UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ANDREA COLLIN KROB

Adaptive Layered Multicast TCP-Friendly
Análise e Validação Experimental

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. José Valdeni de Lima
Orientador

Prof. Dr. Valter Roesler
Co-orientador

Porto Alegre, agosto de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Krob, Andréa Collin

Adaptive Layered Multicast TCP-Friendly: Análise e Validação Experimental / Andréa Collin Krob – Porto Alegre: Programa de Pós-Graduação em Computação, 2009.

95 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2009. Orientador: José Valdeni de Lima; Co-orientador: Valter Roesler.

1. Controle de Congestionamento. 2. IP Multicast. 3. Multimídia. I. Lima, José Valdeni de. II. Roesler, Valter. III. Título.

"Algo só é impossível até que alguém duvide e acabe provando o contrário..."

- Albert Einstein

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACK	*Acknowledge*
AIMD	*Additive Increase Multiplicative Decrease*
ALMTF	*Adaptive Layered Multicast TCP-Friendly*
AQM	*Active Queue Management*
CBQ	*Class-based Queueing*
CODEC	Codificador/Decodificador
CR	*Congestion Representative*
EAD	Educação à Distância
ECN	*Explicit Congestion Notification*
EWMA	*Exponentially Weighted Moving Average*
FIFO	*First In First Out*
FLID-DL	*Fair Layered Increase/Decrease with Dynamic Layering*
FQ	*Fair Queueing*
FTP	*File Transfer Protocol*
GLP	*General Public License*
GMCC	*Generalized Multicast Congestion Control*
HTB	*Hierarchical Token Bucket*
HTTP	*Hypertext Transfer Protocol*
IGMP	Internet *Group Management Protocol*
IP	Internet *Protocol*
IPFW	IPFirewall - versão 4
IPTV	Internet *Protocol Television*
ISP	*Internet Service Provider*
NETEM	*Network Emulator*
NIST	*Nist Network Emulator*
NS-2	*Network Simulator*
PLM	*Packet Pair Receiver-driven cumulative Layered* Multicast

POP3	*Post Office Protocol versão 3*
PP	Pares de Pacotes
QoS	*Quality of Service*
RED	*Random Early Detection*
RLC	*Receiver-driven Layered Congestion Control*
RLM	*Receiver-driven Layered* Multicast
RTO	*Retransmission Timeout*
RTT	*Round Trip Time*
SMCC	*Smooth Multirate Multicast Congestion Control*
TBF	*Token Bucket Filter*
TC	*Traffic Control*
TCP	*Transmission Control Protocol*
TEAR	TCP *Emulation at Receivers*
TTL	*Time to Live*
UDP	*User Datagram Protocol*
UFRGS	Universidade Federal do Rio Grande do Sul
VCM	Variações de Camadas por Minuto
VoD	*Video on Demand*
WF^2Q	*Worst-case Fair Weighted Fair Queueing*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Um dos obstáculos para o uso disseminado do multicast na Internet global é o desenvolvimento de protocolos de controle de congestionamento adequados. Um fator que contribui para este problema é a heterogeneidade de equipamentos, enlaces e condições de acesso dos receptores, a qual aumenta a complexidade de implementação e validação destes protocolos.

Devido ao multicast poder envolver milhares de receptores simultaneamente, o desafio deste tipo de protocolo se torna ainda maior, pois além das questões relacionadas ao congestionamento da rede, é necessário considerar fatores como sincronismo, controle de *feedbacks*, equidade de tráfego, entre outros. Por esses motivos, os protocolos de controle de congestionamento multicast têm sido um tópico de intensa pesquisa nos últimos anos.

Uma das alternativas para o controle de congestionamento multicast na Internet é o protocolo ALMTF (*Adaptive Layered Multicast TCP-Friendly*), o qual faz parte do projeto SAM (Sistema Adaptativo Multimídia). Uma vantagem desse algoritmo é inferir o nível de congestionamento da rede, determinando a taxa de recebimento mais apropriada para cada receptor. Além disso, ele realiza o controle da banda recebida, visando à justiça e a imparcialidade com os demais tráfegos concorrentes.

O ALMTF foi desenvolvido originalmente em uma Tese de doutorado e teve a sua validação no simulador de redes NS-2 (*Network Simulator*). Este trabalho tem como objetivo estender o protocolo para uma rede real, implementando, validando os seus mecanismos e propondo novas alternativas que o adaptem para esse ambiente. Além disso, efetuar a comparação dos resultados reais com a simulação, identificando as diferenças e promovendo as pesquisas experimentais na área.

Palavras-Chave: Controle de congestionamento, IP Multicast, Multimídia.

Adaptive Layered Multicast TCP-Friendly

ABSTRACT

One of the obstacles for the widespread use of the multicast in the global Internet is the development of adequate protocols for congestion control. One factor that contributes for this problem is the heterogeneity of equipments, enlaces and conditions of access of the receivers, which increases the implementation and validation complexity of these protocols.

Due to the number (thousands) of receivers simultaneously involved in multicast, the challenge of these protocols is even higher. Besides the issues related to the network congestion, it is necessary to consider factors such as synchronism, feedback control, fairness, among others. For these reasons, the multicast congestion control protocols have been a topic of intense research in recent years.

The ALMTF protocol (Adaptive Layered Multicast TCP-Friendly), which is part of project SAM, is one of the alternatives for the multicast congestion control in the Internet. One advantage of this algorithm is its ability to infer the network congestion level, assigning the best receiving rate for each receptor. Besides that, the protocol manages the received rate, aiming to achieve fairness and impartiality with the competing network traffic. The ALMTF was developed originally in a Ph.D. Thesis and had its validation under NS-2 simulator.

The goal this work is to extend the protocol ALMTF for a real network, validating its mechanisms and considering new alternatives to adapt it for this environment. Moreover, to make the comparison of the real results with the simulation, being identified the differences and promoting the experimental research in the area.

Keywords: Congestion control, IP Multicast, Multimedia.

1 INTRODUÇÃO

O controle de congestionamento na Internet é realizado fim-a-fim, na camada de transporte, pelo protocolo TCP. Através do algoritmo AIMD (*Additive Increase Multiplicative Decrease*), a taxa de transmissão dos dados é ajustada conforme as condições da rede, sendo possível compartilhar a banda mesmo em enlaces congestionados. Por esse motivo, os fluxos TCP são considerados imparciais e amigáveis (SILVA, 2004).

O protocolo UDP, por outro lado, não realiza nenhum tipo de controle na transmissão dos dados. Por não reduzir a sua taxa de transferência nos momentos de congestionamento, os fluxos UDP são considerados agressivos e não-responsivos, podendo desestabilizar a rede em casos de sobrecarga.

Atualmente, observa-se um constante crescimento no uso de aplicações multimídia na Internet, como videoconferência, ensino à distância, transmissão de rádio e IPTV. A consequência natural desse processo é o aumento do tráfego baseado em UDP, gerando fluxos competitivos que tendem a monopolizar a largura de banda.

Para essas aplicações que potencialmente consomem recursos e envolvem uma grande quantidade de usuários, o IP multicast é a estratégia mais eficiente de transmissão (LI; KALESHI, 2005). Entre as vantagens desse mecanismo está a distribuição escalável para milhares de receptores simultaneamente, sem exigir muitos recursos ou gerar tráfego desnecessário na rede.

Os benefícios do multicast advêm da utilização de uma árvore de encaminhamento, que evita que os pacotes sejam replicados ao longo do caminho, e permite que um único fluxo seja aproveitado por diversos receptores. Por esse motivo, o multicast se adapta totalmente às aplicações multimídia geradoras de tráfego, pois reduz a carga da rede e o processamento dos roteadores intermediários.

Contudo, da mesma forma que as aplicações multimídia em geral, o multicast também utiliza o protocolo UDP na camada de transporte. Considerando a natureza *best-effort* da Internet atual, fica evidente a necessidade de mecanismos de controle apropriados, evitando que algumas aplicações causem um impacto muito grande na rede ou até mesmo um colapso de congestionamento (FLOYD; FALL, 1999).

Um dos obstáculos para o uso disseminado do multicast na Internet global é o desenvolvimento de protocolos de controle de congestionamento adequados. Entre os fatores que contribuem para esse problema está a heterogeneidade de equipamentos, enlaces e condições de acesso dos receptores, o que dificulta um acordo sobre a melhor taxa de transmissão a ser utilizada entre eles (KROB et al, 2007).

Se a taxa enviada pelo transmissor for muito baixa, os receptores com maior largura de banda irão subutilizar os seus recursos. Caso a taxa seja muito alta, os receptores

menos privilegiados podem ficar impossibilitados de receber os dados. Uma maneira de garantir o acesso universal é através da transmissão em multi-taxas, também conhecida como transmissão em camadas, que será detalhada no Capítulo 2.

Manter a estabilidade da transmissão é outro fator que agrega dificuldade aos protocolos (LI et al, 2007). A estabilidade pode ser definida como a habilidade em manter o recebimento dos dados sem que muitas variações sejam percebidas pelo usuário. Isso é uma característica importante para as aplicações multimídia, visto que um comportamento oscilatório e instável é negativo para a transmissão.

O fato do multicast ser utilizado em grande escala também acrescenta novas limitações aos protocolos de controle de congestionamento, pois torna a escalabilidade uma característica fundamental no seu desenvolvimento. Isto se justifica devido à comunicação existente entre os receptores e o transmissor poder gerar um problema conhecido como implosão de *feedbacks*, sendo essencial um mecanismo para controlar o fluxo por parte dos receptores (CARVALHO et al, 2008).

O desafio dos protocolos se torna ainda maior se considerarmos questões como imparcialidade e justiça com os demais tráfegos concorrentes, como o TCP. Devido ao protocolo TCP representar 90% do tráfego total utilizado na Internet (KAMMOUN; YOUSSEF, 2008), é necessário que todos os novos protocolos sejam TCP-*Friendly*, ou seja, que a quantidade de dados transferida em longo prazo não exceda a quantidade de dados transferida por um fluxo TCP sob as mesmas condições (FLOYD; FALL, 1999).

Por essas razões, a área de controle de congestionamento multicast tem sido um tópico de intensa pesquisa nos últimos anos (KAMMOUN; YOUSSEF, 2008), (LI et al, 2007), (BYERS et al, 2006), (PERES et al, 2005), (PAPAZIS et al, 2004), (DETSCH, BARCELLOS, 2003), sendo um dos principais limitadores da expansão dos serviços multicast.

1.1 Motivação e Definição do Problema

Apesar das inúmeras pesquisas existentes na área, o controle de congestionamento no multicast ainda não foi padronizado. Segundo Benslimane (2007), um dos motivos para essa situação é que o controle no multicast é mais complexo de ser realizado que no unicast, haja visto a variedade de fatores que precisam ser considerados, tais como estabilidade, equidade, escalabilidade e até mesmo facilidade de implementação.

Outro motivo que contribui para isso é o fato dos novos protocolos terem sido propostos e validados unicamente através de simuladores de rede (KWON; BYERS, 2003). Apesar de a simulação ser o método que oferece o maior controle sobre o ambiente, as condições de tráfego da Internet são muito complexas para serem modeladas em todos os aspectos, sendo fundamental a avaliação dos protocolos sob condições reais (BAVIER et al, 2007).

Uma das alternativas para o controle de congestionamento multicast na Internet é o protocolo ALMTF (ROESLER, 2003), o qual faz parte do projeto SAM (Sistema Adaptativo Multimídia). Uma vantagem desse algoritmo é inferir o nível de congestionamento da rede, determinando a taxa de recebimento mais apropriada para cada receptor. Além disso, ele realiza o controle da banda recebida, visando à justiça e a imparcialidade com os demais tráfegos concorrentes.

O ALMTF foi desenvolvido originalmente em uma Tese de doutorado, sendo amplamente testado e validado através de simulações no NS-2. O algoritmo foi

comparado com outros protocolos de controle de congestionamento, como o RLM (MCCANNE; JACOBSON; VETTERLI, 1996), RLC (VICISANO; RIZZO; CROWCROFT, 1998) e TFMCC (WIDMER; HANDLEY, 2001), utilizando as seguintes métricas: adaptabilidade em ambientes heterogêneos; escalabilidade; estabilidade; equidade com o próprio tráfego (*fairness*) e equidade com outros tráfegos concorrentes (*friendliness*). O ALMTF se mostrou superior a todos os protocolos avaliados, no entanto, não foi analisado em ambientes reais.

Nesse contexto, o principal problema deste trabalho é estender o protocolo ALMTF para uma rede real, propondo modificações que o adaptem para esse ambiente, de forma que ele possa ser integrado em aplicações multimídia que necessitam realizar a transmissão de vídeo de forma adaptativa, como ensino a distancia, transmissão de shows, etc.

1.2 Detalhamento do Problema

A definição do problema pode ser mais detalhada com a sua decomposição em subproblemas, como segue:

- Identificar as diferenças entre a simulação e o ambiente real;
- Converter o algoritmo original para uma aplicação em C/C++;
- Definir e validar os ambientes de testes a serem utilizados nos experimentos;
- Desenvolver *scripts* para a execução dos testes e captura dos resultados;
- Estudar novas maneiras de adaptar o algoritmo para redes reais;
- Promover a validação experimental das pesquisas na área.

1.3 Objetivos da Dissertação

Com base no problema identificado, foram delimitados alguns objetivos para este trabalho, tais como:

- Validar o protocolo ALMTF em redes locais e de longa distância;
- Analisar o comportamento do ALMTF e verificar a sua viabilidade de uso;
- Comparar os resultados obtidos experimentalmente com a simulação;

1.4 Organização dos Capítulos

O restante desta dissertação está organizado da maneira descrita a seguir. No capítulo 2 são abordados os conceitos fundamentais para o entendimento deste trabalho, englobando as áreas de controle de congestionamento, aplicações multimídia e IP multicast. O capítulo 3 introduz as principais pesquisas na área, destacando o estado da arte dos trabalhos relacionados. O capítulo 4 apresenta o protocolo ALMTF, descrevendo em detalhes os seus mecanismos e as melhorias realizadas no protocolo. A metodologia empregada no desenvolvimento deste trabalho é descrita no capítulo 5, reunindo as métricas de avaliação, os cenários de testes e a descrição completa dos experimentos realizados. O capítulo 6 apresenta os resultados obtidos durante validação do ALMTF em redes reais e a comparação dos resultados experimentais com a simulação original. O capítulo 7 tece as considerações finais deste trabalho, indicando novas possibilidades de pesquisa e trabalhos futuros na área.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz os principais conceitos sobre congestionamento, aplicações multimídia e IP multicast, os quais são base para a compreensão deste trabalho. A seção 2.1 aborda o problema de congestionamento e explica o funcionamento do TCP, visto que é o protocolo modelo na Internet. A seção 2.2 descreve os requisitos das aplicações multimídia e as características das transmissões multicast. A seção 2.3 apresenta o controle de congestionamento em transmissões multicast, as classificações de protocolos existentes e as características de projeto fundamentais no seu desenvolvimento. Por fim, a seção 2.4 apresenta o Projeto SAM, que possui como principal protocolo o ALMTF, foco deste trabalho, e que serviu como motivação para a realização desta dissertação.

2.1 Congestionamento em Redes de Computadores

Segundo Gevros et al (2001), o congestionamento se caracteriza por um estado de sobrecarga na rede, onde a demanda de recursos (geralmente a largura de banda) está próxima ou excede a sua capacidade. Para evitar que ocorram perdas nos momentos de carga demasiada, os pacotes são armazenados nas filas dos roteadores intermediários. Caso a situação persista, os pacotes serão descartados conforme a política de enfileiramento utilizada no roteador.

No caso da Internet, a política de fila mais utilizada é a FIFO (*First In First Out*), sendo que a rede fornece um serviço conhecido como *best-effort* (WIDMER; HANDLEY, 2001). Isso significa que os dados serão entregues tão logo quanto possível, mas sem a reserva de recursos ou garantias de serviço. O uso de tecnologias de QoS na Internet ainda possui restrições de aplicabilidade e custo elevado para implantação, não estando, portanto, disponível para a maioria dos usuários (PAPADIMITRIOUS; TSAOUSSIDIS, 2006).

Pode-se pensar, erroneamente, que uma solução para o congestionamento é aumentar o tamanho das filas dos equipamentos, de forma que seja possível proteger uma quantidade maior de informações. No entanto, o aumento da fila impacta diretamente no atraso fim-a-fim dos pacotes (aumento do RTT), causando problemas para as aplicações multimídia em geral. Outro agravante no uso de filas grandes é que o tempo de resposta ao congestionamento poderá ser significativamente maior, visto que nenhum pacote será perdido até que as mesmas estejam completamente cheias.

Aumentar a capacidade dos enlaces também não resolve o problema de congestionamento (LI; MUNRO; KALESHI, 2005). Segundo Benslimane (2007), o incremento isolado da taxa de transmissão de um enlace pode inclusive ocasionar a redução do *throughput* total da rede.

Essa situação pode ser constatada quando uma rede rápida está conectada a uma rede mais lenta, tornando a taxa de chegada dos dados maior que a taxa de saída. A figura 2.1 exemplifica esse cenário, onde dois provedores (ISP1 e ISP2) estão interconectados através de um enlace de 300 kbit/s.

Os transmissores 0 e 1 se comunicam com os receptores 4 e 5, respectivamente, enviando dados na maior taxa de transmissão possível (100 kbit/s cada um). Devido à soma das taxas ser inferior a capacidade do roteador 2, nenhum congestionamento é detectado e os dois enlaces são totalmente utilizados.

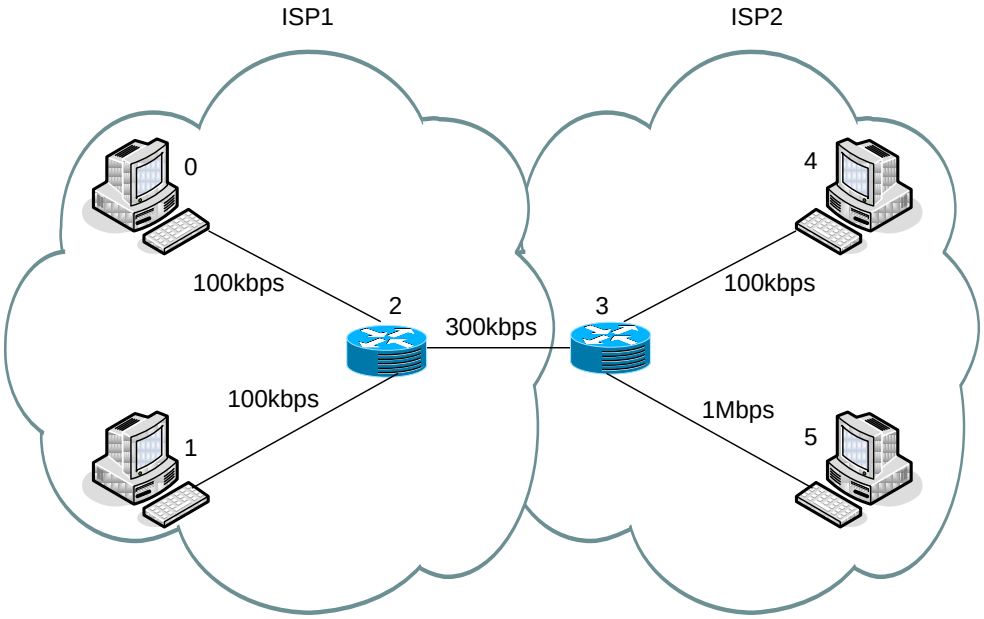

Figura 2.1: Cenário inicial da rede.

Visando uma melhor utilização da rede, o provedor ISP 1 decide fazer um *upgrade* em um de seus enlaces, aumentando a capacidade do *link* entre o transmissor 0 e o roteador 2 para 1 Mbit/s. A figura 2.2 apresenta o comportamento do roteador 2 após essa mudança.

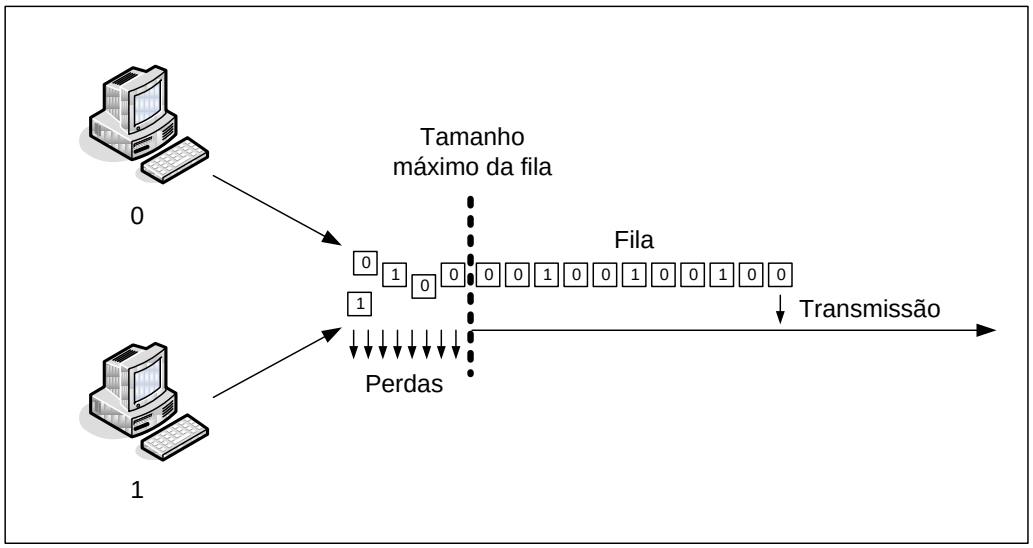

Figura 2.2: Congestionamento após o aumento do enlace.

Cada transmissor incrementa a sua taxa sem se preocupar com os demais *links* de acesso ao roteador. O aumento na capacidade do enlace faz com que a taxa de entrada do roteador seja maior que a sua taxa de saída, provocando o enchimento da fila e a perda de pacotes. Por ter a maior taxa de transmissão, o transmissor 0 monopoliza a fila com seus pacotes, consumindo a banda que deveria ser utilizada pelo transmissor 1. Cria-se, assim, um gargalo desnecessário, reduzindo o *throughput* total da rede.

Um problema similar é conhecido como "efeito fase" e pode ocorrer mesmo se a capacidade de transmissão dos enlaces seja a mesma. Esse fenômeno é observado quando se utiliza uma fila do tipo FIFO, sendo mais freqüente em enlaces de baixa capacidade (com maior propensão para formação de filas). Caso um dos transmissores envie seus dados no momento que a fila está cheia, seus pacotes serão descartados, obtendo um desempenho diferente dos demais que compartilham o mesmo enlace, devido à impossibilidade de se transmitir dois pacotes de fluxos diferentes ao mesmo tempo.

Nota-se, portanto, que o congestionamento é um problema dinâmico e não pode ser resolvido apenas com soluções estáticas (LI; MUNRO; KALESHI, 2005). Uma forma de se evitar os congestionamentos é utilizar, como apoio, funcionalidades de QoS dos roteadores intermediários, de forma que seja possível controlar a taxa de transmissão dos fluxos e punir os mais agressivos. Alguns exemplos desses mecanismos são o RED (*Random Early Drop*) e AQM (*Active Queue Management*).

A desvantagem desse método é que ele requer a manutenção permanente dos fluxos, podendo causar problemas de escalabilidade em equipamentos com elevada carga de dados (WEI; SHIN, 2004). Além disto, por necessitar de modificações na infra-estrutura da rede, não está disponível em toda a Internet atual (LIU et al, 2003).

Outra alternativa é desenvolver protocolos fim-a-fim apropriados, que reduzam as suas taxas de transferência nos momentos de congestionamento e auxiliem as aplicações a utilizarem a banda de forma imparcial e amigável.

Um exemplo clássico é o TCP, principal protocolo de controle de congestionamento fim-a-fim empregado na Internet (CHENG et al, 2007). O TCP utiliza um mecanismo baseado em janela para indicar a quantidade de dados que poderá ser enviado pela origem a cada RTT. Essa janela (*cwnd*) inicia com valor 1 e vai sendo incrementada durante a transmissão. Um limiar chamado *ssthres* controla a forma como *cwnd* se desenvolve: enquanto *cwnd* for menor que *ssthres* (fase conhecida como *slow-start*), *cwnd* dobra de tamanho a cada ACK recebido; quando *cwnd* ultrapassar *ssthres* (fase conhecida como *congestion avoidance*), *cwnd* é acrescida de 1.

Caso seja detectada alguma perda de pacote (indicada pelo recebimento de 3 ACKs duplicados), o transmissor reduz o valor de *ssthres* pela metade, fazendo *cwnd* receber esse novo valor. Se for detectado um congestionamento mais sério (indicado por *timeout*), o valor de *ssthres* é reduzido pela metade e *cwnd* recebe 1 novamente.

O controle de congestionamento do TCP também é conhecido como AIMD, visto que na ausência de perdas a banda é incrementada linearmente e reduzida pela metade, caso contrário. Esse controle permite que os fluxos dividam uniformemente o enlace, desde que o RTT seja idêntico para as duas conexões (BENSLIMANE, 2007).

Embora o TCP ofereça uma resposta rápida para os congestionamentos e seja considerado o principal responsável pela estabilidade da Internet (CHENG et al, 2007), não é atrativo para aplicações multimídia em geral. Um motivo para isso é o fato das

aplicações multimídia não necessitarem confiabilidade (como garantia de entrega, ordenamento, etc.) e exigirem um ajuste suave na taxa de transmissão, uma vez que modificações bruscas causam a degradação na qualidade percebida pelo usuário. Além disso, os mecanismos baseados em janela permitem o envio de tráfego em rajadas, o que pode causar um grande atraso na entrega dos pacotes. Portanto, o TCP raramente é utilizado para transportar tráfego multimídia na Internet.

Uma opção que vem sendo adotada para o tráfego multimídia e multicast é o controle de congestionamento baseado em taxa (INJONG; LISONG, 2007). Nesse caso, os protocolos ajustam a taxa de transmissão dos dados de acordo com o congestionamento observado na rede, tendo como base uma fórmula que modela o *throughput* do protocolo TCP (equação TCP-*Friendly*).

Esse método visa a manter uma compatibilidade com os fluxos TCP unicast, sendo o mais adequado para aplicações multimídia porque não depende de *feedbacks* para realizar a comunicação. Em uma transmissão de rádio ao vivo, por exemplo, o moderador não precisa parar a transmissão se a rede ficar congestionada, o que iria ocorrer se o mecanismo de controle utilizado fosse baseado em janela. O mecanismo baseado em taxa será abordado com mais detalhes posteriormente.

Independentemente da forma utilizada para adaptar a transmissão dos dados, o mecanismo de controle de congestionamento pode ser localizado no transmissor ou no receptor. Se o controle estiver inserido no receptor, existe uma menor interação com o transmissor, gerando menos mensagens de controle e tráfego na rede, o que é vantajoso para as transmissões em grupo de larga escala.

Se o controle de congestionamento for inserido no transmissor, os receptores precisam enviar periodicamente informações sobre a qualidade da comunicação ao transmissor, de forma que seja possível ajustar a sua taxa de transmissão. Por esse motivo, os protocolos que utilizam essa abordagem possuem deficiências com relação à escalabilidade, estando mais suscetíveis ao problema de implosão de *feedbacks*.

2.2 Aplicações Multimídia e Transmissão Multicast

As aplicações multimídia toleram eventuais perdas de pacotes, contudo, exigem um eficiente gerenciamento dos recursos de rede. Isso é necessário porque as aplicações multimídia possuem alguns requisitos específicos, como latência e *jitter* baixos, estabilidade na transmissão, largura de banda mínima, entre outros.

O termo latência representa o tempo que um pacote leva da origem ao destino, enquanto o *jitter* significa a variação estatística da latência. Ambos alteram o fluxo de chegada dos pacotes e devem ser mantidos tão baixo quanto o possível, evitando causar impactos na transmissão (PAPADIMITRIOUS; TSAUSSIDIS, 2006).

Conforme pode ser observado na figura 2.3, as aplicações mais utilizadas em redes de computadores podem ser divididas basicamente em dois tipos (ROESLER, 2003): aplicações com tráfego elástico e aplicações com tráfego inelástico.

As aplicações do primeiro tipo (tráfego elástico) exigem confiabilidade na entrega dos dados sem impor restrições temporais na transmissão. Utilizam geralmente o protocolo TCP para transportar os dados e constituem a classe mais utilizada atualmente na Internet. Exemplos de aplicações dessa modalidade são FTP, HTTP e POP-3.

As aplicações de tráfego inelástico, por sua vez, possuem restrições de tempo e são geralmente utilizadas com o protocolo UDP, sendo subdivididas de acordo com a sua rigidez em relação ao tempo de entrega dos dados. As aplicações do tipo teleconferência envolvem interatividade, portanto, necessitam de rapidez e são mais rígidas com relação à latência e *jitter*. Por outro lado, as aplicações de transmissão unidirecional envolvem apenas um lado da comunicação, possuindo restrições de tempo menores.

Algumas aplicações requerem mais largura de banda que outras, por exemplo, a videoconferência e a transmissão de TV. Conforme explicado anteriormente, somente o aumento na largura de banda dos enlaces não significa um melhor desempenho na utilização da rede. Redes públicas como a Internet estão sempre sujeitas a atrasos e congestionamentos, necessitando de mecanismos que auxiliem as aplicações a usar eficientemente os recursos disponíveis.

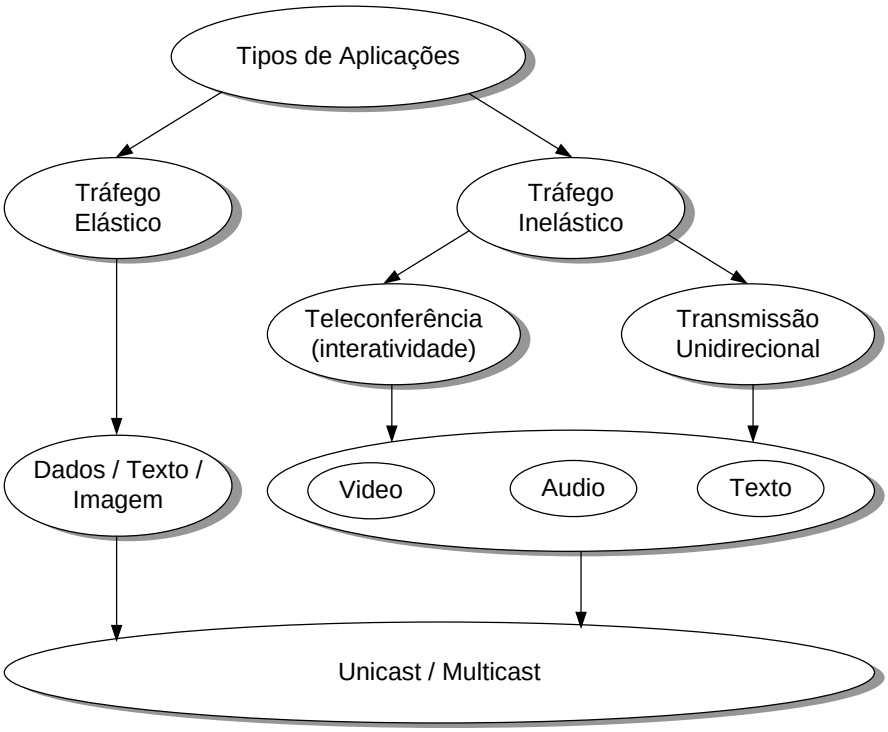

Figura 2.3: Tipos de aplicações em redes de computadores.

O IP multicast oferece a estratégia de transmissão mais apropriada para as aplicações multimídia que envolvem uma grande quantidade de usuários (LI, MUNRO; KALESHI, 2005), sendo capaz de reduzir a quantidade de recursos necessários no transmissor, economizar tráfego na rede e minimizar o processamento dos roteadores intermediários. Por esse motivo, tem sido considerado um componente essencial para muitas aplicações multimídia emergentes na Internet (LIU et al, 2003).

Existem várias características que diferem uma transmissão multicast das transmissões unicast tradicionais, sendo que a mais significante é a forma de envio dos dados, podendo ser verificada nas figuras 2.4 e 2.5.

O multicast (figura 2.4) permite a distribuição de um único fluxo para um grupo de receptores simultaneamente, proporcionando uma melhor utilização da largura de banda da rede. No unicast (figura 2.5), o transmissor precisa gerar um fluxo de pacotes para cada receptor interessado em receber a transmissão, portanto, ele deve gerar *n* vezes o mesmo tráfego, sendo que a rede deve suportar *n* vezes mais largura de banda.

Esses diferenciais são atraentes não apenas para ISPs ou empresas de telecomunicações, mas para muitas necessidades atuais, como as aplicações multimídia em larga escala na Internet. Segundo (CARVALHO et al, 2008), as aplicações de transmissão de vídeo em larga escala ainda são raras e um dos motivos é a carga elevada exigida na rede e na máquina transmissora para atingir um grande número de participantes simultaneamente.

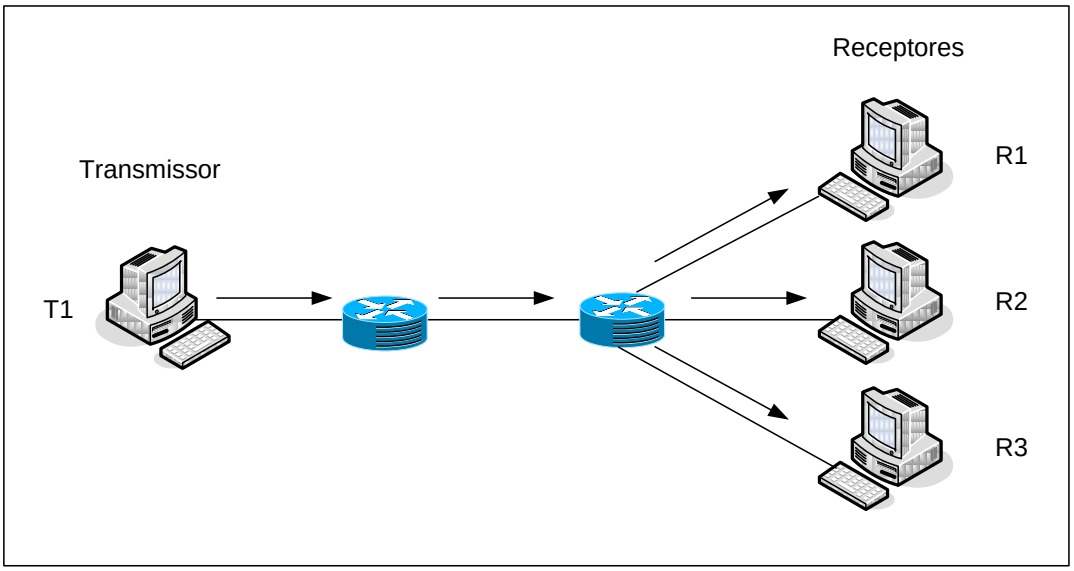

Figura 2.4: Transmissão multicast.

