deve enviar um pacote do tipo GMTP-Response com os campos endereço IP e número de porta do relay preenchidos com as informações do canal de repasse, como ilustrado na Figura 4.12. Nesse pacote, o primeiro bit do campo *opção* deve está ativado para sinalizar ao cliente que a transmissão é unicast e não multicast.

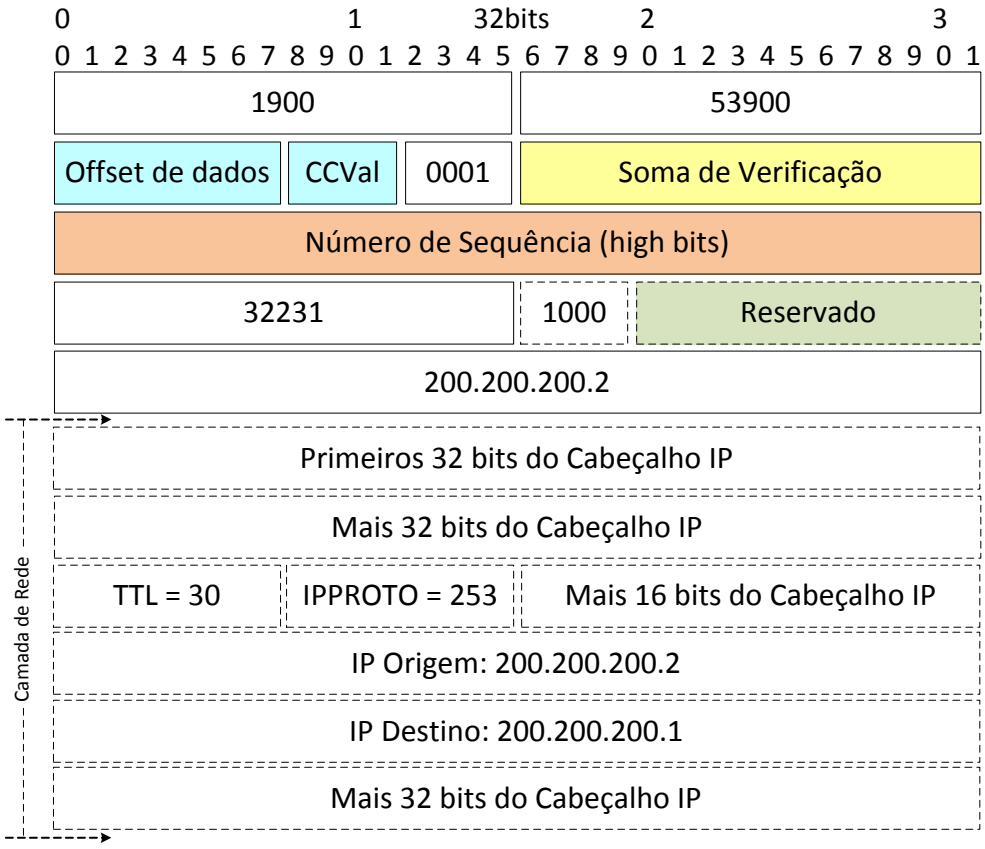


Figura 4.12: Exemplo do Cabeçalho do GMTP-Response quando o relay não está na mesma rede do cliente.

4.2.3 Conexão Rápida

O processo de conexão do GMTP requer que o cliente envie um pacote do tipo GMTP-Request para o canal de controle. Este procedimento objetiva fazer com que o relay tenha conhecimento dos nós GMTP interessados em receber o fluxo de dados repassado pelo relay e, com este conhecimento, permitir que o relay regule a taxa de transmissão a fim de controlar o congestionamento da rede. O problema é que o processo de conexão do GMTP pode demorar devido as tentativas de busca para encontrar um nó relay. Sabendo-se disso, no GMTP adicionou-se um mecanismo que permite um cliente estabelecer uma conexão de forma mais rápida.

A conexão rápida do GMTP é opcional e funciona da seguinte forma. Quando um cliente GMTP se torna um nó relay e começa a enviar dados utilizando o pacote do tipo GMTP-Data, o mesmo pode anunciar no canal de controle suas conexões ativas e qual canal de repasse está sendo utilizado. Neste caso, o relay utiliza o pacote do tipo GMTP-AdvConn para anunciar, através do canal de controle, suas conexões de repasse ativas e clientes interessados em obter o conteúdo multimídia correspondente pode passar a receber pacotes de dados no canal especificado no anúncio do relay. O anúncio do conexão de repasse deve ser enviado a cada $30\,s$.

Na Figura 4.13, ilustra-se o cabeçalho do pacote GMTP-AdvConn para o caso em que um nó relay tem uma conexão de repasse na porta 32231 através do endereço IP 200.200.200.2.

4.3 Algoritmo de Controle de Congestionamento Multicast

Na Seção 3.6.2, apresentou-se uma visão geral do algoritmo de controle de congestionamento para transmissões multicast (GMTP-MCC) empregado no GMTP. Nesta seção, apresenta-se detalhes do funcionamento deste algoritmo, iniciando-se com algumas considerações. Diante das adaptações no algoritmo original do TFRC mencionadas na Seção 3.6.2, para definir um algoritmo de controle de congestionamento em transmissões multicast, deve-se considerar alguns requisitos que serão discutidos a seguir.

1. Uma equação para controle de congestionamento deve ser definida em função de parâmetros obtidos sobre a rede, tais como a taxa dos eventos de perda e o RTT.

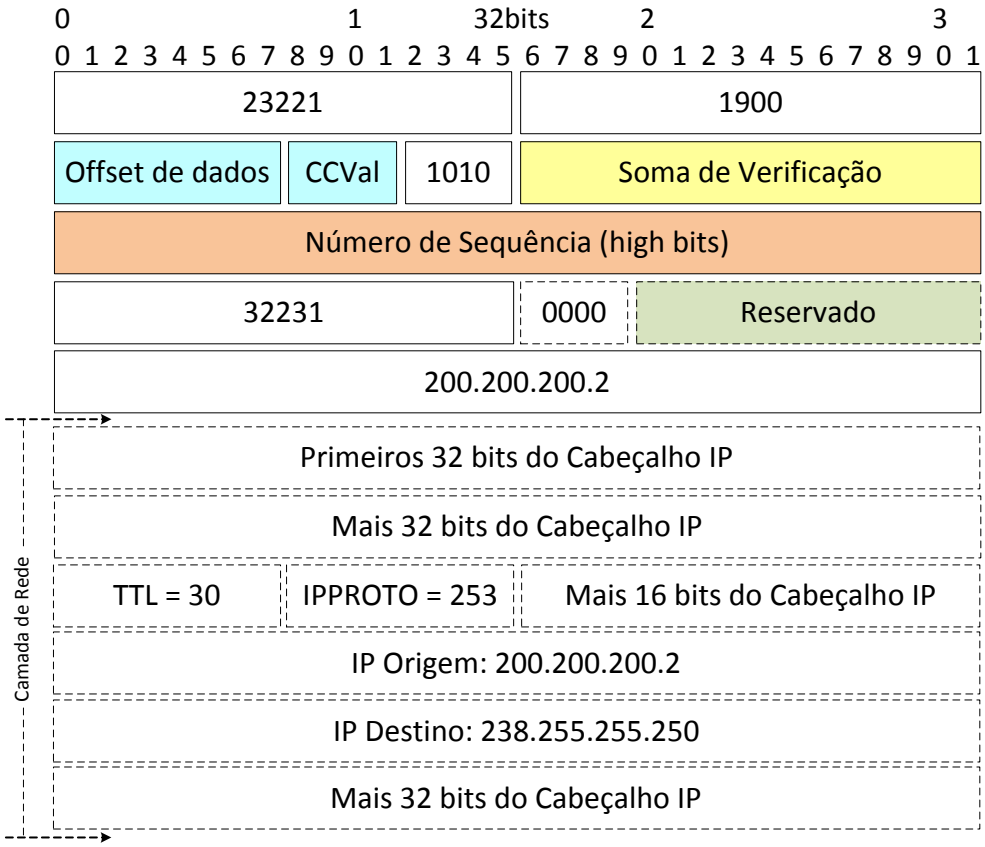


Figura 4.13: Exemplo do Cabeçalho do GMTP-AdvConn para anúncio de conexão de repasse.

2. Como cada receptor deve medir a taxa dos eventos de perda, deve-se definir um filtro que represente uma medição estável da taxa de transmissão para as condições atuais da rede, mas que o algoritmo seja sensível quando tais condições mudarem.

3. Como cada receptor deve estimar o RTT entre ele e o transmissor, deve-se elaborar uma forma efetiva de calcular este parâmetro sem excessivas trocas de dados de controle.

4. Como cada receptor deve calcular a taxa de transmissão e enviá-la para o transmissor, deve-se definir um filtro para determinar qual é a melhor taxa de transmissão dentre as diversas taxas de transmissão enviadas pelos nós receptores.

5. Como cada receptor deve enviar o cálculo da sua taxa de transmissão, deve-se determinar a frequência de envio desse relatórios para o nó transmissor. Essa taxa não pode ser muito alta para não causar o problema da *explosão de feedbacks*, porém, se essa

taxa for muito baixa, o GMTP-MCC irá demorar para reagir às mudanças do estado da rede.

Para as discussões a seguir, considere:

1. $\hat{c}$, uma transmissão GMTP;
2. $\hat{s}$, o nó servidor de $\hat{c}$;
3. $|X|$, a quantidade de elementos de um conjunto X de nós;
4. *link*(e_1, e_2), uma função que relaciona o elemento e_1 a um e_2, com e_1 e e_2 pertencente a dois conjuntos diferentes e e_1 não podendo estar relacionado com mais de um elemento;
5. C, o conjunto de clientes tal que $C = \{c_i / c_i$ é um nó participante de $\hat{c}\}$;
6. S, o conjunto de relays tal que $S = \{s_i / s_i$ é um relay$\}$ e $S \subset C$;
7. W, o conjunto de todos os clientes não relays, ou seja, $W = C - S$;
8. W_i, um subconjunto i de W tal que $W_i = \{\hat{w}_i /$ *link*($\hat{w}_i$, s_i)$\}$. Ou seja, $\bigcap_{i=1}^{n} W_i = \emptyset$, para n correspondendo a quantidade de subconjuntos derivados de W. Isto significa que um cliente $\hat{w}_i$ não pode pertencer a dois subconjuntos W_i e W_j, com $i \neq j$, $1 \leq i \leq n$ e $1 \leq j \leq n$. Na prática, um subconjunto W_i é um grupo multicast e um cliente $\hat{w}_i$ não pode pertencer a dois grupos multicast distintos para receber o mesmo conteúdo, uma vez que cada grupo multicast está relacionados apenas com um relay s_i;
9. R, o conjunto de todos os clientes reporters em uma transmissão $\hat{c}$;
10. R_i, um subconjunto i de R tal que $R_i = \{\hat{r}_i /$ link($\hat{r}_i$, s_i)$\}$ e $R_i \subset W_i$ ou $R_i \subseteq W_i$. Ou seja, $\bigcap_{i=1}^{n} R_i = \emptyset$, para n correspondendo a quantidade de subconjuntos derivados de R. Isto significa que um reporter $\hat{r}_i$ não pode pertencer a dois subconjuntos R_i e R_j, com $i \neq j$, $1 \leq i \leq n$ e $1 \leq j \leq n$. Na prática, isto quer dizer que um reporter $\hat{r}_i$ não pode enviar relatórios sobre congestionamento para mais de um relay s_i.

4.3.1 Determinando a Taxa de Transmissão para s_i

A taxa de transmissão de um nó s_i é definida por T_{s_i} e calculada através da Equação 4.1 em função de $T_{\hat{r}_i}$. No GMTP-MCC, cada elemento de R_i, ou seja, cada reporter $\hat{r}_i$ deve calcular seu valor para $T_{\hat{r}_i}$ e enviá-lo para seu s_i correspondente. O relay s_i, por sua vez, utiliza-se da Equação 4.1 para calcular sua taxa de transmissão e a utilizará para transmitir dados em modo multicast para um grupo de clientes W_i. O cálculo para $T_{\hat{r}_i}$ é feito através da Equação 3.1, em função da taxa de perda p percebida em $\hat{r}_i$ e pelo valor do RTT t_{rtt} entre um certo $\hat{r}_i$ e seu respectivo s_i.

$$T_{s_i} = \frac{\sum_{i=1}^{n} T_{\hat{r}_i}}{n} \tag{4.1}$$

No caso do GMTP-MCC, a estratégia é que o valor da taxa de transmissão para um s_i seja tão próximo ao valor de T quanto possível, tornando-o um algoritmo *TCP-Friendly*. Um fluxo de dados é considerado *TCP-Friendly* quando este não degrada a taxa de transmissão de um fluxo de dados TCP mais do que outro fluxo TCP degradaria quando começasse a ser transmitido na rede.

Como o valor para T é uma aproximação da taxa de transmissão que um fluxo de dados TCP obteria e s_i segue esta condição, então se estiverem sendo transmitidos fluxos de dados TCP concorrentes com fluxos de dados GMTP, o valor calculado para a taxa de transmissão de um fluxo de dados s_i será equivalente aos valores das taxas de transmissão dos fluxos de dados transmitidos através do TCP. Essa preocupação é muito importante no contexto do GMTP porque atualmente o TCP é o protocolo mais utilizado na Internet, sendo de fundamental importância tornar os fluxos de dados transmitidos com o GMTP equinimes com relação aos fluxos de dados transmitidos com TCP.

Desta forma, seja $T_{\hat{r}_{max}}$ o valor da maior taxa de transmissão recebida por um s_i e enviada por algum $\hat{r}_i$, pertencente a R em uma transmissão $\hat{c}$. No GMTP-MCC, partiu-se do pressuposto de que se nenhum nó s_i ou $\hat{s}$ exceder o valor de $T_{\hat{r}_{max}}$ em uma conexão $\hat{c}$, então o fluxo GMTP será *TCP-Friendly*.

Teoricamente, o GMTP seria um protocolo *TCP-Friendly* se $T_{\hat{r}_{max}} = T_{s_i}$ em s_i. Porém, optou-se por utilizar a média aritimética dos valores $T_{\hat{r}_i}$ (Equação 4.1) porque na prática diversos fatores podem alterar drasticamente o estado da rede no instante em que se utilizar

$T_{\hat{r}_{max}}$. Com esta decisão, define-se uma margem de segurança evitando que o GMTP-MCC alcance o limite superior para o valor da taxa de transmissão de um fluxo transmitido com TCP. Além disso, a média aritimética suaviza os valores subsequentes para s_i, mesmo se algum $\hat{r}_x$ envie para s_i um valor $T_{\hat{r}_x}$ muito alto ou muito baixo com relação aos demais valores $T_{\hat{r}_i}$.

4.3.2 Ajuste da Taxa de Transmissão

O ajuste da taxa de transmissão T_{s_i} de um relay s_i ocorre periodicamente de acordo com os valores de cada taxa de transmissão $T_{\hat{r}_i}$ calculada por um nó $\hat{r}_i$. Todo nó $\hat{r}_i$ deve enviar ao nó transmissor o cálculo de $T_{\hat{r}_i}$ e s_i deve ajustar sua taxa de transmissão para o valor obtido através da Equação 4.1.