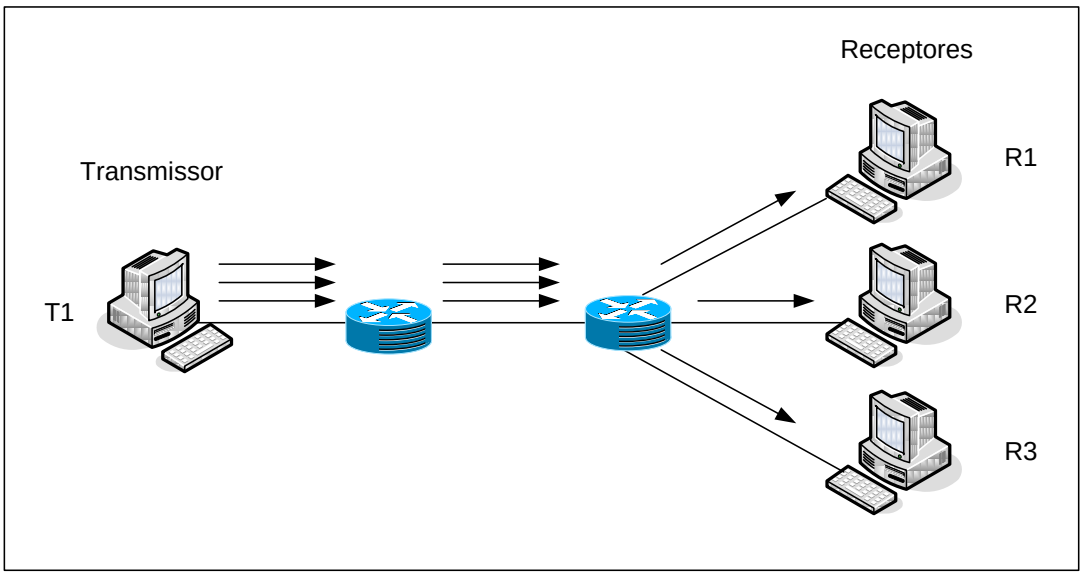

Figura 2.5: Transmissão unicast.

Por essas razões, o conceito de grupo está frequentemente relacionado com as transmissões multicast. Um grupo é um conjunto de entidades que fazem parte da mesma comunicação, como por exemplo, computadores, aplicativos e usuários. Conforme a situação, eles podem ser um grupo de receptores, transmissores ou ambos (BENSLIMANE, 2007).

O protocolo responsável pela composição dos grupos no IP multicast é o IGMP (*Internet Group Management Protocol*). Através dele, é possível que os computadores de uma sub-rede entrem (através da operação de *join*) ou saiam (através da operação de *leave*) dos grupos multicast. Os grupos são abertos e dinâmicos, sendo que os usuários podem entrar ou sair deles a qualquer momento. Maiores informações sobre a última versão do IGMP podem ser encontradas em Cain et al (2002).

2.3 Controle de Congestionamento Multicast

Realizar um controle de congestionamento eficiente é a principal exigência para as aplicações multicast se desenvolverem com segurança na Internet (SEADA; HELMY, 2002). Segundo Widmer et al (2001) e Roesler (2003), o congestionamento no multicast é considerado mais complexo por envolver questões inexistentes nas transmissões unicast, tais como:

- **Escala:** os protocolos devem estar preparados para funcionar com um ou milhares de usuários simultaneamente;
- **Heterogeneidade de enlaces:** a diversidade da rede exige que o protocolo utilize algum método para permitir acesso tanto para os receptores localizados em enlaces rápidos como aos receptores localizados em enlaces mais lentos. Caso contrário, a única taxa a ser transmitida seria equivalente à do receptor mais lento;
- **Taxas de Perda Diferentes:** os receptores localizados em enlaces heterogêneos experimentam taxas de perda diferentes, sendo responsabilidade dos protocolos de controle de congestionamento interpretá-las de forma correta;
- **Tempo de *Leave*:** problema que ocorre devido ao tempo necessário para o roteador confirmar a saída de um participante da transmissão. Enquanto isto, os pacotes continuam chegando e gerando perdas no receptor. Na primeira versão do IGMP, esse processo era realizado via *timeout* e poderia durar até dois minutos. Contudo, na versão 2 foi reduzido para aproximadamente 3 segundos.
- **Sincronismo de *join*:** é recomendado haver um sincronismo entre os receptores cadastrados na mesma sessão multicast (principalmente se estiverem na mesma rede local), pois as tentativas d*e join* de um único receptor podem gerar tráfego e perdas para todos os demais atrás do enlace. Entretanto, o sincronismo deve ser evitado caso as seções sejam diferentes.

Além desses fatores, existem outras dificuldades relacionadas ao controle de congestionamento multicast. Gerenciar a escalabilidade, estabilidade e a equidade simultaneamente é um desses desafios (SEADA; HELMY, 2002), devendo ser tratado de maneira adequada pelos protocolos.

Para que se entenda melhor o processo de controle de congestionamento multicast, alguns conceitos precisam ser descritos. As próximas subseções irão apresentar as classificações de protocolos existentes, os requisitos desejados e as principais dificuldades enfrentadas pelos mecanismos atuais.

2.3.1 Classificação dos Protocolos

Os protocolos de controle de congestionamento multicast podem ser divididos em três categorias: taxa única, multi-taxa ou híbridos. A seguir, cada uma dessas abordagens será descrita em detalhes, destacando-se as vantagens e desvantagens de cada classificação.

2.3.1.1 Protocolos de Taxa Única

Os protocolos de taxa única enviam os dados na mesma (e única) taxa de transmissão para todos os receptores. Para evitar que ocorra congestionamento e permitir que todos possam receber os dados, o transmissor mantém a taxa de envio com base no receptor mais lento, também conhecido como CR (*congestion representative*).

Para ajustar a taxa de transmissão dos dados e descobrir quem é o receptor mais congestionado, o transmissor necessita obter um retorno constante de todos os participantes da comunicação. Por esse motivo, uma dificuldade destes protocolos é evitar a implosão de *feedbacks*. A figura 2.6 exemplifica o funcionamento dos protocolos de taxa única.

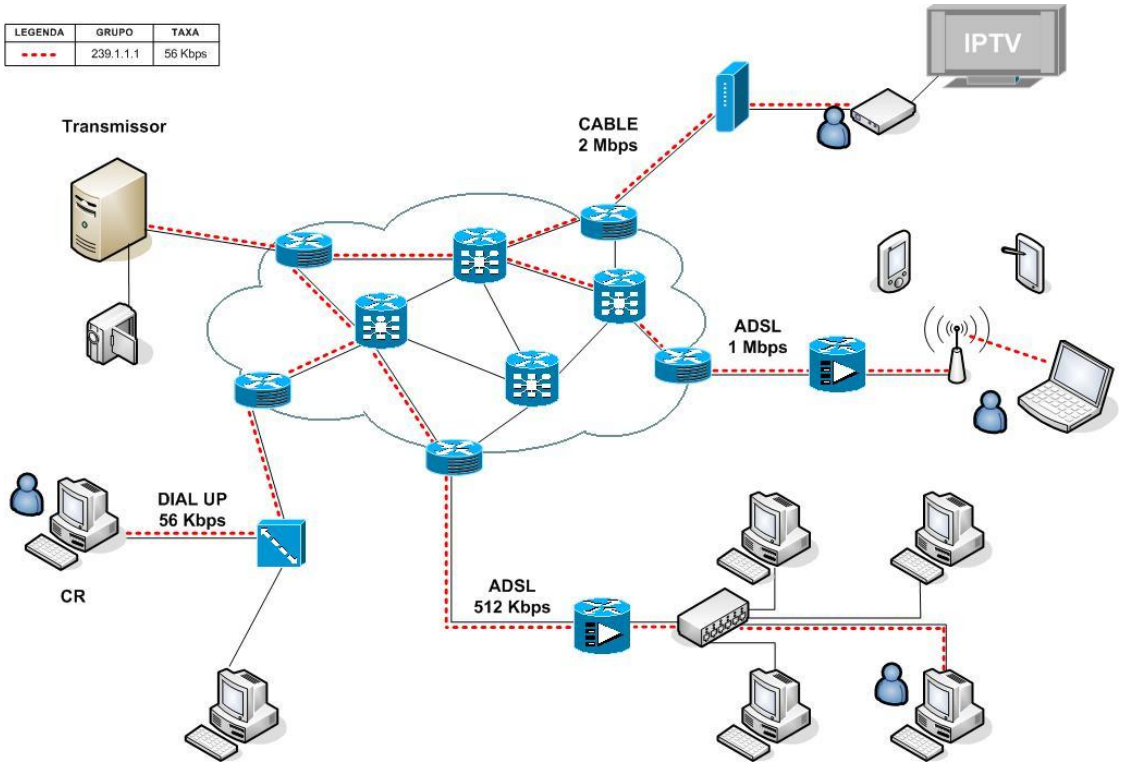

Figura 2.6: Transmissão em Taxa Única.

A linha tracejada representa o fluxo de dados que sai do transmissor e chega a todos os receptores, utilizando o grupo multicast com endereço IP 239.1.1.1. Nesse caso, o CR da transmissão é o usuário que está fazendo o acesso através da conexão *dial-up* de 56 Kbps, pois é o receptor mais lento do grupo.

Conforme pode ser observado, mesmo existindo usuários com larguras de banda maiores, como 512Kbps, 1Mbits ou 2Mbits, um único receptor com baixa velocidade prejudica a transmissão de todos os demais, obrigando o envio dos dados nessa mesma taxa de transmissão.

Os protocolos desta categoria têm como principais vantagens a simplicidade e a facilidade de implementação (JIANG; SHIVKUMAR, 2003) Além disso, devido aos dados serem enviados em uma única taxa de transmissão, apenas um grupo multicast é utilizado, reduzindo o processamento dos roteadores intermediários. Por outro lado, possui como desvantagem a dificuldade em lidar com receptores heterogêneos (LI; MUNRO; KALESHI, 2005), o que prejudica a sua utilização na vida real.

Por esse motivo, os protocolos de taxa-única não são indicados para aplicações que exigem uma largura de banda mínima ou possuem requerimentos de atraso máximo, sendo mais utilizados em transferência de dados ou ambientes onde não haja muitos receptores heterogêneos (LI; YUKSEL; KALYANARAMAN, 2006). Alguns exemplos de protocolos desta categoria são TEAR (RHEE; OZDEMIR; YI, 2000) e TFMCC (WIDMER; HANDLEY, 2001).

2.3.1.2 Protocolos Multi-Taxa

Os protocolos multi-taxa, também conhecidos como protocolos de transmissão em camadas, tem como principal característica a possibilidade de adaptação automática dos receptores. Isso significa que os dados poderão ser recebidos na melhor taxa de transmissão possível, conforme as limitações de rede de cada receptor (LI; MUNRO; KALESHI, 2005).

Nesse esquema, o transmissor gera a codificação do sinal multimídia em várias taxas e transmite cada uma delas em um grupo multicast (camada) separado. Para aumentar ou reduzir a taxa de transmissão dos dados, os receptores utilizam as operações de *join* e *leave* do protocolo IGMP, realizando assim a inscrição ou o desligamento dos grupos multicast. Contudo, devido às camadas serem fixas e pré-configuradas, os receptores devem realizar essas operações frequentemente, adaptando a transmissão para uma situação de congestionamento em tempo real (LI et al., 2007).

A quantidade de camadas mais apropriada para cada receptor varia de acordo com as condições da rede, sendo uma responsabilidade do protocolo de controle de congestionamento multicast. A taxa a ser recebida deve ser calculada considerando os demais fluxos concorrentes, visando a equidade de tráfego durante a transmissão.

As principais vantagens destes protocolos são a escalabilidade e a adaptação em ambientes heterogêneos, visto que viabilizam a transmissão para diferentes receptores sem impor restrições a estes. Por outro lado, devido a fatores como estabilidade e sincronismo, a complexidade desses protocolos é ainda maior.

A figura 2.7 descreve o comportamento de um protocolo multi-taxas na mesma topologia de rede apresentada na figura 2.6. Neste exemplo, o transmissor envia o sinal multimídia em quatro camadas distintas: camada 0, com endereço multicast 239.1.1.1 e taxa de transmissão de 56Kbps; camada 1, com endereço multicast 239.1.1.2 e taxa de transmissão de 256Kbps; camada 3, com endereço multicast 239.1.1.4 e taxa de transmissão de 512Kbps e camada 4, com endereço multicast 239.1.1.4 e taxa de 1024Kbps.

Pode-se observar que todos os receptores conseguem se adaptar automaticamente na melhor taxa de transmissão possível conforme a sua capacidade de rede. O receptor que está utilizando a conexão *dial-up*, por exemplo, se inscreveu em apenas um grupo multicast (camada 0, que corresponde a taxa de 56Kbps), enquanto o receptor que está fazendo o acesso via *cable modem* de 2Mb se inscreveu em todas as camadas, recebendo uma taxa total de 1848Kbps.

O método de transmissão em camadas geralmente é realizado com a codificação em camadas cumulativa. Nesse caso, o transmissor envia uma camada base e diversas camadas de refinamento, onde cada uma adiciona qualidade à anterior (LIU et al, 2003). O resultado dessa técnica pode ser visto na figura 2.8, onde é apresentado o resultado para o vídeo "*carphone*".

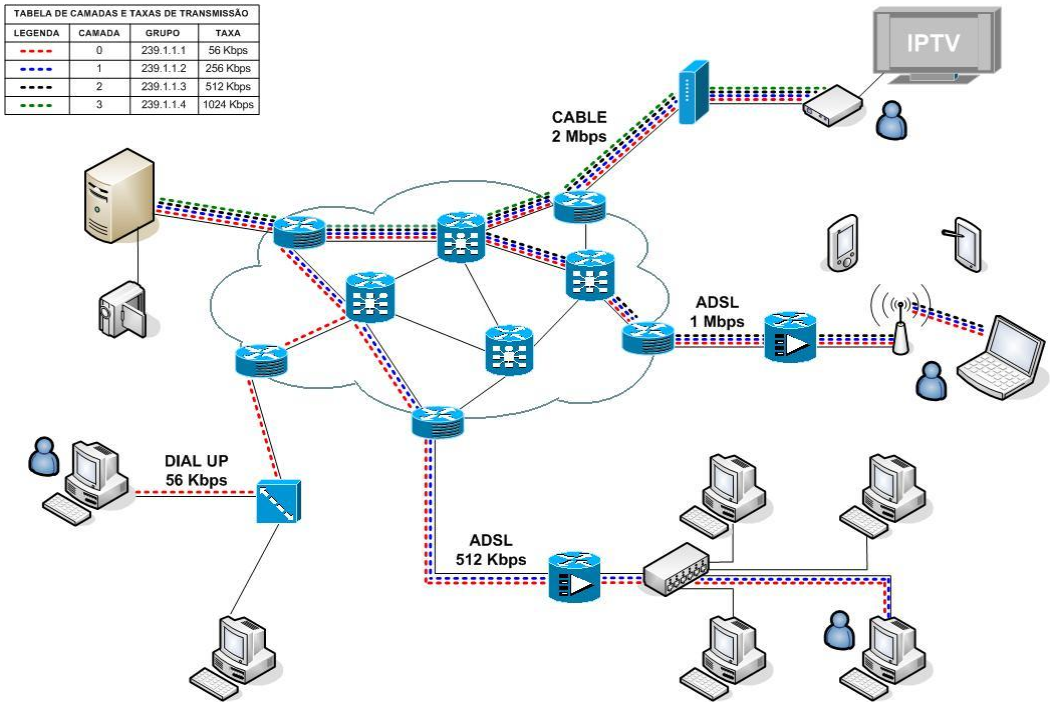

Figura 2.7: Transmissão em multi-taxa.

Caso o usuário se inscreva somente no grupo multicast referente à camada 1, ele vai receber o sinal com artefatos de "blocagem". Em compensação, a banda exigida da rede será muito baixa, na ordem de 20 kbit/s. Por outro lado, caso o usuário se inscreva nos 5 grupos multicast (referentes às 5 camadas), ele irá receber um sinal com qualidade superior, entretanto, o custo de banda na ordem de 300 kbit/s.

Figura 2.8: Transmissão em camadas cumulativas (BRUNO, 2003).

O protocolo ALMTF, foco deste trabalho, é um exemplo de protocolo multi-taxa. Outros exemplos são: RLM (MCCANNE; JACOBSON; VETTERLI, 1996), RLC (VICISANO; RIZZO; CROWCROFT, 1998), FLID-DL (BYERS et al., 2000) e PLM (LEGOUT; BIERSACK, 2000).

2.3.1.3 Protocolos Híbridos

Recentemente, uma nova categoria de protocolo tem sido proposta, unindo os conceitos de taxa-única com transmissão multi-taxa.

Os protocolos híbridos fornecem multi-taxa através de subcamadas independentes de taxa-única (PERES et al, 2005), ou seja, disponibiliza várias camadas que podem apresentar variações na sua taxa interna. Isso permite que, para cada camada existente, a taxa máxima de transmissão seja estabelecida dinamicamente, conforme os participantes cadastrados naquela seção multicast.

A principal vantagem desta abordagem é a redução de operações IGMP nos roteadores intermediários (LI et al., 2007), o que diminui a carga dispensada pelo protocolo na rede. Contudo, esses protocolos são de difícil aplicação prática, visto que necessitam de um codificador que permita a criação de diversas camadas e também a variação dentro delas (DARONCO, 2009). Alguns exemplos de protocolos híbridos são SMCC (KWON; BYERS, 2003) e GMCC (LI et al., 2007).

2.3.2 Características Desejáveis

Os protocolos de controle de congestionamento multicast possuem um conjunto de características desejáveis no seu desenvolvimento. Esta seção irá descrever quais são essas características, bem como as principais dificuldades para aplicá-las na vida real.

2.3.2.1 Estabilidade da Transmissão

Diferente das aplicações de transferência de dados que não são afetadas pelas mudanças drásticas na taxa de transmissão (como FTP, por exemplo), as aplicações multimídia exigem um ajuste suave nas suas taxas, uma vez que modificações bruscas causam a degradação na qualidade percebida pelo usuário.

A instabilidade geralmente é ocasionada por mudanças dinâmicas na rede, como o aumento de fluxos concorrentes, alterações no atraso fim-a-fim, congestionamentos, etc. Nos protocolos com transmissão multi-taxa, a instabilidade gera oscilações entre as camadas adjacentes, ocasionando frequentes operações de *join* e *leave* para ajustar a taxa de transmissão conforme o estado atual da rede.

Esse comportamento deve ser evitado, pois prejudica a aplicação multimídia e todos os fluxos que compartilham o mesmo gargalo. Especialmente as tentativas frustradas de *join*, visto que durante um período (tempo de *leave*) os pacotes continuarão sendo encaminhados para a rede, aumentando o congestionamento e as perdas.

Nota-se, portanto, que é fundamental a utilização de mecanismos para evitar que mudanças temporárias na rede prejudiquem a transmissão e, consequentemente, os demais fluxos.

A dificuldade dos protocolos em manter a transmissão estável se torna maior quando se considera questões como imparcialidade e justiça com os demais tráfegos concorrentes, como o TCP. Isso ocorre porque um fator afeta o outro, criando uma divergência entre os requisitos estabilidade e equidade.

Se o protocolo deseja manter uma estabilidade razoável, deve evitar alterar a sua taxa de transmissão (trocar de camada) com muita frequência. Entretanto, a única forma de se manter a equidade com o TCP é efetuando constantes ajustes na taxa, pois o protocolo incrementa e decrementa a sua largura de banda muito rapidamente.

A quantidade de camadas utilizadas pelo transmissor também interfere nesses fatores. Quanto maior o número de camadas, mais fácil de alcançar a equidade com os outros fluxos. Por outro lado, mais instável fica o protocolo, visto que uma maior quantidade de operações de *join* e *leave* é realizada durante a transmissão (DARONCO, 2009).

Roesler (2003) chegou à conclusão de que é praticamente impossível manter a estabilidade quando existe tráfego TCP concorrente. Considerando-se a importância dos dois requisitos, um desafio a ser alcançado é desenvolver protocolos que consigam chegar a um meio termo, ou seja, satisfazer ambas as características sem comprometer demais as aplicações.

2.3.2.2 Equidade do Tráfego

Um dos principais objetivos dos protocolos de controle de congestionamento multicast é utilizar de forma justa e eficiente os recursos da rede, mantendo a estabilidade da Internet.

Devido à maioria do tráfego ser baseada em fluxos TCP (KAMMOUN; YOUSSEF, 2008), é fundamental que os novos protocolos trabalhem de forma cooperativa a ele, recebendo assim a denominação de protocolos TCP-*Friendly*.

Segundo Floyd e Fall (1999), um protocolo é considerado TCP-*Friendly* se a quantidade de dados transferida em longo prazo não exceder a quantidade de dados transferida por um fluxo TCP sob as mesmas condições. Entretanto, para muitos pesquisadores essa definição inicial é fraca, pois existem diversas variantes do TCP com características de desempenho diferentes.

Outra definição é sugerida por Widmer et al (2001) e diz que um fluxo é considerado TCP-*Friendly* quando ele não reduz o desempenho em longo prazo de uma conexão TCP mais do que outro fluxo TCP reduziria sob as mesmas condições.

No caso do multicast, existe uma indefinição sobre o que seria mais correto considerar TCP-*friendly*, visto que um mesmo pacote é aproveitado por vários receptores (ROESLER, 2003). Por um lado, existe a abordagem que considera o fluxo multicast como um fluxo unicast normal, devendo se comportar como tal. Por outro lado, existe o argumento que um fluxo multicast envia tantos dados quanto *n* fluxos unicast, sendo *n* o número de receptores compartilhando o mesmo enlace, portanto, poderia utilizar uma parcela de banda maior.

Para que seja possível compartilhar a banda de forma amigável com o TCP, os protocolos precisam utilizar mecanismos de controle similares ou saber quanto de banda equitativa a ele cada receptor possui. Conforme comentado na seção 2.2, a utilização de mecanismos baseado em janela não é conveniente para as aplicações multimídia devido à necessidade de ajustes suaves na taxa de transmissão.

Como o comportamento do TCP é conhecido, vários autores calcularam sua vazão analiticamente. Em Padhye et al (1998) e (2000), os autores calcularam um modelo completo e uma aproximação, levando em conta o tamanho da janela, a ocorrência de *timeouts* e os ACKs duplicados. A equação aproximada pode ser observada na figura 2.9.

$$B(p) \approx \min\left(\frac{W\max}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + To\min\left(1{,}3\sqrt{\frac{3bp}{8}}\right)p\left(1+32p^2\right)}\right)$$

Figura 2.9: Fórmula da equação TCP.

Na equação, *B(p)* é a vazão equivalente a uma conexão TCP em função das perdas. *Wmax* é o valor máximo da janela de congestionamento, RTT é o tempo de ida do pacote e a volta do respectivo ACK e *b* é o número de pacotes que o receptor espera para enviar o ACK. A variável *p* corresponde à taxa de perdas ocorridas na transmissão, enquanto *To min* equivale ao tempo de *timeout* (RTO) da sessão TCP.

Pode-se observar que as principais variáveis da equação são a taxa de perdas e o atraso (incluindo o RTT e o RTO). Sabendo-se esses valores, é possível aplicar a fórmula e descobrir a quantidade de dados transferida por um fluxo TCP nas mesmas condições, mantendo a taxa de transmissão abaixo desse valor.

Contudo, é importante observar que competir pelo mesmo gargalo não significa ter as mesmas condições de rede que o TCP (TSAO et al, 2007). A diferença na taxa de perdas observada pelos protocolos, por exemplo, costuma ser diferente e é um dos principais fatores que altera o *throughput* dos protocolos, dificultando a divisão equitativa da banda.

Em Injong e Lisong (2007), os autores provaram que quando dois protocolos competem pelo mesmo gargalo utilizando diferentes taxas de transmissão (como é o caso do TCP com qualquer protocolo de transmissão em camadas), suas taxas de perdas podem ser significativamente diferentes. Nesse caso, fluxos com menor taxa de transmissão observam uma maior quantidade de perdas, independente do protocolo utilizado. Portanto, mesmo empregando a fórmula do TCP, os protocolos baseados em taxa podem obter *throughput* diferente que ele.

Estudos anteriores também compartilham a mesma opinião. Yang e Simon (2000) consideram que em um ambiente FIFO com muitos fluxos concorrentes é razoável assumir que a taxa de perda experimentada por cada fluxo seja independente da taxa de envio do transmissor. Contudo, em ambientes com baixa multiplexação de fluxos, a taxa de perdas de cada receptor irá depender da taxa de envio do transmissor.

Uma dificuldade adicional para se obter equidade de tráfego nas transmissões em camadas é em relação à granularidade da transmissão. Se as camadas tiverem uma diferença de taxa muito grande (granularidade grossa), fica difícil de situar o receptor de forma que ele receba um tráfego similar ao TCP. Se as camadas forem muito fina (granularidade fina), o multicast irá gerar muito tráfego de controle na rede.

Segundo Liu et al (2003), é impossível um protocolo com transmissão em camadas ser totalmente equitativo com o TCP, visto que a sua adaptação depende da granularidade das camadas utilizadas. Portanto, mais importante do que tentar imitar o comportamento do TCP é procurar dividir a banda de forma amigável durante a transmissão, respondendo diante as indicações de congestionamento.

2.3.2.3 Escalabilidade

Conforme verificado na seção anterior, o conhecimento do RTT é fundamental para o cálculo da banda equivalente do TCP. No entanto, para que os protocolos consigam calcular o RTT com precisão e se manter equitativos ao longo do tempo, devem se comunicar em intervalos regulares com o transmissor.

Em ambientes de larga escala, isso gera uma complexidade adicional que é controlar o fluxo de mensagens (pedidos de *feedback*) por parte dos receptores, evitando que o excesso de pedidos gere um problema conhecido como "implosão de *feedbacks*" (CARVALHO et al, 2008), esgotando os recursos do transmissor.

Portanto, um protocolo é considerado escalável quando pode ser utilizado em redes multicast com uma grande quantidade de receptores simultaneamente (WIDMER; HANDLEY, 2001), sem impactar no desempenho do sistema.

A figura 2.10 apresenta o método normalmente utilizado para realizar o cálculo de RTT em transmissões multicast. Os pacotes que saem do transmissor em direção ao receptor representam a transmissão multimídia em questão (pontos "1" da figura).

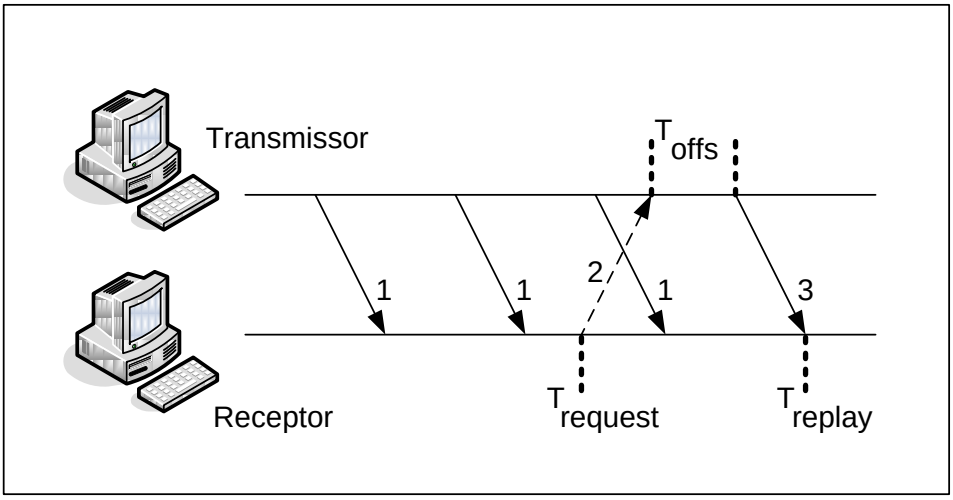

Figura 2.10: Cálculo do RTT.

Periodicamente, cada receptor se comunica com o transmissor visando calcular o seu RTT (ponto "2" da figura 2.10), enviando um pacote com o tempo de saída do mesmo no receptor (*Trequest*). Após receber o pacote, o transmissor retorna enviando o *Trequest* e a diferença de tempo entre a chegada do pedido e a transmissão da resposta (*Toffs*). Quando o pacote chega novamente no receptor (*Treplay*), o RTT é calculado através da fórmula apresentada na figura 2.11(no ponto "3" da figura 2.10).

$$RTT = t_{replay} - t_{request} - t_{offs}$$

Figura 2.11: Fórmula do RTT.

Na tentativa de proporcionar escalabilidade aos protocolos, diversos mecanismos foram sugeridos para estimar o RTT de forma independente ou controlar a quantidade de mensagens enviadas pelos receptores simultaneamente.

Uma proposta para o cálculo do RTT é realizar uma estimativa *one-way*, onde o transmissor adiciona um *timestamp* com o instante exato da transmissão em todos os pacotes enviados (BIAN et al, 2006). Os receptores calculam o RTT através do tempo

de chegada do pacote e o respectivo *timestamp*. A desvantagem dessa técnica está no fato de não gerar um RTT exato quando existe assimetria na rede e diferença entre os relógios do transmissor e receptor, que precisam estar sincronizados.

Atualmente, a solução mais utilizada pelos protocolos para calcular o RTT é através de um cálculo híbrido composto por duas etapas (CARVALHO et al, 2008): a) laço fechado - quando o RTT é calculado com precisão a partir da troca de informações entre transmissor e receptor e b) laço aberto - quando o RTT é calculado com base no *timestamp* enviado pelo transmissor e em parâmetros calculados no laço fechado.

O controle de *feedbacks* em transmissões multicast é realizado basicamente por três mecanismos, que podem ser utilizados em conjunto para aumentar a efetividade do controle das mensagens enviadas:

- ***Timer-Based* + *Supression* -** tem como objetivo limitar o número de mensagens enviadas pelos receptores e evitar que um número muito grande de mensagens chegue ao mesmo tempo no transmissor. Para isso, os receptores recebem do transmissor um período (T) para envio das mensagens e calculam um *timer* aleatório dentro desse intervalo [0 a T]. Caso um receptor receba uma mensagem indicando que outro receptor já enviou um pedido de *feedback* neste intervalo, cancela o seu *timer* e suprime o envio da mensagem dentro deste período;
- ***Representative* -** utiliza um modelo hierárquico, baseado na seleção de um representante para um grupo de receptores. Esse representante é o responsável por concentrar todas as informações do grupo, repassando-as para o transmissor;
- ***Neighboring Measurement* -** técnica utilizada para efetuar a troca de informações entre os usuários de uma mesma rede local. Esse método assume que todos receptores dentro da mesma rede possuem atrasos e banda semelhantes. Sendo assim, utiliza uma cooperação entre os receptores para que uma vez calculado o RTT em qualquer um deles, todos sejam atualizados, otimizando o cálculo e reduzindo a quantidade de mensagens necessárias.

2.4 Projeto SAM

O SAM (Sistema Adaptativo Multimídia) consiste numa ferramenta de transmissão e recepção de sinais multimídia através de redes de computadores. Seu maior benefício é aumentar o desempenho e a acessibilidade da transmissão, utilizando para isso a técnica de transmissão em camadas.

Conforme pode ser visualizado na figura 2.12, o SAM é composto por duas ferramentas: um transmissor multicast (formado pelos blocos “Codificador em camadas” e “Transmissão”) e um receptor (formado pelos blocos “Recepção” e “Decodificador em camadas”).

O transmissor codifica o sinal multimídia em diferentes taxas de transmissão e as envia em grupos multicast separados. No lado do receptor, o protocolo ALMTF realiza o controle de fluxo e congestionamento, determinando a quantidade de camadas mais apropriada para cada um, mantendo a justiça com os demais tráfegos concorrentes.

O ALMTF é o protocolo mais importante do SAM, sendo considerado o componente fundamental do sistema. A adaptação dos usuários é realizada com base na detecção de perdas, equidade e estimativa de banda máxima.

Para inferir a banda máxima da rede, o protocolo ALMTF utiliza o mecanismo de pares de pacotes. Conforme o nível de congestionamento e a banda disponível na rede, ele determina a quantidade de camadas que o receptor poderá suportar.

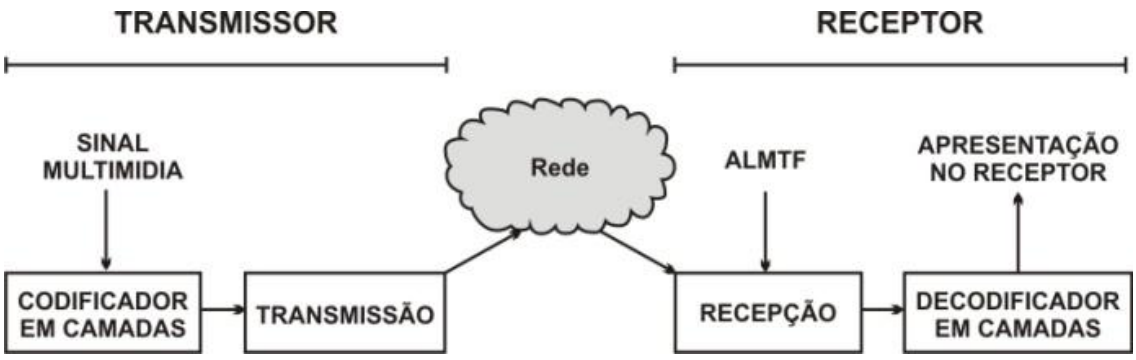

Figura 2.12: Arquitetura do sistema SAM.

No SAM as camadas são cumulativas, ou seja, a próxima adiciona qualidade à anterior e o receptor inscreve-se em tantas camadas quanto a sua condição permitir.

Existe também a possibilidade de utilizar camadas opcionais, que servem para transmissão de fluxos independentes como a dublagem em outro idioma ou taxas alternativas de um mesmo vídeo. Após os dados serem recebidos, os mesmos são decodificados e então entregues ao usuário.

3 TRABALHOS RELACIONADOS

Conforme visto anteriormente, o desenvolvimento de protocolos de controle de congestionamento é uma tarefa árdua, no entanto, indispensável para que o multicast possa ser utilizado na prática. Muitas propostas têm sido sugeridas nos últimos anos, apesar disso, ainda não existem implementações efetivas disponíveis para uso desses protocolos (DARONCO, 2009).

Este capítulo tem como objetivo apresentar uma visão geral da evolução dos principais protocolos de controle de congestionamento multicast, descrevendo o seu funcionamento, vantagens e desvantagens na sua utilização.

3.1 RLM (*Receiver-driven Layered* Multicast)

O protocolo RLM (MCCANNE; JACOBSON; VETTERLI, 1996) foi proposto em 1996, sendo o primeiro mecanismo a utilizar a técnica de transmissão em camadas. Para ajustar a taxa de transmissão dos receptores, o protocolo introduziu o método de experimento de *join*. Seu funcionamento é simples e pode ser observado na figura 3.1.

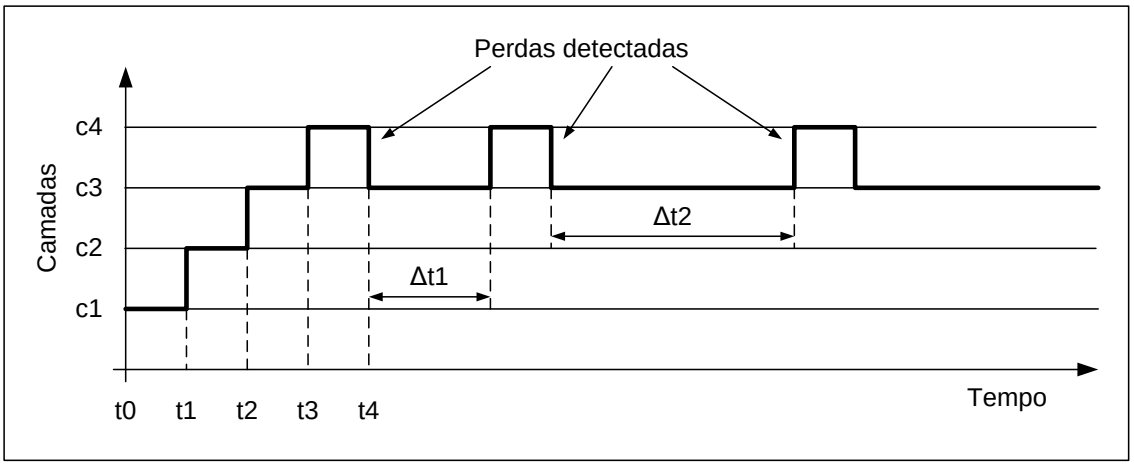

Figura 3.1: Funcionamento do Protocolo RLM.

O receptor inicia a execução e efetua *join* na camada mais básica. Após certo período, caso não ocorra perdas, faz *join* na próxima camada e assim sucessivamente até a ocorrência de perdas, efetuando o *leave* da camada mais alta. A cada intervalo de tempo tenta subir uma camada novamente, sendo que na ocorrência de perdas as tentativas vão ficando mais esparsas (intervalos Δt1 e Δt2).

Na tentativa de solucionar o problema de sincronismo de *join*, utiliza um mecanismo de aprendizado coletivo, onde cada receptor envia uma mensagem para o

grupo avisando quando vai fazer *join* em uma determinada camada. Dessa forma, todos zeram seus contadores de novas tentativas para não interferirem entre si.

O RLM possui várias deficiências. A principal delas é que de tempos em tempos ele gera congestionamento na rede, pois mesmo que a banda já esteja totalmente ocupada, continua tentando se inscrever em novas camadas (ROESLER, 2003). Além disto, é instável e não gera tráfego equitativo com o TCP (KALAIRASAN; SELVAN, 2006).

3.2 RLC (*Receiver-driven Layered Congestion Control*)

O protocolo RLC (VICISANO; RIZZO; CROWCROFT, 1998) foi desenvolvido em 1998, com o propósito de resolver os problemas do RLM. Da mesma forma que o anterior, também utiliza transmissão em camadas e é orientado a receptor.

As principais inovações do RLC são o uso de codificação FEC (*Forward error correlation*) para a transmissão multicast confiável e a utilização de dois métodos simultâneos para adaptar a taxa de transmissão dos receptores: experimentos de *join* sincronizados e testes de *burst* (PUANGPRONPITAG et al, 2003).

O funcionamento do RLC pode ser observado na figura 3.2. No primeiro intervalo, o protocolo envia uma rajada de dados com taxa equivalente ao dobro de pacotes que seriam transmitidos normalmente (teste de *burst*). No segundo intervalo não são enviados pacotes, para manter a média de transmissão. E nos demais, os pacotes são transmitidos conforme a taxa de cada camada. Neste esquema, cada camada tem o dobro da anterior e as perdas geram uma redução multiplicativa na taxa de transmissão dos dados.

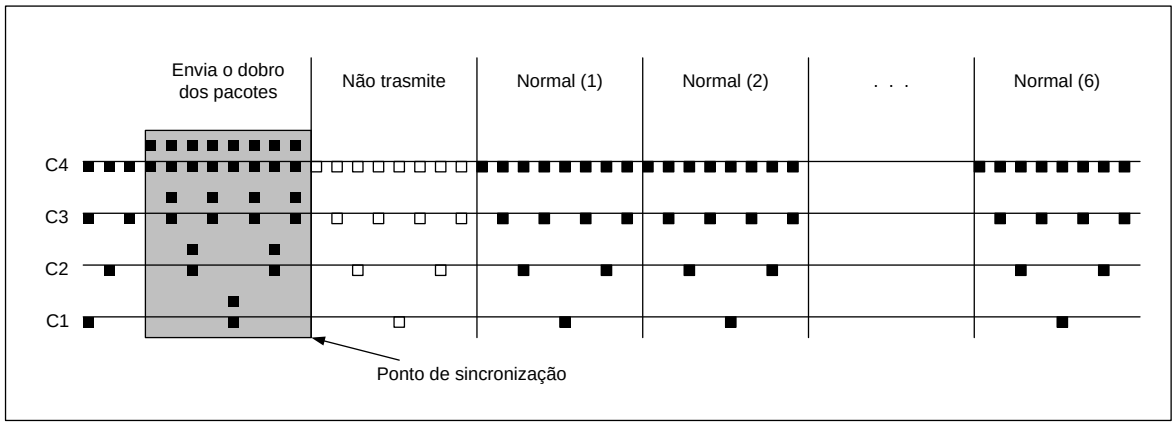

Figura 3.2: Funcionamento do Protocolo RLC.

Para prevenir a ineficiência das ações não coordenadas dos receptores atrás do mesmo gargalo, o protocolo utiliza uma coordenação implícita como ponto de sincronização. Os receptores somente efetuam *join* quando o ponto de sincronismo permitir, sendo que quando ele acontece em uma camada, ocorre em todas as camadas abaixo dela simultaneamente (LI et al, 2005).

Após algumas simulações com o algoritmo, Roesler (2003) verificou que o mesmo pode não funcionar corretamente, devido às rajadas serem absorvidas pelas filas dos roteadores intermediários. Esse problema permite que os receptores subam de camada mesmo quando não existe banda suficiente para isto.

Outro problema verificado por PUANGPRONPITAG et al (2003) é o fato do protocolo não gerar tráfego equitativo com o TCP, além de ter uma resposta lenta aos congestionamentos.

3.3 FLID-DL (*Fair Layered Increase/Decrease with Dynamic Layering*)

Byers et al (2000) desenvolveu o protocolo FLID-DL com base no protocolo RLC. O protocolo tem como objetivo suavizar as mudanças na taxa de transmissão e evitar o problema do tempo de *leave* causado pelas frequentes tentativas de *join*. Para isso, introduziu o conceito de camadas dinâmicas (LI et al, 2005).

Com as camadas dinâmicas, a largura consumida por uma camada diminui ao longo do tempo, então o receptor precisa se inscrever em novas camadas periodicamente para manter sua taxa de recepção. A taxa de cada camada pode ser igualitária (taxas iguais), exponencial (a camada superior tem o dobro da anterior) ou multiplicativa.

O FLID manteve os mecanismos de sincronização e o experimento de *join* do protocolo RLC, contudo, não utiliza a técnica de *burst* (PUANGPRONPITAG et al, 2003).

O protocolo funciona da seguinte maneira: o transmissor divide o tempo em fatias de segundos e todo pacote é marcado com o número da fatia de tempo. Para o receptor, uma nova fatia de tempo começa quando o primeiro pacote da nova fatia de tempo chega. Se durante uma fatia de tempo o receptor detectar perdas, efetua *leave* na camada superior.

Para evitar o problema do sincronismo, os receptores podem tentar subir de camada somente no início de uma nova fatia de tempo, dependendo de um sinal de incremento enviado pelo transmissor nos pacotes. Por exemplo, caso um receptor esteja inscrito na camada “n” e o sinal de incremento for maior que “n”, então poderá subir de camada.

Na tentativa de manter a justiça com os demais fluxos concorrentes, utiliza um conjunto de probabilidades de incremento de camada, de forma que a probabilidade das camadas inferiores seja maior do que para as camadas superiores.

A escalabilidade é alcançada devido a nenhum cálculo de RTT ser realizado pelos receptores, sendo o mesmo fixo em 100ms. No entanto, os próprios autores admitem que isso torna o protocolo ineficiente em ambientes onde os atrasos são variáveis e significativos em relação à latência fim-a-fim dos pacotes.

Simulações realizadas por Roesler (2003) e Kalaiarasan e Selvan (2006) confirmam que o protocolo não é TCP-*Friendly*, podendo obter de 4 a 8 vezes mais vazão que o tráfego TCP concorrente.

3.4 PLM (*Packet Pair Receiver-driven cumulative Layered* Multicast)

O protocolo PLM (LEGOUT; BIERSACK, 2000) também utiliza transmissão em camadas, sendo baseado no receptor. O diferencial deste protocolo é a utilização da técnica de pares de pacotes (PP) para estimativa da banda máxima da rede.

A técnica de PP consiste no envio de pacotes em pares e na observação do espaçamento causado pelo limite físico do enlace entre eles. Com base nesse espaçamento e no tamanho do pacote, é possível estimar a largura de banda entre a origem e o destino.

No entanto, para que a largura de banda seja determinada com precisão, o protocolo PLM exige que todos os roteadores intermediários utilizem filas do tipo WF^2Q (*Worst-case Fair Weighted Fair Queueing*) (LI et al, 2005). Esse tipo de fila agrega muitas facilidades para o controle de congestionamento, como o auxílio para manter a justiça com os demais fluxos concorrentes.

Os receptores ajustam a taxa de transmissão dos dados conforme o resultado obtido através do PP, sendo que a análise de subida ou descida das camadas é realizada apenas uma vez por segundo.

Se a estimativa de banda resultar em um valor menor que o utilizado, deixa uma camada. Caso contrário, poderá efetuar uma tentativa de *join*. Contudo, a subida é realizada somente em determinados intervalos de tempo, considerando o valor mínimo de estimativas de banda. Isso torna o algoritmo bastante conservador, tendo mais facilidade para reduzir do que para aumentar a taxa de transmissão (ROESLER, 2003).

O cálculo de RTT é realizado através do intervalo de tempo entre uma operação de *join* e o recebimento do primeiro pacote referente à nova camada. Apesar de melhorar a escalabilidade, essa técnica possui vários problemas, como não contabilizar o atraso entre o transmissor e o primeiro roteador da rede e não funcionar quando o tráfego multicast já se encontrar nos roteadores intermediários (CARVALHO et al, 2008).

Além disso, experimentos realizados com outros tipos de filas mostraram que o protocolo causa a degradação do tráfego TCP concorrente (KALAIRASAN; SELVAN, 2006). Portanto, o fato de nem todos os roteadores utilizar filas do tipo WF^2Q inviabiliza a sua utilização na prática.

3.5 TFMCC (*TCP-Friendly Multicast Congestion Control*)

O TFMCC (WIDMER; HANDLEY, 2001) foi desenvolvido utilizando um mecanismo baseado em taxa e com adaptação no transmissor, sendo, portanto, um protocolo de taxa única. Seu objetivo é melhorar a estabilidade da transmissão, provendo uma mudança suave e sem oscilações no recebimento dos dados.

Para ajustar a taxa de envio, o transmissor necessita de um retorno contínuo dos receptores, de forma que seja possível adaptar a transmissão com base no receptor mais lento do grupo.

Visando a evitar problemas de escalabilidade, o protocolo não permite que os receptores enviem retornos quando a taxa calculada por ele for maior do que a taxa sendo transmitida pelo transmissor. Os receptores calculam a sua taxa de transmissão usando a fórmula da equação do TCP apresentada na figura 3.3.

$$B(p) \approx \frac{s}{tRTT\sqrt{\frac{2p}{3} + \left(12\sqrt{\frac{3p}{8}}\right) \times p\left(1 + 32p^2\right)}}$$

Figura 3.3: Equação utilizada pelo TFMCC.

Na figura 3.3, *tRTT* representa o tempo de RTT, que é calculado através do *timestamp* dos pacotes recebidos, *p* representa as perdas, sendo contabilizadas através de eventos de perda em vez de pacotes perdidos (nesse caso, todas as perdas no mesmo RTT são consideradas como uma única perda) e *s* é o tamanho do pacote (em *bytes*).

O protocolo funciona da seguinte maneira: cada receptor avalia a sua taxa de perdas e o RTT até o transmissor. Com base nos valores observados, os receptores calculam a taxa equitativa (conforme a equação do TCP) e enviam o resultado para o transmissor. O transmissor ajusta a taxa de envio do grupo com base nos retornos recebidos dos receptores, mantendo a taxa baseada no receptor mais lento do grupo.

Segundo Roesler (2003), o protocolo TFMCC é razoavelmente justo e tem uma baixa instabilidade na recepção dos dados, se tornando uma boa opção para as aplicações do tipo *streamings* de vídeo. No entanto, não utiliza de maneira eficiente os recursos disponíveis na rede, não agradando usuários em ambientes heterogêneos.

3.6 SMCC (*Smooth Multirate Multicast Congestion Control*)

O protocolo SMCC (KWON; BYERS, 2003) foi o primeiro protocolo híbrido a tentar combinar os benefícios dos protocolos de taxa única (controle suave da taxa e TCP-*Friendly*) com o uso de múltiplas taxas.

As camadas empregadas são cumulativas, como na maioria dos protocolos multi-taxa. No entanto, utiliza o protocolo TFMCC como mecanismo de controle interno dentro de cada camada, ajustando a taxa de cada sessão de forma independente. A taxa de transmissão interna de cada camada é baseada no receptor mais lento daquela sessão, sendo alterada conforme os participantes do grupo.

Isso possibilita que as camadas utilizem taxas dinâmicas, melhorando a adaptabilidade dos usuários sem prejudicar a estabilidade da transmissão.

O protocolo trabalha da seguinte maneira: cada receptor calcula o seu *throughput* conforme a equação do TCP apresentada na seção 3.5. Se a banda obtida via equação for maior que a taxa máxima da camada onde ele está inscrito, ele se inscreve na próxima camada. Caso a banda seja menor e estiver abaixo do limite mínimo da camada atual, o receptor deixa-a.

Uma desvantagem deste método é que ele requer a configuração estática da taxa mínima e máxima de cada camada. Portanto, permite que o receptor se adapte automaticamente, mas sempre dentro dos limites impostos pela aplicação. Além disso, o protocolo pode requerer mais camadas do que o necessário para o receptor atingir o *throughput* desejado (LI et al., 2007).

3.7 GMCC (*Generalized Multicast Congestion Control*)

O protocolo GMCC (LI et al., 2007) é um protocolo híbrido desenvolvido com base no SMCC. As principais inovações do protocolo são:

- É totalmente adaptável. A taxa de transmissão de cada camada pode ser ajustada sem limites rígidos. Além disso, permite a criação e a finalização automática das camadas, melhorando a adaptação dos receptores;
- A quantidade de camadas utilizadas varia conforme as necessidades dos receptores, mas o transmissor pode controlar o *throughput* total da aplicação multicast limitando o número de camadas que serão usadas. Por exemplo, se somente uma camada for permitida, o protocolo irá trabalhar como um esquema de controle de congestionamento multicast de taxa única;
- Não é integrado com nenhum mecanismo específico, como o TFMCC, podendo ser substituído por qualquer outro protocolo de taxa-única.

Em qualquer seção GMCC sempre haverá uma camada básica na qual o transmissor envia os pacotes. As demais camadas só são ativadas quando solicitadas por um receptor, evitando assim o desperdício de banda. As camadas utilizadas são sempre cumulativas.

O protocolo trabalha da seguinte maneira: em cada camada, o transmissor escolhe o receptor mais congestionado (CR) e ajusta a taxa de envio de acordo com esse receptor. Quando algum receptor detectar que está muito menos congestionado que o CR da camada superior, deduz que ele pode receber uma taxa maior e se inscreve em uma nova camada. Por outro lado, caso detecte que ele é o CR em mais de uma camada, deixa a camada mais alta na qual estava inscrito.

Para detectar quem é o usuário mais congestionado, este protocolo utiliza a métrica TAF (*Throughput Attenuation Factor*), que indica o nível de congestionamento de cada receptor (LI et al., 2007). Os receptores podem também decidir se fazem *join* ou *leave* nas camadas com base em estatísticas. As estatísticas podem ser usadas apenas se um número mínimo de amostras TAF foi coletado e se cada camada tem um CR.

Uma limitação do GMCC é que ele não é TCP-*Friendly* considerando a soma de todas as camadas recebidas pelo usuário. Os autores consideraram a equidade por fluxo uma aproximação razoável para utilização na Internet.

No entanto, a principal deficiência do protocolo GMCC é a sua limitação de uso na prática, visto que necessita de mecanismos que permitam a criação das camadas em tempo real, bem como a variação interna dentro delas (DARONCO, 2009).

4 PROTOCOLO ALMTF

Segundo Roesler (2003), o principal objetivo do ALMTF é transmitir de forma equitativa com o TCP, porém, mantendo a estabilidade durante a transmissão. Este capítulo analisa em detalhes o algoritmo ALMTF, apresentando os mecanismos utilizados pelo protocolo para tentar alcançar esse objetivo.

O controle de fluxo e de congestionamento é realizado por dois mecanismos complementares: um baseado em janela (imitando o comportamento do TCP) e outro baseado em taxa (utilizando a equação do TCP). O mecanismo de janelas é o principal, enquanto a equação é utilizada como apoio na tomada de decisões.

Da mesma forma que o TCP, o ALMTF é executado a cada RTT. Ou seja, em redes rápidas ele é executado mais vezes, gerado um número maior de alterações na sua taxa de transmissão. Em redes mais lentas (com um RTT maior), o algoritmo leva mais tempo para decidir se deve ou não alterar a sua taxa, resultando em um comportamento mais estável.

As principais variáveis utilizadas pelo algoritmo são:

- **Bwjanela**: representa a taxa máxima (em bits/s) que o receptor poderá receber, inferida pelo mecanismo de janelas. A cada execução, o algoritmo atualiza a variável *bwjanela*, e decide se deve fazer *join* ou *leave* com base nesta variável;
- **Cwnd:** determina o tamanho da janela de congestionamento atual do protocolo (em pacotes). Está diretamente relacionada com a variável *bwjanela*;
- **Bweq**: representa a taxa máxima (em bits/s) que o receptor poderá utilizar, inferida pelo mecanismo da equação do TCP. Este valor reflete a taxa de transmissão utilizada por um fluxo TCP nas mesmas condições;
- **RTT**: determina o tempo entre duas execuções consecutivas do ALMTF. Também é utilizado para calcular o valor de *Bweq*, visto que o RTT é um dos principais componente da fórmula da equação;
- **Bwmax**: representa a banda máxima da rede, inferida pelo método de pares de pacotes. Quando utilizado, limita o valor de *bwjanela* e *bweq* neste valor, evitando que o receptor utilize uma banda maior que a disponível na rede.

O ALMTF possui duas fases: *start-state* e *steady-state*. Durante a primeira fase, o algoritmo visa a alcançar a banda equitativa com os demais fluxos concorrentes, e faz isso incrementando *bwjanela* em 50% a cada RTT. Após a primeira perda ser detectada,

assume que ultrapassou o seu valor equitativo, então reduz *bwjanela* em 30% e vai para a fase *steady-state*.

Durante a segunda fase, a meta do algoritmo é manter a equidade e a estabilidade ao longo da transmissão. O ALMTF assume que é possível alcançar este objetivo sendo dez vezes menos agressivo que o TCP, tanto para aumentar como para reduzir a sua taxa de transmissão. Portanto, incrementa *cwnd* em 0,1 a cada RTT (em vez de 1) e reduz 5% a cada perda (em vez de 50%).

Após iniciar as variáveis e se cadastrar na primeira camada, o algoritmo chama a sub-rotina ALM-IE, que é executada a cada RTT. Esta sub-rotina é repetida consecutivamente, de acordo com o RTT calculado pelo receptor. Caso o RTT seja maior, a sub-rotina é executada menos vezes, refletindo o que acontece com uma conexão TCP.

Como o algoritmo não possui conhecimento prévio do estado da rede, algumas variáveis iniciam com um valor fixo, com é o caso de *bweq* (que inicia com 1000 bits/s) e o RTT (que inicia com 500ms). Por definição, a variável *cwnd* inicia com 1 pacote.

A cada RTT, as variáveis são atualizadas e o algoritmo decide se deve aumentar ou reduzir a sua taxa de transmissão. Caso a banda inferida pelo mecanismo de janela seja maior que a banda atual recebida pelo receptor, faz *join* na próxima camada. Caso a banda seja menor que taxa utilizada, o algoritmo deve deixar a camada mais alta. Contudo, o algoritmo somente irá descer de camada se a banda obtida pelo mecanismo de equação também for menor que a taxa atual, caso contrário, ele manterá a mesma taxa.

Com o objetivo de deixar o protocolo mais estável, sempre que houver a troca de camada (subida ou descida), a variável *bwjanela* é atualizada com a média aritmética das suas camadas adjacentes. Dessa forma, não fica muito próxima da banda necessária para trocar de camada novamente. Além disso, existe uma proteção que impede que o algoritmo suba mais de uma camada por segundo.

Em determinadas ocasiões o ALMTF ignora perdas, como após uma condição de congestionamento, onde é necessário um tempo para que a rede consiga se estabilizar. Este tempo foi definido em 1 segundo.

As próximas seções descrevem os mecanismos do ALMTF em seu projeto original.

4.1 Controle de Fluxo e Congestionamento

Para criar um mecanismo baseado em janela semelhante ao TCP, o ALMTF utiliza a variável *cwnd*, que representa o tamanho da janela de congestionamento e reflete a variável *bwjanela*, indicando a quantidade de banda permitida para o receptor se cadastrar.

A relação entre *cwnd* e *bwjanela* é dada pela função *win2rate()*, que efetua o seguinte cálculo:

$$bwjanela = \frac{cwnd \times packetsize \times 8}{RTT}$$

Este cálculo representa a banda utilizada em uma conexão TCP equivalente para um determinado tamanho de janela. Por exemplo, se o RTT é 100 ms e o tamanho de pacote é 500 *bytes*, tendo *cwnd* igual a 2 pacotes, a banda será igual a 80 kbit/s.

Em alguns momentos é necessário efetuar o inverso, ou seja, passar de taxa (bit/s) para o equivalente em janela de congestionamento (pacotes). Isso acontece, por exemplo, quando é necessário adaptar a banda da janela de acordo com o valor obtido via equação do TCP. Para que *cwnd* reflita o valor da banda da janela é utilizada a função *rate2win()* a seguir:

$$cwnd = \frac{bwjanela \times RTT}{packetsize \times 8}$$

Por exemplo, se a variável *bwjanela* for 400 kbit/s, com tamanho do pacote de 500 bytes e RTT de 100ms, o valor equivalente para *cwnd* será 10.

Sempre que o mecanismo baseado em janela permitir, o receptor pode efetuar *join* na camada superior. Caso o mecanismo descubra que não possui banda para sustentar a camada atual, o receptor consulta o mecanismo de equação antes de deixar uma camada. Se o valor inferido pela equação for maior que a banda atual, ele mantém a taxa.

Uma proteção é empregada para evitar que os valores de *bwjanela* e *bweq* fiquem muito distantes entre si. Neste caso, se o valor inferido pelo mecanismo baseado em janela for maior que o dobro do mecanismo da equação, o protocolo altera o valor de *cwnd* para o dobro do valor da equação.

O mecanismo baseado na equação é semelhante ao empregado no protocolo TFMCC (WIDMER; HANDLEY, 2001). A cada pacote recebido, é executada uma função que retorna o valor estimado de uma conexão TCP na mesma circunstância. A figura 4.1 mostra a fórmula utilizada pelo ALMTF.

$$B(p) \approx \frac{packetsize}{RTT\sqrt{\frac{2p}{3} + \left(1{,}3\sqrt{\frac{3p}{8}}\right) \times 4 \times RTT \times p\left(1 + 32p^2\right)}}$$

Figura 4.1: Fórmula da equação TCP.

Como se pode verificar, é necessário saber o tamanho do pacote (*bytes*), o valor do RTT e as perdas (*p*). O ALMTF utiliza um tamanho de pacote fixo, definido no arquivo de configuração. O cálculo de RTT será descrito a seguir e o cálculo das perdas é baseado no número de seqüência dos pacotes, sendo detalhado na seção 4.3.

4.2 Cálculo do RTT e Intervalo de *Feedbacks*

O receptor se comunica em intervalos regulares com o transmissor, a fim de calcular o RTT com precisão. Contudo, o número de mensagens enviadas é limitado, a fim de garantir escalabilidade e evitar o problema de implosão de *feedbacks*.

O cálculo do RTT pode ser realizado de duas maneiras: por estimativa ou com precisão. A estimativa do RTT (técnica conhecida como laço aberto ou RTO) é realizada a cada pacote de dados recebido do transmissor. Neste caso, utiliza-se o *timestamp* contido nos pacotes para calcular a variação do RTT.

Para calcular o RTT com precisão (técnica conhecida como laço fechado), o ALMTF utiliza um método no qual os relógios do transmissor e receptor não precisam estar sincronizados. No início da transmissão e a cada intervalo entre *feedbacks*, o receptor envia ao transmissor um pacote denominado “Solicitação de Echo”. Neste

pacote, os seguintes campos são utilizados: a) *Timestamp*: instante em que o receptor enviou o pacote e b) *Num_nodereceptor*: identificação do receptor que enviou o pacote.

O transmissor recebe o pacote e responde a solicitação preenchendo no cabeçalho do próximo pacote de dados os seguintes campos: a) *Timestamp*: instante que o transmissor está enviando o pacote; b) *Timestamp_offset*: diferença de tempo entre a chegada do pedido e a transmissão da resposta; c) *Timestamp_echo*: cópia do *timestamp* obtido no pacote do receptor e d) *Num_nodereceptor*: identificação do receptor que solicitou o cálculo de RTT.

As respostas aos pedidos de *feedbacks* são sempre enviadas na camada zero. É necessário que o pacote seja enviado nesta camada para que todos os receptores possam receber o pacote. Através desta informação, o receptor está apto a calcular o RTT de forma precisa no momento que chega o pacote, conforme mostra a figura 4.2 a seguir.

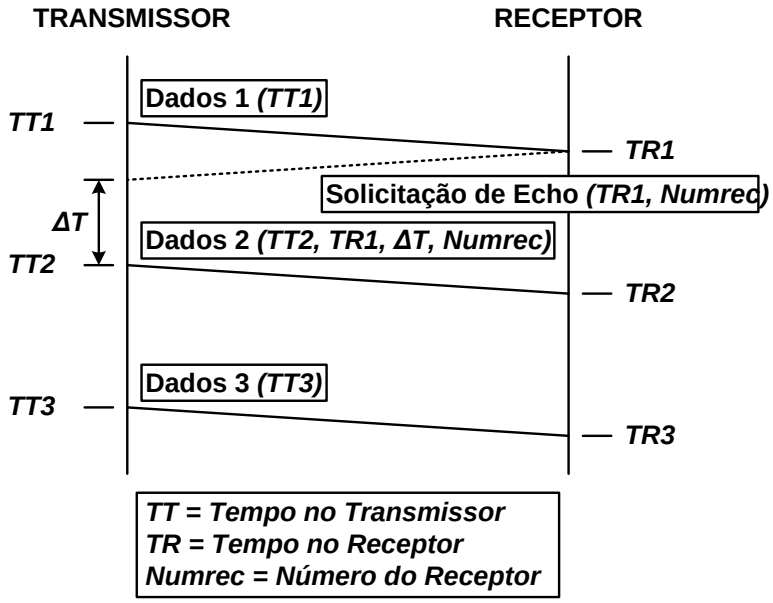

Figura 4.2: Cálculo do RTT.

Na figura 4.2, o transmissor envia três pacotes de dados (Dados 1, Dados 2 e Dados 3). No cabeçalho de todos os pacotes o transmissor envia seu *timestamp*, representado por TT1, TT2 e TT3. Na chegada do primeiro pacote, o receptor envia um pacote de solicitação de *echo*, contendo no cabeçalho seu *timestamp* (TR1) e sua identificação (*numrec*).

Quando o transmissor recebe a solicitação, ele armazena seu tempo de chegada e calcula a diferença de tempo entre a chegada da solicitação e a transmissão do novo pacote (ΔT). No próximo pacote de dados, o transmissor envia, além do seu *timestamp*, o tempo do receptor (TR1), a identificação do receptor e ΔT, conforme visto no pacote "Dados 2" da figura 4.2.

Para aumentar a estabilidade das medições de RTT, o ALMTF utiliza um filtro que evita variações bruscas. Neste caso, se o RTT calculado for menor que o RTT atual (ou seja, se a banda tende a aumentar), utiliza 90% do valor anterior e 10% do novo, evitando aumentos bruscos. Caso o RTT calculado seja maior que o RTT atual (a banda tende a diminuir), só altera o valor se a diferença for maior que 25% do atual, evitando que ocorram reduções de camadas devido a pequenas alterações na rede.

Para controlar os *feedbacks* os receptores devem aguardar um intervalo de tempo antes de enviar novamente suas solicitações da seguinte maneira: até 60 receptores, o total de mensagens deve ser limitado a uma por segundo. Se existir apenas um receptor, o mesmo vai enviar uma mensagem por segundo ao transmissor, obtendo 60 atualizações por minuto. Se existirem 60 receptores, cada um deles irá enviar uma mensagem a cada 60 segundos, obtendo uma atualização por minuto. Entretanto, acima de 60 receptores, mantém-se a mesma atualização por minuto, a fim de evitar uma espera muito grande para o cálculo do RTT com precisão.

4.3 Cálculo das Perdas e Detecção de *Timeout*

A detecção de perdas é realizada com base no número de seqüência existente em cada pacote. Caso o receptor detecte uma perda, ele incrementa uma variável (*nunloss*), que reflete a quantidade de perdas durante um determinado tempo.

Para realizar o cálculo das perdas utilizado no mecanismo da equação, o ALMTF considera os oito últimos eventos de perda. Quando ocorre uma perda, o algoritmo primeiro verifica se já passou o RTT, se passou, inicia um novo evento de perda, caso contrário, mantém o mesmo evento (as perdas dentro do mesmo RTT são consideradas como uma única ocorrência de perda).

O cálculo das perdas pode ser observado na figura 4.3. Pode-se ver que durante o evento de perda 1 ocorreram duas perdas e chegaram 13 pacotes (visto através do número de círculos preenchidos, que representam os pacotes recebidos), durante o evento de perda 2 ocorreram três perdas e chegaram 24 pacotes, e durante o evento de perda 3 ocorreu apenas uma perda e chegaram 27 pacotes.

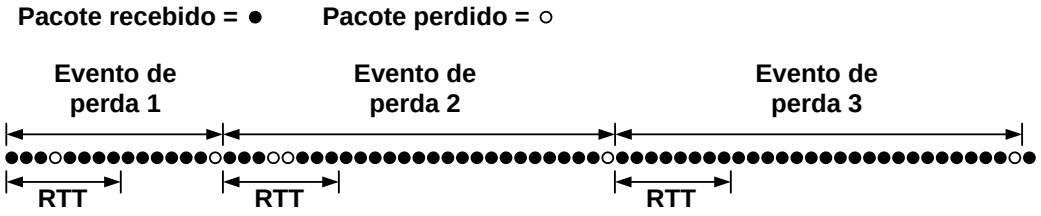

Figura 4.3: Criação do vetor “Evento de Perda”.

Através desse método, é gerado um vetor de oito entradas contendo o número de pacotes recebidos em cada evento de perdas, sendo que o valor inicial do vetor refere-se ao evento mais recente. No caso da figura 4.3, o vetor seria [27; 24; 13; 0; 0; 0; 0; 0].

No vetor de perdas, é dado um maior peso para os eventos mais recentes. O filtro utilizado se baseia nos 8 tempos anteriores, definindo um vetor da seguinte forma: [5; 5; 5; 5; 4; 3; 2; 1]. O valor de *p* será o inverso da média do valor dos eventos de perdas.

Para detectar *timeout*, o ALMTF se baseia no tempo esperado para chegada de 10 pacotes. Caso o receptor fique o tempo equivalente a 10 pacotes sem receber dados, considera que houve um *timeout* e reduz a variável *cwnd* para 1.

4.4 Inferência de Banda Máxima

O ALMTF utiliza a técnica de pares de pacotes (PP) para determinar a banda máxima que os receptores podem utilizar durante a transmissão.

O mecanismo se baseia no envio de pacotes aos pares (pelo transmissor) e na observação do espaçamento entre eles (no receptor). Sabendo-se o tamanho do pacote e o espaçamento entre eles, é possível descobrir a largura de banda entre a origem e o destino. A descrição completa deste método pode ser encontrada em Roesler (2003).

No entanto, o mecanismo de PP pode obter medições bastante variadas. Para evitar modificações bruscas, o ALMTF utiliza um filtro para obter o valor com maior probabilidade de estar correto.

Para isso, as dez primeiras medidas são simplesmente inseridas numa fila. O primeiro valor de banda máxima só é obtido após as dez primeiras medições, sendo a média entre os dez primeiros valores. Após este período, cada medida nova deve estar entre ±30% da média atual, caso contrário será descartada.

Estando dentro dessa faixa, a nova medida substitui a mais antiga da fila e a média é recalculada, obtendo-se o novo valor de *bwmax*. Caso ocorra dez descartes em seqüência (durante dez medições o valor obtido não estava entre ±30% da média atual), considera-se que a topologia da rede mudou por algum motivo e o processo é reiniciado do zero.

4.5 Sincronismo e Dessincronismo

Conforme explicado na seção 2.3, a sincronização entre os receptores da mesma sessão multicast pode melhorar a estabilidade da transmissão. No entanto, os receptores que estiverem recebendo sessões multicast diferentes não devem estar sincronizados.

O ALMTF utiliza pontos de sincronização para efetuar as tentativas de *join* dos receptores. Neste caso, mesmo que os receptores possuam banda suficiente para subir de camada, isto somente será realizado em instantes específicos, visando à sincronização dos receptores dentro da mesma sessão.

O sincronismo é realizado através da variável *sincjoin*, enviada pelo transmissor no do cabeçalho dos pacotes. A figura 4.4 a seguir mostra o valor de *sincjoin* em uma transmissão com cinco camadas. Os círculos representam a camada máxima na qual os receptores poderão efetuar tentativas de *join* em cada execução do algoritmo.