Sendo assim, a frequência que s_i deve ajustar sua taxa de transmissão T_{s_i} é determinada por cada nó em R_i. Toda vez que $\hat{r}_x$ enviar para s_i seu novo valor $T_{\hat{r}_x}$, s_i deve recalcular sua taxa de transmissão T_{s_i} utilizando a Equação 4.1, mantendo-se o restante dos valores inalterados de $T_{\hat{r}_i}$ neste cálculo.

Note que um nó $\hat{r}_x$ deve enviar um novo valor para $T_{\hat{r}_x}$ todas as vezes que um novo intervalo de perda for determinado. 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. Na Seção 4.3.3, explica-se com mais detalhes como funciona o processo que determina um intervalo de perda.

Uma outra observação com relação ao ajuste da nova taxa de transmissão de s_i está relacionado ao fato de um nó $\hat{r}_i$ se desconectar ou perder repentinamente sua conexão. Caso isto aconteça com algum nó $\hat{r}_x$, seu valor $T_{\hat{r}_x}$ deverá ser desconsiderado no cálculo da nova taxa de transmissão T_{s_i}. Existem duas formas que um relay s_i pode perceber a desconexão de um ou mais nós $\hat{r}_i$. A primeira forma é quando um nó $\hat{r}_x$ envia explicitamente um pedido de desconexão para s_i, tal processo é discutido na Seção 3.7.1, ao passo que a segunda forma é quando um contador de tempo de manutenção de conexão, mantido pelo relay s_i se expira, tal processo é discutido na Seção 3.7.2. Com esta medida, evita-se utilizar um taxa de transmissão T_{s_i} incorreta, portanto não correr o risco de utilizar uma taxa de transmissão não condizente com o estado atual da rede.

4.3.3 Taxa de Eventos de Perda p

A taxa de eventos de perda p, definido na Equação 3.1, é determinada por cada GMTP Reporter $\hat{r}_i$ de forma similar ao TFRC. No caso do GMTP-MCC, cada $\hat{r}_i$ agrega as perdas de pacotes que ocorrem dentro de um evento de perda, definido por uma ou mais perdas de pacotes no espaço de tempo de um RTT. O número de pacotes entre eventos de perdas consecutivos é chamado de intervalo de perda. Para o cálculo de p, utiliza-se a média dos tamanhos dos intervalos de perda, calculada através da média ponderada dos m mais recentes intervalos de perdas $l_k, \ldots, l_{k-m+1}$ seguindo a Equação 4.2. O conjunto de todos os intervalos de perda é chamado de *histórico de perdas*.

$$l_{avg} = \frac{\sum_{i=0}^{m} w_i \times l_{k-i}}{\sum_{i=0}^{m} w_i} \tag{4.2}$$

Os pesos w_i são escolhidos de tal forma que os intervalos de perdas mais recentes recebem pesos mais altos, decrescendo-os gradualmente até 1 para os intervalos de perdas mais antigos. Por exemplo, para 8 intervalos de perda, pode-se utilizar os pesos $w = [5, 5, 5, 5, 4, 3, 2, 1]$. Ao utilizar-se da média ponderada para o cálculo da média dos tamanhos dos intervalos de perda, obtem-se mudanças mais suaves para o valor de l_{avg} à medida que os tamanhos dos intervalos de perdas se tornam mais antigos. Para grandes valores de m, obtem-se mudanças mais suaves para p ao longo do tempo, porém isto também reduz a capacidade de resposta e portanto a equidade do protocolo. No TFRC, recomenda-se utilizar valores de m entre 8 e 32 e por este motivo no GMTP é considerada esta recomendação. A Equação 4.2 é definida na RFC 3448 [40] e foi mantida no GMTP. O protocolo DCCP também utiliza essa mesma abordagem no algoritmo de controle de congestionamento CCID-3 [55].

Uma vez definido como determina-se a média dos tamanhos dos intervalos de perda, a taxa dos eventos de perda p é definido pelo inverso de l_{avg}, definido na Equação 4.3. Como um intervalo de perda é definido em função do número de pacotes entre de eventos de perdas consecutidos, o mais recente evento de perda não pode influenciar na taxa do evento de perda, por isto utilizou a função *max* no denominador da Equação 4.3.

$$l_{avg} = \frac{1}{max(l_{avg}(k), l_{avg}(k-1))} \tag{4.3}$$

4.3.4 Cálculo do RTT

O cálculo do RTT realizado no GMTP-MCC é feito apenas pelos nós reporters e funciona da seguinte forma. Um nó $\hat{r}_i$ transmite ao seu respectivo relay $\hat{s}_i$ um pacote de controle e inicia um marcador de tempo. Ao receber uma resposta do nó relay $\hat{s}_i$, o nó reporter $\hat{r}_i$ pára o marcador de tempo e utiliza este tempo chamado de $RTT_{instant}$ para calcular o valor do próximo RTT de acordo com a Equação 4.4.

$$RTT = \beta \times RTT_{instant} + (1 - \beta) \times RTT \tag{4.4}$$

Note que no GMTP-MCC não se utiliza o valor de RTT instantâneo ($RTT_{instant}$) como o valor do RTT, mas sim utiliza-se de um mecanismo para suavizar as mudanças do RTT ao longo do ciclo de vida de uma conexão. Desta forma, procura-se evitar que valores absurdos de $RTT_{instant}$ – muito baixos ou muito altos com relação aos valores medidos anteriormente – influenciem demasiadamente na taxa de transmissão $T_{\hat{s}_i}$.

O mecanismo mencionado anteriormente para suavizar as medições do valor de RTT é chamado de Médias Móveis Exponencialmente Ponderadas ou *Exponentially Weighted Moving Average* (EWMA). O EWMA foi primeiramente utilizado para índices financeiros de medição de risco, onde a série de retornos diários com n observações é ponderada por um fator de decaimento. As observações mais recentes no tempo são ponderadas com um peso maior que as observações mais antigas. O peso de uma observação decai exponencialmente com n. Em seguida, utilizou-se EWMA em medições de tempo do RTT em protocolos como o TCP. Como trata-se de uma estratégia conhecida para medição de RTT, no GMTP-MCC manteve-se o mesmo mecanismo, principalmente por já ter sido exaustivamente testado e utilizado. Tanto no TCP quanto no caso do GMTP, utiliza-se $\beta = 0.25$ para o cálculo do valor de RTT através da Equação 4.4 [61].

Porém, diferentemente do mecanismo de medição de RTT no TCP e no TFRC, os nós reporters $\hat{r}_i$ são os responsáveis pela medição do RTT e não o nó transmissor. Um aspecto importante na medição do RTT está relacionado com o início de uma conexão GMTP, pois não se sabe o valor para $RTT_{instant}$ até o final do processo de estabelecimento de uma conexão. Nesse caso, deve-se utilizar um valor consideravelmente alto para evitar taxas de transmissões $T_{\hat{s}_i}$ muito maiores do que a rede tem capacidade de suportar. No GMTP,

utiliza-se o valor inicial de $RTT_{instant}$ igual a 150 *ms*. Quando um nó $\hat{r}_i$ enviar um pedido de conexão utilizando o pacote do tipo GMTP-Request, o mesmo deve realizar a sua primeira medição do valor de $RTT_{instant}$, iniciando-se o marcador de tempo para o cálculo do RTT quando enviar o primeiro GMTP-Request e parando-o quando receber o pacote do tipo GMTP-Response. Em seguida, deve-se acionar o mecanismo de cálculo de $T_{\hat{r}_i}$, caso o respectivo nó $\hat{r}_i$ seja eleito como reporter.

4.4 Considerações sobre Implementação

O núcleo do protocolo GMTP foi implementado no simulador NS-2 e a versão atual do protocolo já permite a execução de transmissão de dados. Com o desenvolvimento preliminar do protocolo GMTP no simulador NS-2, permitiu-se a execução de diversas simulações a fim de avaliar o comportamento do protocolo considerando diversas configurações.

Em linhas gerais, a implementação no referido simulador de rede permite a comunicação entre os nós através dos modos de transmissão multicast e unicast. Foram implementados os tipos de pacotes do GMTP e os processos de estabelecimento de conexão, incluindo o processo de uso de nós relays, troca de dados em modos multicast e unicast e os algoritmos para controle de congestionamento, incluindo o uso de nós reporters.

Contudo, não foi implementado o arcabouço de extensão para permitir o desenvolvimento de novos algoritmos para o processo de conexão, descoberta e seleção de nós, adaptação de fluxo de dados e tolerância a falhas. Tal implementação será feita no núcleo do sistema operacional Linux, juntamente com todos os mecanismos básicos para o funcionamento do GMTP.

A proposta é de implementar um arcabouço de extensão para as funções previstas no GMTP, permitindo-se o desenvolvimento e adição de novos algoritmos para as funcionalidades supracitadas, de modo que torne o GMTP flexível para permitir que qualquer aplicação os utilizem.

Na prática, um cenário desejado para essa proposta de implementação do GMTP é que o desenvolvedor possa configurar quais algoritmos deseja utilizar em sua aplicação, permitindo-se que estes sejam alterados em modo de execução da da mesma. Neste caso, suponha um algoritmo de descoberta de nós chamado *DN-1*, um algoritmo para controle

de congestionamento chamado *CC-2*, um algoritmo de tolerância a desconexão *TD-3*; um algoritmo de adaptação de fluxo *AF-4* e um algoritmo para reciprocidade *R-2*. As implementações de tais algoritmos serão feitas na camada de transporte, acoplando-as em forma de módulos do sistema operacional ao protocolo GMTP. Em seguida, as aplicações podem selecionar quais algoritmos melhor se adequa as suas necessidades, com o GMTP sendo responsável por:

1. carregar o conjunto de algoritmos *A = {DN-1, CC-2, TD-3, AF-4, R-2}*;
2. definir os parâmetros iniciais para cada um dos algoritmos em A, definidos pela aplicação;
3. executar funções preliminares para ajustes iniciais, tais como informar aos nós participantes de uma transmissão quais dos algoritmos estão sendo utilizados (o conjunto *A*) e quais outros estão disponíveis;
4. executar os algoritmos em momentos apropriados de acordo com os eventos de rede, notificando a aplicação caso necessário e desejável pela aplicação;
5. descarregar os algoritmos quando não forem mais necessários e informar aos nós parceiros.

Desta forma, um nó servidor poderá solicitar que seus nós clientes carreguem um determinado módulo, dependendo da sua disponibilidade nos nós clientes. Neste caso, o GMTP controlará todo o processo de carregamento dos mesmos.

Com isso, o GMTP se tornará um protocolo extensível que gerencia quais algoritmos devem ser executados em cada ponto de extensão. Esses algoritmos podem ser adicionados ao protocolo através de módulos do sistema operacional, carregáveis utilizando-se comandos como o *modprobe* (no Linux, por exemplo) e manipulados (passagem de parâmetros) pela aplicação através de uma API de programação, por exemplo, utilizando-se as primitivas *setsockopt()* e *getsockopt()* da especificação *BSD Socket API* [106].

Para que as aplicações possam utilizar o GMTP, o protocolo deve ser compatível com todas as funções prevista na especificação *BSD Socket API*, são elas: *socket()*, *bind()*, *listen()*, *connect()*, *accept()*, *send()*, *recv()*, *write()*, *read()*, *sendto()*, *recvfrom()*, *close()*, *select()*, *setsockopt()*, *getsockopt()* e *pull()*.

Outros trabalhos podem ser desenvolvidos para tornar o GMTP compatível com o padrão de *sockets* do sistema operacional Windows, conhecido pelo nome de *winsock*. Todavia, por ser um sistema operacional de código fechado, a implementação do GMTP só será possível após sua padronização em forma de RFC.

4.5 Sumário do Capítulo

Neste capítulo, apresentou-se uma visão técnica do GMTP. Discutiu-se o uso e as aplicabilidades dos diferentes tipos de pacotes do GMTP, onde foram abordados exemplos para a execução das funcionalidade de tal protocolo.

Em seguida, apresentou-se discussões acerca do processo de estabelecimento de conexão do GMTP e uma formalização do algoritmo para controle de congestionamento em modo de transmissão multicast empregado no GMTP através do uso de teoria de conjuntos.

Por fim, apresentou-se algumas considerações importantes quanto a implementação do GMTP em sistemas operacionais, tais como o Linux (*BSD Socket API*) e Windows (*Winsock API*).

Capítulo 5

Métodos, Simulações e Experimentos

Neste capítulo é descritos o método estatístico utilizado para a obtenção das métricas estabelecidas para analisar o desempenho do protocolo GMTP e as metodologias adotadas para obtenção dos valores finais para cada uma das métricas obtidas.

Para isto, apresenta-se um método estatístico baseado na teoria da probabilidade, que possibilita calcular a quantidade de ensaios necessários para um determinado tratamento de simulação e assim obter um nível de confiança de 95 % nos valores apresentados. Com este método, foi possível realizar comparações quanto ao desempenho do GMTP frente a outros protocolos tradicionais, como o DCCP e o TCP. Tais discussões comparativas são apresentadas no Capítulo 6.

5.1 Tratamentos

Neste trabalho, considerou-se a análise do protocolo GMTP em confronto com o protocolo DCCP e o TCP.

De acordo com os objetivos deste trabalho, considera-se desnecessária uma análise de desempenho do GMTP em confronto com protocolos tradicionais como o UDP. Isto porque em diversos trabalhos anteriores, inclusive na dissertação de mestrado do autor desta proposta de tese [23], já foram apresentadas avaliações comparativas entre o DCCP, o TCP e o UDP. No estado atual, procura-se avaliar o comportamento do GMTP com relação a sua capacidade de escalabilidade diante de grandes quantidades de nós receptores e de sua equidade

diante de múltiplos fluxos de dados. Por este motivo, descartou-se a necessidade de avaliar o desempenho do GMTP em confronto com o UDP, uma vez que este último não implementa qualquer solução para controle de congestionamento, compartilhamento de conexão etc.

Para os tratamentos que apresentam resultados do confronto entre o protocolo GMTP e o DCCP, cujo principal objetivo é apresentar a capacidade de escalabilidade de ambos os protocolos, as simulações foram executadas de forma isolada, primeiramente o DCCP e em seguida o GMTP.

O tempo de duração da execução de cada ensaio foi de 400 s, onde cada ensaio foi repetido a quantidade de vezes necessárias até atingir um intervalo de confiança de 95 %, de acordo com as definições estabelecidas na Seção 5.3.

Definiu-se dois tratamentos, um com confrontos GMTP vs. DCCP vs. TCP (Tratamento 1) e o outro com confrontos entre GMTP vs. DCCP (Tratamento 2). O objetivo do Tratamento 1 é averiguar a capacidade de convergência e equidade dos fluxos transmitidos utilizando o GMTP analizando a vazão obtida por esses fluxos. Por outro lado, o objetivo para o Tratamento 2 é averiguar a escalabilidade do GMTP no que diz respeito a quantidade de nós receptores interessados em um mesmo fluxo de dados transmitido por um nó servidor. Esta avaliação foi realizada aumentando-se a quantidade de nós receptores gradativamente em uma transmissão de vídeo $1 \rightarrow n$, coletando-se valores para as métricas de vazão, carga de dados transmitida e perda de pacotes, atraso e qualidade do vídeo transmitido. A metodologia adotada neste trabalho para obtenção de cada uma das métricas mencionadas anteriormente será explicada na Seção 5.2.

Os Tratamentos 1 e 2 foram executados em simulações de rede no NS-2 [22] cuja topologia da rede foi definida como uma árvore binária completa, segundo a Figura 5.1. Além disso, alguns fatores foram pré-definidos e são descritos a seguir.