No primeiro intervalo de tempo (^*sinc1*), a variável *sincjoin* possui valor 2. Isto significa que todos os receptores que estiverem em camadas abaixo desta (camada 0 ou camada 1) poderão efetuar tentativas de *join*, caso possuam banda para isto. Contudo, os receptores que estiverem em camadas superiores (camada 3, 4 ou 5) não poderão realizar tentativas durante esta execução.

No próximo intervalo de tempo (^*sinc2*) a variável *sincjoin* possui valor 3. Portanto, todos os receptores que estiverem abaixo desta camada (camada 0, 1 ou 2) poderão efetuar novas tentativas, e assim por diante.

Para implementar o mecanismo de sincronismo, o ALMTF divide o número de camadas utilizado na transmissão por quatro e depois multiplica o valor resultante por um vetor de distribuição. O vetor de distribuição empregado no algoritmo possui os seguintes valores: [2; 3; 2; 4; 2; 3; 2; 5].

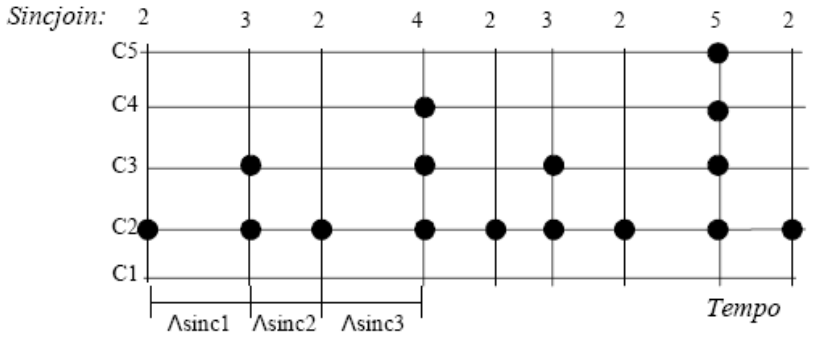

Figura 4.4: Pontos de sincronização no ALMTF.

A idéia do protocolo é fazer com que os pontos de sincronização aconteçam mais frequentemente nas camadas mais baixas, possibilitando que os usuários com menos banda realizem mais tentativas de *join*.

Os receptores que não tiverem banda para subir de camada ficam um período ignorando as perdas (no caso, 1 segundo, que é o tempo de estabilização estipulado no protocolo), pois poderá haver outros usuários aumentando as suas taxas e gerando perdas no enlace.

Para evitar a sincronização dos receptores de sessões diferentes, os transmissores enviam seus pontos de sincronismo em intervalos de tempos aleatórios, sendo que o intervalo médio para envio do novo sincronismo é de um segundo.

4.6 Melhorias Realizadas

Após o estudo aprofundado de diversos protocolos de controle de congestionamento multicast em camadas, percebeu-se que uma das principais preocupações dos algoritmos é manter a escalabilidade com o aumento de receptores.

Essa preocupação é justificável devido à necessidade de comunicação periódica entre os receptores e o transmissor, fundamental para o cálculo de RTT e a banda equivalente do TCP. Em uma sessão com um grande número de receptores cadastrados, o excesso de mensagens pode sobrecarregar o transmissor e ocasionar uma implosão de *feedbacks*, influenciando diretamente na escalabilidade da transmissão.

O principal mecanismo responsável pela escalabilidade é, portanto, o controle de *feedbacks*. O objetivo desse mecanismo é impedir que o transmissor fique sobrecarregado com o excesso de mensagens recebidas simultaneamente dos receptores e, ao mesmo tempo, evitar que os receptores permaneçam um grande intervalo de tempo sem uma estimativa precisa do RTT.

Conforme visto anteriormente, o ALMTF não utiliza nenhuma técnica especial para gerenciar os pedidos de *feedbacks* dos receptores, tendo seu controle baseado apenas no intervalo entre os envios de mensagens em cada receptor. Esse intervalo possui um limite mínimo de 1s e vai crescendo linearmente com a quantidade de receptores, até chegar ao intervalo máximo de um minuto entre cada atualização de RTT por receptor.

No entanto, o intervalo ótimo entre as atualizações de RTT depende das características do canal de comunicação entre o receptor e o transmissor. Se esse canal variar de forma significativa, uma sessão com mais de 60 receptores e intervalo médio

de atualização de um minuto fará com que o RTT fique desatualizado, ocasionando problemas de divisão equitativa de banda com os demais tráfegos concorrentes.

Além disso, o ALMTF não possui nenhum controle efetivo na taxa de envio das mensagens. Por esse motivo, se for utilizado com muitos receptores simultâneos, poderá perder eficiência e até mesmo deixar o transmissor sem recursos.

Com o objetivo de minimizar esses problemas e possibilitar a sua utilização em larga escala, foram realizadas modificações nos mecanismos de cálculo de RTT e controle de *feedbacks*. A principal alteração no mecanismo de cálculo de RTT é que além dos dois métodos tradicionais de cálculo (laço aberto e laço fechado) é possível também atualizar o RTT através da troca de informações entre os receptores da mesma rede local, reduzindo a quantidade de mensagens enviadas ao transmissor.

O mecanismo de controle de *feedbacks* sofreu várias modificações e agora é composto pela integração de novos métodos, como *timer* com supressão e cooperação entre os receptores da mesma rede. As adaptações visam à redução do número de mensagens enviadas ao transmissor e também a atualização mais freqüente do RTT.

Resumidamente, o mecanismo funciona da seguinte maneira: o transmissor divide o intervalo de controle em períodos de tempo (τ). A cada início de período, ele utiliza as mensagens recebidas durante o período anterior para calcular τ e $\boldsymbol{\lambda}$ (parâmetro utilizado no *timer*), enviando esta informação para todos os receptores.

Ao receber as informações de início de período, cada receptor programa um *timer* (***t***) dentro do intervalo [0, τ], utilizando a distribuição exponencial truncada apresentada na figura 4.5 (Nonnenmacher; 1999), em conjunto com o parâmetro $\boldsymbol{\lambda}$ enviado pelo transmissor.

$$f_z(z) = \begin{cases} \dfrac{1}{e^{\lambda} - 1} \cdot \dfrac{\lambda}{\tau} \cdot e^{\left(\frac{\lambda z}{\tau}\right)} \;, \quad 0 \le z \le \tau \\ \\ 0 \;, \quad \text{para qualquer outro valor de } z \end{cases}$$

Figura 4.5: Distribuição exponencial utilizada para o cálculo do *timer*.

Caso o receptor receba alguma mensagem indicando que outro receptor já enviou um pedido de *feedback*, ele suspende o seu *timer* e suprime o envio da mensagem dentro deste período. Caso contrário, aguarda a expiração do *timer* e envia o seu pedido, junto com uma mensagem de supressão para os receptores que estiverem na mesma rede local.

Ao receber o primeiro pedido de *feedback* dentro do período, o transmissor envia uma mensagem de supressão a todos os receptores e em seguida responde ao pedido, incluindo na transmissão as informações recebidas do receptor, o instante de tempo em que o pedido foi recebido e o instante de tempo da resposta. Esse procedimento é repetido a cada pedido de *feedback* recebido dentro do mesmo período.

Após receber a resposta do seu pedido de *feedback*, o receptor efetua o cálculo do RTT de forma precisa, bem como os parâmetros necessários para o RTT em laço aberto, e envia essa informação para todos na rede local. Dessa forma, os receptores dentro da mesma rede que tiveram seus pedidos suprimidos também terão o RTT atualizado.

O intervalo de tempo entre dois períodos é dividido em intervalos menores, denominados τ_{ow} (período *one-way*). O transmissor envia o seu *timestamp* a cada *one-way*, permitindo que os receptores estimem o RTT em laço aberto. Os métodos utilizados no cálculo do RTT e no controle de *feedbacks* serão detalhados a seguir.

4.6.1 Cálculo do RTT

O cálculo do RTT com precisão é realizado através da fórmula abaixo. Nesta expressão, o RTT é o resultado da soma dos atrasos receptor-transmissor (t_{RT}) e transmissor-receptor (t_{TR}), sendo t_T^P e t_T^R o instante de chegada do pedido de *feedback* e o instante de partida da resposta. As variáveis t_R^P e t_R^R representam os instantes de partida do pedido de *feedback* e chegada da resposta no receptor, enquanto t_{diff} é a diferença entre os relógios do transmissor e receptor.

$$RTT = t_{RT} + t_{TR}\ ,\ onde:$$
$$t_{RT} = t_T^P - t_R^P + t_{diff} \quad (1)$$
$$t_{TR} = t_R^R - t_T^R - t_{diff} \quad (2)$$

Somando-se (1) e (2) temos a fórmula final do RTT, que pode ser visualizada na expressão da figura 4.6.

$$RTT = t_R^R - t_R^P - (t_T^R - t_T^P)$$

Figura 4.6: Cálculo do RTT em laço fechado.

Isto prova que o método é independente de sincronismo de relógio entre transmissor e receptor, porém, como comentado anteriormente, uma estimativa constante baseada nesta fórmula poderia afetar a escalabilidade do mecanismo. Por essa razão, o cálculo de RTT em laço fechado deve ser realizado em intervalos maiores de tempo.

Para garantir uma atualização mais freqüente do RTT é utilizado o cálculo em laço aberto (RTO), que se baseia no *timestamp* do transmissor e no parâmetro α. O parâmetro α reflete a assimetria da rede (t_{assi}) e a diferença entre os relógios do transmissor e receptor, sendo calculado conforme descrito na figura 4.7.

$$t_{assi} = t_{TR} - t_{RT}\ ,\ substituindo\ t_{TR}\ e\ t_{RT}:$$
$$t_{assi} = (t_R^R - t_T^R - t_{diff}) - (t_T^P - t_R^P + t_{diff})$$
$$\alpha = 2.t_{diff} + t_{assi} = (t_R^R - t_T^R - t_T^P + t_R^P)$$

Figura 4.7: Cálculo do parâmetro α.

Portanto, o cálculo do RTO pode ser realizado conforme a figura 4.8 a seguir.

$$\text{Com RTT} = t_{RT} + t_{TR} \quad e \quad t_{assi} = t_{TR} - t_{RT} \ ,$$

$$\text{Podemos substituir: } t_{RT} = t_{TR} - t_{assi}.$$

$$\text{Assim sendo, RTO} = (t_{TR} - t_{assi}) + t_{TR} = 2\,t_{TR} - t_{assi}$$

$$\text{Substituindo } t_{TR} \text{ por } {}^{(2)} \text{ temos:}$$

$$\text{RTO} = 2.(T_R^R - t_T^R - t_{diff}) - t_{assi}$$

$$\text{RTO} = 2.(t_R^R - t_T^R) - (2\,t_{diff} + t_{assi})$$

$$\text{RTO} = 2.(t_R^R - t_T^R) - \alpha$$

Figura 4.8: Cálculo do RTT em laço aberto (RTO).

Como se pode observar, o RTO é calculado utilizando apenas o *timestamp* do transmissor, o instante de chegada do pacote no receptor e o parâmetro α. Contudo, este método possui apenas a atualização de t_{TR}, enquanto t_{RT} somente é atualizado durante o cálculo do RTT em laço fechado. Sendo assim, o cálculo do RTT com precisão não deve ter intervalos muito distantes também.

Uma das formas de reduzir esse intervalo de atualização é utilizando a cooperação entre os receptores de uma rede local. Partindo da premissa que as condições da rede e o atraso receptor-transmissor são os mesmos, os valores do RTT obtidos por um receptor podem ser aproveitados por todos os demais membros da rede.

Para suavizar o valor do RTT obtido com essa técnica, adaptou-se a média ponderada utilizada no ALMTF (EWMA - *Exponentially weighted moving average*) conforme mostra a figura 4.9. Se o RTT em questão for proveniente da técnica de laço fechado, o parâmetro θ = 0,1. Caso for um RTT de laço aberto ou na mesma rede local, o parâmetro θ = 0,05.

$$RTT_{atual} = (1-\theta) \cdot RTT_{atual} + (\theta) \cdot RTT_{calculado}$$

Figura 4.9: Média utilizada para suavizar o RTT.

4.6.2 Controle de *Feedbacks*

O mecanismo de controle de *feedbacks* é similar ao proposto por Nonnenmacher (1999), sendo baseado no *timer* de distribuição exponencial apresentado na figura 4.5. Conforme sugerido originalmente, o período $\boldsymbol{\tau}$ utiliza o maior RTT calculado a cada período no transmissor, sendo o valor de $\boldsymbol{\tau}$ igual a 10 vezes esse RTT, visando à redução do número de *feedbacks*.

Para aprimorar o mecanismo, o *timer* é utilizado em conjunto com técnicas de supressão. Nonnenmacher (1999) demonstrou que *f(z)* tem a sua taxa de supressão otimizada se o parâmetro λ for igual a 10, contudo, uma alta taxa de supressão significa um maior intervalo sem atualização do RTT nos receptores. Por esse motivo, **λ** é calculado em função do número de receptores, conforme sua proposta original, apresentada na figura 4.10. Nessa figura, R é o número total de receptores presentes na sessão multicast.

$$\lambda = 1{,}1 \cdot \ln(R) + 0{,}8$$

Figura 4.10: Cálculo do parâmetro λ.

Entretanto, o mecanismo implementado diretamente dessa forma não é justo, pois os receptores em enlaces mais lentos ou em distâncias maiores tendem a iniciar seus *timers* defasados dos demais. Para minimizar esse problema, foi acrescentado ao *timer* (*t*) um intervalo determinístico de tempo (t_{delay}), refletindo a diferença entre o maior caminho transmissor-receptor e o último intervalo de tempo calculado em cada receptor, conforme a expressão a seguir.

$$t_s = t + t_{delay} \text{ , com } t_{delay} = t_{TRmax} - t_{TRatual}$$

Esta alteração permite equilibrar o início do intervalo aleatório para receptores localizados em ambientes heterogêneos, evitando que as diferenças de velocidade e distância interfiram na taxa de supressão. Contudo, devido à aleatoriedade da expressão utilizada no *timer*, ainda poderia ocorrer que um receptor permaneça um longo período sem uma estimativa precisa do RTT.

Para solucionar o problema foi utilizado um fator de prioridade dinâmico (*fp*), que incrementa a cada intervalo de período em que o receptor não calcule o RTT com precisão, conforme mostra a figura 4.11:

$$fp = \begin{cases} 1, & RTT\ calculado \text{ (laço fechado)} \\ fp + 1, & RTT\ obtido\ da\ LAN \\ 2\,fp, & \text{RTO (laço aberto)} \end{cases}$$

Figura 4.11: Incremento do fator de prioridade.

O método funciona da seguinte maneira: *fp* inicia com um valor máximo (fp_{max}) e após o primeiro cálculo de RTT efetivo *fp* recebe 1, tendo seu valor dobrado a cada intervalo sem atualização (período *one-way*). Caso o receptor obtenha o RTT através da rede local, o valor de *fp* é incrementado de 1.

A utilização de prioridades acrescenta ao *timer* t_s uma nova parcela, proporcional ao intervalo de período τ e ponderado com o *timer* t_s pelo parâmetro γ, com $0 < \gamma < 1$. Sendo assim, o *timer* final é obtido pela seguinte expressão:

$$timer = (1-\gamma) \cdot (t_s + t_{delay}) + \frac{\gamma.\tau}{f_p}$$

Figura 4.12: Cálculo final do *timer*.

O valor de γ utilizado no ALMTF foi 0,5, pois é uma ponderação média entre o período total dividido pelo fator de prioridade e o *timer* efetivamente calculado.

A comunicação entre os receptores da mesma rede local também foi utilizada para otimizar o mecanismo de supressão. Portanto, quando um receptor envia um pedido de *feedback* para o transmissor, envia também uma mensagem de supressão dentro da rede local. Devido a essa mensagem ser muito mais rápida que a enviada pelo transmissor, melhora a taxa de supressão entre os receptores na mesma rede.

Para realizar a comunicação interna, foi utilizado um fluxo multicast limitado a rede local (TTL=1) e separado do fluxo de dados. Isto permite que várias outras informações possam ser repassadas dentro da rede, como taxa de perdas, sincronismo de camadas, banda disponível, entre outras, integrando de forma mais eficiente os mecanismos relacionados ao controle de congestionamento.

5 METODOLOGIA

Conforme mencionado anteriormente, o objetivo principal deste trabalho é estender o protocolo ALMTF para uma rede real, propondo novas alternativas que o adaptem para este ambiente. Para isso, foi realizada a implementação do protocolo e sua validação em redes reais, além da comparação dos resultados experimentais com a simulação original.

Para a validação do protocolo, foram criadas duas aplicações na linguagem C/C++ para GNU/Linux: um transmissor multicast e um receptor de dados com o algoritmo ALMTF integrado.

A ferramenta transmissora é responsável por realizar a transmissão multicast em multi-taxa. Para isso, ela gera os pacotes nas taxas especificadas em cada camada, enviando-as em grupos multicast separados. O receptor recebe os dados, calcula o RTT, a quantidade de perdas e outras informações utilizadas pelo ALMTF.

Os parâmetros da aplicação ficam armazenados em um arquivo de configuração. Após a leitura do arquivo, o transmissor irá adquirir conhecimento dos grupos multicast e das taxas que devem ser utilizadas, podendo iniciar a transmissão. Para cada camada é criada uma *thread*, sendo que, no momento da transmissão, é adicionado o número de seqüência (dentro da camada) e o *timestamp* aos pacotes. A tabela 5.1 apresenta um exemplo dos parâmetros utilizados nas aplicações.

Tabela 5.1: Grupos multicast e suas taxas de transmissão.

Camada	*Grupo*	*Taxa*
0	239.1.1.1	30 kbit/s
1	239.1.1.2	60 kbit/s
2	239.1.1.3	120 kbit/s
3	239.1.1.4	240 kbit/s
4	239.1.1.5	480 kbit/s
5	239.1.1.6	960 kbit/s

As simulações da Tese foram efetuadas iniciando a transmissão na camada número 1. Contudo, os experimentos deste trabalho iniciaram o primeiro grupo multicast sempre na camada número 0, conforme pode ser observado na tabela 5.1.

As informações enviadas pelo transmissor não são dados multimídia, pois o codificador em camadas ainda está em desenvolvimento e não foi integrado a este trabalho. Contudo, isso é irrelevante para o funcionamento do algoritmo, pois a análise

será realizada com base nas taxas recebidas, sem necessidade de exibir o conteúdo dos dados nos receptores.

Os receptores lêem um arquivo de configuração igual ao utilizado pelo transmissor e descobrem a taxa de transmissão de cada camada, bem como a banda necessária para se inscrever em cada uma delas. A inscrição em uma nova camada (*join*) cria uma *thread* específica para o recebimento dos dados. A desassociação (*leave*) faz com que a *thread* seja finalizada, parando de receber o tráfego associado a ela.

Visando a facilitar a comparação dos resultados experimentais com a simulação, as métricas utilizadas foram as mesmas empregadas originalmente na Tese. Contudo, devido ao número de equipamentos necessários para realizar os experimentos, alguns testes foram modificados. Um exemplo é a métrica de escalabilidade, que teve sua validação inicial com mais de cem computadores e foi reformulada neste trabalho.

Os experimentos foram realizados em redes locais (ambiente controlado) e de longa distância (Rede Nacional de Pesquisa - RNP), permitindo uma avaliação mais ampla e realística do protocolo. Os cenários utilizados nos experimentos estão descritos na seção 5.2. Todos os testes foram repetidos no mínimo três vezes, a fim de verificar a consistência destes.

Para se extrair os resultados dos experimentos, foram desenvolvidos vários arquivos de *log* que permitiram verificar detalhadamente o funcionamento do algoritmo. Diversos *scripts* e programas auxiliares também foram implementados, filtrando dos *logs* apenas as informações pertinentes para cada avaliação. Com base nos dados desses arquivos, foi possível montar os gráficos e as análises que serão apresentadas no capitulo 6. A seção 5.3 fornece a relação completa de todos os experimentos realizados.

5.1 Descrição das Métricas

Os experimentos visam analisar o protocolo ALMTF nos seguintes aspectos:

- **Adaptabilidade:** descreve a capacidade do protocolo em se adaptar ao número de camadas correto, mesmo quando os receptores se encontram em ambientes heterogêneos;
- **Escalabilidade:** descreve o comportamento do protocolo com o aumento do número de receptores. Serão analisadas questões como o número de pedidos de *feedback*, intervalos de atualização e a precisão no cálculo de RTT;
- **Estabilidade:** descreve a capacidade do protocolo em manter uma transmissão estável ao longo do tempo, ou seja, se os dados são recebidos de maneira constante ou se ocorrem muitas variações na taxa dos receptores;
- **Equidade**: descreve a equiparação entre a quantidade de dados transferida pelo algoritmo e pelos demais fluxos concorrentes. Essa métrica será analisada com outras sessões ALMTF (*fairness*) e com tráfego TCP e UDP (*friendliness*). Também será verificado o comportamento do protocolo começando antes ou depois dos outros tráfegos, visto que alguns protocolos possuem problemas nesse critério.

Sabendo-se da dinâmica dos tráfegos em uma rede como a Internet, diversos parâmetros foram modificados durante os experimentos, visando a analisar o protocolo sob diferentes situações. Os parâmetros avaliados foram:

- **Quantidade de fluxos concorrentes:** nos testes do ambiente controlado, foram realizadas variações com 1, 2 e 3 fluxos simultaneamente (ALMTF, TCP ou UDP). Os testes realizados na RNP já possuem, naturalmente, inúmeros fluxos simultâneos, não sendo possível precisar a quantidade exata destes fluxos;
- **Quantidade de receptores:** as métricas de adaptabilidade e equidade foram analisadas com 1, 2 e 3 receptores simultaneamente. Para verificar a escalabilidade, foram realizados experimentos integrando o ambiente controlado com a RNP, atingindo assim um total de 25 receptores;
- **Largura de banda:** através da limitação dos enlaces dos receptores, o protocolo foi analisado com diferentes larguras de banda, por exemplo, 1.2 Mbit/s, 2.5 Mbit/s e 5 Mbit/s;
- **Atraso dos enlaces:** como o atraso influencia diretamente no RTT, foram realizados experimentos com atraso de 1ms e 10ms;
- **Tipo de camadas:** foram utilizados dois tipos de camadas cumulativas: exponenciais e fixas. Nas camadas exponenciais, a taxa da camada superior é o dobro da camada atual, realizando um aumento de banda exponencial. As camadas fixas possuem sempre a mesma taxa de transmissão, proporcionando um aumento de banda linear;
- **Taxa das camadas:** o ALMTF foi analisado com as mesmas taxas da simulação original, que foram de 30 kbit/s, 60 kbit/s, 120 kbit/s, 240 kbit/s, 480 kbit/s e 960 kbit/s (totalizando 1.890 kbit/s). As camadas fixas são de 100 kbit/s cada. Também foram utilizadas taxas mais altas, adequando o protocolo para os ambientes atuais. Nesse caso, as camadas exponenciais começam com 75 kbit/s e vão até 2.400 kbit/s (totalizando 4.625 kbit/s), enquanto as camadas fixas são de 250 kbit/s cada.
- **Tamanho da Fila:** visando a fidelidade com a simulação original, o roteador foi configurado para trabalhar com uma fila do tipo FIFO, com capacidade de armazenamento para 20 e 60 pacotes. Contudo, outros tamanhos de fila também foram utilizados.
- **Pares de Pacotes (PP):** foram realizados experimentos com e sem o mecanismo integrado, visando analisar o impacto da sua utilização na rede.

5.2 Cenário dos Experimentos

Para avaliar o protocolo em ambiente real controlado, foi utilizado um roteador executando GNU/Linux com várias interfaces de rede, configurado para realizar o roteamento dos pacotes multicast e a limitação da largura de banda dos receptores.

O sistema operacional utilizado no roteador foi o GNU/Linux com a distribuição Fedora versão 9, recompilado especialmente para realizar essas funções através dos módulos *mrouted*, *tc* (*traffic control*), *Netem* (*network emulator*) e TBF (*token bucket filter*).

O *mrouted* é o serviço responsável por realizar o roteamento dos pacotes multicast entre as interfaces de rede. Foi utilizada a última versão do software (*mrouted* 3.9 beta 3), que possui suporte ao protocolo IGMP versão 2. O TBF é uma disciplina de fila que

trabalha em conjunto com a ferramenta *tc*, ambas nativas do Linux, dispensando o mesmo tratamento para todos os pacotes. Ou seja, não possui QoS, priorização ou classes, sendo a solução ideal para limitar a velocidade de uma interface de rede (HUBERT et al, 2009). O *Netem* (NETEM; 2007) faz parte da ferramenta *tc* e foi utilizado como apoio, pois permite emular várias características de um enlace WAN, como atraso, perdas, etc.

Para evitar possíveis interferências do sistema operacional (como a concorrência pelo processador, por exemplo), cada transmissor foi executado em um equipamento separado e os dados foram enviados em endereços multicast distintos. Em uma transmissão multicast com cinco camadas, por exemplo, o primeiro transmissor utilizaria os grupos 239.1.1.1 a 239.1.1.5, o segundo transmissor os grupos 239.1.1.6 a 239.1.1.10 e assim por diante.

Além disso, na tentativa de se evitar o efeito fase, explicado na seção 2.1, foi adicionada uma randomização aos pacotes enviados pelo transmissor, fazendo com que os mesmos sejam distribuídos uniformemente dentro de cada intervalo de tempo, respeitando as taxas utilizadas em cada camada. Por exemplo, se a taxa da camada zero é 30 kbit/s e o tamanho do pacote é de 1000 bytes, o transmissor deverá enviar aproximadamente 4 pacotes por segundo, ou seja, um pacote a cada 250ms.

A topologia 1 apresentada na figura 5.1 foi utilizada para avaliar o protocolo nas métricas de adaptabilidade e estabilidade em ambientes heterogêneos. Para isso, foi utilizado um único transmissor ALMTF e três receptores, cada um em uma rede isolada e com larguras de banda diferenciada. Neste ambiente, todos os receptores recebem a mesma sessão multicast (ou seja, fazem parte do mesmo grupo).

A adaptabilidade é alcançada através da limitação de banda das estações, enquanto a estabilidade é verificada através da monitoração das taxas de recebimentos dos dados em cada receptor.

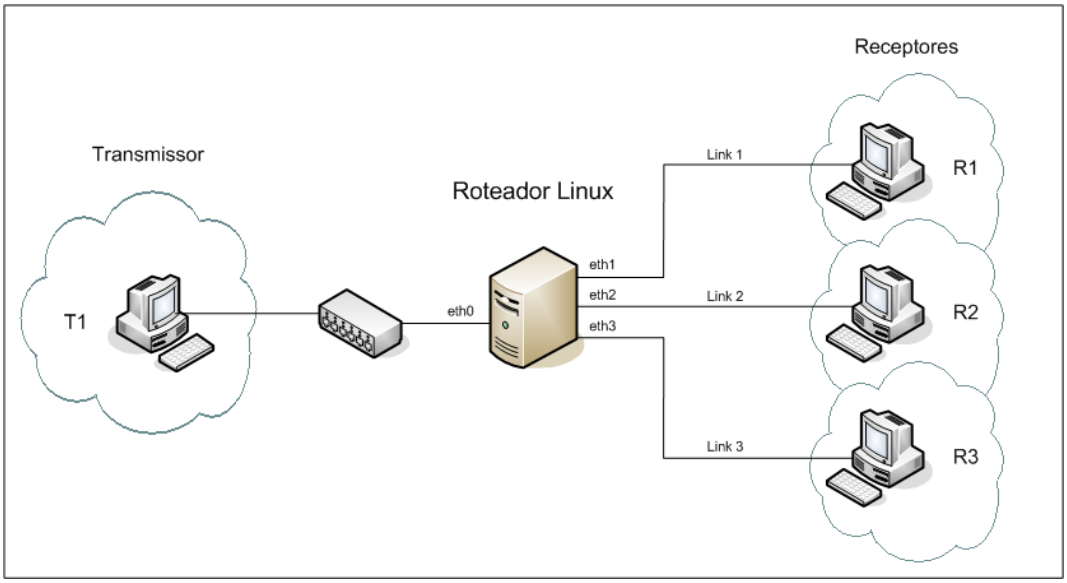

Figura 5.1: Topologia 1 - adaptabilidade e estabilidade em ambiente heterogêneo.

A topologia 2 apresentada na figura 5.2 foi utilizada para verificar as métricas de equidade e estabilidade ao longo da transmissão. Neste caso, todos os receptores compartilham o mesmo enlace, sendo possível analisar o comportamento do ALMTF na presença de outros fluxos concorrentes.

Para isso, foram utilizados vários transmissores simultaneamente, sendo que cada transmissor se conecta a um único receptor. Sendo assim, os receptores recebem sessões multicast diferentes (não fazem parte do mesmo grupo). A equidade foi analisada limitando a largura de banda (B) e o atraso (A) do enlace principal, e observando as taxas recebidas em cada receptor.

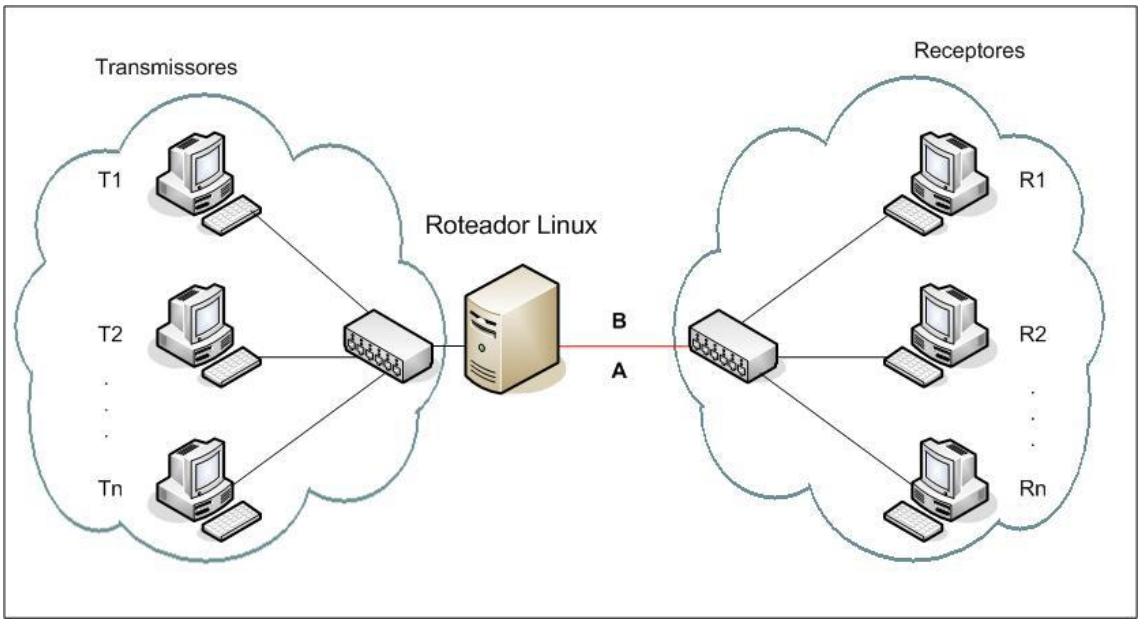

Figura 5.2: Topologia 2 - equidade e estabilidade no mesmo enlace.

A estabilidade foi verificada com base no número de variações de camadas por minuto (VCM) ocorrido em cada receptor. Quanto menor o VCM, maior a capacidade do algoritmo em manter a transmissão estável ao longo do tempo.

Para facilitar a análise dos resultados, foi desenvolvido um programa na linguagem C que captura do *log* todas as trocas de camadas realizadas durante a execução, calculando o VCM médio de cada fluxo. Cada tentativa frustrada de *join* conta duas variações, uma para subir de camada e outra para voltar à anterior. Esse programa também foi utilizado para analisar qual seria o VCM dos fluxos TCP ou UDP concorrentes, caso eles utilizassem uma transmissão em camadas.

Todos os programas e *scripts* desenvolvidos neste trabalho, bem como a própria implementação do protocolo ALMTF estão disponíveis para *download* no site do Projeto SAM[1].

A topologia 3 apresentada na figura 5.3 foi utilizada para verificar a escalabilidade do protocolo com o aumento no número de receptores. Visando a criação de um ambiente híbrido e com mais computadores, a rede local foi utilizada em conjunto com a rede da RNP, totalizando 25 receptores simultâneos.

A tabela 5.2 descreve os equipamentos utilizados na topologia 3. Os testes foram realizados utilizando a rede do Instituto de Informática da UFRGS e os POPs (Pontos de Presença) da RNP em diferentes locais, todos com IP multicast habilitado. Os POPs envolvidos neste experimento foram POP-RJ, POP-PI, POP-PR e POP-RS.

[1] http://www.inf.ufrgs.br/prav/projetos_sam.html

Tabela 5.2: Computadores utilizados nos experimentos de escalabilidade.

Localidade	*Quantidade*	*Plataforma*
Rio de Janeiro	1	GNU/Linux
Rio Grande do Sul	1	GNU/Linux
Piauí	1	GNU/Linux
Paraná	1	GNU/Linux
UFRGS	08	GNU/Linux
UFRGS	14	Windows

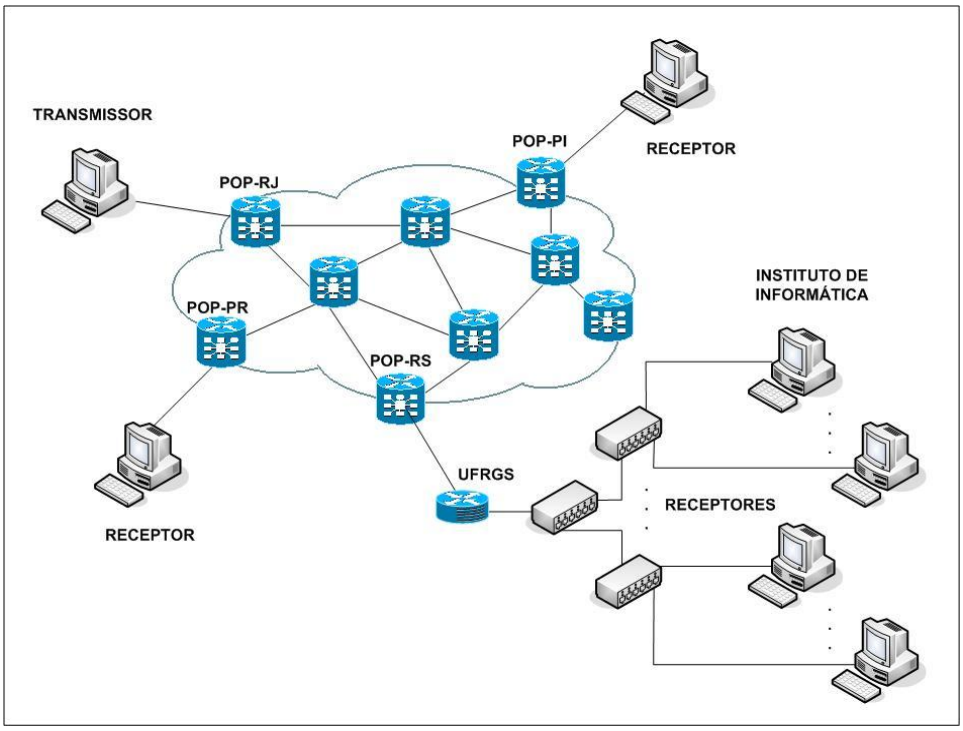

Figura 5.3: Topologia 3 - escalabilidade em ambiente controlado e na RNP.