- Número de computadores receptores por rede: 10
- Largura de banda da rede local: 100 *Mbps*
- Latência da rede local: 1 *ms*
- Largura de banda do backbone: 100 *Mbps*
- Latência do backbone: 5 *ms*

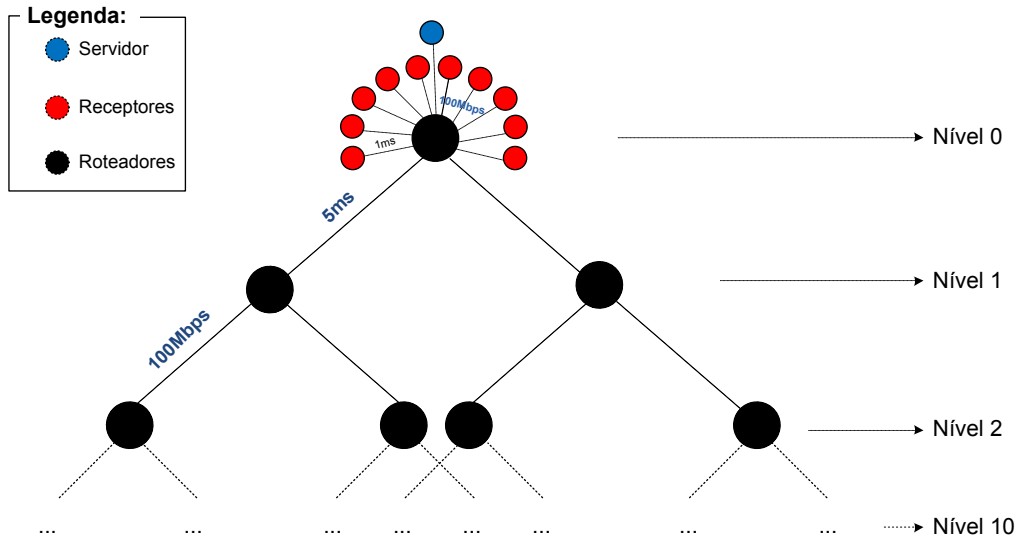


Figura 5.1: Topologia da rede definida para as simulações realizadas. Cada rede é representada por um roteador e com 10 nós em cada rede.

- Tamanho da fila dos roteadores do backbone: 3000 pacotes
- Duração da simulação: 400 *s*

De acordo com a topologia definida, cada nó da árvore representou um roteador, cada um com 10 nós receptores TCP, DCCP e/ou GMTP conectados a ele. Para o caso do Tratamento 2, os ensaios foram executados à medida em que se aumentava o nível da árvore. Por exemplo, r ensaios foram executados para 10 nós receptores e 1 roteador, pois o nível da árvore *L* foi igual a 0 (zero); em seguida r outros ensaios foram executados utilizando-se 30 nós receptores e 3 roteadores, pois L=1; em seguida, outros r ensaios foram executados utilizando-se 70 nós receptores e 7 roteadores, pois L=2; e assim por diante até L=9, quando utilizou-se 10.230 nós receptores e 1.023 roteadores. Deve-se utilizar $n = 2^{L+1} - 1$ para se obter a quantidade *n* de roteadores dado um nível *L* da topologia de rede utilizada.

Os fluxos de dados foram transmitidos da seguinte forma: um nó localizado na raiz da árvore transmitiu o mesmo conteúdo multimídia para todos os outros nós conectados à rede, simulando uma típica transmissão multimídia $1 \rightarrow n$ e um tráfego de comportamento equivalente a um vídeo MPEG-2.

5.2 Métricas Selecionadas e Métricas Derivadas

Com relação aos tratamentos que envolveram transmissões de fluxos de dados dos protocolos TCP e GMTP, foi estudada apenas a métrica de vazão, ao passo que tratamentos que envolveram a análise de desempenho do GMTP com relação do protocolo DCCP, foram analisadas as métricas de vazão, perda de pacote e a latência, que através desta última foi possível calcular o *jitter* médio para uma determinada transmissão. Além disso, atráves da vazão e da quantidade de pacotes perdidos, pode-se obter a carga efetiva de dados transmitidos.

Para cada métrica selecionada, foram coletados seus valores instantâneos e para cada uma delas algumas considerações são discutidas a seguir.

5.2.1 Vazão Média, Carga Efetiva Média, Latência Média e Jitter

Para um determinado tratamento, a média final da vazão e da carga efetiva transmitida pelo TCP foi obtida através da média aritmética das médias das vazões obtidas em cada ensaio *r*, ou seja, através das Equação 5.1 e 5.2, onde *n* é o total de ensaios de um determinado tratamento. Assim, temos:

$$\mu_{vazaotcp} = \frac{\sum_{r=1}^{n} vazão\_média_r}{n} \tag{5.1}$$

$$\mu_{cargatcp} = \frac{\sum_{r=1}^{n} carga\_média_r}{n} \tag{5.2}$$

No entanto, para obter as médias da vazão e carga efetiva dos fluxos GMTP, o procedimento foi um pouco diferente. Considerando que os fluxos GMTP foram sempre iniciados 50 *s* após o fluxo TCP, é preciso definir um mecanismo que não penalize GMTP, já que eles deixaram de transmitir por 50 *s*. Assim, dado que:

$$\mu_{vazãogmtp} = \frac{\sum_{r=1}^{n} vazão\_média_r}{n}$$

onde $vazão\_média_r$ é obtida através da média aritmédia das vazões em cada segundo de cada ensaio, tem-se que a vazão para os fluxos do protocolo GMTP é obtida através da Equação 5.3.

$$\mu_{vazao-final-GMTP)} = \mu_{vazãogmtp} + S \times (\frac{\mu_{vazãogmtp}}{T}) \tag{5.3}$$

Onde,

- S, o tempo de atraso para iniciar os fluxos UDP ou DCCP ($S = 50\,s$);
- T, o tempo total do ensaio ($T = 400\,s$).

Assim, as médias são normalizadas para não penalizar nenhum dos protocolos, com base na Equação 5.3.

De forma equivalente, pode-se obter a carga média efetivamente transmitida e da latência média. Note que para o confronto GMTP $\times$ DCCP, a vazão média e carga média são obtidas através das Equações 5.1 e 5.2, respectivamente.

Jitter

O cálculo para obter o valor médio do *jitter* para um fluxo transmitido é bastante similar ao cálculo da vazão média. Este valor pode ser obtido através da Equação 5.5. Esta equação foi obtida da seguinte forma:

$$\mu_{\textit{jitter-parcial-gmtp}} = \frac{\sum_{r=1}^{n} \textit{jitter\_médio}_r}{n} \tag{5.4}$$

onde,

$$\textit{jitter\_médio}_r = \frac{\sum_{k=1}^{Q} V_k}{Q}$$

Logo,

$$\mu_{\textit{jitter-final-gmtp}} = \mu_{\textit{jitter-parcial-gmtp}} + S \times (\frac{\mu_{\textit{jitter-parcial-gmtp}}}{T}) \tag{5.5}$$

Sendo,

- Q, quantidade de intervalos ($Q = T - 1$) entre cada medição do ensaio, ou seja, entre dois segundos quaisquer consecutivos;
- V, valor da variação do atraso entre pacotes de um mesmo fluxo, por exemplo para $\textit{instante}_1 = 10,3\,ms$ e $\textit{instante}_2 = 11,2\,ms$, $V = 0,9\,ms$;

- T, o tempo total do ensaio ($T = 400\,s$).

5.3 Metodologia Estatística para o Cálculo Final das Métricas Estudadas

Os resultados apresentados neste trabalho, por exemplo, para determinar que um protocolo obteve melhor desempenho que outro em termos da vazão média, foram baseados em amostras dos dados coletados em cada ensaio de um tratamento. A metodologia adotada baseia-se no conceito de intervalo de confiança [51], considerando $\rho = 95\,\%$ (nível de confiança) e portanto $\alpha = 5\,\%$ (nível de significância, ou erro).

Determinando o Intervalo de Confiança para $\rho = 95\,\%$

O princípio do intervalo de confiança é baseado no fato de que é impossível determinar uma média perfeita μ para uma população de infinitas amostras N, considerando um número finito n de amostras $\{x_1, ..., x_n\}$. Porém, em termos probabilísticos é possível determinar um intervalo em que μ estará dentro dele com probabilidade igual a ρ e que estará fora dele com probabilidade igual a α.

Para determinar o valor mínimo c_1 e um valor máximo c_2 deste intervalo, chamado de intervalo de confiança, considera-se uma probabilidade $1 - \alpha$, tal que o valor μ esteja dentro desde intervalo de confiança, para n ensaios de um determinado tratamento. Assim, temos a seguinte relação:

$$Probabilidade\{c_1 \leq \mu \leq c_2\} = 1 - \alpha \tag{5.6}$$

onde,

- (c_1, c_2) é o intervalo de confiança;
- α é o nível de significância, expresso como uma fração e tipicamente perto de zero, por exemplo, $0,05$ ou $0,1$;
- $(1 - \alpha)$ é o coeficiente de confiança; e

- ρ = 100 * $(1-\alpha)$, é o nível de confiança, tradicionalmente expresso como porcentagem e tipicamente perto de 100 %, por exemplo, 90 % ou 95 %.

Assim, através do *Teorema do Limite Central*[1] [51], se um conjunto de amostras $\{x_1, ..., x_n\}$ são independentes, tem uma média $\bar{x}$ e pertencem a uma mesma população N, com média μ e desvio padrão σ, então a média das amostras tende a distribuição normal com $\bar{x} = \mu$ e desvio padrão $\sigma/\sqrt{n}$:

$$\bar{x} \simeq N(\mu, \frac{\sigma}{\sqrt{n}}) \tag{5.7}$$

Então, tendo como base a relação 5.6 e o *Teorema do Limite Central* (5.7), obtem-se o intervalo de confiança (c_1, c_2) para $\rho = 95\%$ e $\alpha = 0.05$ da seguinte forma:

$$(\mu - z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}, \mu + z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}) \tag{5.8}$$

onde,

- μ é a média para n ensaios;
- $z_{1-\alpha/2}$ é igual a 1.96. Esse valor determina 95 % para o nível de confiança, como definido na Tabela A.2, do Apêndice A, da referência [51];
- n é igual ao número de ensaios; e
- s é o desvio padrão das médias para as n ensaios.

Com relação ao valor 1.96 para o termo $z_{1-\alpha/2}$, também chamado de quantil, este é baseado no *Teorema do Limite Central* e por ser freqüentemente utilizado, encontra-se na tabela de *Quantis da Unidade de Distribuição Normal*. Esta tabela pode ser encontrada no apêndice A, Tabela A.2, da referência [51]. Para determinar este valor, temos:

$$z_{1-\alpha/2} = (1 - 0.05)/2 = 0.975 \tag{5.9}$$

O valor correspondente ao resultado da Equação 5.9, que será o valor da variável z, é igual a 1.96, segundo a tabela *Quantis da Unidade de Distribuição Normal*.

[1]Teorema do Limite Central: expressa o fato de que qualquer soma de muitas variáveis aleatórias independentes e com mesma distribuição de probabilidade tende a distribuição normal.

Portanto, baseando-se nos intervalos de confiança para cada média das métricas calculadas de acordo com a Seção 5.2.1, é possível realizar comparações com estes valores segundo o tratamento realizado para 95 % de confiança com 5 % de erro.

Determinando o Valor de n para obter $\rho = 95\,\%$

O nível de confiança depende da quantidade n de amostras coletadas para um dado tratamento. Assim, quanto maior o valor de n, maior será o nível de confiança. Entretanto, obter uma quantidade grande de amostras exige mais esforço e tempo. Portanto, é importante definir o valor de n de tal forma que consiga-se poupar esforço e tempo, porém mantendo o nível de confiança desejado, ou seja, $\rho = 95\,\%$.

Para iniciar o processo, utilizamos uma quantidade pequena $n_{base} = 3$ de amostras preliminares, por exemplo, 3 valores da vazão para um determinado fluxo transmitido. O objetivo é obter um valor alto para a variância, a qual é utilizada para determinar o valor de n ensaios necessárias para 95 % de nível de confiança.

Como vimos através da relação 5.8, temos que o intervalo de confiança para uma quantidade n de amostras é defindo da seguinte forma:

$$\mu \pm z \times \frac{s}{\sqrt{n}} \tag{5.10}$$

Assim, para um nível de confiança $\rho = 95\,\%$ e $\alpha = 0.05$, o intervalo de confiança é:

$$(\mu(1 - 0.05), \mu(1 + 0.05)) \tag{5.11}$$

Então, igualando os intervalos de confiança 5.11 ao intervalo de confiança 5.10 (geral), obtemos a Equação 5.12.

$$\mu \pm z \times \frac{s}{\sqrt{n}} = \mu(1 \pm 0.05) \tag{5.12}$$

Portanto, organizando a expressão para isolar a variável n, para cada tratamento, foram executados n ensaios, já contando com os 3 ensaios iniciais (n_{base}), através da Equação 5.13, para um nível de confiança $\rho = 95\,\%$, o que implica em $z = 1.96$ (a partir da Equação 5.9).

$$n = \left(\frac{1.96 \times s}{0.05 \times \mu}\right)^2 \tag{5.13}$$

5.4 Sumário do Capítulo

Neste capítulo apresentou-se a metodologia adotada para uma análise de desempenho do protocolo GMTP, a ser apresentada no próximo capítulo.

Foram descritos um método estatístico utilizado para a obtenção das métricas estabelecidas para analisar o desempenho do protocolo GMTP e discussões sobre o cálculo para obtenção dos valores finais para cada uma das métricas avaliadas. Apresentou-se um método estatístico baseado na teoria da probabilidade, que possibilita calcular a quantidade de ensaios necessários para um determinado tratamento de simulação. Na metodologia discutida, determinou-se um nível de confiança de 95 % para os valores obtidos para cada métrica estudada. Com este método, foi possível realizar comparações quanto ao desempenho do GMTP frente a outros protocolos tradicionais, como o DCCP e o TCP. Além disso, apresentou-se o cálculo para se obter a quantidade de repetições (ensaios) necessário para conseguir o nível de confiança estabelecido.

Definiu-se dois tratamentos e seus fatores. No Tratamento 1, definiu-se confrontos GMTP vs. DCCP vs. TCP, ao passo que no Tratamento 2 definiu-se confrontos entre GMTP vs. DCCP. O objetivo do Tratamento 1 é averiguar a capacidade de convergência e equidade dos fluxos transmitidos utilizando o GMTP analizando a vazão obtida por esses fluxos. Por outro lado, o objetivo para o Tratamento 2 é averiguar a escalabilidade do GMTP no que diz respeito a quantidade de nós receptores interessados em um mesmo fluxo de dados transmitido por um nó servidor. Esta avaliação foi realizada aumentando-se a quantidade de nós receptores gradativamente em uma transmissão de vídeo $1 \rightarrow n$, coletando-se valores para as métricas de vazão, carga de dados transmitida e perda de pacotes, atraso e qualidade do vídeo transmitido.

Os Tratamentos 1 e 2 foram executados em simulações de rede no NS-2, cuja topologia da rede foi definida como uma árvore binária completa. Por fim, discutiu-se o mecanismo estatístico para obtenção dos valores finais para as métricas estudadas e o cálculo do Nível de Confiança, fixado em 95 %.

Capítulo 6

Análise de Desempenho do GMTP

Neste capítulo, apresentam-se os resultados preliminiares do uso do protocolo GMTP para a transmissão de fluxos de dados multimídia em cenários com muitos nós receptores. Os resultados estão organizados de acordo com três métricas avaliadas, a escalabilidade quanto ao número de nós clientes interessados por um mesmo fluxo de mídia, a taxa de transmissão e o atraso. Essas métricas foram escolhidas segundo as sugestões de avaliação de protocolos apresentadas no documento *Metrics for the Evaluation of Congestion Control Mechanisms* (RFC 5166) [33].

6.1 Análise da Escalabilidade do Número de Clientes

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

Observando-se o gráfico na Figura 6.1, nota-se que no nível 9 da topologia de rede utilizada, fazem-se necessárias 10230 conexões simultâneas dos clientes ao servidor. Seguindo-se esta constatação, observa-se que ao utilizar o protocolo GMTP são realizadas, no máximo 1023 conexões (pior caso). O pior caso acontece se cada nó localizado em cada rede local distinta não encontrar um nó relay, sendo necessário um cliente por rede local estabelecer uma conexão direta com o servidor. Este pior caso é muito difícil de acontecer, pois o mecanismo de busca por nós relays empregado no GMTP cuidará de encontrá-los, evitando-se

conexões desnecessárias no servidor, exceto se o nó relay estiver 5 saltos de distância do cliente interessado pelo fluxo de dados.

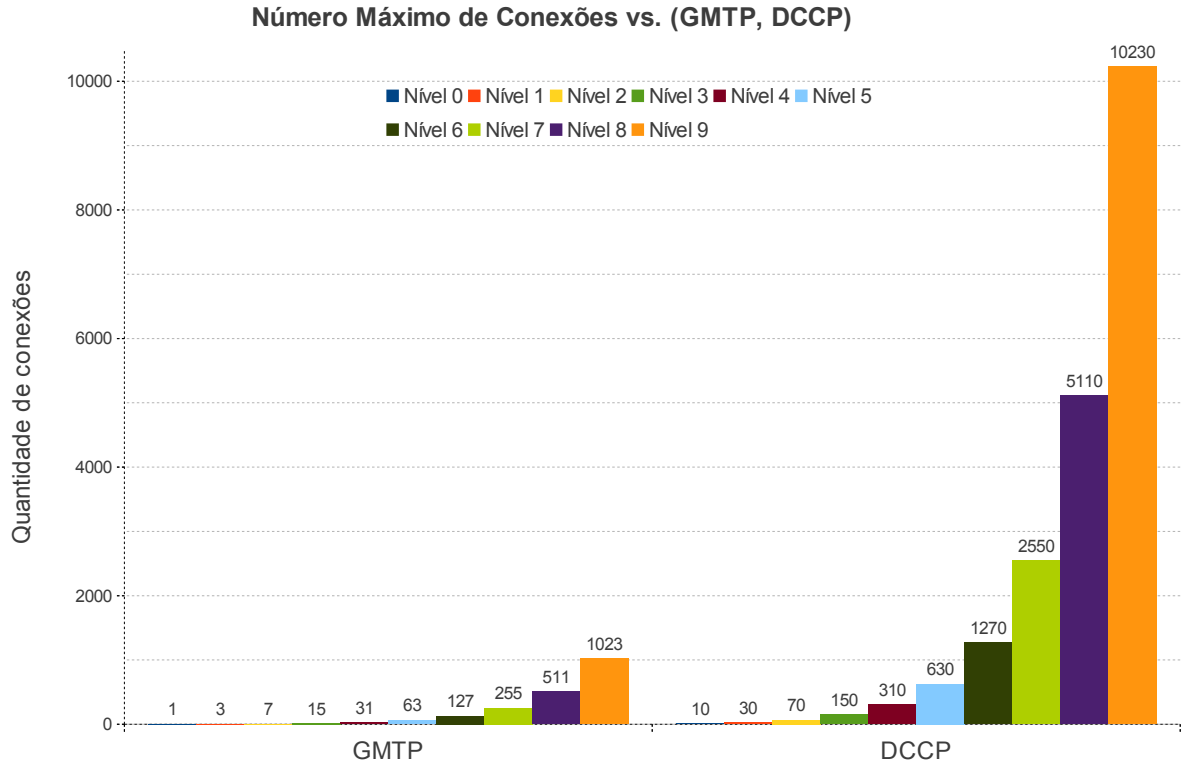


Figura 6.1: Gráfico da quantidade de conexões simultâneas ao servidor com o uso dos protocolos GMTP e o DCCP.

Nas simulações de rede executadas, avaliou-se a evolução do protocolo GMTP quanto à disponibilidade de nós relays e o uso destes por parte dos clientes. No caso da topologia de rede utilizada nas simulações, constatou-se apenas duas conexões ao nó servidor e em cada nó relay (melhor caso). Isto porque a quantidade de níveis da árvore da topologia de rede foi pequena (apenas 10 níveis), então os clientes localizados em níveis mais extremos da topologia da rede não alcançou o limiar de atraso necessário para ativar o mecanismo empregado no GMTP de conexão direta ao servidor, sem o uso de nós relays. Este aspecto do GMTP foi discutido no final da Seção 3.4. Neste contexto, faz-se necessário mais estudos de simulações a fim de atingir o limiar de atraso configurado pela aplicação e então avaliar o comportamento do GMTP no tocante à quantidade de conexões simultâneas que serão necessárias ao nó servidor caso aconteça esta situação. Mais adiante na Seção 6.3 serão discutidos outros resultados relacionados ao atraso com o uso do GMTP.

Dando continuidade às discussões acerca da escalabilidade do GMTP no quesito de nú-

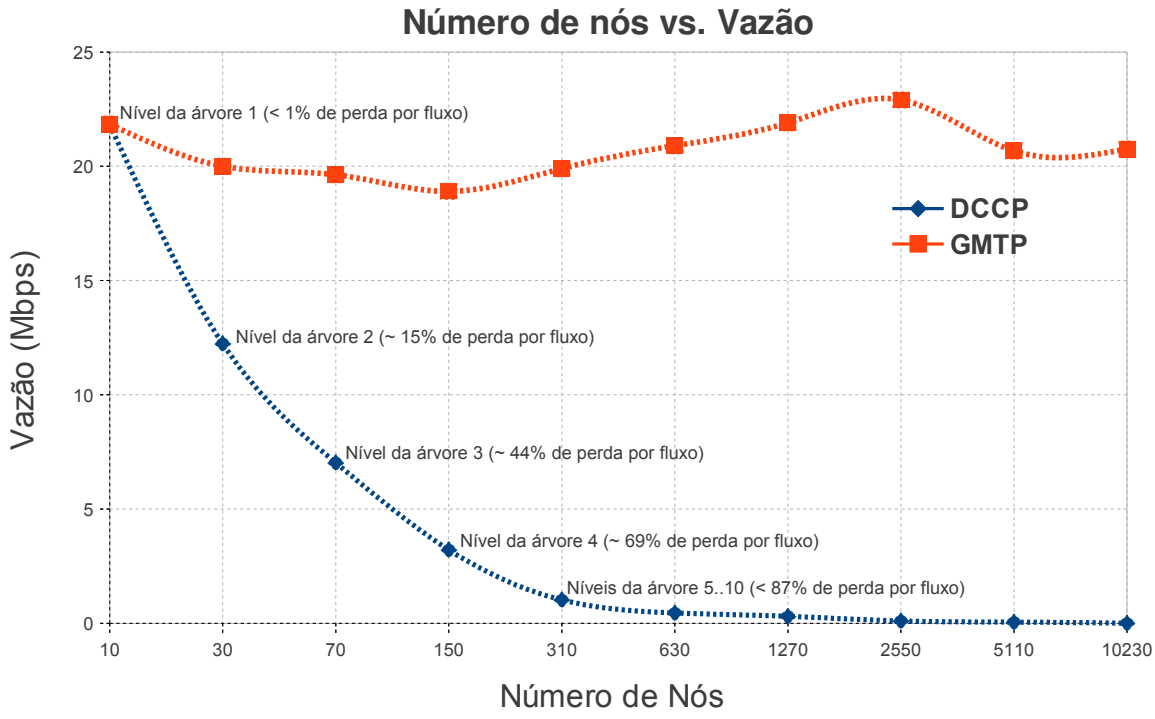


Figura 6.2: Gráfico da taxa média de recepção e perda de pacotes pela quantidade de nós clientes entre o nível 1 (10 clientes) e o nível 9 (10230 clientes), utilizando-se os protocolos GMTP e DCCP.

mero de clientes interessados pelo mesmo fluxo de dados, no gráfico da Figura 6.2, observa-se um comparativo do comportamento do uso do GMTP e a taxa de transmissão média obtida por cada nó cliente à medida que aumentou-se o número de clientes (níveis 0-9). Constata-se claramente que quando se compara o comportamento do GMTP e do DCCP, com o uso do GMTP, melhora-se sobremaneira a taxa efetiva de recepção de dados por parte dos nós clientes, ao passo que a quantidade de perda de pacotes de dados por parte do DCCP aumenta exponencialmente. Observa-se que mesmo aumentando a quantidade de nós clientes, por exemplo, entre 150 e 2550, a taxa de transmissão média continuou sendo satisfatória, sendo crescente à medida que aumentou-se a quantidade de nós clientes. Isso é justificado pelo fato de que quanto mais nós DCCP na rede, aumentam-se as chances de mais nós clientes se tornaram relays, o que consequentemente aumentam as chances de existirem mais clientes recebendo fluxos de dados em modo multicast e não somente em modo unicast.

6.2 Análise da Taxa de Transmissão

O estudo da métrica *taxa de transmissão* ocorreu através da análise do comportamento dos protocolos GMTP, DCCP e TCP para a transmissão de fluxos de dados multimídia no cenário apresentado no Capítulo 5.

No gráfico apresentado na Figura 6.3, observa-se o comportamento dos protocolos estudados quando utilizados em uma transmissão de vídeo, segundo as considerações estabelecidas na Seção 5.1. No eixo das abscissas representa-se o tempo de simulação em segundos, ao passo que no eixo das ordenadas representa-se a taxa de transmissão obtida a cada segundo. Note que os valores plotados no gráfico refletem a taxa de recepção para um dos ensaios efetuados do Tratamentos 1. A quantidade de repetições realizadas para cada protocolo, assim como um sumário dos valores de todas as métricas estudadas são apresentadas na Seção 6.4.

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

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

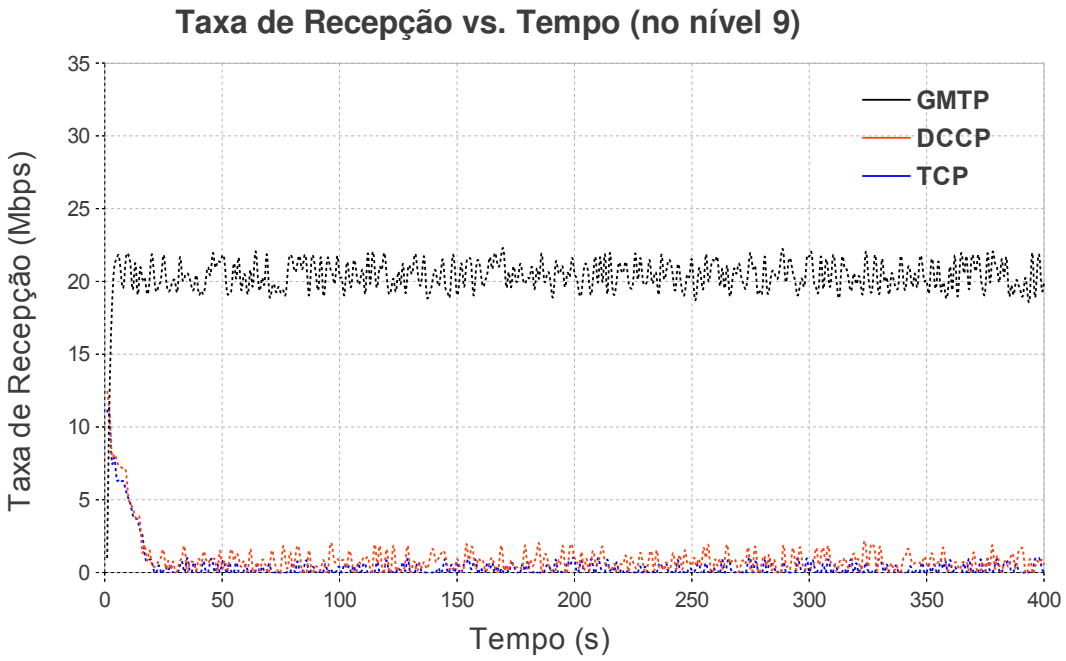


Figura 6.3: Gráfico da taxa de recepção para 10230 clientes (nível 9), considerando-se o uso dos protocolos GMTP, DCCP e TCP em uma transmissões de vídeo.

6.3 Análise do Atraso Fim-a-Fim

Após uma avaliação do comportamento do GMTP com relação a métrica taxa de transmissão e escalabilidade do protocolo, apresenta-se agora discussões acerca da métrica *atraso*. No gráfico ilustrado na Figura 6.4, apresenta-se a evolução do atraso com relação aos protocolos GMTP e o DCCP, considerando-se a topologia de rede estudada. No eixo das abscissas, representa-se a distância entre um grupo de clientes (os níveis da árvore da topologia de rede utilizada) até o servidor localizado na raíz da árvore, ao passo que no eixo das ordenadas representa-se a média do atraso de recepção observado em cada nó GMTP.

Note que, à medida que a distância em saltos aumenta entre um nó cliente (níveis mais altos da topologia de rede) e o servidor localizado na raíz, o atraso de recepção aumenta exponencialmente para cada cliente DCCP. Não observa-se este fato ao utilizar o protocolo GMTP, onde o aumento do atraso de recepção é linear e suave. De fato, isto ocorre por conta do uso da combinação do modo de transmissão unicast e multicast empregado no GMTP. Ao utilizar o protocolo DCCP, muitas conexões são abertas ao servidor, o que resulta em múltiplos fluxos com conteúdos iguais que são transmitidos na rede, fazendo-se mal uso do

canal de transmissão e elevando-se o nível de congestionamento da rede. A consequência disso é uma inundação de pacotes de dados no roteador que, quando estes não os destartam, demoram demasiadamente para processá-los, o que aumenta o atraso fim-a-fim.