Conforme se pode observar, o ambiente foi bastante variado, contando com 5 pontos distanciados geograficamente e redes rápidas e lentas. O POP-PI foi escolhido especialmente por ser um ponto mais lento, possuindo um enlace de 34 Mbit/s.

A escalabilidade foi avaliada considerando questões como a quantidade de pedidos de *feedback* chegando ao transmissor por unidade de tempo, o tempo médio entre as atualizações de RTT nos receptores e o erro percentual do cálculo de RTT com relação ao valor estimado na rede.

A figura 5.4 apresenta a topologia utilizada nos demais experimentos realizados na RNP. O objetivo principal dos testes foi verificar a estabilidade do ALMTF na existência de muitos fluxos concorrentes. Dois conjuntos de POPs foram utilizados nos experimentos. O primeiro é composto dos POPs RJ, PE, PR e RS; e o segundo dos POPs RJ, PI, PR e TO. Em todos os testes, o transmissor ficou localizado no POP-RJ.

A figura 5.5 apresenta o panorama geral da RNP um pouco antes da realização de um dos experimentos. Conforme pode ser observado, os POPs utilizados possuem enlaces com capacidades e condições de tráfegos bastante diferenciadas, sendo a alternativa ideal para validação do protocolo em ambientes reais.

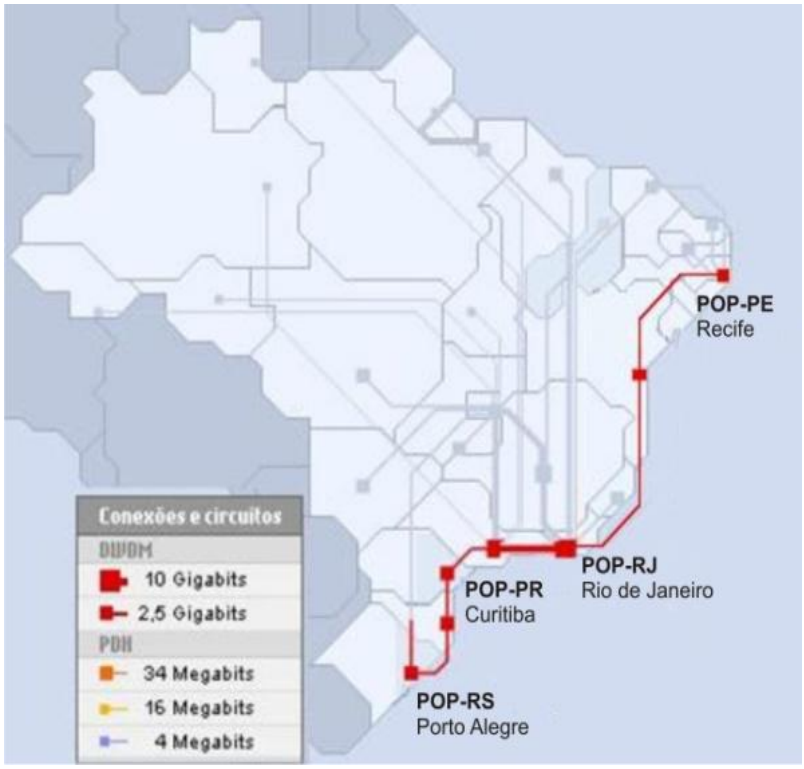

Figura 5.4: Topologia 4 – estabilidade na RNP.

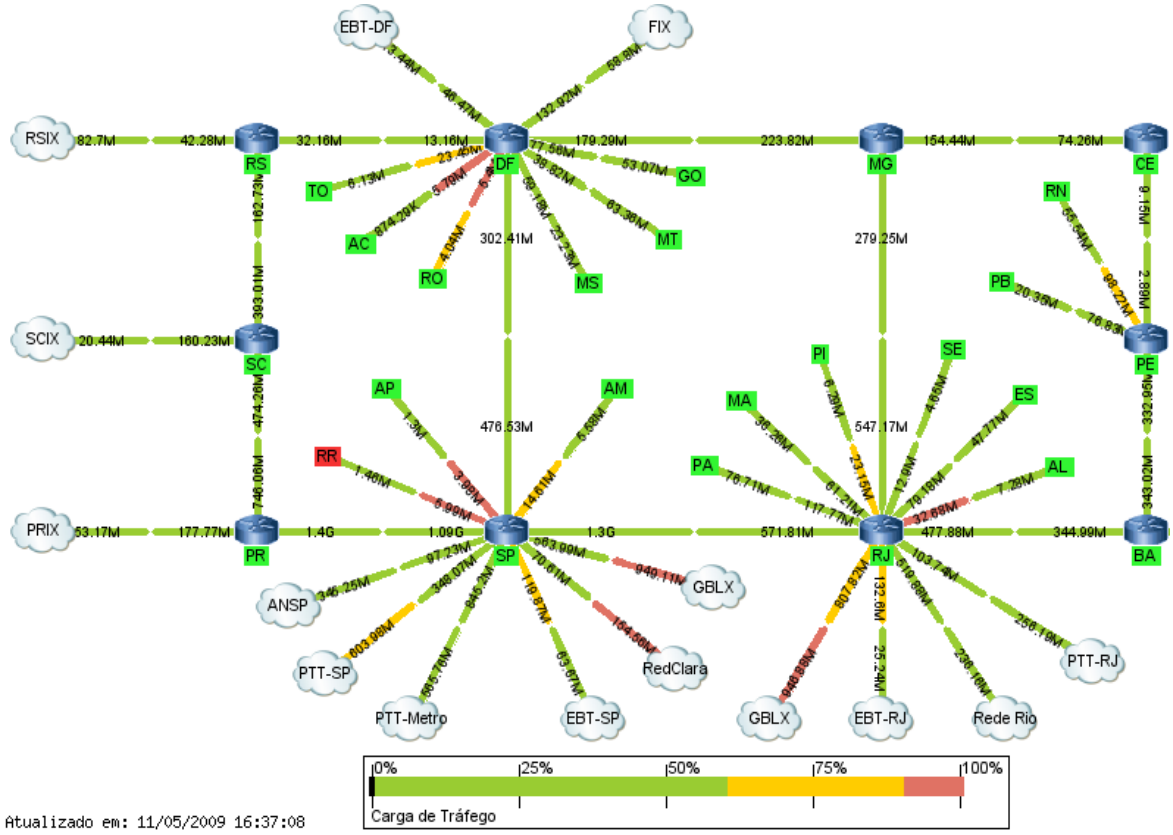

Figura 5.5: Panorama geral da RNP.

5.3 Experimentos Realizados

Esta seção tem por objetivo descrever os experimentos realizados durante este trabalho. Todos os testes efetuados tiveram, no mínimo, 500s de duração e foram executados automaticamente através de um *script*.

Após a digitação dos parâmetros no *script* (quantidade de receptores, número de tráfegos ALMTF, TCP ou UDP, tempo de duração dos fluxos, etc.), ele disparava e finalizava as aplicações nos tempos certos em cada receptor, automatizando desde o processo de execução até a coleta dos resultados.

Quando necessário, a geração de tráfego concorrente (TCP ou UDP) foi realizada através do software Iperf (TIRUMALA et al, 2008). O Iperf é uma ferramenta utilizada para análise de desempenho de redes, sendo distribuída na forma de licença GPL. Além de gerar tráfego na rede, este software pode ser usado para realizar medições ativas, como banda máxima, *jitter* e perdas.

Os experimentos realizados tiveram como principal objetivo validar o protocolo ALMTF em redes reais e comparar os resultados obtidos com a simulação original. Para que isto fosse possível, vários testes foram realizados utilizando os mesmos parâmetros especificados em Roesler (2003). A relação completa dos testes efetuados será descrita a seguir.

5.3.1 Adaptabilidade em ambientes heterogêneos

A métrica de adaptabilidade foi analisada através da topologia 1, onde cada receptor ficava localizado em uma rede separada e possuía larguras de banda diferentes. Os experimentos realizados foram os seguintes:

- **Adaptabilidade 1**
  - Quantidade de receptores: 3;
  - Enlaces: R1 = 2,1 Mbit/s; R2 = 1,05 Mbit/s e R3 = 525 Kbit/s;
  - Atraso físico dos enlaces: 10ms;
  - Tipo de camada: exponenciais da Tese;
  - Tipo de fila = FIFO com 20 pacotes;
  - Sem o mecanismo de Pares de Pacotes (PP).
- **Adaptabilidade 2:** igual ao "1", mas com PP.
- **Adaptabilidade 3:** igual ao "1", mas com 1ms de atraso.
- **Adaptabilidade 4:** igual ao "1", mas com 22 camadas fixas de 100k + PP.
- **Adaptabilidade 5:** igual ao "4", mas com fila de 60 pacotes.

5.3.2 Equidade de tráfego

A métrica de equidade foi analisada através da topologia 2, onde todos os receptores faziam parte da mesma rede e disputam o mesmo gargalo. Os experimentos realizados seguiram os mesmos parâmetros das simulações, contudo, não foi utilizado um limite máximo para a janela de congestionamento dos protocolos ALMTF e TCP.

As simulações utilizaram janela de congestionamento máxima de 30 e 60 pacotes (para os dois protocolos), mas como em situações reais não será possível alterar os

parâmetros dos demais fluxos concorrentes, decidiu-se não aplicar esta limitação e verificar o comportamento normal dos protocolos.

Os experimentos realizados foram:

- **Equidade 1**
  - Quantidade de receptores simultâneos: 2 ALMTF;
  - Banda do enlace principal: 1Mbit/s;
  - Atraso físico do enlace: 10ms;
  - Tipo de camada: exponenciais da Tese;
  - Tipo de fila = FIFO com 20 pacotes;
  - Com o mecanismo de PP.
- **Equidade 2:** igual ao "1", mas com um ALMTF começando antes do outro.
- **Equidade 3:** 2 ALMTF, camadas fixas de 250 kbit/s, enlace de 1.2 Mbit/s e fila de 60 pacotes.
- **Equidade 4**: igual ao "3", mas com enlace de 1.65 Mbit/s.
- **Equidade 5**: igual ao "4", mas com um ALMTF começando antes do outro e sem PP.
- **Equidade 6**: 3 ALMTF, enlace de 1.60Mbit/s, camadas fixas de 250 kbit/s, fila de 60 pacotes e PP.
- **Equidade 7**: igual ao "6", mas com cada fluxo começando 100s antes do outro e enlace de 1.2Mbit/s.
- **Equidade 8**: 2 ALMTF e 2 TCP, enlace de 2Mbit/s, camadas exponenciais da Tese, fila de 20 pacotes e PP.
- **Equidade 9**: igual ao "8", mas com 1 ALMTF e 1 TCP e enlace de 1.2 Mbit/s.
- **Equidade 10:** igual ao "9", mas com Fila de 60 pacotes, camadas fixas de 250kbit/s e sem PP.
- **Equidade 11:** 1 ALMTF e 1 TCP, enlace de 4.2 Mbit/s, camadas exponenciais novas e PP. TCP começando 100s depois do ALMTF.
- **Equidade 12:** 2 ALMTF e 2 UDP, enlace de 1.2 Mbit/s, fila de 60 pacotes, camadas fixas de 250 kbit/s.
- **Equidade 13:** igual ao "12", mas com 1 ALMTF e 2 UDP.

5.3.3 Estabilidade na transmissão

A métrica de estabilidade foi verificada em todos os experimentos anteriores com a topologia 1 e 2. Visando analisar o comportamento dos protocolos em redes de longa distância, também foram realizados testes com a topologia 4, utilizando a rede da RNP. Nesse caso, foi utilizado um número maior de camadas e taxas de transmissão mais altas, visando aumentar a utilização dos enlaces.

Nos experimentos com camadas fixas, foram utilizadas 50 camadas de 250kbit/s, totalizando uma taxa de 12.500kbit/s. As camadas exponenciais utilizadas podem ser visualizadas na tabela 5.3. Os experimentos realizados foram os seguintes:

- **Estabilidade 1:** ALMTF com PP e camadas fixas;
- **Estabilidade 2:** ALMTF com PP e camadas exponenciais;
- **Estabilidade 3:** ALMTF sem PP e camadas fixas;
- **Estabilidade 4:** ALMTF sem PP e camadas exponenciais.

Tabela 5.3: Camadas exponenciais utilizadas na RNP.

Camada	***Grupo***	***Taxa***
0	239.1.1.1	30 kbit/s
1	239.1.1.2	60 kbit/s
2	239.1.1.3	120 kbit/s
3	239.1.1.4	240 kbit/s
4	239.1.1.5	480 kbit/s
5	239.1.1.6	960 kbit/s
6	239.1.1.7	1920 kbit/s
7	239.1.1.8	3840 kbit/s
TOTAL		7550 kbit/s

5.3.4 Escalabilidade

Conforme explicado anteriormente, não é possível utilizar os mesmos testes efetuados no simulador para analisar a métrica da escalabilidade, visto que seriam necessários mais de cem receptores simultaneamente. Por esse motivo, os experimentos desta métrica foram modificados para analisar os seguintes fatores:

- **Pedidos de *feedback*:** representa a quantidade de pedidos de *feedback* por unidade de tempo chegando ao transmissor;
- **Taxas de atualização do RTT:** apresenta o tempo médio entre as atualizações do RTT em cada protocolo;
- **Erro de estimativa:** verifica o erro percentual no cálculo de RTT com relação ao valor estimado pela rede. Esse erro é calculado a partir da diferença (RTT – RTO) / RTT, sendo RTO o RTT *one way* calculado, conforme detalhado na seção 2.3.2.3. Desta forma, um erro percentual negativo significa um RTO menor que o RTT, já um erro percentual positivo significa um RTO maior.

Todos os experimentos foram realizados utilizando a topologia 3, que envolve a rede do Instituto de Informática da UFRGS e a rede da RNP. Os testes contaram com 10 e 25 receptores simultaneamente, sendo executados por aproximadamente 30 minutos cada um.

5.4 Análise dos Resultados

Os resultados dos experimentos foram analisados utilizando basicamente quatro métodos: gráfico de camadas, gráfico de banda, VCM e o gráfico de *feedbacks*.

O gráfico de camadas é utilizado para analisar a métrica de adaptabilidade e mostra a variação das camadas cadastradas pelos receptores ao longo do tempo. Para gerar esse gráfico, criou-se um *script* em C que captura do log a camada atualmente inscrita pelo receptor a cada instante de execução. Para evitar que ocorra sobreposição quando vários receptores estão inscritos na mesma camada, criou-se um segundo *script* que lê o arquivo gerado pelo programa anterior e aplica um deslocamento (definido como parâmetro de entrada) em todas as camadas.

O gráfico de banda é utilizado para analisar a métrica de equidade e reflete a banda utilizada por cada um dos receptores durante a transmissão. Para gerar esse gráfico, criou-se um log específico que registra a quantidade de pacotes recebidos por segundo, sendo possível calcular a banda (kbit/s) recebida em cada receptor.

A métrica de estabilidade foi analisada através do VCM médio dos protocolos. Conforme explicado anteriormente, o VCM revela a quantidade de variações de camada realizada por minuto durante a execução.

O gráfico de *feedback* é utilizado para analisar a métrica de escalabilidade e representa a média de pedidos de atualização de RTT efetuado pelos receptores por unidade de tempo. Esta métrica também é avaliada com o auxílio de algumas tabelas contendo informações como o intervalo de tempo entre as atualizações e o erro médio nos cálculos. Todos esses resultados foram gerados a partir de logs específicos na aplicação.

6 VALIDAÇÃO EXPERIMENTAL

Este capítulo é aborda a validação do protocolo ALMTF em redes reais e a comparação dos resultados experimentais com a simulação. A metodologia utilizada nos experimentos foi definida no capítulo 5, e os resultados a seguir analisam o protocolo com relação às métricas de adaptabilidade, equidade estabilidade e escalabilidade na transmissão.

6.1 Adaptabilidade em ambientes heterogêneos

Para analisar o comportamento do protocolo em ambientes heterogêneos, foi utilizada a topologia 1 apresentada anteriormente na figura 5.1. Os parâmetros empregados nos experimentos foram os mesmos utilizados na simulação, contudo, por restrições na quantidade de interfaces do roteador, foi utilizada uma sub-rede a menos.

Neste ambiente, os receptores foram configurados com as seguintes limitações de banda: receptor 1 = 2,1Mbit/s, receptor 2 = 1,05 Mbit/s e receptor 3 = 525 Kbit/s. Os demais parâmetros foram utilizados conforme explicado a seguir.

O primeiro experimento foi realizado com seis camadas do tipo exponencial (30 kbit/s, 60 kbit/s, 120 kbit/s, 240 kbit/s, 480 kbit/s e 960 kbit/s), atraso de 10ms, fila de 20 pacotes e sem a utilização do mecanismo de PP. A figura 6.1 apresenta o resultado do experimento em ambiente controlado e a figura 6.2 o mesmo teste obtido através do simulador NS-2.

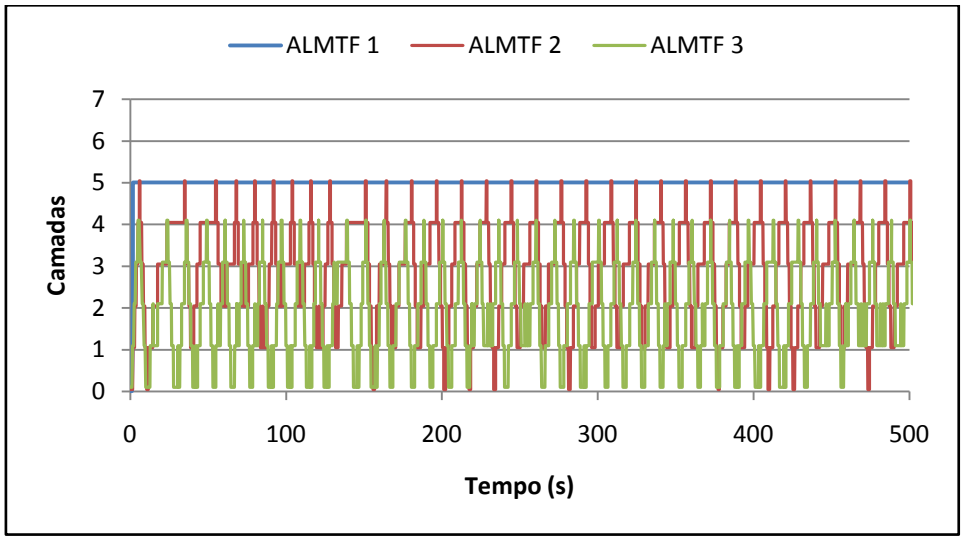

Figura 6.1: Adaptabilidade 1 - ALMTF sem PP em ambiente real.

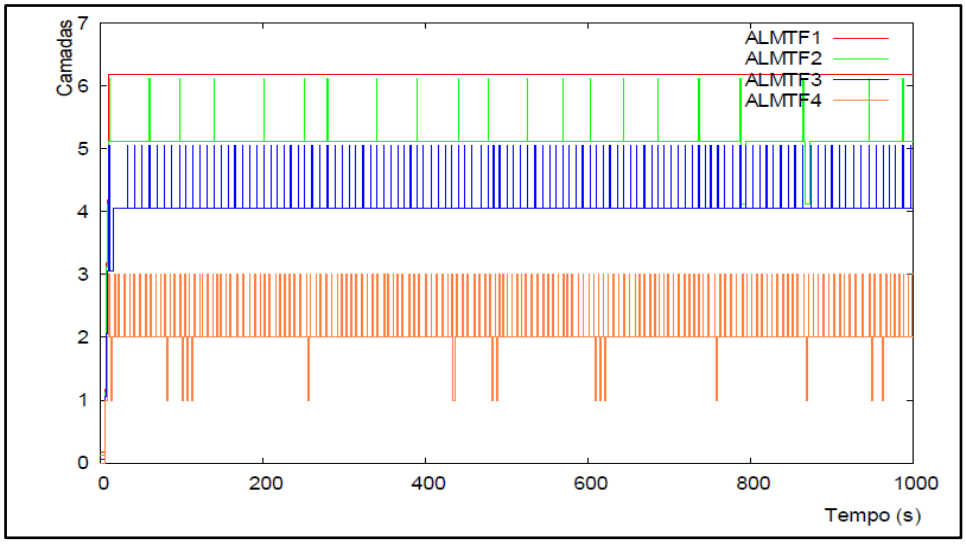

Figura 6.2: Simulação 1 - ALMTF sem PP no simulador.

Conforme pode ser verificado na figura 6.1, o receptor ALMTF1 se cadastrou em todas as camadas e permaneceu estável durante a execução, pois a sua largura de banda (2,1 Mbit/s) é superior a soma de todas as camadas utilizadas (no caso, 1.890Kbit/s). No entanto, os receptores ALMTF2 e ALMTF3 subiam e desciam constantemente de camada, diferente do observado na simulação.

Esse comportamento é explicado devido ao tempo de *leave* do protocolo IGMP. Como os receptores não sabem a banda disponível na rede (não estão utilizando o mecanismo de PP), é natural que o protocolo aumente a sua taxa até a detecção de perdas e após efetue *leave* da camada superior. Contudo, mesmo após o envio da mensagem de *leave* ao roteador, os receptores continuam recebendo os pacotes multicast por um determinado período, gerando perdas sucessivas no enlace.

Quando um roteador multicast recebe uma mensagem de *leave*, ele não interrompe imediatamente o encaminhamento do tráfego. Ao invés disso, ele aguarda que mensagens adicionais cheguem do mesmo grupo para então confirmar o processo (DAVIES; 2004). Esse problema não ocorreu durante a simulação porque o NS-2 não reflete o tempo de *leave* do protocolo IGMP. Nesse caso, a partir do momento exato em que o protocolo envia a mensagem de *leave* para os roteadores intermediários, nenhum pacote multicast adicional é recebido pelo receptor.

Uma alternativa empregada para evitar esse problema é a modificação do protocolo IGMP nos roteadores, de forma a reduzir o tempo de *leave*. Esse recurso é conhecido como *fast-leave* e foi sugerido por Rizzo (1998). Quando utilizado, o *fast-leave* permite que o roteador primeiro remova a interface de rede da sua tabela de roteamento para depois verificar se ainda existem receptores ativos ou interessados na respectiva transmissão.

Contudo, seu uso não é indicado em redes reais, pois caso exista mais de um receptor atrás da mesma interface poderá ocorrer perda de pacotes inadvertidamente. Além disso, os desenvolvedores do *mrouted* sugerem que essa modificação não seja realizada, pois pode causar problemas de incompatibilidade com a implementação do IGMP utilizada pelo software (*mrouted* 3.9 beta 3). Portanto, os experimentos seguiram o padrão recomendado, sem qualquer alteração no roteador.

Caso a mensagem de *leave* fosse recebida na porta de um *switch*, seria desejável que o tráfego multicast cessasse imediatamente. Um recurso conhecido como IGMP *snooping* permite a saída rápida dos grupos nestes equipamentos, caso as portas estejam conectadas diretamente aos *hosts* (no caso, os participantes da transmissão). Contudo, esse recurso pode ser muito prejudicial e até mesmo inviabilizar a comunicação, caso as portas estiverem conectadas a outros roteadores ou *switches* (DAVIES; 2004).

O problema do tempo de *leave* pode ser observado já nos primeiros dez segundos do experimento, conforme pode ser verificado através da figura 6.3, que apresenta o mesmo resultado do experimento anterior com duração reduzida.

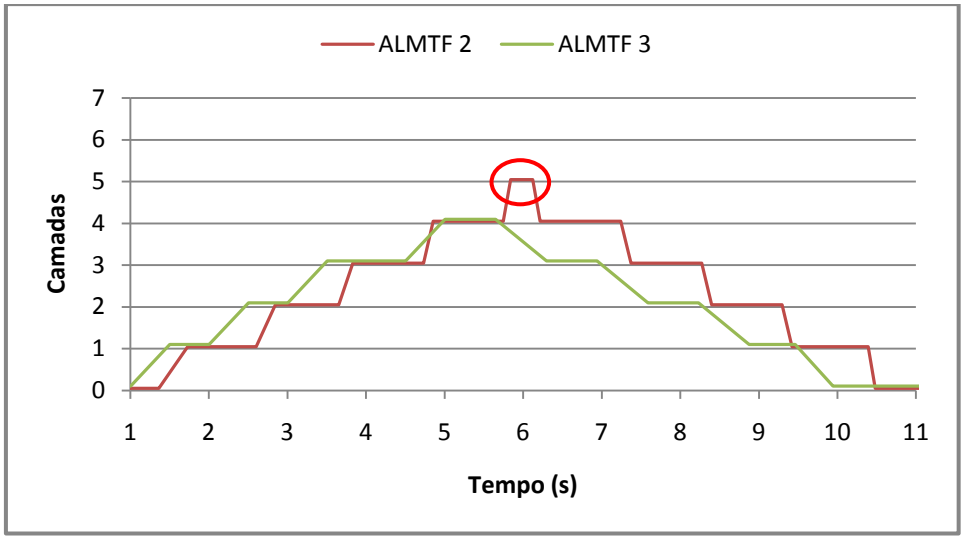

Figura 6.3: Problema de *leave*.

O receptor ALMTF2, por exemplo, estava se cadastrando normalmente em todas as camadas. Quando alcançou a sua camada ideal (camada 4), estava recebendo um total de 930kbit/s, sem gerar congestionamento na rede. No entanto, a sua tentativa de *join* na camada superior (destacado com a elipse na figura 6.3) gerou um tráfego adicional de 960kbit/s na rede, excedendo a capacidade do enlace. Após se descadastrar dessa camada, os pacotes continuaram chegando ao roteador por um determinado período de tempo, impedindo a estabilização na camada correta.

Segundo Benslimane (2007), independente da versão do protocolo IGMP utilizada, o tempo mínimo para realizar o desligamento de um grupo multicast é de três segundos. Isto ocorre porque na prática, o processo somente é finalizado após o envio de três mensagens consecutivas de *query*, uma por segundo.

Nota-se, portanto, que o problema do tempo de *leave* é uma limitação real para os protocolos de controle de congestionamento multicast que utilizam a técnica de transmissão em camadas, devendo ser tratado para evitar que o mesmo inviabilize as aplicações. Apesar dessa dificuldade já ter sido identificada, o fato da maioria dos protocolos serem validados apenas em simuladores faz com que eles não estejam aptos a lidar com essa situação, como foi o caso do ALMTF.

Uma alternativa que minimiza esse problema é o desenvolvimento de técnicas que auxiliem os protocolos a evitar as tentativas frustradas de *join*, como a utilização de mecanismos de aprendizado ou pares de pacotes. Se os receptores souberem a capacidade máxima do enlace, por exemplo, não precisariam testar a rede para realizar sua adaptação acima deste valor.

A figura 6.4 apresenta o resultado do mesmo experimento utilizando a técnica de PP, enquanto a figura 6.5 mostra o resultado obtido via simulação.

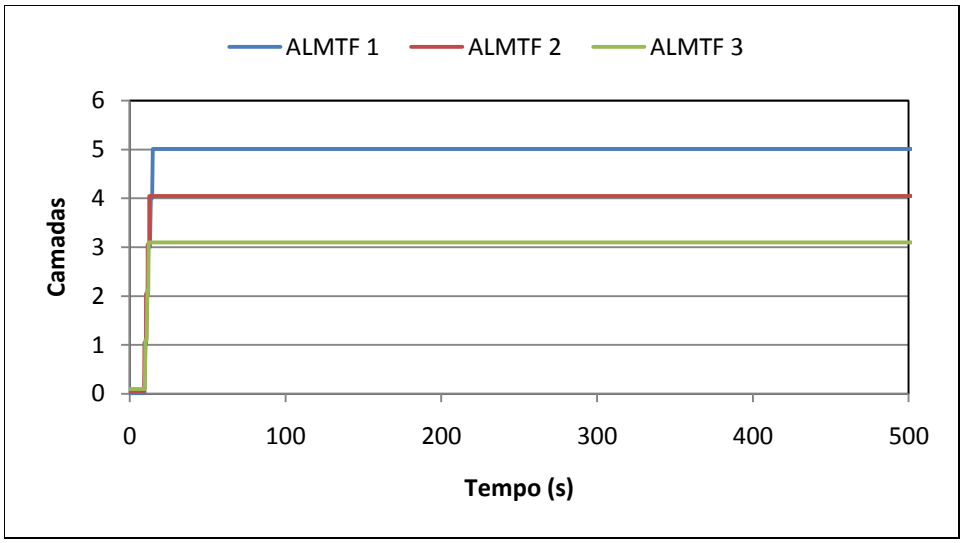

Figura 6.4: Adaptabilidade 2 - ALMTF com PP.

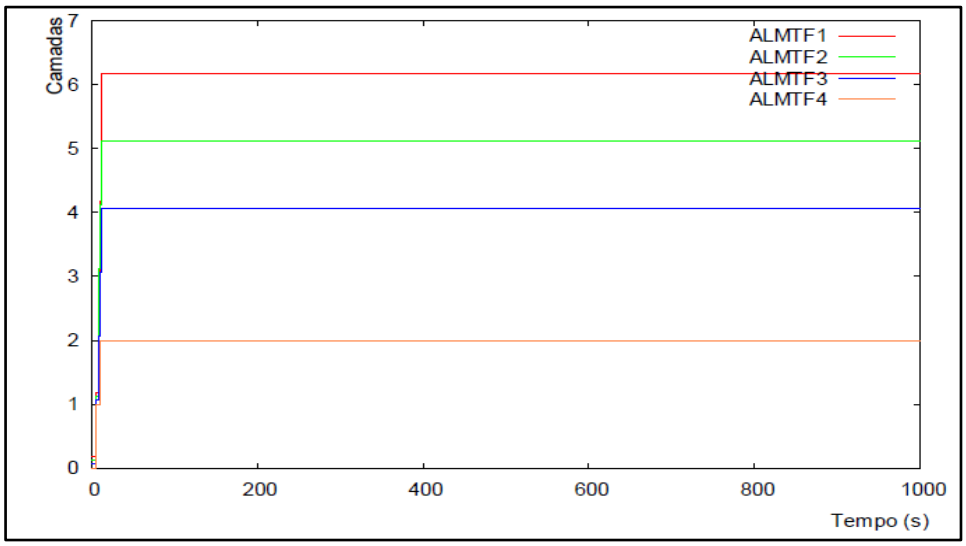

Figura 6.5: Simulação 2 – ALMTF com PP.

Conforme pode ser verificado em ambas as figuras, a utilização do mecanismo de PP permitiu que os receptores descobrissem a banda máxima da rede e se adaptassem na camada ideal sem efetuar tentativas desnecessárias de join, garantindo estabilidade total à transmissão. Como os receptores são os únicos usuários do enlace e não existem tentativas de subir de camada, a adaptabilidade é consistente com a banda disponível em cada enlace.

Sendo assim, o receptor ALMTF1 se inscreveu nas 6 camadas (camadas 0 + 1 + 2 + 3 + 4 + 5), recebendo uma banda total de 1.890kbit/s, o que é consistente com o seu enlace de 2.1Mbit/s. O receptor ALMTF2 se inscreveu em 5 camadas (camadas 0 + 1 + 2 + 3 + 4), recebendo uma banda de 930Kbit/s, também coerente com o seu enlace de 1,05Mbit/s. Por fim, o receptor ALMTF3 se inscreveu até a camada 3 (0 + 1 + 2 + 3), pois o seu enlace é de somente 525Kbit/s.

É importante observar que o mecanismo de PP só infere a banda máxima da rede, não a banda equitativa para cada fluxo durante a transmissão. Por esse motivo, é fundamental que os protocolos utilizem outros mecanismos em conjunto para evitar as frequentes tentativas de *join* na presença de fluxos concorrentes.

A figura 6.6 mostra o resultado do mesmo experimento realizado na figura 6.1 (experimento 1), mas com um atraso de 1ms no enlace (ao invés de 10ms). A figura 6.7 apresenta o resultado equivalente no simulador.

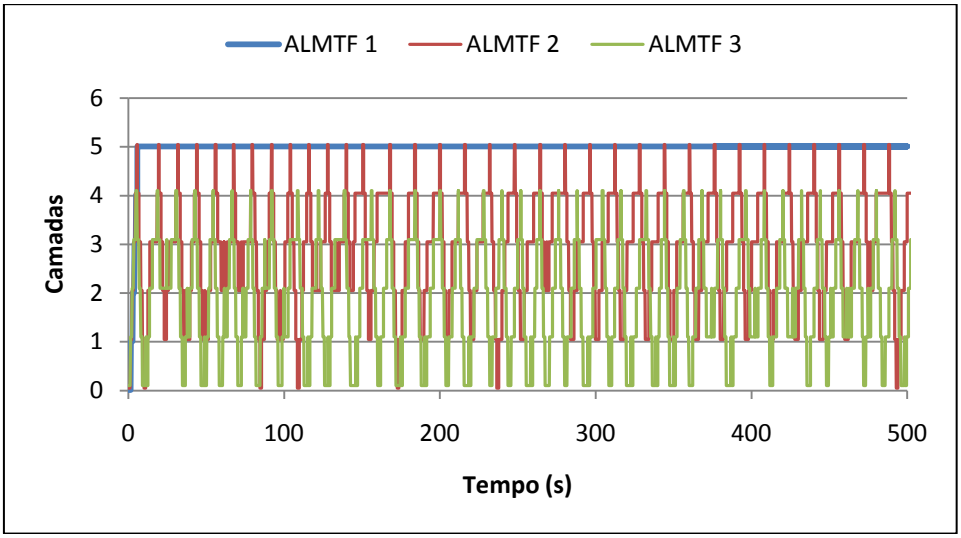

Figura 6.6: Adaptabilidade 3 – ALMTF sem PP e 1ms de atraso.