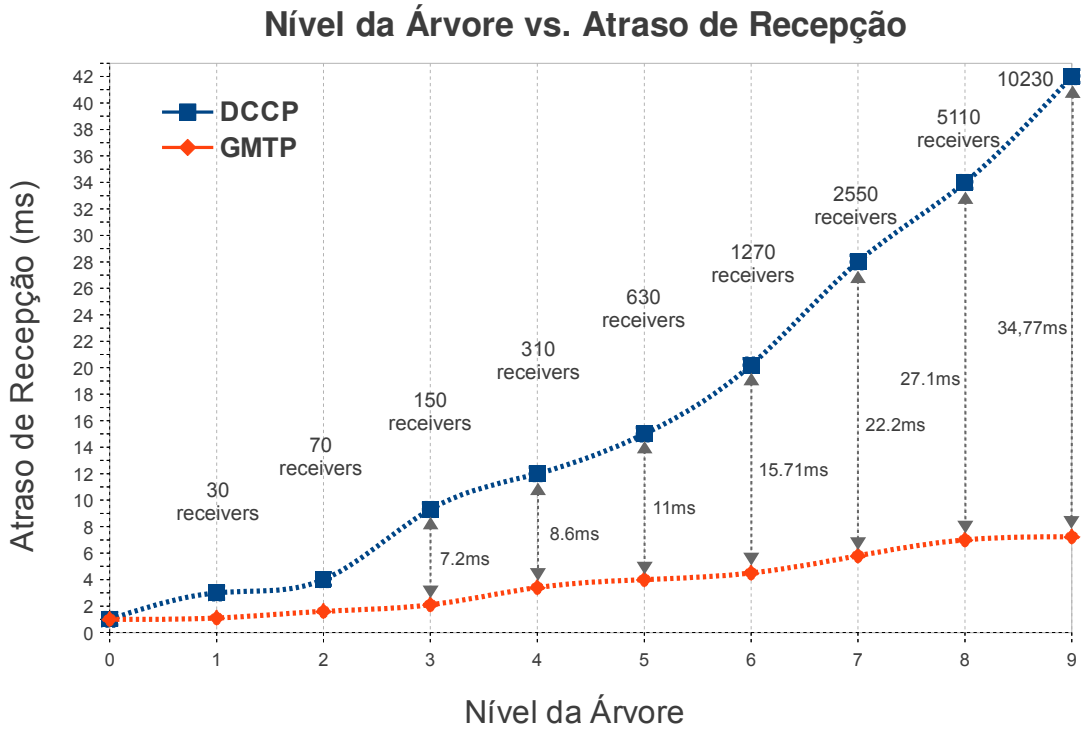


Figura 6.4: Gráfico do atraso médio de recepção de pacotes de dados pela quantidade de nós receptores, entre o nível 1 (10 clientes) e o nível 10 (10230 clientes) da topologia de rede estudada, considerando-se o uso dos protocolos GMTP e DCCP.

6.4 Compilação dos Resultados

Na Tabela 6.1, apresenta-se um sumário das métricas taxa de transmissão, quantidade de dados transmitidos e perdidos, atraso e o número de repetições dos tratamentos para os protocolos GMTP, DCCP e TCP. Considera-se importante salientar a quantidade de repetições que foram realizadas durante a execução dos tratamentos. Os valores apresentados na coluna *n* da Tabela 6.1 expressa a quantidade de vezes que um determinado tratamento foi repetido, utilizando-se todo o suporte estatístico discutido na Seção 5.3.

De acordo com os valores apresentados na Tabela 6.1, constata-se que para as métricas avaliadas, o GMTP promove melhorias significativas se comparado ao uso de protocolos

tradicinais, tais como o DCCP e o TCP em cenários de aplicações estudados neste trabalho. Observou-se melhorias da taxa de transmissão média obtida por um fluxo GMTP de quase 211 vezes mais, comparando-se ao DCCP e de quase 320 vezes mais comparando-se ao TCP. Além disso, obteve-se melhoras bastante expressivas com o uso do GMTP com relação às métricas de quantidade de dados efetivamente transmitidos e o atraso.

Considerando-se a taxa de transmissão média obtida por cada fluxo de dados GMTP, conseguiu-se transmitir, em média, 29103,8 *MBytes*, com perda apenas de 724 *MBytes*. Comparando-se o desempenho do GMTP aos protocolos DCCP e TCP, o resultado é bastante satisfatório. Com o uso do DCCP, transmitiu-se efetivamente apenas 87,62 *MBytes*, de 141,62 *MBytes* originados do servidor. Já com o uso do TCP, transmitiu-se apenas 28,15 *MBytes*, de 141,62 *MBytes* originados do servidor. Este resultado obtido com o uso do TCP é muito ruim porque tal protocolo implementa retransmissão de dados e a perda de dados causa a redução da taxa de transmissão e o aumento do atraso fim-a-fim devido ao acumulo de dados que precisa ser retransmitido.

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

Tabela 6.1: Sumário da taxa de transmissão, quantidade de dados transmitidos e perdidos e o atraso para os protocolos GMTP, DCCP e TCP.

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

6.5 Sumário do Capítulo

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

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

Durante todo o processo de discussão acerca dos resultados obtidos, comparou-se o desempenho do GMTP em relação aos protocolos DCCP e o TCP, onde constatou-se melhorias significativas para as métricas estudadas do uso do protocolo GMTP em transmissões de conteúdos multimídia para 10230 nós receptores.

De acordo com os resultados apresentados ao longe deste capítulo e discussão apresentada na Seção 6.4, constatou-se que o desempenho do GMTP foi significamente superior em todas as métricas estudadas se comparado aos outros protocolos analisados.

Capítulo 7

Trabalhos Relacionados

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

A compilação dos trabalhos a seguir foi realizada baseando-se em informações obtidas e adaptadas de publicações encontradas em diversas fontes (revistas, conferências, livros, teses e dissertações etc.) disponíveis na literatura.

Na Seção 7.1 apresenta-se uma breve discussão acerca de protocolos multimídia padronizado. Em seguida, na Seção 7.2 discute-se o estado da arte acerca de protocolos de transporte, dando-se ênfase às propostas de protocolos de transporte que utilizam uma abordagem P2P. Por fim, na Seção 7.4 apresenta-se um sumário comparativo sobre os protocolos apresentados e uma breve discussão sobre a proposta diferenciada promovida no GMTP.

7.1 Protocolos Multimídia Padronizados

Para permitir a criação, encerramento e controle de conferências multimídia, como videoconferência, audioconferência ou mesmo uma simples transmissão de dados de vídeo de uma parte a outra, em tempo real, padrões de comunicação foram desenvolvidos e disponibilizado publicamente. Nesta perspectiva, a seguir apresentam-se os protocolos de rede que merecem

destaque, principalmente por serem empregado na maioria das aplicações multimídia existentes.

O processo de desenvolvimento de padrões abertos iniciou com um grupo de estudos do ITU-T (*International Telecommunication Union – Telecommunication Section*), em 1996. O ITU-T especificou o padrão H.323 [66], que estabelece uma arquitetura de comunicação destinada ao controle de conferências de voz, vídeo e dados sobre redes TCP/IP. O H.323 se tornou largamente utilizado, uma vez que, a época da especificação de sua segunda versão, em 1998, não havia qualquer padrão aberto e aceito pelo mercado capaz de atender às aplicações multimídia. Embora as transmissões multimídia em tempo real por multicast já estivessem sendo realizadas no Mbone [4] há algum tempo, não havia ainda qualquer padrão aberto para controle de conferências multimídia.

Num período equivalente a maturação do H.323, a IETF iniciou o desenvolvimento de uma arquitetura de comunicação mais flexível e poderosa que o H.323. Alicerçada sobre o protocolo SIP (*Session Initiation Protocol*) [87], a arquitetura SIP teve logo potencial reconhecido, uma vez que supria todos os pontos fracos da arquitetura H.323, como a demora no estabelecimento de conexão (canais H.225 e H.245) e a complexidade de operação.

Devido à natureza da pilha de protocolos TCP/IP, voltada à transmissão de dados pelo paradigma do "melhor esforço", sem qualquer cumprimento a requisitos de tempo, novos protocolos foram propostos para as necessidades de transmissão de dados em tempo real. O protocolo em destaque neste sentido é o RTP (*Real Time Protocol*) [90], um protocolo de transporte para aplicações de tempo real. O RTP atualmente constitui a base das transmissões multimídias em tempo real na Internet, sendo o padrão adotado em praticamente todas as soluções multimídia baseadas em IP. O RTP oferece um mecanismo para que as aplicações tenham controle sobre número de seqüência e marcas de tempo de cada pacote de dados transmitido, não oferecendo qualquer mecanismo de retransmissão de informações perdidas.

Em paralelo ao RTP, as aplicações multimídia utilizam o *Real Time Streaming Protocol* (RTSP) [91]. O RTSP é um protocolo a nível de aplicação desenvolvido pela IETF para controle na transferência de dados com propriedades de tempo real. Tal protocolo torna possível a transferência, sob demanda, de dados em tempo real como áudio e vídeo. Ele serve para estabelecer e controlar um único ou vários fluxos sincronizados de mídias contínuas pertencentes a uma apresentação.

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

Tanto o RTP quanto o RTSP são propostas desenvolvidas para suprir as carências encontradas em protocolos de transporte tradicionais da Internet, como o UDP e o TCP. Apesar de novos protocolos de transporte padronizados pela IETF, tais como o DCCP e o SCTP (*Stream Control Transmission Protocol*) [53; 50], o uso dos protocolos RTP/RTSP em conjunto com o protocolo UDP é predominante nas aplicações multimídia encontradas na Internet.

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

7.2 Protocolos de Transporte de Dados Multimídia

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

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

O *Peer-to-Peer Epi-Transport Protocol* (PPETP) é um protocolo distribuído que utiliza uma abordagem P2P para transmissão de mídias em tempo real, sendo proposto para operar em redes com nós heterogêneos [11].

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

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

O PPETP utiliza uma abordagem de transmissão do tipo *push* onde os nós iniciam e finalizam as conexões utilizando pacotes de controle e trocam dados em modo unicast. Ao desejar receber um fluxo de dados, um nó B envia um pedido de conexão ao servidor PPETP (Figura 7.1). Em seguida, o servidor responde com uma informação que determina qual nó o cliente B deve solicitar os dados da transmissão, além de solicitar que o nó B obtenha um arquivo de configuração dos parâmetros de conexão em um servidor de configuração chamado de *starting point*. O servidor de configuração pode ser um nó diferente do servidor gerador do fluxo de dados multimídia. O nó B então requisita os dados ao nó determinado pelo servidor. No PPETP utiliza-se o protocolo UDP para transmissão de dados por padrão, embora permite-se o uso de outros protocolos, como o TCP e possivelmente o DCCP.

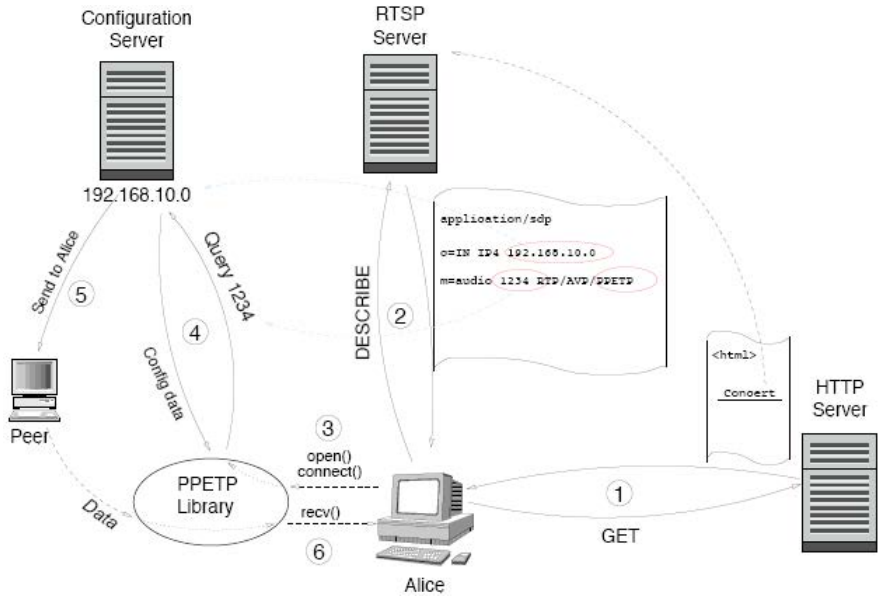


Figura 7.1: Arquitetura e funcionamento do protocolo PPETP.

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

consigam contribuir com o sistema.

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

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

Considerações sobre o trabalho

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

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

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

2. O fluxo de dados de controle é centralizado no servidor. Isto significa que para que um nó A comece a receber um fluxo de dados de um outro nó, primeiro o nó A precisa solicitar ao servidor o acesso ao conteúdo para em seguida efetivamente começar a recebê-lo. No GMTP isto não acontece, pois qualquer nó pode funcionar como servidor, o que ocorre de forma transparente para a aplicação.

3. O PPETP atualmente utiliza o protocolo UDP que, como já discutido, possui diversas desvantagens para a aplicação e para a rede, principalmente em situação de congestionamento na rede. O GMTP é um protocolo de transporte e portanto não necessita de nenhum outro protocolo da sua própria camada. Além disso, o GMTP possui um arcabouço para adicionar novos algoritmos de controle de congestionamento, tanto para as transmissões em modo unicast quanto para as transmissões em modo multicast.

4. O PPETP não suporte transmissão de dados em modo multicast. No PPETP os dados são transmitidos entre os nós em modo unicast, apesar dos autores mencionarem que o protocolo funciona em modo multicast, pelo menos no ponto de vista da aplicação. O termo multicast empregado nesse contexto é apenas para dá a idéia que poucos fluxos são transmitidos a partir do nó transmissão, mas que todos os nós receptores os recebem através da rede sobreposição criada pelo PPETP. No GMTP utiliza-se um mecanismo híbrido de transmissão: sempre que possível usa-se o modo multicast, caso contrário usa-se o modo unicast.

5. No PPETP alguns mecanismos bastante utilizados em sistemas de transmissão de mídias em tempo real, como o de descoberta de nós, deve ser implementado na camada de aplicação. Apesar dessa abordagem do PPETP funcionar por ser flexível para a camada de aplicação, a mesma limita o uso desses mecanismos à própria aplicação, impedindo que outras aplicações façam uso dos mesmos. No GMTP procurou-se adicionar tal funcionalidade dentro do próprio protocolo, permitindo-se o reúso desses mecanismos em diferentes aplicações. Desta forma, é possível que um algoritmo para descoberta de nós seja implementado no GMTP, em forma de componente, e qualquer outra aplicação reutilizar tal mecanismo. Com isto, o GMTP permite a interioperabilidade entre diferentes aplicações a nível de camada de transporte.

6. Para que o PPETP funcione efetivamente nas aplicações serão necessárias diversas alterações em protocolos da camada de aplicação, como no RTSP e no SDP (*Session Description Protocol*). Todas as modificações necessárias estão listadas no documento disponível na referência [11].

7.2.2 PPSP/Swift – *P2P Streaming Protocol / The Generic Multiparty Transport Protocol*

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

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

O funcionamento básico do PPSP ocorre da seguinte forma 7.2. Um nó transmissor *Peer-P* notifica ao tracker a transmissão realizada por ele (passo 1). O *tracker* então transmite uma mensagem para os *Peer-M* e *Peer-D* interessados em receber o conteúdo multimídia para se juntarem ao grupo (passos 2 e 4). O *Peer-P* transmite o conteúdo para o *Peer-M*, que repassa para o *Peer-D*. Em seguida, outros *peers* se registram como clientes interessado o *tracker*

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

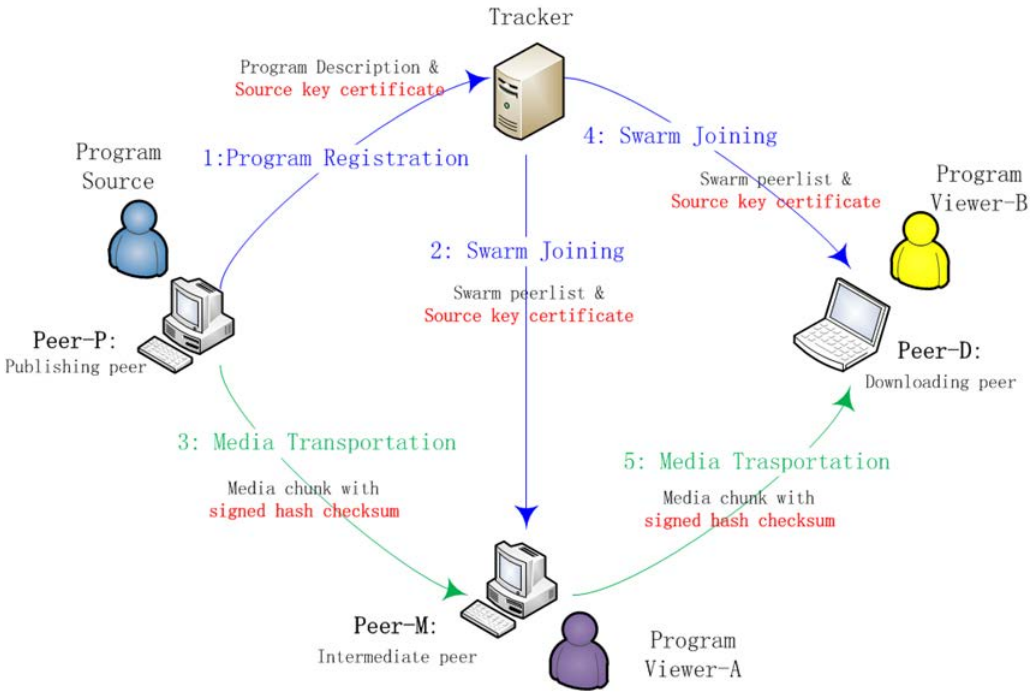


Figura 7.2: Arquitetura e funcionamento do protocolo PPSP/Swift.

transporte UDP com suporte de controle de congestionamento chamado de LEDBAT [88; 92], o mesmo adotado no BitTorrent.

Considerações sobre o trabalho

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

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

1. Ausência de suporte para extensão para recursos da aplicação, como por exemplo, descoberta, controle de congestionamento e tolerância à falhas. O GMTP é extensível nesse aspecto.

2. Não suporta compartilhamento de conexão com suporte a transmissão em modo multicast.

3. Menciona o uso futuro do algoritmo para controle de congestionamento TFRC, porém não possui suporte a controle de congestionamento em grupo. No GMTP isso é feito

utilizando o algoritmo MCC, apresentado nas Seções 3.6 e 4.3.

4. A transmissão de conteúdo com o transporte de *chunks* é interessante em aplicações para compartilhamento de arquivos, onde o tempo de resposta não é requisito fundamental para a qualidade de serviço no ponto de vista do usuário que o utiliza. O uso dessa abordagem em aplicações de transmissão de mídia em tempo real não é uma estratégia interessante devido a complexidade de indexar e remontar os pacotes de dados de acordo com cada *chunck*. Esses procedimentos podem onerar o tempo em que um pacote de dados é entregue para a camada de aplicação, gerando-se um atraso no fluxo contínuo de dados para a camada de aplicação.

5. Uso do protocolo UDP, apesar de fornecer mecanismo para controle de congestionamento. Esta prática quebra a idéia da organização dos protocolos em camadas funcionais, onde uma camada fornece serviços para a camada superior e, obviamente, usufrui de serviços da camada inferior. Mecanismos para controle de congestionamento devem ser implementados na camada de transporte e não na camada de aplicação. Isso limita o uso os recursos implementados no Swift apenas para aplicações que utilizam sua implementação, a *libswift*, disponível em forma de biblioteca de software. O GMTP é um protocolo de transporte e portanto independente de qualquer outro, além de implementar e suportar à adição de seus próprios algoritmos para controle de congestionamento.

7.3 Protocolos de Aplicação para Transmissão de Mídias

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

7.3.1 PDTP – *Peer Distributed Transfer Protocol*

O protocolo *Peer Distributed Transfer Protocol* (PDTP) [6] surgiu em 2002 com a promessa de prover um método para transferência de arquivos e mídia em tempo real similar ao BitTorrent. O uso do protocolo foi perdendo força e no final de 2007 foi descontinuado. Sua

implementação de referência era conhecida pelo nome de DistribuStream [1].

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

O PDTP especifica um conjunto de nós chamados de *hubs*, que tem como responsabilidade prover o mapa da rede, listagem de diretórios e serviço de arquivos. O serviço de arquivo é similar ao esquema de *seed* do BitTorrent, com a diferença do uso de outro conjunto de nós chamados de *Piece Proxies* (PP). Os PPs fazem download e cache de pedaços de arquivos armazenados nos nós hubs e então servem estes pedaços na rede sob demanda, reduzindo o consumo de banda dos *hubs*. Segundo os autores, o BitTorrent resolve esse problema com o uso de multiplos *seeds*, porém se não existir nenhum *seed* disponível para um *torrent* o conteúdo fica inacessível. Na Figura 7.3 ilustra-se a organização geral dos nós PDTP.

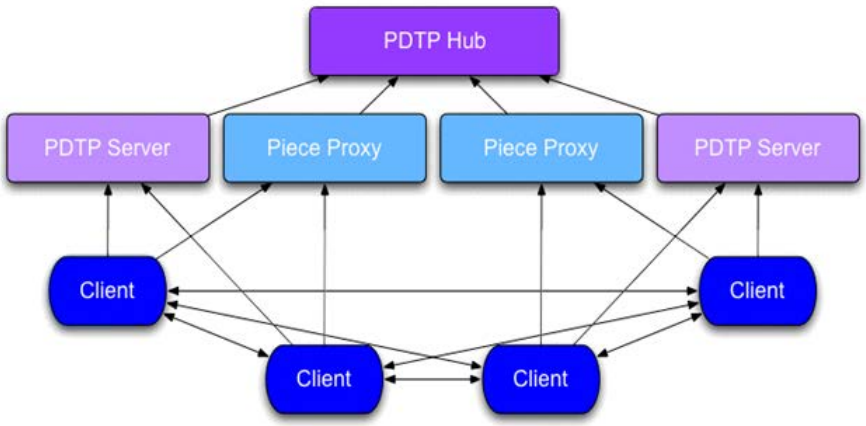


Figura 7.3: Organização dos Nós PDTP.

Considerações sobre o trabalho

A proposta do protocolo PDTP tem um ponto positivo porque organiza os nós interessados por um mesmo conteúdo de forma hierárquica e o conjunto de comandos disponíveis do

[1]DistribuStream: `http://freecode.com/projects/distribustream`

protocolo é similar a protocolos tradicionais, como o HTTP e o FTP.

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

7.3.2 CPM – *Cooperative Peer Assists and Multicast*

No *Cooperative Peer Assists and Multicast* (CPM) [37] propõe-se uma abordagem unificada para prover suporte eficiente de transmissão de vídeos sob de demanda para ser utilizada por provedores de serviços. O CPM é um protocolo de aplicação que suporta transmissão em modo multicast, cache de dados nos nós clientes, compartilhamento de dados entre os cliente, onde o servidor utiliza modo de transmissão unicast.

Na Figura 7.4 ilustra-se a visão geral do funcionamento do CPM através de um diagrama de sequência. Primeiramente o cliente conecta o servidor para saber sobre a existência de algum grupo multicast, e então passa a receber o conteúdo em modo multicast. Caso não exista um grupo multicast para o conteúdo de interesse, o cliente solicita, através de um servidor de diretórios a lista de nós que detém o conteúdo de interesse e então inicia a transferência. Caso não exista nenhum nó com o conteúdo requisitado, o cliente requisita o conteúdo diretamente para o servidor.

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

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

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

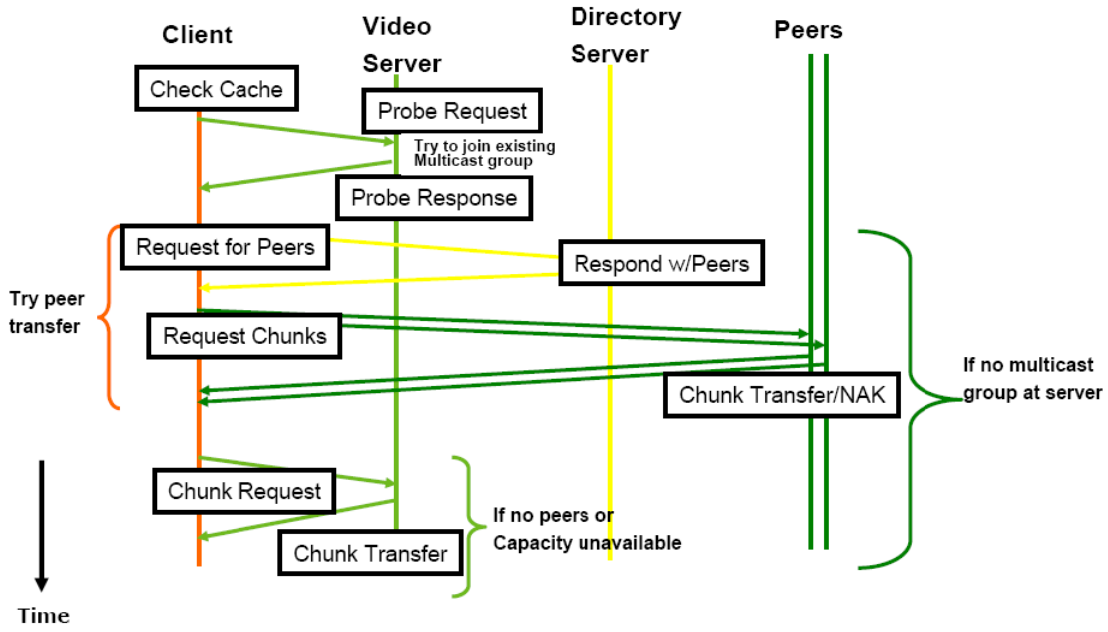


Figura 7.4: Diagrama de sequência do CPM (*Cooperative Peer Assists and Multicast*).

interesse pelo mesmo conteúdo.

Considerações sobre o trabalho

A capacidade para transmitir o conteúdo em modo híbrido é um aspecto positivo para o CPM. A seguir enumeram-se os pontos fracos identicados.

1. Para utilizar o modo de transmissão multicast, o cliente tem que ter rota multicast diretamente para o servidor, pois apenas este pode iniciar o processo de transmissão utilizando este modo. No GMTP esse mecanismo é segmentado e qualquer nó pode transmitir em modo multicast.

2. O mecanismo de transmissão de conteúdo quebra os segmentos em pedaços, o que torna o gerenciamento mais complexo devido ao espalhamento dos pedaços entre os nós participantes da transmissão. Isto pode gerar atrasos na reprodução do conteúdo no cliente devido a necessidade de localizar cada pedaço individualmente, embora o uso dessa abordagem pode aumenta a velocidade de download. No GMTP a existe a idéia de quebrar os segmentos em pedaços menores. Tal abordagem é deixada a cargo da aplicação.

3. O uso de servidor de diretórios para consultar a lista de nós que mantém o conteúdo multimídia desejado não faz muito sentido para sistemas de transmissão de mídia ao vivo. No GMTP não utiliza-se este tipo de solução.

4. Como o CPM foi desenvolvido com foco em transmissão de vídeo sob demanda, os nós só podem começar a repassar o conteúdo se previamente o este já tenha reproduzido o vídeo no passado. No caso de sistemas transmissão de vídeo em tempo real esta abordagem é completamente inútil. No GMTP qualquer nó é capaz de realizar o repasse de conteúdo, inclusive em modo multicast.

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

No *Hybrid Delivery System with Adaptive Content Management for IPTV Networks* (HySAC) [60] propõe-se uma nova arquitetura e um sistema adaptativo e híbrido que utiliza um esquema chamado de pre-população para distribuição de vídeos sob demanda em redes IPTV. Os autores do HySAC criticam o protocolo CPM ao afirmarem que tal abordagem não utiliza os recursos de rede de forma otimizada, uma vez que o conteúdo de mídia não é armazenado de modo pré-planejado de acordo com a demanda dos nós clientes, o que eleva o consumo de recursos de rede e provê uma baixa qualidade na transmissão do conteúdo multimídia ao usuário final.

Diferente do CPM, o HySAC provê a arquitetura ilustrada na Figura 7.5. Para evitar que a grande quantidade de usuários concorrents sobrecarregue os servidores de mídia, os autores do HySAC propõem um sistema adaptativo de transmissão de mídia que otimiza o processo de entrega de dados baseado na popularidade do conteúdo e nos recursos de rede disponíveis. O conteúdo é categorizado em diferentes classes e o modo de entrega do conteúdo é baseado na popularidade do mesmo. A popularidade de um conteúdo é computada baseando-se no interesse dos usuários e no número de requisições que chegam ao servidor. Os servidores HySAC categorizam os conteúdos e os servidores de indexação e descoberta utilizam Tabelas Dinâmicas de Hash (DHT) para encontrar os servidores que armazenam o conteúdo. Quando a localização de um conteúdo muda, os servidores de indexação são atualizados e quando um novo conteúdo é adicionado, os servidores cuidam da replicação do mesmo de acordo com a

sua popularidade.

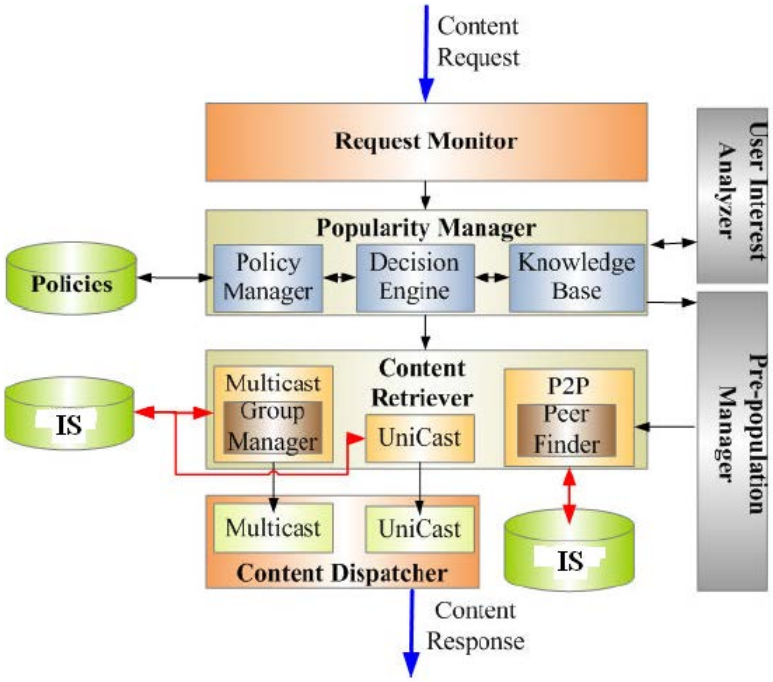


Figura 7.5: Arquitetura do HySAC (*Hybrid Delivery System with Adaptive Content Management for IPTV Networks*).

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

Considerações sobre o trabalho

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

1. A decisão do modo de transmissão é baseado na popularidade do vídeo. Considerando-se transmissões de mídia em tempo real, a forma o como HySAC implementa o me-

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

2. Da mesma forma que outras soluções, o HySAC utiliza o protocolo UDP para transmissão de dados da aplicação multimídia. Não foi encontrado nenhuma menção a respeito do uso de algoritmos para controle de congestionamento, ao contrário do GMTP.

3. Trata-se de um sistema de transmissão e não de um protocolo de rede propriamente dito. Isto significa que a proposta do HySAC servirá apenas para clientes que seguem sua especificação.