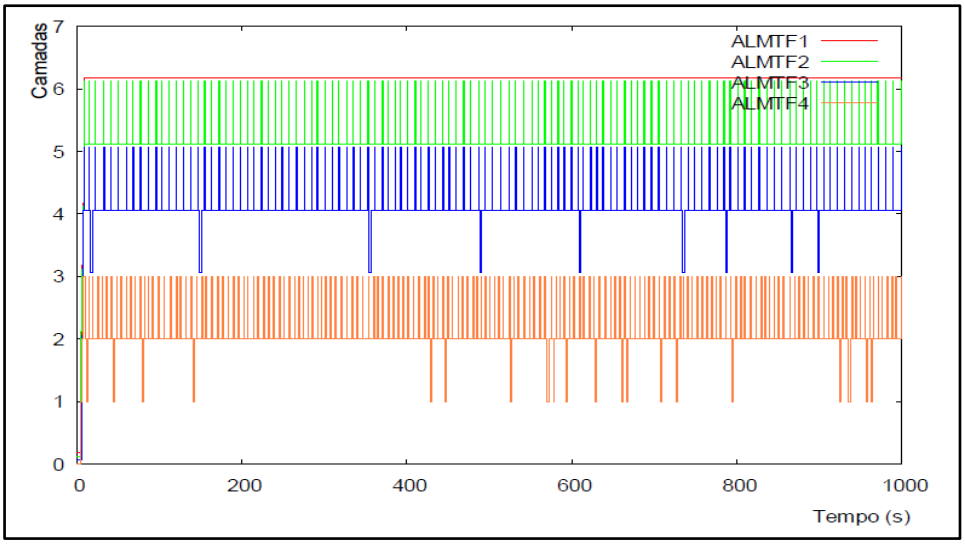

Figura 6.7: Simulação 3 – ALMTF sem PP e 1ms de atraso.

Conforme esperado, o problema do tempo de *leave* se torna ainda pior em redes mais rápidas (com um atraso menor), pois o ALMTF é executado mais vezes por segundo, tendo mais chances de aumentar ou reduzir a sua taxa de transmissão.

A granularidade das camadas também influencia nessa questão, pois quanto maior a diferença na taxa, maior a sobrecarga que estará sendo gerada na rede em casos de uma tentativa frustrada de *join*.

O quarto experimento apresenta um exemplo desse fato e foi realizado com camadas fixas de 100kbit/s em vez de camadas exponenciais. Para que todos os receptores pudessem se adaptar foram utilizadas 22 camadas, totalizando uma taxa de transmissão máxima de 2.2 Mbit/s. A figura 6.8 apresenta o resultado obtido com a utilização do mecanismo de PP, enquanto a figura 6.9 mostra o mesmo teste no simulador.

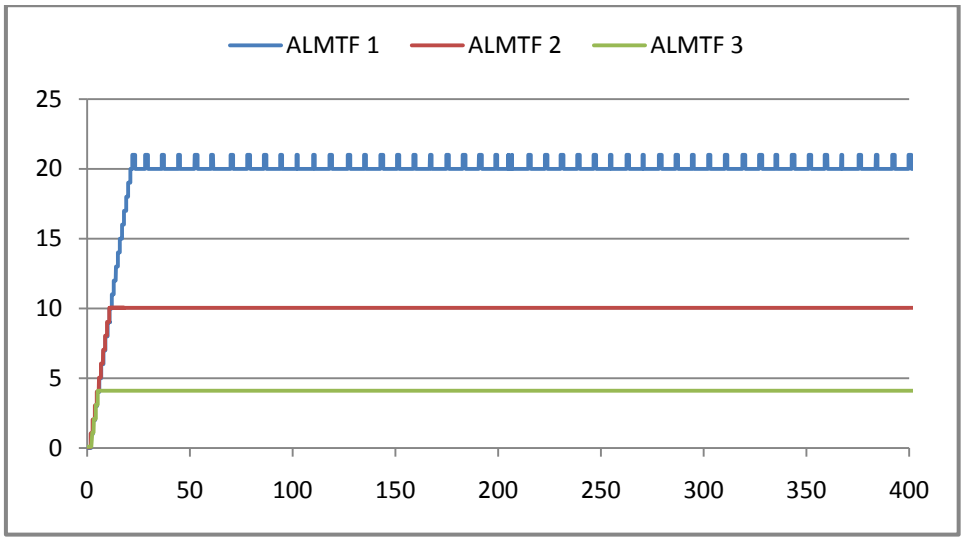

Figura 6.8: Adaptabilidade 4 – ALMTF com camadas fixas e PP.

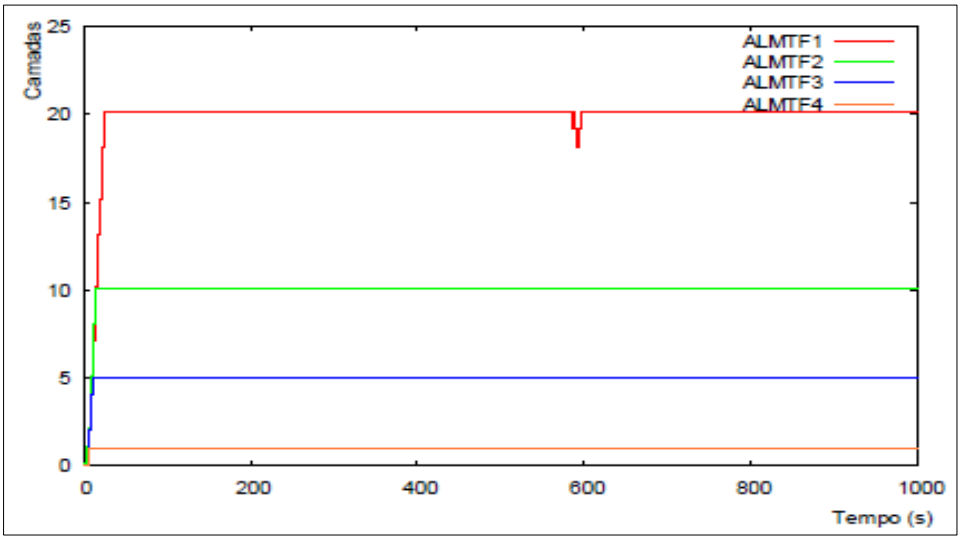

Figura 6.9: Simulação 4 – ALMTF com camadas fixas e PP.

Conforme mostra a figura 6.8, os receptores conseguiram se adaptar ao número correto de camadas, no entanto, mesmo utilizando o mecanismo de PP, o receptor ALMTF1 teve sucessivas tentativas de *join*. Isto ocorreu porque o mecanismo de PP deste receptor inferiu um pouco de banda a mais que o disponível no enlace, ficando próximo a 2.2 Mbit/s (banda necessária para atingir a camada 22).

No entanto, mesmo com a imprecisão do mecanismo de PP, o cadastramento em uma camada superior não provocou um problema de *leave* tão evidente quanto nos experimentos anteriores, pois a taxa excedida em cada tentativa é de apenas 100kbit/s (granularidade fina), sendo absorvida pela fila do roteador intermediário.

A figura 6.10 apresenta o mesmo experimento anterior, mas com o aumento da fila do roteador para 60 pacotes. Conforme se pode observar, a fila acomodou todos os pacotes sem gerar perdas, permitindo que o receptor ALMTF1 se estabilizasse na última camada.

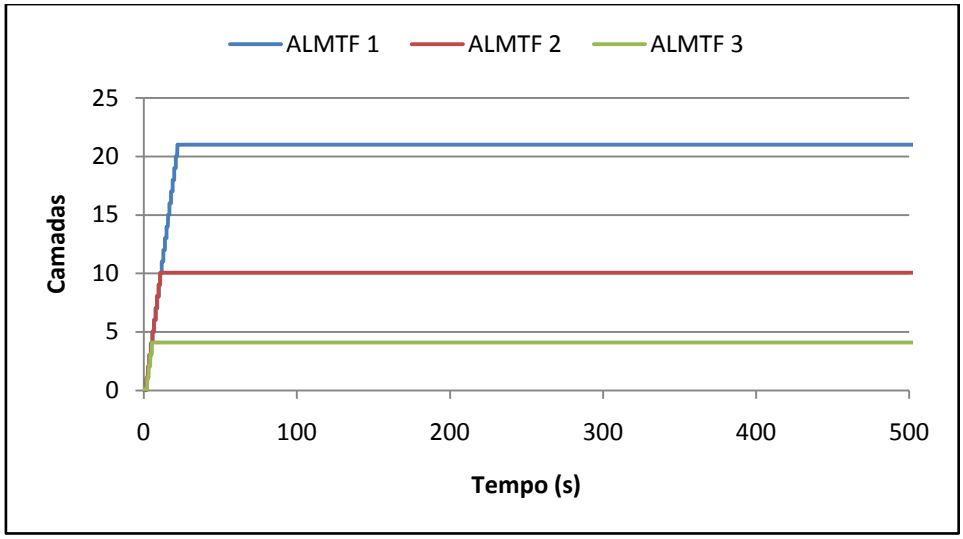

Figura 6.10: Adaptabilidade 5 – ALMTF com camadas fixas, PP e fila de 60 pacotes.

Outros experimentos foram realizados com camadas fixas e PP. Observou-se que quanto menor a granularidade e mais próximas as camadas estiverem entre si, maior a instabilidade do ALMTF, devido às frequentes trocas de camadas.

Outra diferença observada está na configuração da fila do roteador no ambiente real e no simulador. Enquanto o tamanho da fila é normalmente configurado em bytes, o NS-2 utiliza uma fila baseada em pacotes. Por esse motivo, a alteração de um simples parâmetro como o tamanho do pacote pode interferir nos resultados, exigindo um ajuste manual no tamanho da fila dos roteadores.

6.2 Equidade de Tráfego

A métrica de equidade foi analisada através da topologia 2 (apresentada na figura 5.2) e tem por objetivo avaliar o comportamento do protocolo ALMTF na presença de fluxos concorrentes.

Nesse cenário, os receptores estão localizados na mesma sub-rede, mas recebendo transmissões multicast diferentes. A ligação entre os equipamentos é realizada através de um *switch* que possui um controle do tráfego multicast por porta (IGMP *snooping*), portanto, os receptores recebem somente os grupos em que estão cadastrados.

O efeito do *leave* é suavizado devido ao tempo de descadastramento dos grupos no *switch* ser menor do que no roteador Linux, ficando no máximo em 2 segundos. Após este período, os pacotes continuam chegando ao equipamento, mas são descartados automaticamente sem onerar o receptor.

Os parâmetros empregados nos experimentos foram os mesmos utilizados na simulação, porém, novos experimentos foram realizados para analisar o comportamento do protocolo sobre outras condições.

O experimento da figura 6.11foi realizado com os seguintes parâmetros:

- **Quantidade de receptores simultâneos**: 2 ALMTF;

- **Banda do enlace principal:** 1Mbit/s;
- **Atraso físico do enlace:** 10ms;
- **Tipo de camada:** exponenciais (30 kbit/s, 60 kbit/s, 120 kbit/s, 240 kbit/s, 480 kbit/s e 960 kbit/s);
- **Tamanho da Fila**: 20 pacotes;
- **Mecanismo de PP:** sim.

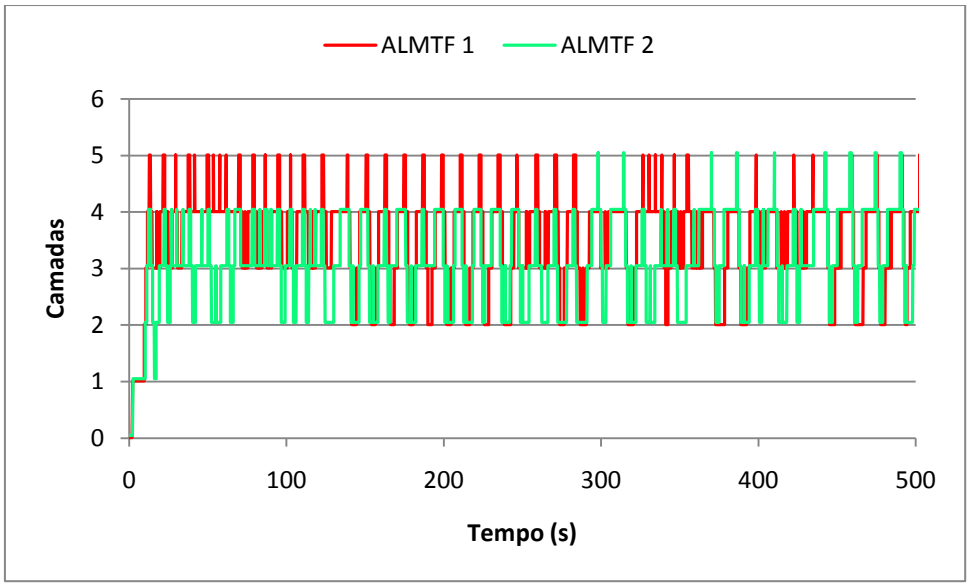

Figura 6.11: Equidade 1 – 2 ALMTF começando juntos.

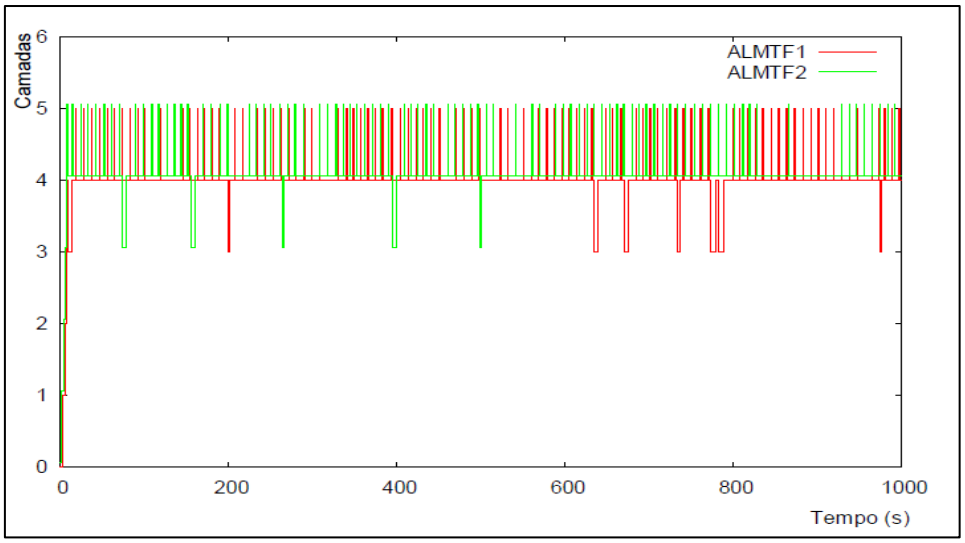

Figura 6.12: Simulação 1 – 2 ALMTF começando juntos.

Pode-se observar que os receptores ALMTF1 e ALMTF2 ficaram normalmente na camada correta (camada 3), que corresponde a uma banda acumulada de 450 kbit/s para cada um. Entretanto, quando um dos receptores efetua *join* na camada superior é gerado um tráfego adicional de 480 kbit/s na rede, ocasionando o enchimento da fila e perdas no enlace.

Esse comportamento pode ser verificado na figura 6.11, pois logo após um receptor realizar uma tentativa frustrada de *join* o outro cai devido às perdas geradas pelo excesso de tráfego. Este problema pode ser contornado com o aumento da fila do roteador, conforme será apresentado posteriormente. Na figura 6.12 isto não ocorre, visto que o simulador não reflete o problema de *leave* e pára o encaminhamento dos pacotes multicast imediatamente evitando as perdas.

O experimento apresentado na figura 6.13 foi realizado com os mesmos parâmetros do anterior, mas com um fluxo iniciando 100s antes do outro. O objetivo deste teste é verificar a capacidade do ALMTF em se adaptar após estar em regime permanente. A figura 6.14 representa o resultado equivalente do experimento no simulador.

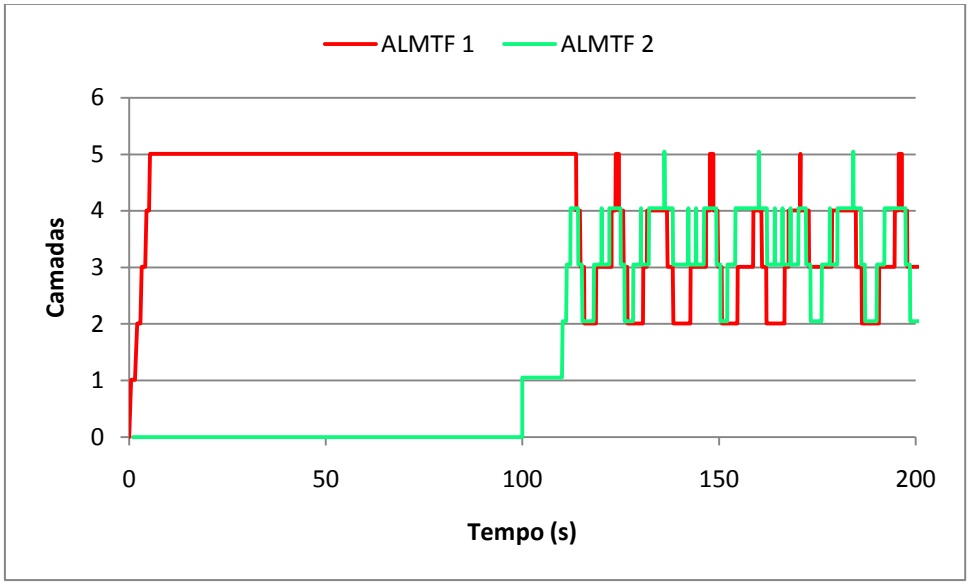

Figura 6.13: Equidade 2 – 2 ALMTF começando em momentos diferentes.

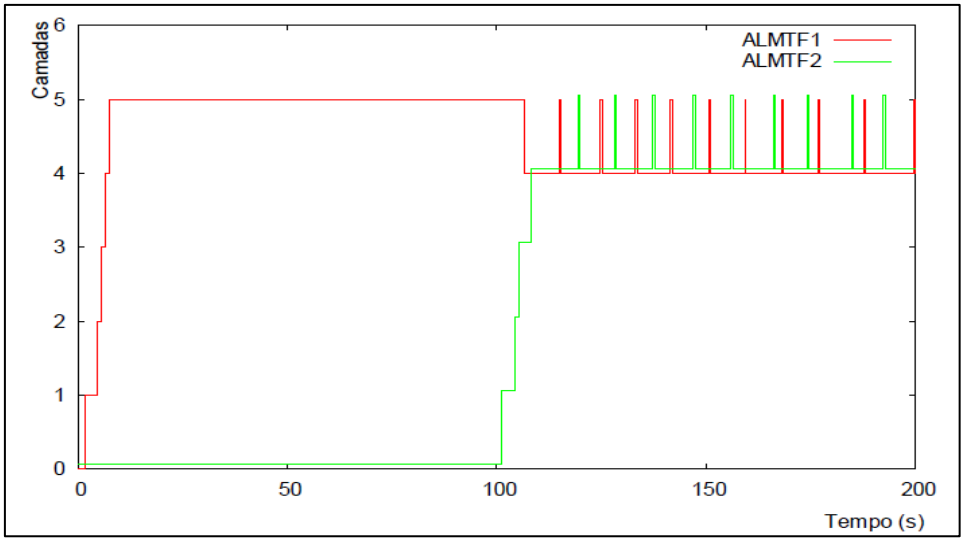

Figura 6.14: Simulação 2 – 2 ALMTF começando em momentos diferentes.

O receptor ALMTF1 inicia a sua execução e se cadastra rapidamente até a quarta camada, recebendo uma taxa total acumulada de 930 kbit/s, coerente com o enlace de 1 Mbit/s. Devido à utilização do mecanismo de PP, o receptor não realiza nenhuma tentativa de *join* na camada superior, permanecendo estável na camada ideal.

Com a entrada do receptor ALMTF2 no tempo 100s, ocorrem perdas e os dois fluxos tendem a se adaptar, dividindo o restante da banda. No entanto, as frequentes tentativas de *join* fazem com que as perdas sejam constantes no enlace, impedindo os receptores de se estabilizar. Com exceção desta questão, percebe-se que os resultados foram do experimento e da simulação foram similares.

A figura 6.15 mostra a visão da banda transferida em um experimento realizado com dois ALMTF disputando um gargalo de 1.2 Mbit/s. Neste teste foram utilizadas camadas fixas de 250 kbit/s e PP, de forma que cada fluxo pudesse utilizar uma banda equitativa de 500 kbit/s cada um. Com o objetivo de minimizar o problema de *leave*, a fila foi ajustada para 60 pacotes.

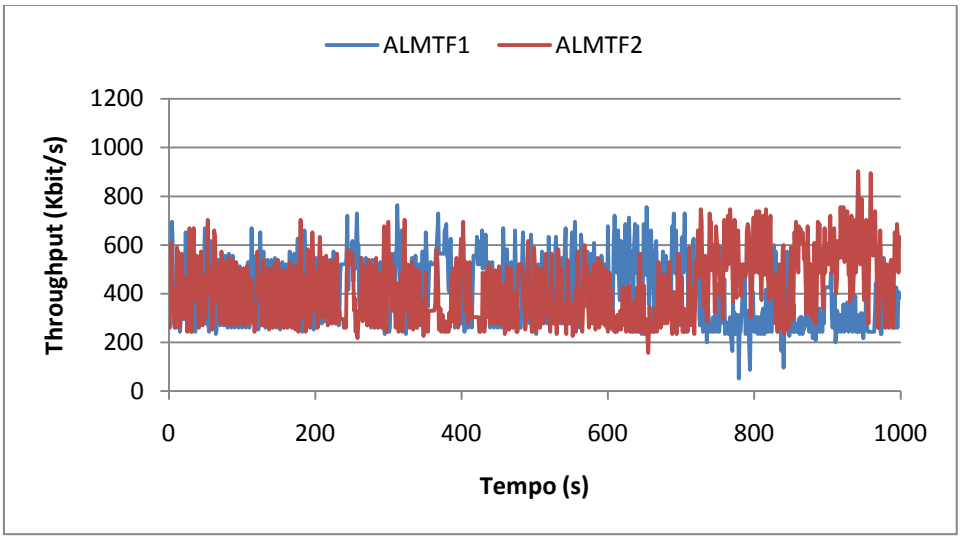

Figura 6.15: Equidade 3 – 2 ALMTF começando juntos.

Conforme pode ser observado no gráfico acima, os receptores se adaptaram corretamente em torno de duas camadas, recebendo em média 500 kbit/s cada um. A figura 6.16 mostra o *throughput* total dos receptores durante a transmissão.

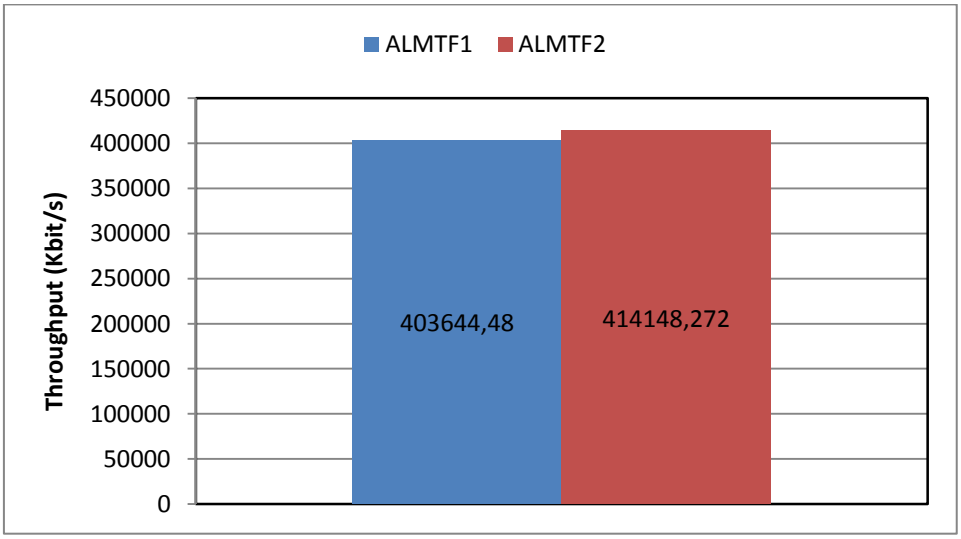

Figura 6.16: Equidade 3 - *Throughput* de cada receptor.

Pode-se verificar que o ALMTF consegue dividir a banda de forma equitativa com seu próprio tráfego, mantendo um *throughput* bastante aproximado nos dois receptores.

A figura 6.17 mostra o mesmo experimento anterior, mas com o aumento da capacidade do enlace principal para 1.65 Mbit/s. Desta forma, os receptores podem se cadastrar em até 3 camadas cada um, aumentando as suas taxas de transmissão. A figura 6.18 exibe o *throughput* dos receptores após os 1000s de experimento.

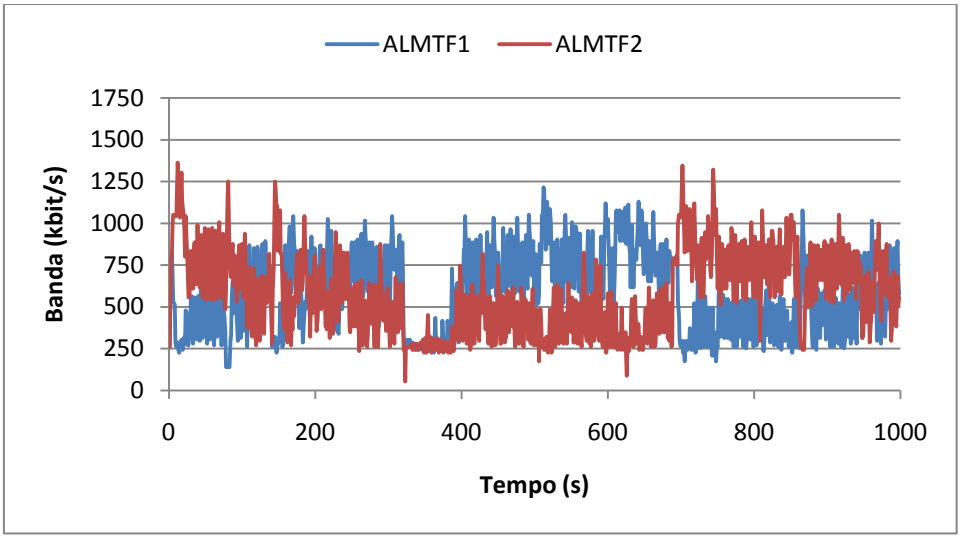

Figura 6.17: Equidade 4 – 2 ALMTF dividindo um enlace de 1.65Mbit/s.

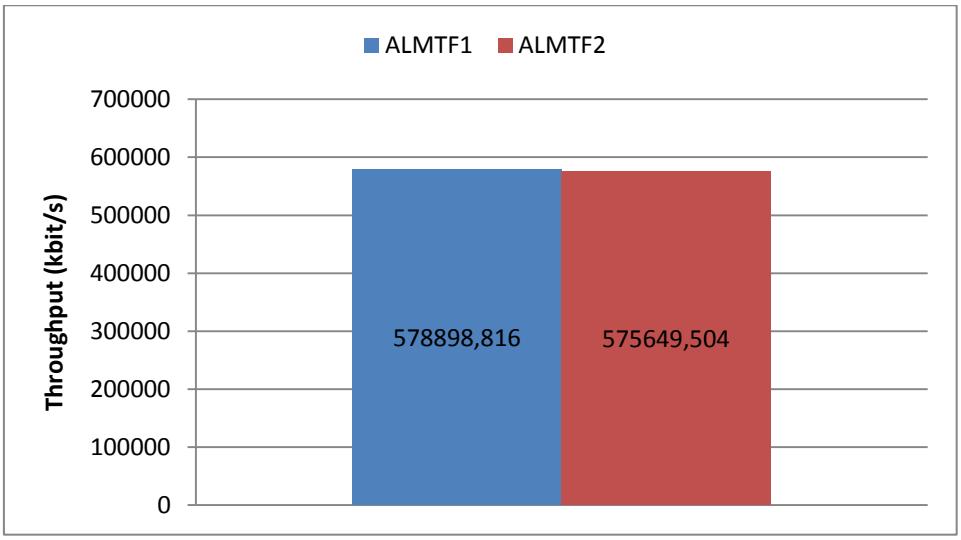

Figura 6.18: Equidade 4 - *Throughput* de cada receptor.

Percebe-se que o resultado é bastante similar ao anterior. Mesmo com o aumento da capacidade do enlace, os dois receptores conseguiram compartilhar o enlace de forma adequada, mantendo o *throughput* quase igual durante toda a transmissão.

No entanto, observou-se que próximo aos 300s o mecanismo de pares de pacotes apresentou problemas e inferiu uma banda máxima bem mais baixa que a real, fazendo com que os receptores permanecessem inscritos apenas na primeira camada. Esse problema durou apenas alguns segundos e logo normalizou, fazendo com que os fluxos voltassem a se adaptar.

A figura 6.19 apresenta o resultado do mesmo experimento, mas com um ALMTF começando 100s antes do outro e sem o mecanismo de PP. O *throughput* total dos receptores pode ser analisado na figura 6.20.

Para gerar o gráfico da figura 6.20, foi contabilizado apenas o tempo em que os receptores executaram ao mesmo tempo, ou seja, do tempo 100s em diante. Da mesma maneira que os dois experimentos anteriores, a fila foi mantida em 60 pacotes para minimizar o efeito do problema de *leave*.

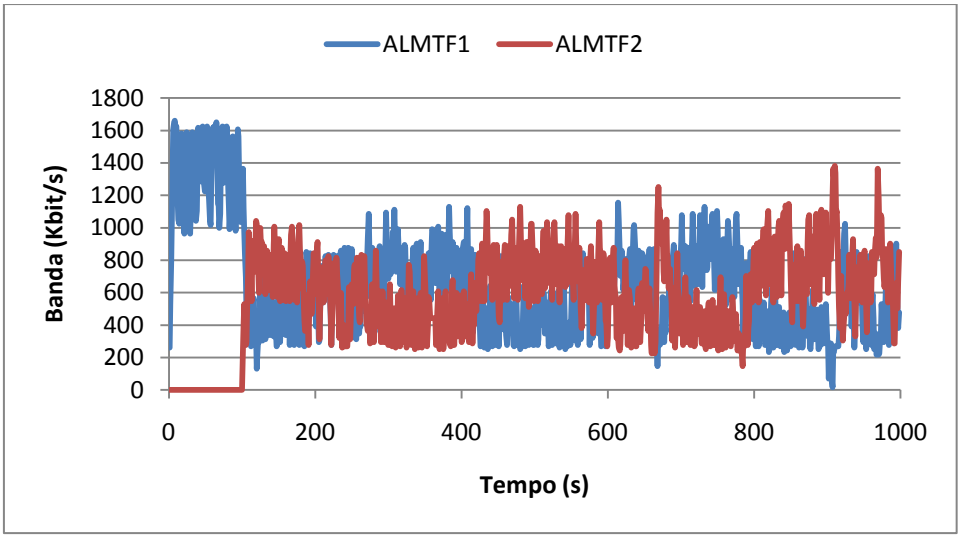

Figura 6.19: Equidade 5 - 2 ALMTF em momentos diferentes.

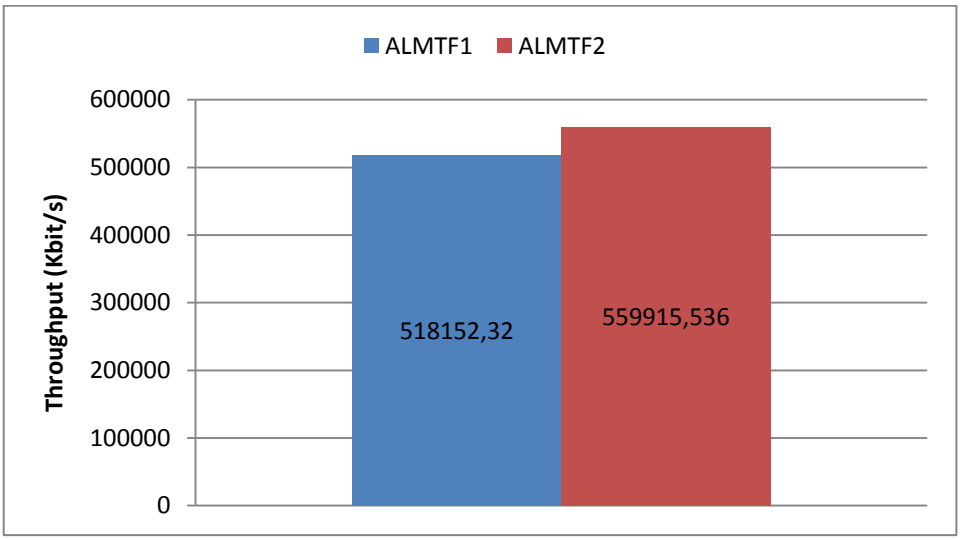

Figura 6.20: Equidade 5 - *Throughput* de cada receptor.

Devido ao teste ser realizado sem o mecanismo de PP, o receptor ALMTF1 iniciou a sua execução e permaneceu entre as camadas seis e sete, utilizando uma banda aproximada de 1.400 kbit/s. Com o início do receptor ALMTF2 no tempo 100s, o receptor ALMTF1 reduziu a sua taxa de transmissão e logo após os dois se adaptaram corretamente, dividindo a banda de forma amigável entre si. A figura 6.20 confirma a equidade de tráfego no final dos 1000s de transmissão.

A figura 6.21 apresenta o resultado do experimento com 3 receptores simultâneos. Os receptores iniciam a execução no mesmo instante e compartilham um enlace de 1.60Mbit/s utilizando o mecanismo de PP. Foram utilizadas 32 camadas fixas de

250kbit/s e fila de 60 pacotes. A figura 6.22 mostra a banda total recebida por cada um durante toda a transmissão.

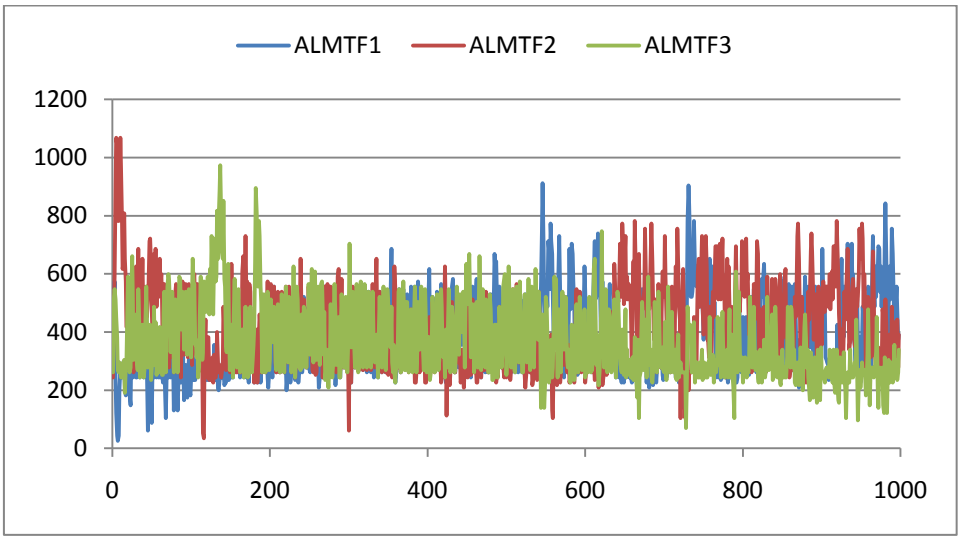

Figura 6.21: Equidade 6 – 3 ALMTF começando juntos.