7.3.4 Outras propostas

Além das propostas de protocolos ou sistemas para distribuição de conteúdos multimídia, vários outros foram propostas foram encontradas durante o levantamento bibliográfico, com menor nível de similaridade com o trabalho aqui proposto. Por exemplo, vale ressaltar as tecnologias de distribuição de conteúdo conhecidas e utilizadas na indústria, tais como: GridMedia [112], CoolStreaming [103], AnySee [48], PPLive [42] e o ZIGZAG [97]. Estas tecnologias têm sido utilizadas com sucesso para a aplicações multimídia, contudo tais soluções foram concebidas para distribuição de conteúdo multimídia considerando um determinado propósito, com pouca flexibilidade no ponto de vista de extensibilidade e reúso dos mecanismos desenvolvidos.

7.4 Sumário Comparativo

Nas seções anteriores, presentou-se as considerações sobre cada trabalho com comparações entre o respectivo trabalho e o protocolo GMTP. Na Tabela 7.1 apresenta-se um sumário das comparações apresentadas anteriormente. Os critérios de comparação são apresentados a seguir.

- Localizado na Camada de Transporte (CT)

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

Tabela 7.1: Tabela comparativa dos protocolos para transmissão de mídia em tempo real. **Legenda:** X = Suporta; $\frac{X}{2}$ = Suporta parcialmente, apenas para algoritmos de controle de congestionamento; S = Requer intervenção da aplicação; Ind. = Indefinido.

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

7.5 Sumário do Capítulo

Neste capítulo apresentou-se uma avaliação crítica acerca de um conjunto de trabalhos sobre protocolos de transporte para distribuição de conteúdos multimídia. O foco principal foi discutir sobre trabalhos com propostas semalhantes ao protocolo GMTP , entendendo-see que nenhum trabalho contempla uma solução com a visão apresentada no Capítulo 3, mesmo as consideradas mais similares, as quais também estão em processo de especificação.

De fato, apenas dois trabalhos discutidos apresentam propostas realmente próximas ao protocolo GMTP. Esses trabalhos são o PPETP e o PPSP/Swift. No caso do PPETP trata-se de um protocolo com uma proposta de cunho mais acadêmico, onde foi possível encontrar dois artigos publicados e um *Internet-Draft* disponível online. Já os autores do PPSP/Swift apresentam-o de forma mais técnica, não sendo disponibilizado nenhum artigo científico, apenas textos técnicos em forma de *Internet-Draft*.

Capítulo 8

Considerações Finais e Planejamento

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

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

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

Apesar da existência de sistemas que exploram eficientemente o cooperativismo entre os nós participantes de uma transmissão, tais sistemas continuam à mercê de protocolos de

transportes tradicionais (TCP, UDP, DCCP e SCTP), que não foram projetados para transmitir dados multimídia em larga escala.

8.1 Estado atual de desenvolvimento da tese

Como discutido no Capítulo 1, o problema central abordado nesse trabalho é concernente à carência de soluções para dar suporte ao transporte eficiente de dados em sistemas de transmissão de mídias ao vivo baseados em arquiteturas P2P. As soluções existentes, detalhadas e comparadas ao GMTP no Capítulo 7, são paleativas porque são disseminadas em forma de sistemas ou protocolos de aplicação que, embora possam ser executadas com sucesso na Internet, é impossível evitar a pulverização e fragmentação das mesmas, aumentando-se a complexidade de implantação de uma solução unificada em larga escala na Internet.

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

Inicialmente, como detalhado nos Capítulos 3 e 4, apresentou-se um cenário geral de funcionamento do GMTP, considerando-se uma topologia de rede híbrida *mesh*-árvore. Ainda, apresentou-se uma visão geral do GMTP com destaque para a sua arquitetura, discutindo sua flexibilidade em compartilhar as boas práticas (algoritmos) entre diferentes aplicações. Além disso, deve-se enfatizar a abstração proporcionada pelo GMTP no que se refere a complexidade para construir uma infra-estrutura de comunicação entre os nós participantes de uma transmissão de mídias em redes P2P. Para tanto, permite-se a adição de novos mecanismos para descoberta e seleção de nós parceiros, tolerância à desconexão de nós e adaptação e controle de congestionamento.

De acordo com os resultados apresentados no Capítulo 6, seguindo as metodologias discutidas no Capítulo 5, constatou-se que o protocolo GMTP apresenta um desempenho satisfatório no que diz respeito ao uso eficiente dos recursos de rede. Neste contexto, discutiu-se

sobre o desempenho do GMTP quanto às métricas de taxa de transmissão, atraso e escalabilidade do número de nós receptores, concluindo-se que o GMTP é capaz de compartilhar o canal de transmissão com outros fluxos de dados (TCP), ao passo que ainda mantém taxas de transmissão em níveis aceitáveis para uma transmissão multimídia.

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

8.2 Trabalhos Futuros e Cronograma

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

1. **melhorias gerais na especificação do GMTP:** faz-se necessário definir um diagrama de estados e transições de tal protocolo, assim como na estrutura do cabeçalho dos pacotes definidos;

2. **especificar e implementar o arcabouço de extensão:** definir os pontos de extensão do protocolo, bem como a forma como novos algoritmos são adicionados, carregados e descarregados do protocolo. Isto inclui a definição das mensagens que devem ser trocadas entre os nós para notificação de mudanças ou pedido de execução de algum algoritmo específico;

3. **desenvolvimento de um protótipo executável:** implementar o protocolo GMTP no sistema operacional Linux de acordo com as especificações discutidas ao longo deste documento (capítulos 3 e 4)

4. **aplicação modelo:** implementar uma aplicação que utilize os recursos disponíveis no GMTP. Após avaliar a viabilidade, pretende-se também alterar uma implementação de algum sistema existente para utilizar o GMTP;

5. **experimentos para validação dos protótipos desenvolvidos:** executar experimentos para analisar o funcionamento do GMTP e da aplicação desenvolvidos;

6. **redação de *Internet-Drafts*:** redigir duas propostas de RFC sobre o GMTP e submetê-las para a IETF. A primeira proposta contemplará discussões acerca dos principais problemas encontrados em sistemas de transmissão multimídia em tempo real, bem como as motivações para a disponibilização de um protocolo de transporte com suporte aos requisitos de tais sistemas. Já a segunda proposta de RFC contemplará a especificação do GMTP e seus detalhes de implementação, incluindo diagramas de sequência e pseudo-algoritmos. O título da primeira proposta será *Problem Statement for Global Media Transmission Protocol (GMTP)*, ao passo que o da segunda proposta será *Global Media Transmission Protocol (GMTP) - Protocol Specification*;

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

8.2.1 Cronograma

Na Tabela 8.1 apresenta-se o cronograma das atividades a serem desenvolvidas no período de 1 ano e 2 meses.

Tabela 8.1: Cronograma de Atividades.

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

Bibliografia

[1] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch. Performance analysis of modern tcp variants: A comparison of cubic, compound and new reno. In *Communications (QBSC), 2010 25th Biennial Symposium on*, pages 80 –83, may 2010.

[2] Sachin Agarwal, Jatinder Pal Singh, Aditya Mavlankar, Pierpaolo Bacchichet, and Bernd Girod. Performance of P2P live video streaming systems on a controlled testbed. In *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, TridentCom '08, pages 6:1–6:10, Innsbruck, Austria, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ACM ID: 1390584.

[3] AINA. Assigned internet protocol numbers, 7 2011. `http://www.iana.org/assignments/protocol-numbers`. Último acesso: 22 de Novembro de 2011.

[4] K.C. Almeroth. The evolution of multicast: from the mbone to interdomain multicast to internet2 deployment. *Network, IEEE*, 14(1):10 –20, jan/feb 2000.

[5] Amazon. Introducing Amazon Cloud Player for Web & Android. Online publication in Amazon.com, 4 2011. `http://www.amazon.com/b?ie=UTF8&node=2658409011`.

[6] A. Arcieri. Peer distributed transfer protocol, 7 2004. `ftp://ftp.good.net/dl/bd/convention.cdroms/defcon12/Arcieri/draft-arcieri-peer-distributed-transfer-protocol.txt`. Último acesso: 22 de Novembro de 2011.

[7] Vlad Balan, Lars Eggert, Saverio Niccolini, and Marcus Brunner. An Experimen-

tal Evaluation of Voice Quality over the Datagram Congestion Control Protocol. In *INFOCOM 07*, volume 1, pages 455–463, 5 2007.

[8] A. Banerjea, D. Ferrari, B. A Mah, M. Moran, D. C Verma, and Hui Zhang. The tenet real-time protocol suite: design, implementation, and experiences. *IEEE/ACM Transactions on Networking*, 4(1):1–10, February 1996.

[9] R. Bell, J. Bennett, Y. Koren, and C. Volinsky. The million dollar programming prize. *IEEE Spectrum*, 46(5):28–33, May 2009.

[10] Jameson Berkow. Netflix Proclaimed "King" of North American Internet Use, 5 2011. Último acesso: 22 de Novembro de 2011.

[11] Riccardo Bernardini, Roberto Rinaldo, and Roberto Fabbro. Peer-to-Peer Epi-Transport protocol, 7 2011. `http://tools.ietf.org/html/draft-bernardini-ppetp-02`. Último acesso: 22 de Novembro de 2011.

[12] S. Bhatti, M. Bateman, and D. Miras. A comparative performance evaluation of dccp. In *Performance Evaluation of Computer and Telecommunication Systems, 2008. SPECTS 2008. International Symposium on*, pages 433–440, 6 2008.

[13] Irena Bojanova and Augustine Samba. Analysis of cloud computing delivery architecture models. In *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, pages 453–458, Biopolis, Singapore, March 2011.

[14] Thomas Bonald, Laurent Massoulié, Fabien Mathieu, Diego Perino, and Andrew Twigg. Epidemic live streaming: optimal performance trade-offs. In *Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '08, pages 325–336, Annapolis, MD, USA, 2008. ACM. ACM ID: 1375494.

[15] S. Bradner. Key words for use in rfcs to indicate requirement levels, 3 1997. `http://www.ietf.org/rfc/rfc2119.txt`. Último acesso: 22 de Novembro de 2011.

[16] Jack Brassil and Henning Schulzrinne. Structuring internet media streams with cueing protocols. *IEEE/ACM Trans. Netw.*, 10(4):466–476, August 2002. ACM ID: 581865.

[17] E. Brosh, S. A Baset, V. Misra, D. Rubenstein, and H. Schulzrinne. The Delay-Friendliness of TCP for Real-Time traffic. *IEEE/ACM Transactions on Networking*, 18(5):1478–1491, October 2010.

[18] Dave Caputo and Tom Donnelly. Global Internet Phenomena Spotlight - Netflix Rising. Technical report, Sandvine Incorporated ULC, 5 2011. `http://www.sandvine.com/downloads/documents/05-17-2011_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Spotlight%20-%20Netflix%20Rising.pdf`.

[19] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: high-bandwidth multicast in cooperative environments. *SIGOPS Oper. Syst. Rev.*, 37(5):298–313, October 2003. ACM ID: 945474.

[20] Jeng-Yuh Chang and Xiao Su. An evaluation of transport protocols in peer-to-peer media streaming. In *Networking, Architecture, and Storage, 2008. NAS '08. International Conference on*, pages 241 –247, june 2008.

[21] Yi Cui, Yanchuan Cao, Liang Dai, and Yuan Xue. Optimizing P2P streaming throughput under peer churning. *Multimedia Systems*, 15(2):83–99, November 2008.

[22] DARPA Group. Network Simulator 2. `http://www.isi.edu/nsnam/ns/`. Último acesso: 22 de Novembro de 2011.

[23] Leandro Melo de Sales. Avaliação Experimental do Protocolo DCCP para Transmissão de Conteúdos Multimídia em Redes Sem Fio 802.11g e na Internet. Master's thesis, Universidade Federal de Campina Grande, 4 2008.

[24] Leandro Melo de Sales, Hyggo O. Almeida, Angelo Perkusich, and Marcello Sales Jr. An Experimental Evaluation of DCCP Transport Protocol: A Focus on the Fairness and Hand-off over 802.11g Networks. In *In Proceedings of the 5th IEEE Consumer Communications and Networking Conference*, pages 1149–1153, 1 2008.

[25] Leandro Melo de Sales, Hyggo O. Almeida, Angelo Perkusich, and Marcello Sales Jr. On the Performance of TCP, UDP and DCCP over 802.11g Networks. In *In*

Proceedings of the SAC 2008 23rd ACM Symposium on Applied Computing Fortaleza, CE, pages 2074–2080, 1 2008.

[26] Leandro Melo de Sales, Hyggo Oliveira, and Angelo Perkusich. Multimedia content distribution of real time controlled and non-reliable datagrams between peers. In *Proceedings of IEEE Globecom 2011. 2nd IEEE Workshop on Multimedia Communications & Services*, volume 1, pages 131–146, 5 2011. A ser publicado.

[27] Leandro Melo de Sales, Hyggo Oliveira, Angelo Perkusich, and Arnaldo Carvalho de Melo. Measuring dccp for linux against tcp and udp with wireless mobile devices. In Andrew J. Hutton and C. Craig Ross, editors, *Proceedings of the Linux Symposium*, volume 2, pages 163–178, 7 2008. `http://www.linuxsymposium.org/2008/ols-2008-Proceedings-V1.pdf`.

[28] Leandro Melo de Sales, Hyggo Oliveira, Angelo Perkusich, and Rafael A. Silva. Distribuição de conteúdo multimídia em tempo real com transporte de fluxos controlados e não confiáveis entre pares. In *Proceedings of Simpósio Brasileiro de Redes de Computadores 2011. VII Workshop de Redes Dinâmicas e Sistemas P2P*, volume 1, pages 131–146, 5 2011. `http://sbrc2011.facom.ufms.br/files/workshops/wp2p/ST04_1.pdf`.

[29] D. Eastlake, J. Schiller, and S. Crocker. Randomness requirements for security, 3 2005. `http://www.ietf.org/rfc/rfc4086.txt`. Último acesso: 22 de Novembro de 2011.

[30] Hala ElAarag, Andrew Moedinger, and Chris Hogg. TCP friendly protocols for media streams over heterogeneous wired-wireless networks. *Comput. Commun.*, 31(10):2242–2256, June 2008. ACM ID: 1380037.

[31] B. Fallica, Yue Lu, F. Kuipers, R. Kooij, and P. Van Mieghem. On the quality of experience of SopCast. In *The Second International Conference on Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08*, pages 501–506. IEEE, September 2008.

[32] Sally Floyd. Highspeed tcp for large congestion windows, 12 2003. `http://www.ietf.org/rfc/rfc3649.txt`. Último acesso: 22 de Novembro de 2011.

[33] Sally Floyd. Metrics for the evaluation of congestion control mechanisms, 3 2008. `http://www.ietf.org/rfc/rfc5166.txt`. Último acesso: 22 de Novembro de 2011.

[34] Sally Floyd, Mark Handley, and Eddie Kohler. Problem Statement for the datagram congestion control protocol (DCCP), 2006. `http://www.ietf.org/rfc/rfc4336.txt`. Último acesso: 22 de Novembro de 2011.

[35] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-Based Congestion Control for Unicast Applications. *ACM SIGCOMM Computer Communication Review*, 30(4):43–56, October 2000.

[36] Torrent Freak. Paramount Pictures to Release Film on Bittorrent. Online publication in the Torrent Freak Website, 3 2011. `http://torrentfreak.com/paramount-pictures-partner-with-bittorrent-release-movie-110317/`.