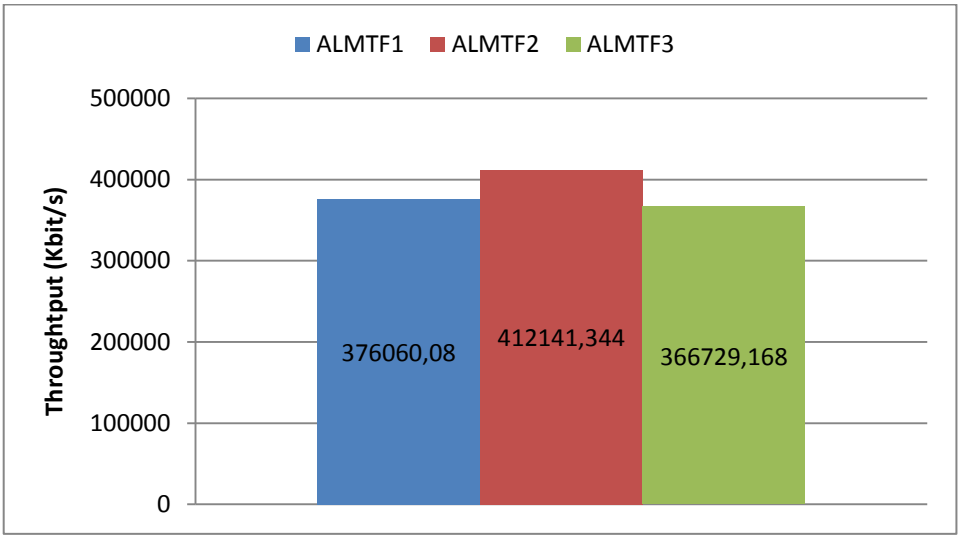

Figura 6.22: Equidade 6 – *Throughput* dos 3 ALMTF começando juntos.

É possível observar que mesmo com o aumento do número de receptores, o protocolo ALMTF continua dividindo o enlace de forma justa entre si. A vazão equitativa para os receptores é de 500kbit/s (duas camadas) para cada um, e a vazão para os três fluxos foi: ALMTF1 = 376,81kbit/s, ALMTF2 = 412,96kbit/s e ALMTF3 = 378,10kbit/s.

A vazão dos fluxos foi menor que a vazão ideal devido às perdas de pacotes. Observou-se que o uso do mecanismo de PP acarreta uma maior utilização da fila do roteador, visto que são enviados dois pacotes por vez em vez de um, e com isso um aumento nas perdas. A figura 6.22 confirma que o ALMTF transmite de forma eqüitativa com seu próprio tráfego, pois o *throughput* dos fluxos continuou bastante similar.

A figura 6.23 apresenta o experimento anterior, mas com os três receptores começando a execução em momentos diferentes e compartilhando um enlace um pouco menor, no caso de 1.2Mbit/s.

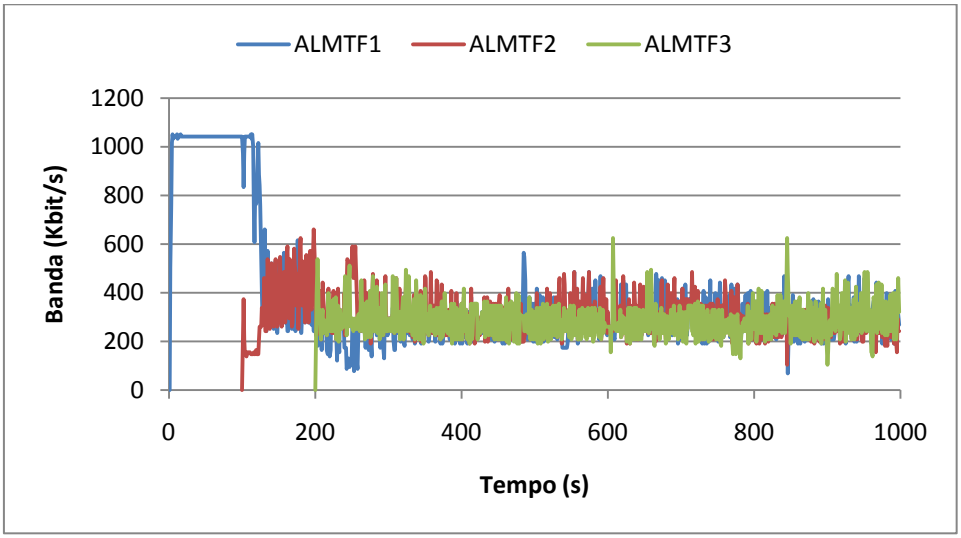

Figura 6.23: Equidade 7 - 3 ALMTF começando em momentos diferentes.

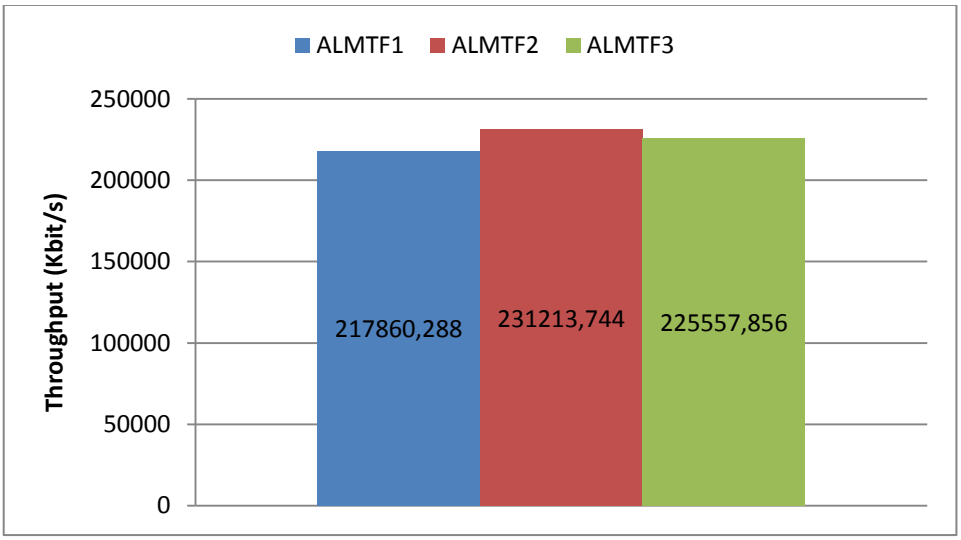

Figura 6.24: Equidade 7 - *throughput* total dos 3 fluxos ALMTF.

O receptor ALMTF1 inicia sua execução no tempo 0s e sobe até a camada três, recebendo uma banda total de 1Mbit/s. Devido ao mecanismo de PP, não realiza tentativas de *join*, pois sabe que a soma da próxima camada (1.250 Mbit/s) é maior que a sua capacidade máxima de rede (1.2 Mbit/s).

No tempo 100s, o receptor ALMTF2 inicia a sua execução e imediatamente faz com que o ALMTF1 reduza sua taxa de transmissão. Os dois receptores utilizam duas camadas cada um (aproximadamente 500 kbit/s), compartilhando adequadamente o enlace entre si. O receptor ALMTF3 inicia no tempo 200s e novamente os fluxos voltam a se adaptar, consumindo aproximadamente entre uma e duas camadas cada um.

Através dos experimentos anteriores, percebe-se que o protocolo ALMTF consegue dividir a banda de forma amigável com seu próprio tráfego, tendo resultados aproximados da simulação e satisfazendo com sucesso o quesito *fairness*.

No entanto, o mesmo não ocorre com os experimentos envolvendo tráfego TCP concorrente. Todos os testes realizados obtiveram resultados significativamente diferentes da simulação, onde o TCP ocupa a maior parte da banda e o ALMTF não consegue sustentar mais do que algumas poucas camadas.

O experimento visualizado na figura 6.25 apresenta um exemplo desta situação, onde dois fluxos ALMTF e dois fluxos TCP compartilham um enlace de 2 Mbit/s. O teste foi realizado com as camadas exponenciais da Tese, PP e fila de 20 pacotes. A figura 6.26 apresenta o resultado do mesmo teste na simulação.

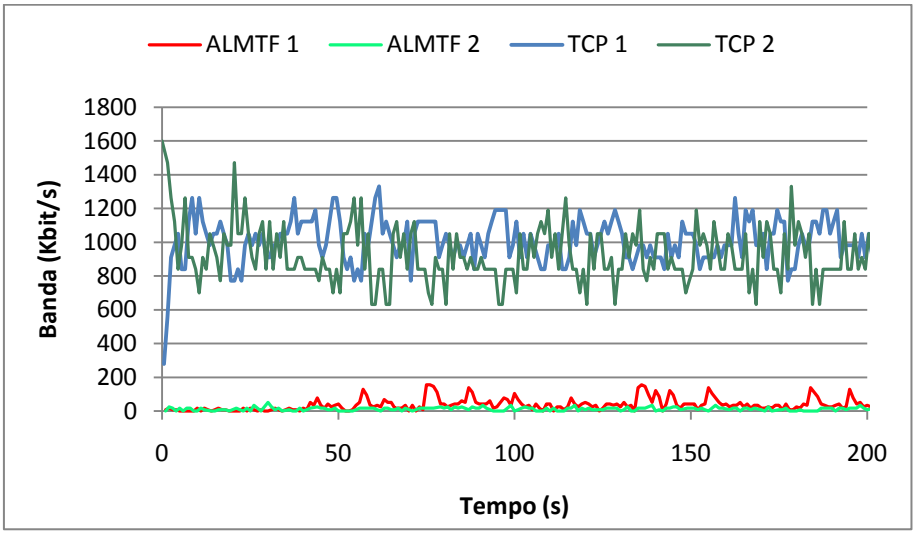

Figura 6.25: Equidade 8 - Equidade para 2 fluxos ALMTF e 2 TCP.

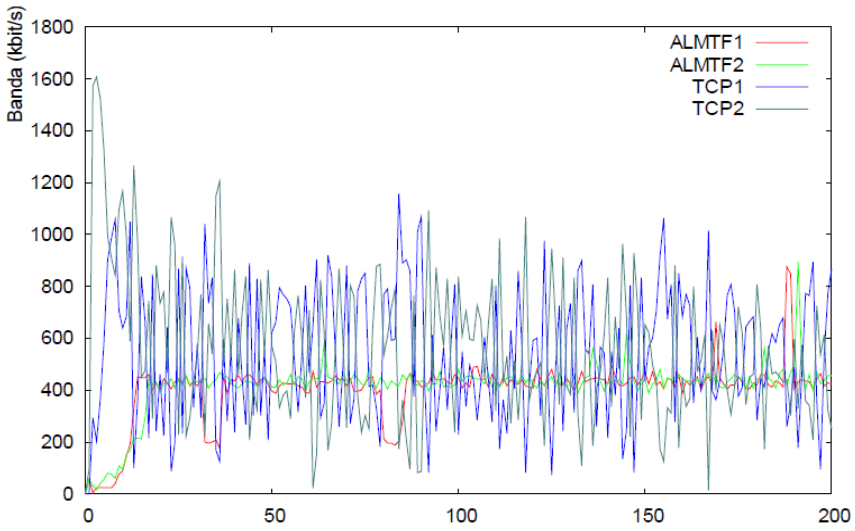

Figura 6.26: Simulação 8 - Equidade para 2 fluxos ALMTF e 2 TCP.

Pode-se observar que os resultados foram bastante diferentes. Enquanto na simulação os receptores ALMTF se cadastraram até a camada quatro e obtiveram aproximadamente 500kbit/s cada um (banda equitativa ideal para eles no enlace), no ambiente real eles ficaram a maioria do tempo na primeira camada, sendo que em

determinados momentos a taxa recebida era menor ainda que os 30 kbit/s da camada base.

Foi possível verificar através dos logs que o fato do TCP estar transmitindo em uma taxa de transmissão maior fez com ele monopolizasse a fila do roteador com os seus pacotes. Por esse motivo, as perdas do ALMTF foram constantes e ele se estabilizou na camada mais baixa, recebendo uma taxa muito inferior que a sua parcela de banda equitativa.

Outros testes envolvendo fluxos ALMTF e TCP foram realizados na Tese, mas com 5 e 10 tráfegos concorrentes de cada protocolo. Devido à quantidade de equipamentos necessários para montar esse cenário, eles não foram repetidos neste trabalho, contudo, novos experimentos foram efetuados para investigar o problema sob outras condições.

O experimento da figura 6.27 mostra os fluxos ALMTF e TCP compartilhando um enlace de 1.2 Mbit/s. Da mesma forma que o anterior, este teste foi realizado com as mesmas camadas exponenciais da Tese, PP e fila de 20 pacotes. Portanto, cada fluxo poderia utilizar em torno de 600kbit/s cada um.

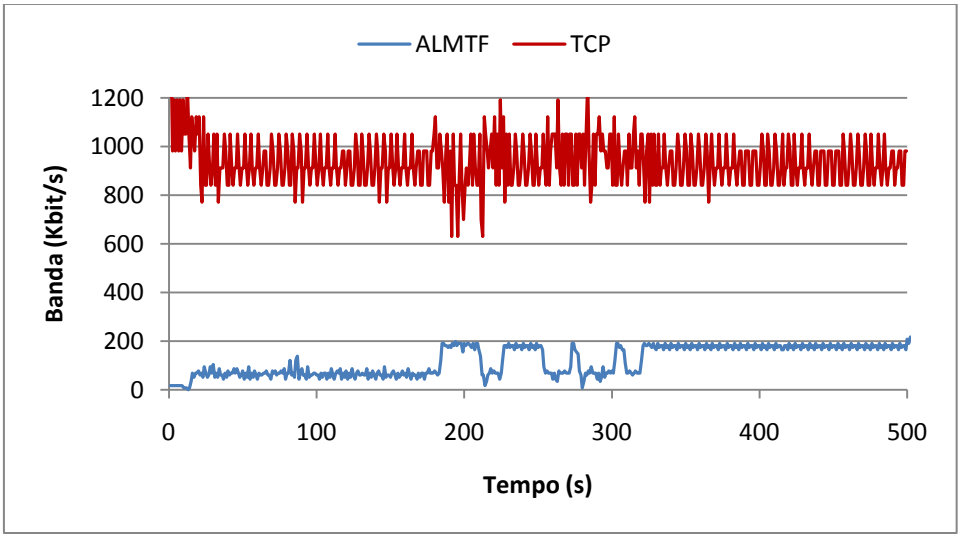

Figura 6.27: Equidade 9 - 1 ALMTF e 1 TCP.

A diferença no comportamento dos protocolos pode ser observada desde o 1s da experimentação. Enquanto o ALMTF está cadastrado na primeira camada, utilizando em torno de 30 kbit/s, o TCP já está ocupando todo o restante do enlace. Mesmo com várias tentativas de subir de camada, o ALMTF se mostrou muito sensível às perdas, obtendo um *throughput* médio de apenas 120kbit/s contra os 950kbit/s do fluxo TCP concorrente.

Para verificar a influência do tamanho da fila do roteador, um novo teste foi realizado, desta vez com camadas fixas de 250kbit/s e fila de 60 pacotes. Para desonerar ainda mais a fila, o mecanismo de PP não foi utilizado. A figura 6.28 apresenta o resultado do experimento, onde os fluxos TCP e ALMTF iniciam no mesmo instante e compartilham um enlace de 1.2Mbit/s durante um período de 1000s.

O ALMTF inicia a sua execução e se cadastra na primeira camada, recebendo uma taxa de 250kbit/s. No mesmo intervalo de tempo, o TCP infere a banda disponível na rede e se estabiliza em uma taxa próxima a 1000kbit/s. Conforme pode ser observado, o ALMTF permaneceu toda a sua execução na camada base, sem compartilhar o enlace de

forma adequada com o TCP. Portanto, o aumento da fila do roteador não interfere nos resultados, apenas causa uma latência maior na entrega dos dados e minimiza as perdas de pacotes.

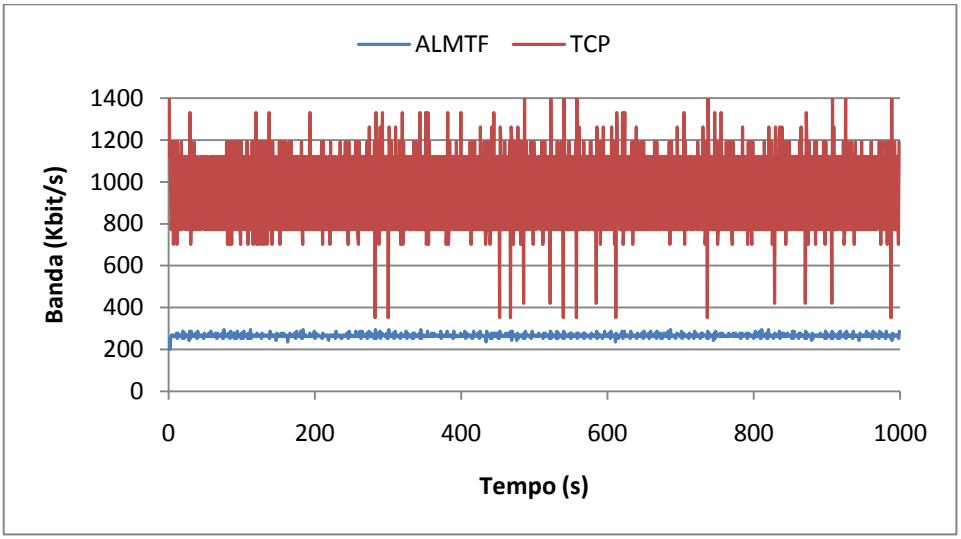

Figura 6.28: Equidade 10 - 1 ALMTF e 1 TCP começando juntos.

Injong e Lisong (INJONG, R.; LISONG, X; 2007) descreveram algumas razões para a diferença de *throughput* encontrada entre o TCP e os demais protocolos de controle de congestionamento baseados no mecanismo da equação. Após um estudo aprofundado das limitações desse mecanismo, eles provaram que, mesmo estando sob o mesmo gargalo e com as mesmas condições de rede, a diferença na taxa de envio dos fluxos faz com que eles experimentem taxas de perdas diferentes, ocasionando uma significante diferença no *throughput* observado pelos protocolos.

Apesar de o ALMTF utilizar duas técnicas diferentes para realizar o controle de congestionamento e o mecanismo da equação não ser o principal deles, existem várias inconsistências devido à utilização conjunta desses mecanismos. Um exemplo é a proteção existente sobre o método da janela, forçando que a taxa utilizada pelo algoritmo seja no máximo duas vezes maior ou menor que o valor obtido pelo método da equação. Isso faz com que a banda seja diretamente controlada pelo mecanismo de equação e provoca variações bruscas na taxa da janela, tornando o protocolo menos estável e equitativo.

A figura 6.29 apresenta o resultado do experimento com o aumento da largura de banda do enlace para 4.2Mbit/s. O teste foi realizado com camadas do tipo exponenciais novas e PP. Neste caso, o TCP começou a sua execução 100s após o ALMTF já estar em regime permanente.

Devido à utilização do mecanismo de PP, o ALMTF inicia sua execução e logo se estabiliza na quinta camada, recebendo uma banda de aproximadamente 2500kbit/s. Como a taxa da próxima camada exige uma banda maior que a disponível no enlace (no caso, 4625kbit/s), o algoritmo permanece estável sem efetuar novas tentativas de *join*.

No instante 100s inicia o tráfego TCP concorrente, fazendo com que o fluxo ALMTF desça imediatamente de camada para realizar a adaptação da banda com o novo tráfego. Contudo, após reduzir a sua taxa de transmissão para aproximadamente 500kbit/s, o protocolo não consegue sustentar novamente uma nova camada, permanecendo estável na terceira.

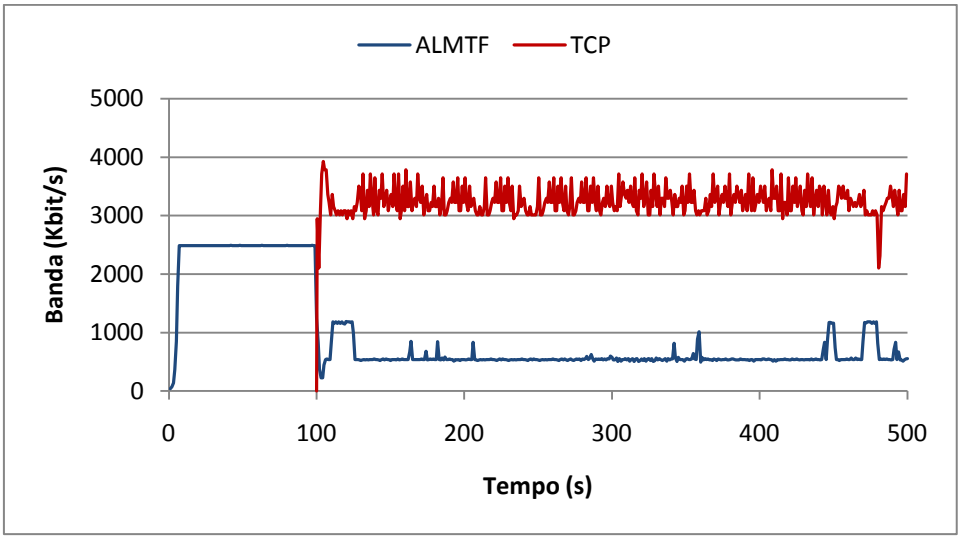

Figura 6.29: Equidade 11 - ALMTF começando 100s antes.

Diversos outros experimentos foram efetuados, analisando variações na limitação do enlace, tamanho da fila, quantidade de camadas, granularidade da transmissão, etc. De forma geral, os parâmetros que alteram os resultados são a taxa de transmissão do enlace e a granularidade das camadas.

Observou-se que quanto maior a taxa de transmissão do enlace, melhor o comportamento do protocolo ALMTF. Ainda que ele não compartilhe a banda da forma desejada com o TCP, seu desempenho é melhor do que em gargalos pequenos. Com relação à granularidade das camadas, uma forma de tornar o protocolo mais competitivo com o TCP é utilizando camadas com taxas mais altas, como por exemplo, camadas fixas de 500kbit/s. No entanto, camadas altas resultam em saltos bruscos na qualidade e dificuldade de adaptação com outros tipos de fluxos, devendo ser analisadas e usadas com cautela.

Em relação ao compartilhamento de banda com fluxos UDP concorrentes, o ALMTF efetuou uma divisão equitativa do enlace, como mostra a figura 6.30. Neste teste, um fluxo UDP de 500kbit/s foi transmitido juntamente com dois fluxos ALMTF. A figura 6.31 mostra o resultado equivalente no simulador.

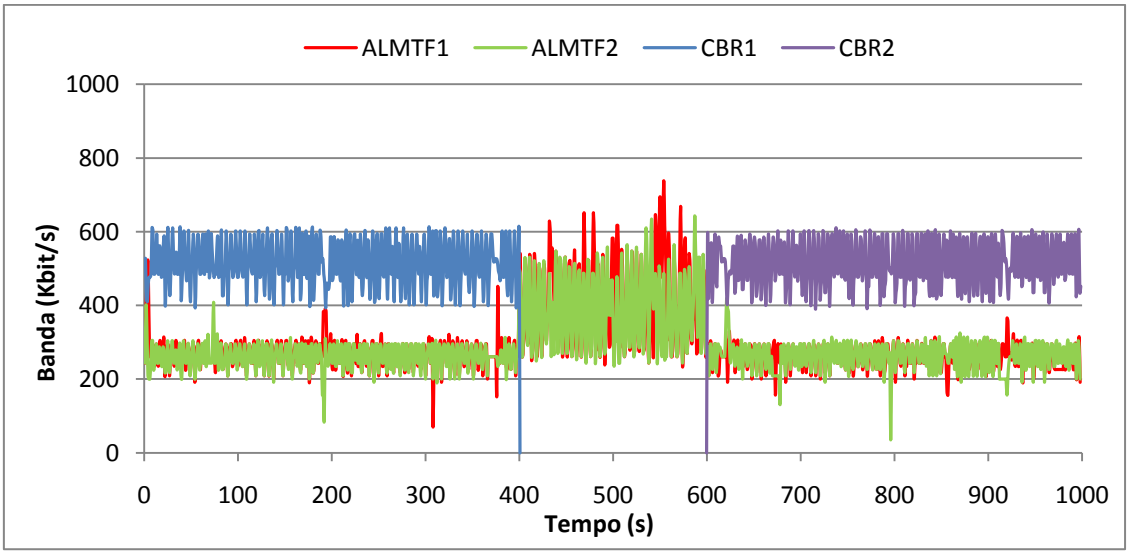

Figura 6.30: Equidade 12 - Equidade com fluxos UDP concorrentes.

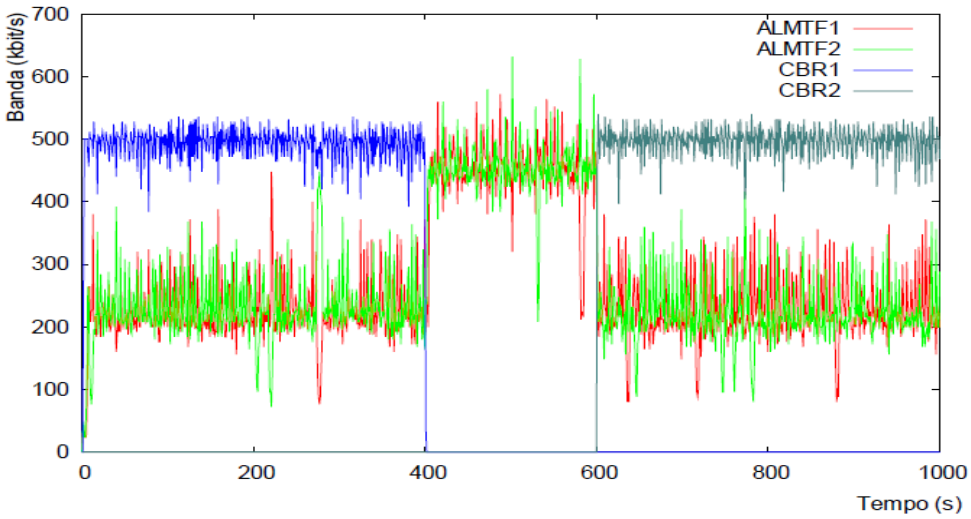

Figura 6.31: Simulação 12 – Equidade com fluxos UDP concorrentes.

Conforme pode ser verificado nas figuras, o fluxo UDP não se adapta e força os dois fluxos ALMTF a dividirem o restante da banda entre si. Quando o primeiro fluxo UDP termina, no instante 400s, os fluxos ALMTF voltam a subir, compartilhando a banda total do enlace (aproximadamente 500kbit/s cada um). O início do segundo fluxo UDP, no instante 600s, provoca perdas faz com que os fluxos ALMTF voltem a se adaptar, voltando a dividir o restante da banda adequadamente.

A figura 6.32 apresenta o mesmo experimento, mas com apenas um fluxo ALMTF. Da mesma forma que no teste anterior, o ALMTF dividiu a banda de forma equitativa com fluxos do mesmo tipo, aproveitando todo o enlace nos momentos em que estava rodando sozinho na rede.

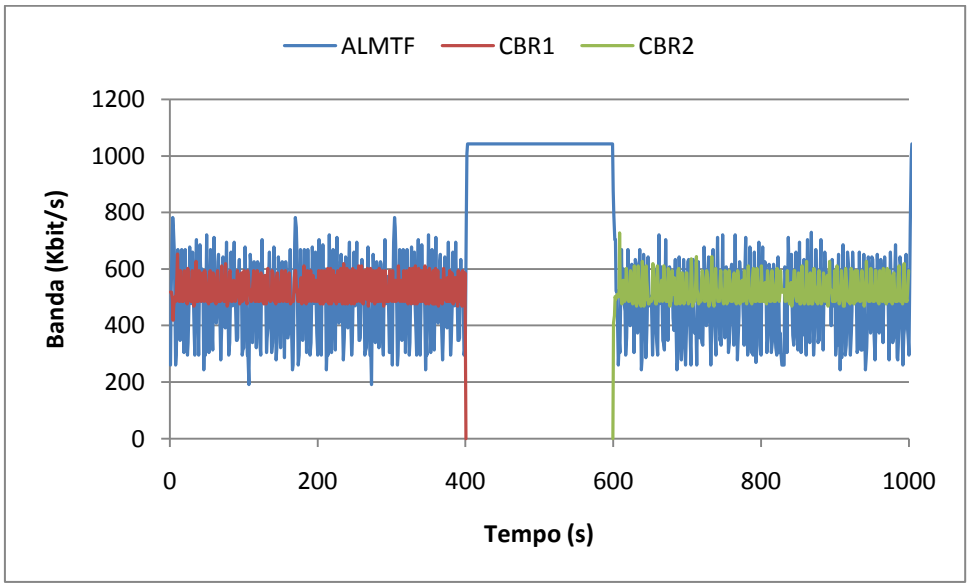

Figura 6.32: Equidade 13 - 1 ALMTF e 1 fluxo UDP.

De forma geral, os experimentos apresentados nesta seção mostram que o protocolo ALMTF consegue dividir a banda de forma equitativa com seu próprio tráfego e também com fluxos UDP concorrentes. Contudo, possui problemas para dividir a banda com fluxos TCP, devendo ser estudado e modificado para evitar a subutilização do enlace neste tipo de comunicação.

A dificuldade de se obter equidade com tráfegos TCP concorrentes não é uma novidade para os protocolos de controle de congestionamento multicast, como mostra a figura 6.33, retirada de Roesler (2003). Esta figura apresenta o resultado da simulação realizada com dois fluxos do algoritmo em questão e dois fluxos TCP, concorrendo por um enlace de 2 Mbit/s. Neste caso, a banda equitativa ideal para cada fluxo seria 500kbit/s.

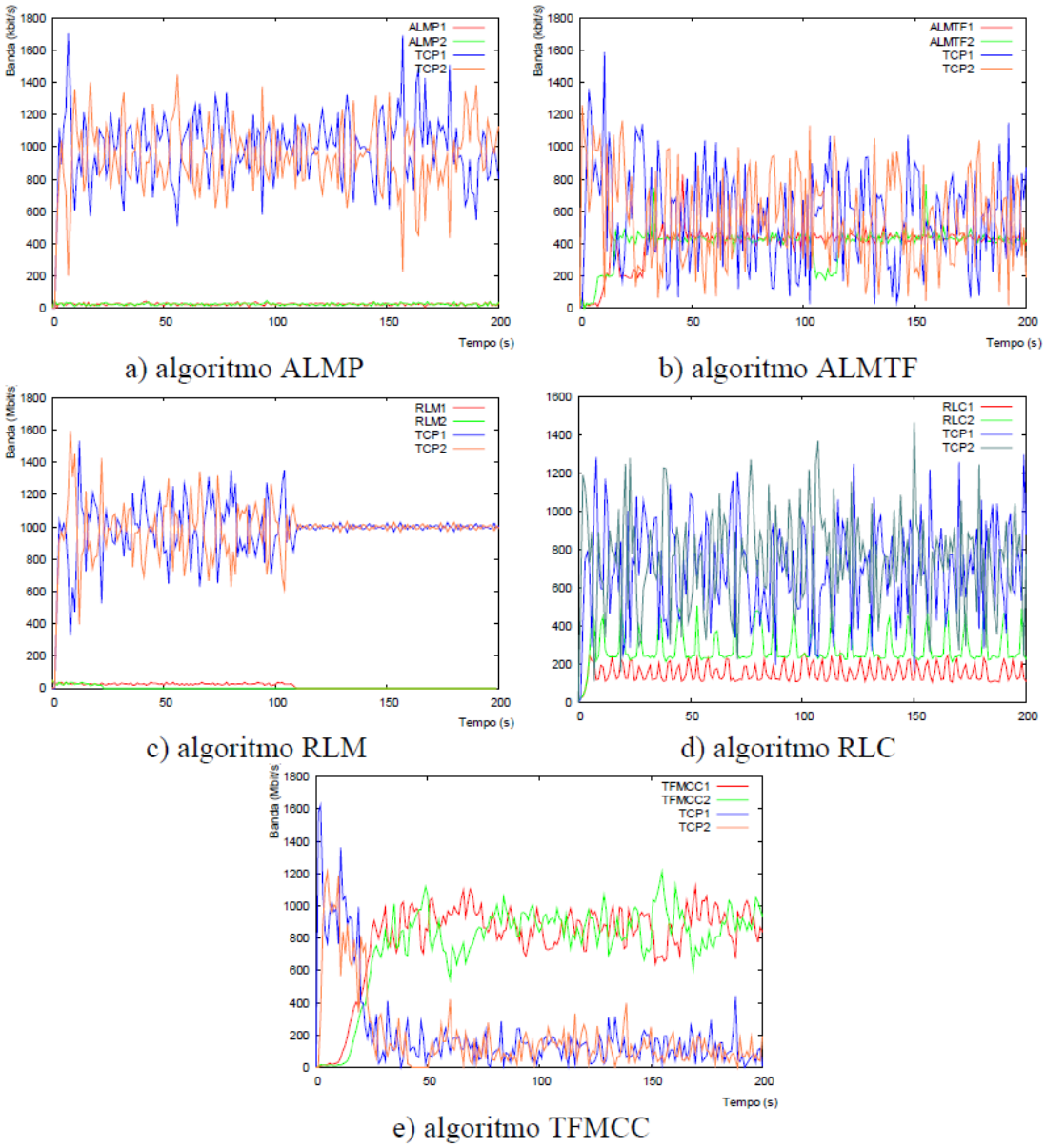

Figura 6.33: Comparação de outros protocolos com dois fluxos TCP (Roesler, 2003).

Conforme pode ser verificado, alguns protocolos não são TCP-*Friendly* e utilizam muito mais banda que o TCP, como é o caso do TFMCC (WIDMER; HANDLEY, 2001) (gráfico b). Os outros não conseguem se adaptar e dividir a banda de forma adequada na presença de fluxos TCP concorrentes, como foi o caso dos protocolos ALMP (Roesler, 2003) (gráfico a), RLM (MCCANNE; JACOBSON; VETTERLI, 1996) (gráfico c), RLC (VICISANO; RIZZO; CROWCROFT, 1998) (gráfico d) e o próprio ALMTF em ambientes reais.

6.3 Estabilidade na Transmissão

A métrica de estabilidade tem por objetivo verificar se o protocolo tem capacidade para manter uma transmissão estável ao longo do tempo ou se ocorrem muitas variações nas taxas recebidas pelos receptores.

A estabilidade do ALMTF foi analisada em todos os experimentos anteriores (ambiente real controlado) e também na RNP (ambiente real de longa distância. Para isso, foi verificada a quantidade de variações de camada por minuto (VCM) realizada por cada receptor, conforme definido na seção 5.2.