[37] V. Gopalakrishnan, B. Bhattacharjee, K.K. Ramakrishnan, R. Jana, and D. Srivastava. Cpm: Adaptive video-on-demand with cooperative peer assists and multicast. In *INFOCOM 2009, IEEE*, pages 91 –99, april 2009.

[38] Yang Guo, Chao Liang, and Yong Liu. A survey on peer-to-peer video streaming systems. *PeertoPeer Networking and Applications*, 1(1):18–28, 2008.

[39] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42:64–74, July 2008.

[40] M. Handley, S. Floyd, J. Padhye, and J. Widmer. Tcp friendly rate control (tfrc): Protocol specification, 1 2003. `http://www.ietf.org/rfc/rfc3448.txt`. Último acesso: 22 de Novembro de 2011.

[41] Garrett Hardin. The Tragedy of the Commons. *Science*, 162(3859):1243 – 1248, 3 1968.

[42] X Hei, C Liang, J Liang, Y Liu, and KW Ross. Insights into PPLive: a measurement study of a Large-Scale P2P IPTV system. In *In Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[43] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and K. W Ross. A measurement study of a Large-Scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.

[44] Xiaojun Hei, Yong Liu, and Keith Ross. IPTV over P2P streaming networks: the mesh-pull approach. *IEEE Communications Magazine*, 46(2):86–92, February 2008.

[45] D. P Hong, C. Albuquerque, C. Oliveira, and T. Suda. Evaluating the impact of emerging streaming media applications on TCP/IP performance. *IEEE Communications Magazine*, 39(4):76–82, April 2001.

[46] C. -M Huang and M. -S Lin. Multimedia streaming using partially reliable concurrent multipath transfer for multihomed networks. *IET Communications*, 5(5):587–597, March 2011.

[47] Qi Huang, Hai Jin, and Xiaofei Liao. P2P live streaming with Tree-Mesh based hybrid overlay. In *International Conference on Parallel Processing Workshops, 2007. ICPPW 2007*, pages 55–55. IEEE, September 2007.

[48] Qi Huang, Hai Jin, Ke Liu, Xiaofei Liao, and Xuping Tu. Anysee2: an auto load balance P2P live streaming system with hybrid architecture. In *Proceedings of the 2nd international conference on Scalable information systems*, InfoScale '07, pages 30:1–30:2, Suzhou, China, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ACM ID: 1366843.

[49] Alexandru Iosup, Simon Ostermann, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema. Performance analysis of cloud computing services for Many-Tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945, June 2011.

[50] J. R Iyengar, P. D Amer, and R. Stewart. Concurrent multipath transfer using

SCTP multihoming over independent End-to-End paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, October 2006.

[51] Raj Jan. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Inc, 1 edition, 3 1991.

[52] John Jannotti, David K Gifford, Kirk L Johnson, M. Frans Kaashoek, and Jr. O'Toole. Overcast: reliable multicasting with on overlay network. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation - Volume 4*, OSDI'00, pages 14–14, San Diego, California, 2000. USENIX Association. ACM ID: 1251243.

[53] A. Jungmaier. *SCTP for Beginners*. Computer Network Tecnology Group, 1 edition, 9 2003.

[54] E. Kohler and J. Lai. Efficiency and Late Data Choice in a User-kernel Interface for Congestion-Controlled Datagrams. In *12th Annual SPIE Conference on Multimedia Computing and Networking (MMCN 05)*, pages 925–931, San Jose, Califórnia, 2005. `http://www.cs.ucla.edu/~kohler/pubs/lai04efficiency.pdf`. Último acesso, Abril 2008.

[55] Eddie Kohler and Mark Handley. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC), 3 2006. `http://www.ietf.org/rfc/rfc4342.txt`. Último acesso: 12/04/2011.

[56] Eddie Kohler, Mark Handley, and Floyd. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. In *IETF Online RFC*, 3 2006. `http://www.ietf.org/rfc/rfc4341.txt`. Último acesso: 12/04/2011.

[57] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP), 3 2006. `http://www.ietf.org/rfc/rfc4340.txt`. Último acesso: 22 de Novembro de 2011.

[58] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: congestion control without reliability. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 27–38, Pisa, Italy, 2006. ACM. ACM ID: 1159918.

[59] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion Control Without Reliability. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–38, New York, NY, USA, 2006. ACM Press.

[60] C. Kulatunga, G. Kandavanam, A.I. Rana, S. Balasubramaniam, and D. Botvich. Hysac: A hybrid delivery system with adaptive content management for iptv networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5, june 2011.

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

[62] N. Leavitt. Network-Usage Changes Push Internet Traffic to the Edge. *Computer*, 43(10):13 –15, 10 2010.

[63] Zhenjiang Li, Yao Yu, Xiaojun Hei, and Danny H. K Tsang. Towards low-redundancy push-pull P2P live streaming. In *Proceedings of the 5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, QShine '08, pages 13:1–13:7, Hong Kong, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ACM ID: 1535589.

[64] Chao Liang, Yang Guo, and Yong Liu. Hierarchically clustered P2P streaming system. In *IEEE GLOBECOM 2007-2007 IEEE Global Telecommunications Conference*, pages 236–241, Washington, DC, USA, November 2007.

[65] Sébastien Linck, Emmanuel Mory, Julien Bourgeois, Eugen Dedu, and François Spies. Video Quality Estimation of DCCP Streaming over Wireless Networks. In *14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, volume 1, pages 455–463, 2006.

[66] Hong Liu and P. Mouchtaris. Voice over IP Signaling: H.323 and Beyond. In *Communications Magazine, IEEE*, volume 28, pages 142 – 148, 10 2000.

[67] Yaning Liu, Hongbo Wang, Yu Lin, Shiduan Cheng, and G. Simon. Friendly P2P: Application-Level congestion control for Peer-to-Peer applications. In *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*, pages 1–5. IEEE, December 2008.

[68] Yongxiang Liu, K. N Srijith, L. Jacob, and A. L Ananda. TCP-CM: a transport protocol for TCP-friendly transmission of continuous media. In *Performance, Computing, and Communications Conference, 2002. 21st IEEE International*, pages 83–91. IEEE, 2002.

[69] R. Lo Cigno, A. Russo, and D. Carra. On some fundamental properties of P2P push/-pull protocols. In *Second International Conference on Communications and Electronics, 2008. ICCE 2008*, pages 67–73. IEEE, June 2008.

[70] Thomas Locher, Remo Meier, Stefan Schmid, and Roger Wattenhofer. Push-to-Pull Peer-to-Peer live streaming. In Andrzej Pelc, editor, *Distributed Computing*, volume 4731, pages 388–402. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[71] Steven H. Low, Larry L. Peterson, and Limin Wang. Understanding tcp vegas: a duality model. *J. ACM*, 49:207–235, March 2002.

[72] V. Lucas, J. -J Pansiot, D. Grad, and B. Hilt. Fair multicast congestion control (M2C). In *IEEE INFOCOM Workshops 2009*, pages 1–6. IEEE, April 2009.

[73] Lin Ma and Wei Tsang Ooi. Congestion control in distributed media streaming. In *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 1397–1405. IEEE, May 2007.

[74] N. Magharei, R. Rejaie, and Y. Guo. Mesh or Multiple-Tree: a comparative study of live P2P streaming approaches. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 1424–1432, Anchorage, AK, USA, 2007.

[75] Laurent MassouliÃ© and Andrew Twigg. Rate-optimal schemes for Peer-to-Peer live streaming. *Perform. Eval.*, 65(11-12):804–822, November 2008. ACM ID: 1453585.

[76] D. Meyer. Administratively scoped ip multicast, 7 1998. `http://www.ietf.org/rfc/rfc2365.txt`. Último acesso: 22 de Novembro de 2011.

[77] Emir Mulabegovic, Dan Schonfeld, and Rashid Ansari. Lightweight streaming protocol (LSP). In *Proceedings of the tenth ACM international conference on Multimedia*, MULTIMEDIA '02, pages 227–230, Juan-les-Pins, France, 2002. ACM. ACM ID: 641051.

[78] P. Navaratnam, N. Akhtar, and R. Tafazolli. On the Performance of DCCP in Wireless Mesh Networks. In *MobiWac '06: Proceedings of the International Workshop on Mobility Management and Wireless Access*, pages 144–147, New York, NY, USA, 2006. ACM Press.

[79] Frederic Nivor. Experimental Study of DCCP for Multimedia Applications. In *International Conference On Emerging Networking Experiments And Technologies (CoNEXT'05)*, pages 272–273. ACM, 2005.

[80] Hideki Otsuki and Takashi Egawa. A retransmission control algorithm for Low-Latency UDP stream on StreamCode-Base active networks. In Naoki Wakamiya, Marcin Solarski, and James Sterbenz, editors, *Active Networks*, volume 2982, pages 92–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[81] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, New York, NY, USA, 1998. ACM Press.

[82] Ju-Won Park, Jong Won Kim, and R. P Karrer. TCP-ROME: a Transport-Layer approach to enhance quality of experience for online media streaming. In *16th International Workshop on Quality of Service, 2008. IWQoS 2008*, pages 249–258. IEEE, June 2008.

[83] Larry L. Peterson and Bruce S. Davie. *Computer Networks, A System Approach*. Morgan Kaufmann, 5 ed. edition, 3 2011.

[84] Darshan Purandare and Ratan Guha. An alliance based peering scheme for peer-to-peer live media streaming. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, P2P-TV '07, pages 340–345, Kyoto, Japan, 2007. ACM. ACM ID: 1326328.

[85] RFC1631. The IP Network Address Translator (NAT), 5 1994. `http://www.ietf.org/rfc/rfc1631.txt`. Último acesso: 22 de Novembro de 2011.

[86] R Rodrigues and P Druschel. Peer-to-peer streaming systems. *Communications of the ACM*, 53(10):3–39, 2010.

[87] J. Rosenberg, H. Schulzrinne, and G. Camarillo. SIP: Session Initiation Protocol, 6 2002. `http://www.ietf.org/rfc/rfc3261.txt`. Último acesso: 22 de Novembro de 2011.

[88] D. Rossi, C. Testa, S. Valenti, and L. Muscariello. Ledbat: The new bittorrent congestion control protocol. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pages 1 –6, 8 2010.

[89] Leandro Sales, Hyggo Almeida, and Angelo Perkusich. The DCCP Protocol in Three Steps. *Linux Magazine*, 1(93):58–64, 8 2008. `http://www.linux-magazine.com/content/view/full/36244/\%28offset\%29/3`.

[90] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, 7 2003. `http://www.ietf.org/rfc/rfc3550.txt`. Último acesso: 22 de Novembro de 2011.

[91] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp), 4 1998. `http://www.ietf.org/rfc/rfc2326.txt`. Último acesso: 22 de Novembro de 2011.

[92] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. Low extra delay background transport (LEDBAT), 10 2011. `http://tools.ietf.org/id/`

`draft-ietf-ledbat-congestion-09.txt`. Último acesso: 22 de Novembro de 2011.

[93] Luiz Fernando Gomes Soares. *Redes de Computadores: das LANs, MANs e WANs às Redes ATM*. Campus, 2 edition, 1995.

[94] R. Stewart, J. Stone, D. Otis, K. Morneault, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol (SCTP), 9 2007. `http://www.ietf.org/rfc/rfc4960.txt`. Último acesso: 22 de Novembro de 2011.

[95] Shigeki Takeuchi, Hiroyuki Koga, Katsuyoshi Iida, Youki Kadobayashi, and Suguru Yamaguchi. Performance Evaluations of DCCP for Bursty Traffic in Real-Time Applications. In *2005 Symposium on Applications and the Internet*, volume 1, pages 142–149, 2005.

[96] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound tcp approach for high-speed and long distance networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1 –12, april 2006.

[97] D. A Tran, K. A Hua, and T. Do. ZIGZAG: an efficient peer-to-peer scheme for media streaming. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1283– 1292 vol.2. IEEE, April 2003.

[98] A. Vakali and G. Pallis. Content delivery networks: status and trends. *Internet Computing, IEEE*, 7(6):68–74, nov.-dec. 2003.

[99] S. Venot and Lu Yan. Peer-to-Peer media streaming application survey. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2007. UBICOMM '07*, pages 139–148. IEEE, November 2007.

[100] Alex Borges Vieira and Sergio Vale Aguiar Campos. *Transmissão de mídia contínua ao vivo em P2P: modelagem, caracterização e implementação de mecanismo de resiliência a ataques*. PhD thesis, Universidade Federal de Minas Gerais - UFMG, 3 2010. `http://dspace.lcc.ufmg.br/dspace/handle/1843/SLSS-85BNKG`.

[101] Long Vu, Indranil Gupta, Klara Nahrstedt, and Jin Liang. Understanding overlay characteristics of a large-scale peer-to-peer IPTV system. *ACM Trans. Multimedia Comput. Commun. Appl.*, 6(4):31:1–31:24, November 2010. ACM ID: 1865115.

[102] Zhonghua Wei and Jianping Pan. Modeling BitTorrent-Based P2P video streaming systems in the presence of NAT devices. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, June 2011.

[103] Susu Xie, Bo Li, G.Y. Keung, and Xinyan Zhang. Coolstreaming: Design, theory, and practice. *IEEE Transactions on Multimedia*, 9(8):1661–1671, December 2007.

[104] T. K Yan and H. P Dommel. Multimedia-Aware congestion control for video streaming over the internet. In *Second International Conference on Digital Telecommunications, 2007. ICDT '07*, pages 6–6. IEEE, July 2007.

[105] Lei Ye and Zhijun Wang. A qos-aware congestion control mechanism for dccp. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 624–629, 7 2009.

[106] S. Zeadally and Jia Lu. A performance comparison of communication apis on solaris and windows operating systems. In *Information Technology: Coding and Computing [Computers and Communications], 2003. Proceedings. ITCC 2003. International Conference on*, pages 336–340, 4 2003.

[107] Chao Zhang, P. Dhungel, Di Wu, and K. W Ross. Unraveling the BitTorrent ecosystem. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1164–1177, July 2011.

[108] Meng Zhang, Jian-Guang Luo, Li Zhao, and Shi-Qiang Yang. A peer-to-peer network for live media streaming using a push-pull approach. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 287–290, Hilton, Singapore, 2005. ACM. ACM ID: 1101206.

[109] Meng Zhang, Qian Zhang, Lifeng Sun, and Shiqiang Yang. Understanding the power of Pull-Based streaming protocol: Can we do better? *IEEE Journal on Selected Areas in Communications*, 25(9):1678–1694, December 2007.

[110] Meng Zhang, Li Zhao, Yun Tang, Jian-Guang Luo, and Shi-Qiang Yang. Large-scale live media streaming over peer-to-peer networks through global internet. In *Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming*, P2PMMS'05, pages 21–28, Hilton, Singapore, 2005. ACM. ACM ID: 1099388.

[111] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y. -S.P Yum. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2102– 2111 vol. 3. IEEE, March 2005.

[112] Li Zhao. GridMedia+: a P2P streaming system for live and On-Demand video. In *2009 6th IEEE Consumer Communications and Networking Conference*, pages 1–2, Las Vegas, Nevada, USA, January 2009.

[113] Li Zhao, Jian-Guang Luo, Meng Zhang, Wen-Jie Fu, Ji Luo, Yi-Fei Zhang, and Shi-Qiang Yang. Gridmedia: A practical Peer-to-Peer based live video streaming system. In *2005 IEEE 7th Workshop on Multimedia Signal Processing*, pages 1–4. IEEE, November 2005.