Para calcular o VCM de cada receptor, foi desenvolvido um programa em C (parser.cpp) que lê um arquivo de log e contabiliza todas as subidas e descidas de camada durante a execução, fornecendo a média de acordo com a duração de cada experimento. As tentativas frustradas de *join* contam como duas variações, pois o algoritmo sobe, gera perdas e desce em seguida. No entanto, diferentemente da simulação, os resultados mostram todas as variações ocorridas durante o experimento, não apenas as variações após a entrada do algoritmo em regime permanente.

Nas tabelas apresentadas a seguir, a primeira coluna identifica o experimento realizado, a segunda associa a figura ao experimento e a terceira apresenta os valores de VCM para cada receptor. Para facilitar a comparação dos resultados, nos experimentos onde existem mais de um receptor foi utilizado o mesmo padrão que a Tese, ou seja, foi fornecido o pior e o melhor resultado, bem como a média dos valores obtidos.

A tabela 6.1 apresenta o número de variações de camadas por minuto obtido através dos experimentos de adaptabilidade, enquanto a tabela 6.2 apresenta os mesmos resultados obtidos no simulador.

Tabela 6.1: Estabilidade em ambiente real controlado.

ADAPTABILIDADE				
Descrição	**Figura**	**VCM**		
		ALMTF1	**ALMTF2**	**ALMTF3**
Adaptabilidade 1	Figura 6.1	0,012	34	44,3
Adaptabilidade 2	Figura 6.4	0,012	0,08	0,01
Adaptabilidade 3	Figura 6.6	0,012	35,5	46,2
Adaptabilidade 4	Figura 6.10	0,04	0,02	0,08

Tabela 6.2: Estabilidade no simulador.

ADAPTABILIDADE				
Descrição	**Figura**	**VCM**		
		ALMTF1	**ALMTF2**	**ALMTF3**
Adaptabilidade 1	Figura 6.2	0,0	2,0	13,9
Adaptabilidade 2	Figura 6.5	0,0	0,0	0,0
Adaptabilidade 3	Figura 6.7	0,0	13,9	15,2
Adaptabilidade 4	Figura 6.9	0,0	0,0	0,0

Os experimentos “Adaptabilidade 2” e “Adaptabilidade 4” foram realizados utilizando o mecanismo de PP e por esse motivo tiveram um VCM desprezível, conforme o esperado, confirmando o resultado obtido na simulação. No entanto, os experimentos “Adaptabilidade 1” e “Adaptabilidade 3” obtiveram um VCM significativamente maior.

Isso é explicado pelo fato do simulador não apresentar o problema de *leave* ocorrido nos protocolos de transmissão em camadas, mascarando os resultados que seriam obtidos em ambientes reais. Em ambos os testes, o receptor ALMTF1 foi o único que obteve um VCM menor, e isto ocorreu somente porque a banda desse receptor era maior que a taxa de todas as camadas, o que possibilitou a sua estabilização.

Esses resultados reforçam duas premissas da Tese:

- Os fluxos com menos banda possuem um VCM mais alto e isso se deve ao maior número de tentativas de subir camada proveniente dos receptores inscritos nas camadas mais baixas;
- A utilização mecanismo de PP evita que o algoritmo tente subir camadas além da sua banda máxima, proporcionando uma maior estabilidade ao sistema.

A tabela 6.3 apresenta o número de variações de camadas por minuto obtido através dos experimentos de equidade, enquanto a tabela 6.4 apresenta os resultados equivalentes no simulador.

Tabela 6.3: Estabilidade em ambiente real controlado.

EQUIDADE				
Descrição	**Figura**	**VCM**		
		Pior	**Melhor**	**Média**
Equidade 1	Figura 6.11	29,5	27,5	28,5
Equidade 2	Figura 6.13	26	22,7	24,3
Equidade 3	Figura 6.15	44,6	45,4	45
Equidade 4	Figura 6.17	51	46	48,5
Equidade 5	Figura 6.19	48,3	47,9	48,1
Equidade 6	Figura 6.21	47	45	46
Equidade 7	Figura 6.23	24	17,5	20,7
Equidade 8 2 ALMTF 2 TCP	Figura 6.25	 8,1 114	 4,5 107	 6,3 110,5
Equidade 9 1 ALMTF 1 TCP	Figura 6.27	 - -	 17,5 165,33	 17,5 165,33
Equidade 10 1 ALMTF 1 TCP	Figura 6.28	- -	 22,3 170	 22,3 170
Equidade 11 1 ALMTF 1 TCP	Figura 6.29	- -	 28,5 178	 28,5 178

Tabela 6.4: Estabilidade no simulador.

EQUIDADE				
Descrição	**Figura**	**VCM**		
		Pior	**Melhor**	**Média**
Equidade 1	Figura 6.12	12,0	11,1	11,5
Equidade 8 2 ALMTF	Figura 6.26	3,4	3,2	3,3
2 TCP		48,5	44,6	46,5

A tabela 6.4 apresenta menos resultados que a anterior, pois foram os únicos testes disponibilizados na Tese repetidos via experimentação. Os demais resultados da Tese foram baseados em testes com 5 e 10 fluxos simultâneos para cada protocolo, o que impediu a sua realização visto que seria necessário um cenário com 10 e 20 computadores respectivamente.

Contudo, pode-se observar que os valores de VCM obtidos em ambiente real são geralmente bem maiores que no simulador. O problema do tempo de *leave* é uma das razões para essa diferença, no entanto, não é a única. Conforme explicado anteriormente, após os primeiros experimentos já foi possível identificar alguns pontos falhos no protocolo e seus mecanismos.

Um exemplo é a utilização conjunta dos mecanismos de janela e equação para realizar o controle de congestionamento. A proteção existente do método da equação sobre o método de janela faz com que ocorram variações bruscas na taxa, tornando o protocolo menos estável e equitativo.

Esse problema se estende também à variável *cwnd*, visto que ela reflete a banda da janela e sofre conseqüências com a sua alteração. Portanto, nem sempre as modificações em *cwnd* resultarão no comportamento esperado pelo algoritmo, podendo haver aumentos maiores que 50% na fase *start*-state e reduções acima de 30% na fase *steady-state.* As falhas tornam-se mais aparentes quando não é utilizado o mecanismo de PP, pois durante a sua utilização o algoritmo não leva em consideração valores maiores do que o inferido por ele.

Outro fator que afeta a estabilidade é a forma como está implementado o vetor de perdas do ALMTF. Nos testes realizados na simulação, o vetor de perdas era continuamente atualizado, mesmo durante o período de estabilização do protocolo. No entanto, se o protocolo está em estabilização e deve ignorar as perdas, seria mais eficiente se elas não fossem contabilizadas no vetor durante este período.

Com esses resultados, foi possível confirmar mais algumas premissas da Tese:

- A estabilidade do ALMTF é bem maior que a de fluxos TCP equivalentes;
- A estabilidade do protocolo diminui com o aumento de receptores.

A tabela 6.5 apresenta o número de variações de camadas por minuto obtido através dos experimentos em longa distância da RNP. A primeira parte dos resultados apresenta os valores encontrados nos POPs PE, PR e RS, A parte inferior mostra o resultado dos POPs PI, PR e TO. Em todos os testes, o transmissor ficou localizado no POP-RJ.

Os experimentos realizados foram os seguintes:

- **Estabilidade 1:** ALMTF com PP e camadas fixas;
- **Estabilidade 2:** ALMTF com PP e camadas exponenciais;

- **Estabilidade 3:** ALMTF sem PP e camadas fixas;
- **Estabilidade 4:** ALMTF sem PP e camadas exponenciais.

Tabela 6.5: Estabilidade na RNP.

ESTABILIDADE			
Descrição	**VCM**		
	POP-PE	**POP-RS**	**POP-PR**
Estabilidade 1	44,7	47,1	18,3
Estabilidade 2	5,2	4	1,8
Estabilidade 3	45,4	47,5	11
Estabilidade 4	7,2	4,7	1,7
Descrição	**VCM**		
	POP-PI	**POP-PR**	**POP-TO**
Estabilidade 1	43,5	1,1	64,5
Estabilidade 2	19,5	0,3	29,4
Estabilidade 3	45,3	1,13	69,5
Estabilidade 4	21,2	0,41	29,4

Conforme pode ser observado nos experimentos “Estabilidade 2” e “Estabilidade 4”, os testes realizados com camadas exponenciais deixaram o protocolo mais estável. Esses testes foram realizados utilizando oito camadas que iniciavam com 30kbit/s (camada zero) e chegavam em 3840kbit/s (camada sete), somando uma taxa total de 7550kbit/s.

Isto confirma mais uma premissa da Tese:

- Quanto mais próximas estiverem as camadas entre si, menor a estabilidade do algoritmo, pois as tentativas de subir camada serão realizadas de forma mais freqüente.

Contudo, é importante observar que independente da granularidade das camadas, o VCM dos fluxos foi bastante alto, o que não é favorável para aplicações multimídia em geral. Uma forma de minimizar a oscilação gerada pelo algoritmo seria desenvolver um mecanismo de estabilidade que aprenda com as falhas anteriores, reduzindo a probabilidade de *joins* sem sucesso durante a execução. Outra alternativa seria implementar uma fila de recepção para os pacotes, de forma que eles fossem buferizados antes da entrega, garantindo assim uma maior estabilidade a transmissão.

6.4 Escalabilidade

A métrica de escalabilidade foi analisada através da topologia 3, apresentada anteriormente na figura 5.3. Os quesitos escolhidos para verificar o comportamento do ALMTF estão listados a seguir e foram repetidos antes e após as melhorias realizadas no algoritmo.

- **Pedidos de *feedbacks*:** quantidade de pedidos de *feedbacks* por unidade de tempo chegando ao transmissor;
- **Taxa de atualização do RTT:** tempo médio entre atualizações do RTT;
- **Erro de estimativa:** verifica o erro percentual no cálculo do RTT em relação ao valor estimado na rede.

As figuras 6.34 e 6.35 apresentam o resultado com relação ao número médio de pedidos de *feedbacks* por segundo, para 10 e 25 receptores respectivamente. O intervalo de obtenção do número médio de pedidos de *feedback* é de 60 segundos e a duração do teste é de 30 minutos.

Como pode-se observar em ambas as situações, as alterações realizadas no ALMTF aumentaram levemente a média de pedidos de *feedbacks* enviados ao transmissor. Esse comportamento em relação ao algoritmo original pode ser explicado devido ao número de receptores utilizados no experimento, pois sendo esse número inferior à janela de intervalo máxima entre pedidos de *feedback* utilizado no ALMTF (60 segundos), o número médio de pedidos tende a se manter próximo a um.

Contudo, essa janela máxima e a falta de um mecanismo de supressão poderiam tornar o protocolo inviável para utilização em larga escala, o que é evitado após as modificações realizadas, pois os novos mecanismos fazem com que a taxa de supressão cresça com o aumento de receptores, como é demonstrado em Nonnenmacher (1999).

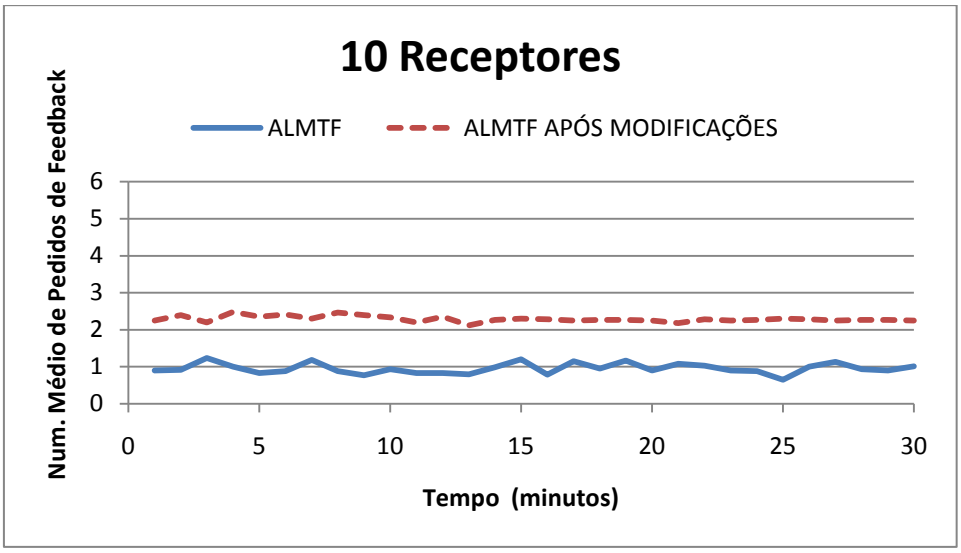

Figura 6.34: Número médio de pedidos de *feedbacks* para 10 receptores.

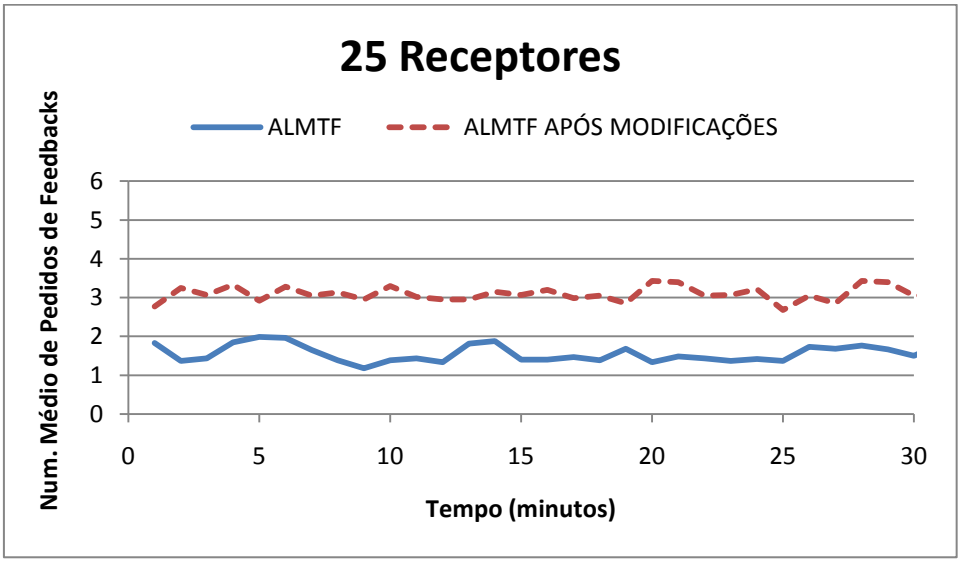

Figura 6.35: Número médio de pedidos de *feedbacks* para 25 receptores.

Com o objetivo de verificar a influência dos novos mecanismos de supressão de *feedbacks* para a rede local, os testes foram repetidos somente na rede da UFRGS, para 10 e 22 computadores. Os resultados são apresentados nas figuras 6.36 e 6.37 respectivamente.

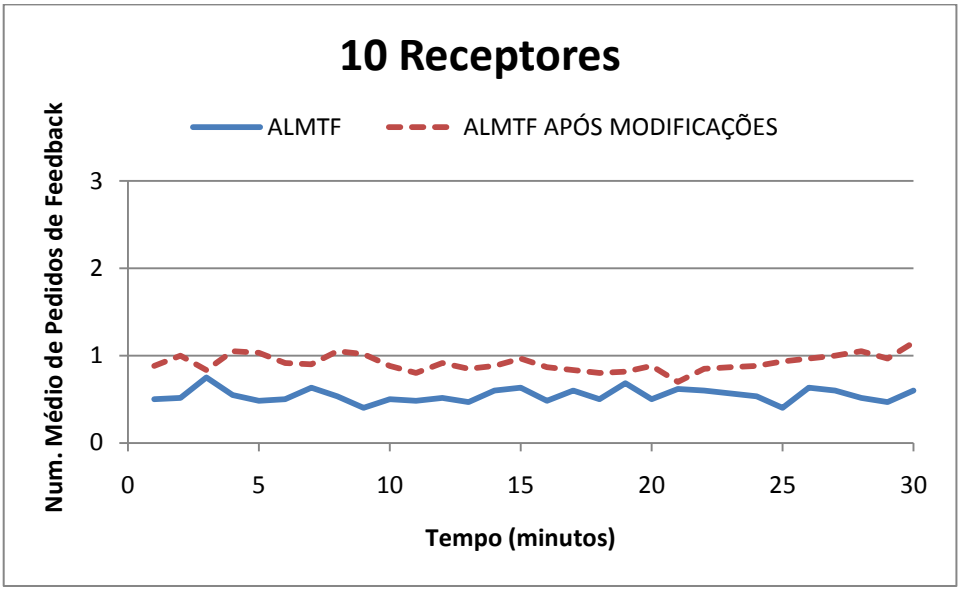

Figura 6.36: Número médio de pedidos de *feedbacks* para 10 receptores.

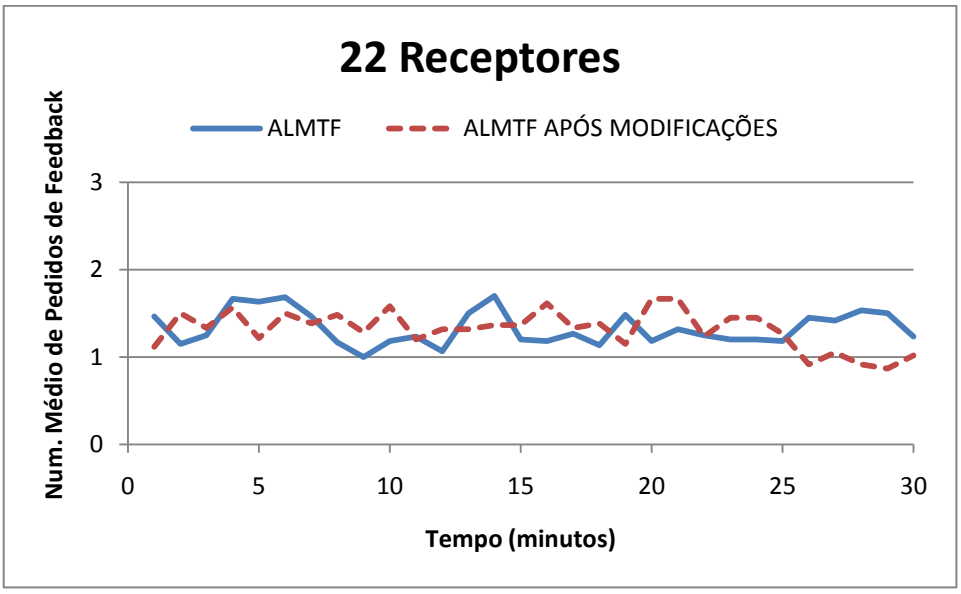

Figura 6.37: Número médio de pedidos de *feedbacks* para 22 receptores.

Devido a troca de informações entre receptores da mesma rede local ocorrer a uma velocidade muito superior à troca de mensagens entre receptores e transmissor, foi possível aumentar a taxa de supressão e reduzir o número médio de pedidos de *feedbacks* ao transmissor.

Esses resultados demonstram o ganho obtido com a utilização do novo mecanismo de controle de *feedbacks*, visto que aproxima aos valores obtidos pelo ALMTF original mesmo com um número de receptores dentro da sua janela máxima de intervalo.

A Tabela 6.6 apresenta, em segundos, os resultados obtidos para o intervalo de atualização de RTT em laço fechado, para 10 e 25 receptores respectivamente. Nas tabelas a seguir, os dados mostram os valores máximo, mínimo e médio de cada experimento.

Tabela 6.6: Intervalo de atualização para 10 e 25 receptores.

Protocolo	**10 Receptores**			**25 Receptores**		
	Máx.	**Mín.**	**Méd.**	**Máx.**	**Mín.**	**Méd.**
ALMTF	20,7	1,07	6,9	37,4	1,30	14,5
ALMTF após melhorias	17,1	0,03	3,1	14,7	0,01	3,3

A partir desses resultados pode-se observar o ganho significativo no tempo de atualização do RTT obtido pelo protocolo ALMTF após as melhorias realizadas nos mecanismos de cálculo de RTT.

Esse ganho está relacionado à utilização da cooperação entre receptores da mesma rede local, o que permite uma atualização mais freqüente do RTT. Percebe-se, portanto, que foi possível aumentar a eficiência no cálculo garantindo escalabilidade ao protocolo.

Um detalhe interessante da tabela 6.6 é que após as melhorias, o tempo máximo de intervalo de atualização foi menor para 25 receptores do que para 10 receptores. Isso pode ser explicado devido ao fato dos receptores utilizarem aprendizado dos vizinhos, minimizando assim o tempo máximo de espera.

Além disso, observa-se também alguns efeitos devido a perdas de pacotes, provavelmente no enlace *wireless* da UFRGS. No ALMTF original, a fórmula de intervalo máximo é igual a “Número de Receptores + 1” (para até 60 receptores). No entanto, pode-se verificar que o ALMTF apresenta valores máximos maiores que isso, demonstrando perdas nos enlaces.

A Tabela 6.7 mostra os resultados obtidos para o erro percentual. O erro é calculado sobre a estimativa *one-way* em relação ao RTT calculado efetivamente, conforme explicado na definição das métricas.

Tabela 6.7: Erro para 10 e 25 receptores.

Protocolo	**10 Receptores (%)**			**25 Receptores (%)**		
	Máx	**Mín.**	**Méd**	**Máx**	**Mín.**	**Méd.**
ALMTF	26,8	0,0	0,2	8,0	-14,4	0,2
ALMTF após melhorias	5,9	-11,8	1,5	19,8	0,0	1,9

Observa-se que o erro permanece baixo para ambos os algoritmos, ou seja, antes e depois das melhorias existe uma boa aproximação entre o RTT calculado e o RTT real.

7 CONCLUSÃO

O multicast é a estratégia de transmissão mais apropriada para as aplicações multimídia que envolvem uma grande quantidade de usuários, pois economiza tráfego na rede e minimiza o processamento dos roteadores intermediários. Contudo, ainda existem vários obstáculos para o seu uso disseminado na Internet global, sendo um deles o desenvolvimento de protocolos de controle de congestionamento adequados.

Apesar das inúmeras pesquisas na área, o controle de congestionamento multicast ainda não foi padronizado. Entre os fatores que contribuem para esse problema estão questões como a estabilidade, escalabilidade e equidade da transmissão, o que aumenta a complexidade de implementação e validação desses protocolos.

O objetivo deste trabalho foi estender o protocolo ALMTF para uma rede real, implementando, validando os seus mecanismos e propondo novas alternativas que o adaptem para este ambiente. Além disso, efetuar uma comparação dos resultados obtidos experimentalmente com a simulação, identificando as diferenças e promovendo as pesquisas experimentais na área.

As principais contribuições deste trabalho foram:

- Validação do protocolo ALMTF em redes locais e de longa distância;
- Comparação dos resultados reais com a simulação;
- Modificação dos mecanismos de cálculo de RTT e controle de *feedbacks*.

Através dos resultados obtidos, conclui-se que o protocolo ALMTF se adapta em ambientes heterogêneos e transmite de forma equitativa com seu próprio tráfego, bem como com tráfegos UDP concorrentes. No entanto, diferente dos resultados obtidos na simulação, possui deficiências com relação ao compartilhamento de banda com o protocolo TCP.

Da mesma forma que a maioria dos protocolos existentes, o ALMTF não conseguiu se adaptar e dividir a banda de forma adequada na presença de fluxos TCP, utilizando uma taxa muito inferior que a sua parcela equitativa de banda. Portanto, para que ele possa ser utilizado em redes como a Internet, são necessárias modificações no seu mecanismo de controle de congestionamento atual.

No simulador, a situação ideal foi encontrada utilizando um mecanismo dez vezes menos agressivo que o TCP. Contudo, em ambientes reais essa situação não é favorável, sendo necessário tornar o algoritmo mais competitivo. Além disto, a utilização conjunta

dos mecanismos de equação e janela ocasionou diversas inconsistências, aumentando a complexidade do ALMTF.

Resultados diferentes também foram observados com relação à estabilidade na transmissão. De forma geral, o ALMTF é mais instável na prática do que no simulador. Contudo, isto é justificável devido ao problema do tempo de *leave* do protocolo IGMP, inexistente no NS-2 e presente em ambientes reais.

Apesar dessas diferenças, os resultados obtidos em redes locais e de longa distância confirmaram várias premissas da Tese, tais como:

- A utilização mecanismo de PP evita que o algoritmo tente subir camadas além da sua banda máxima, proporcionando uma maior estabilidade ao sistema;
- Os fluxos com menos banda possuem um VCM mais alto e isso se deve ao maior número de tentativas de subir camada proveniente dos receptores inscritos nas camadas mais baixas;
- A estabilidade dos fluxos ALMTF é muito maior que a de fluxos TCP equivalentes;
- A estabilidade do ALMTF diminui com o aumento de receptores;
- Quanto mais próximas estiverem as camadas entre si, menor a estabilidade do algoritmo, pois as tentativas de subir camada serão realizadas de forma mais frequente.

Outra contribuição deste trabalho foi a reformulação dos mecanismos de cálculo de RTT e controle de *feedbacks*. Conforme visto anteriormente, o ALMTF não utilizava nenhuma técnica para monitorar os pedidos de *feedbacks* dos receptores, tendo seu controle baseado apenas no intervalo entre os envios de mensagens em cada receptor.

Além disso, o protocolo não possui nenhum controle efetivo na taxa de envio das mensagens. Por esse motivo, se fosse utilizado em larga escala poderia perder eficiência e até mesmo deixar o transmissor sem recursos.

Conforme demonstrado na seção 6.4, as melhorias realizadas aumentaram a eficiência dos mecanismos, garantindo escalabilidade ao protocolo. Devido à troca de informações entre os receptores da mesma rede local, foi possível otimizar a taxa de supressão e reduzir o número médio de pedidos de *feedbacks* enviados ao transmissor. Essas melhorias tendem a beneficiar também a equidade de tráfego com o TCP, visto que os receptores terão uma atualização mais frequente do RTT.

7.1 Dificuldades encontradas

Durante a fase de validação, existiram algumas dificuldades as quais necessitaram um pouco mais de tempo para serem resolvidas. A principal delas foi a definição do cenário de teste a ser utilizado nos experimentos em ambiente controlado (topologias 1 e 2).

Para analisar as métricas de adaptabilidade e equidade, por exemplo, era necessário limitar a largura de banda dos receptores. Apesar da existência de inúmeros softwares para isto, moldar a taxa dos fluxos multicast não foi uma tarefa trivial. As seções 7.1.1 a 7.1.5 descrevem todos os ambientes de teste configurados e testados durante este trabalho, destacando os problemas encontrados em cada um.

Outra dificuldade verificada na prática foi com relação ao tempo de *leave* do protocolo IGMP. O tempo mínimo de 3 segundos para que os roteadores parem de encaminhar os pacotes multicast indicou uma limitação real para os protocolos que utilizam a técnica de transmissão em camadas. Algumas alternativas para evitar esse problema e minimizar a quantidade de operações de *join*/*leave* são a utilização de mecanismos de aprendizado e camadas dinâmicas, ao invés de camadas fixas.

7.1.1 FreeBSD, IPFW e Dummynet

O primeiro cenário foi configurado com um roteador FreeBSD e os módulos IPFW[1] (IPFW; 2007) e *Dummynet*[2] (RIZZO; 2007), responsáveis pela limitação da banda dos receptores. Embora esse ambiente funcione perfeitamente com fluxos unicast, observou-se que a limitação de banda não funcionou no multicast.

Com essas ferramentas, a limitação da banda é realizada através do endereço IP ou do endereço de rede do receptor. Contudo, como os pacotes multicast são endereçados para um grupo, eles não caiam na limitação imposta pelo roteador.

Outro problema encontrado neste cenário foi com relação ao mecanismo de PP. Em todos os testes realizados, o PP inferiu sempre a metade da banda real do receptor, dificultando a análise dos resultados. Por esses motivos, este ambiente foi descartado e iniciou-se a implantação do próximo cenário, que será descrito a seguir.

7.1.2 *Switch* Gerenciável

O segundo cenário de teste foi configurado utilizando um *switch* gerenciável[3] com recursos de limitação de banda por porta. Após o estudo e a configuração do equipamento, verificou-se que ele limitava corretamente o tráfego multicast, no entanto, alterava o espaçamento original entre os pacotes transmitidos, fazendo com que o mecanismo de PP não funcionasse adequadamente.

Além disso, o *switch* possuía um *buffer* para o armazenamento dos pacotes, o que evitava que os dados fossem perdidos quando se utilizada a limitação. Por esse motivo, introduzia uma grande latência na transmissão, o que ocasionava valores altos no RTT e, consequentemente, problemas nos mecanismos de janela e equação.

7.1.3 NISTNet, HTB e CBQ

O terceiro cenário de teste foi composto por um roteador Linux e ferramentas específicas para limitação de banda, como o NISTNet (NIST; 2008), HTB (*Hierarchical Token Bucket*) (PAYNTER; 2008) e CBQ (*Class-based Queueing*) (KUZNETSOV; 2008).

O NISTNet é um emulador de redes que permite gerar diversos efeitos nas interfaces de rede, como atraso, *jitter*, limitação de banda, perdas, duplicação de pacotes, entre outros. Os escalonadores HTB e CBQ fazem parte da implementação de QoS[4] nativa do Linux e possuem como principal objetivo o controle do tráfego de saída de uma interface.

---

[1] *IPFW* é um filtro de pacotes (*firewall*) nativo no *kernel* de sistemas operacionais BSD.

[2] *Dummynet* é um programa para limitação de banda em sistemas FreeBSD.

[3] O *switch* utilizado nos testes foi um DLINK modelo DES-3526.

[4] As funcionalidades de QoS são baseadas no pacote *iproute2*.

Todas essas ferramentas foram instaladas e testadas isoladamente, tendo como principal restrição o fato de não possibilitar a limitação de banda dos tráfegos multicast. A limitação podia ser configurada através dos endereços IP de origem ou destino dos receptores (unicast), no entanto, não reconhecia os grupos multicast.

7.1.4 PlanetLab

O PlanetLab é um laboratório virtual para o desenvolvimento de novas aplicações para a Internet. O ambiente é resultado de um consórcio de inúmeras instituições, que atualmente conta com mais de 700 computadores espalhados em 25 países (REDIGOLO et al; 2007).

Visando a realização de experimentos em larga escala, a próxima tentativa foi a utilização deste ambiente. Para que isso fosse possível, foi necessário submeter um projeto e obter uma autorização junto a uma das instituições participantes do consórcio (no caso, a instituição escolhida foi a RNP).

A única exigência para o uso do ambiente foi a utilização de aplicações GNU/Linux. Até o momento isso não tinha sido realizado, pois o algoritmo rodava nas extremidades da rede (em máquinas com plataforma Windows), sendo apenas o núcleo (roteador) baseado em Linux ou BSD. Portanto, foi realizada a migração do código para essa plataforma, além da revisão e validação das ferramentas neste ambiente.

No entanto, após a configuração inicial do PlanetLab e a realização dos primeiros experimentos, descobriu-se que ele funcionava apenas com multicast no nível de aplicação (e não na camada de rede), o que impossibilitava a validação correta do algoritmo. Por este motivo, este cenário de testes também foi descartado.

7.1.5 TBF e *Traffic Control*

A solução para o ambiente de testes foi encontrada através da configuração de um roteador Linux com várias interfaces de rede e os módulos *mrouted*, TC (*Traffic Control*) e TBF (*Token Bucket Filter*).

O TBF é outro escalonador nativo do Linux, que utiliza o modelo de balde com *tokens* para moldar o tráfego de saída. Contudo, apresenta a vantagem de limitar a velocidade integral de uma interface de rede. Desta forma, todos os pacotes eram afetados, independente do tráfego utilizado nas aplicações (multicast ou unicast).

É possível observar, portanto, que a maioria das ferramentas existentes não está apta para trabalhar adequadamente com tráfegos multicast, o que agrega uma dificuldade adicional para a validação experimental dos protocolos de controle de congestionamento.

7.2 Trabalhos Futuros

Como extensões ao que foi desenvolvido até o momento, existem dois pontos importantes: a modificação do mecanismo de controle de congestionamento e o desenvolvimento de uma técnica de aprendizado para melhorar a estabilidade do protocolo.

Com relação ao mecanismo de controle de congestionamento, durante esta pesquisa foi realizada a separação dos mecanismos de equação e janela, observando-se melhores resultados com a utilização isolada do mecanismo de equação. Contudo, estudos adicionais são necessários para adaptar a formula da equação utilizada no protocolo, visando um cálculo mais correto da banda equivalente do TCP.

A implementação de um mecanismo que aprende com as falhas anteriores é uma alternativa para deixar o protocolo mais estável, minimizando o problema causado pelas tentativas frustradas de *join*.

O mecanismo poderia funcionar da seguinte maneira: a cada tentativa de *join* sem sucesso, o protocolo armazena o estado atual da rede no momento da falha. Nas próximas tentativas de subida, o protocolo irá analisar se existe algum estado de rede parecido com o atual. Se houver ele não sobe, caso contrário, tem permissão para subir.

REFERÊNCIAS

BAVIER et al. PlanetLab-VINI: Uma infra-estrutura de redes virtualizada. In: Workshop de P&D do Projeto GIGA/RNP- RJ, 2007.

BENSLIMANE, Abderrahim. **Multimedia Multicast on the Internet.** France: Hadback, 2007.

BIAN, J. et al. Congestion Control Protocol Based on Delay Parameters for Layered Multicast Communication. **IEEE/ICCT**, Conference on Communication, p.1-4, 2006.

BRUNO, G. **VEBIT: um novo algoritmo para codificação de vídeo com escalabilidade**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

BYERS, J. et al. Fine-Grained layered Multicast with STAIRS. In: IEEE/ACM Transactions on Networking, 2006. **Proceedings**. . . New York, v.14, n. 1, 2006, p. 81-93.

BYERS, John et al. FLID-DL: Congestion control for layered multicast. In: INTERNATIONAL WORKSHOP ON NETWORKED GROUP COMMUNICATION, NGC, 2., 2000, USA. **Proceedings**… New York: ACM, 2000.

CAIN et al**. Internet Group Management Protocol, Version 3**. RFC 3376. California: IETF, 2002. http://www.rfc-editor.org/rfc/rfc3376.txt.

CARVALHO, J. S. A. et al. FeedC: um novo mecanismo de cálculo de RTT e controle de feedback para transmissões multimídia de larga escala. In: WEBMEDIA, 2008. **Proceedings**... [S.1.: s.n.], 2008.

CHEN et al. Extending TCP congestion control to multicast. **Computer Networks**. 51, 11 (Aug. 2007), 3090-3109. DOI= http://dx.doi.org/10.1016/j.comnet.2007.01.004.

DARONCO, L. C. **Avaliação Subjetiva de Qualidade Aplicada à Codificação de Vídeo Escalável**. 2009. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

DAVIES, Guy. **Designing and Developing Scalable IP Networks**. Inglaterra: Wiley, 2004.

DETSCH, A; BARCELLOS, M. P. **Controle de Congestionamento com Suporte a ECN em Protocolos Multicast de Taxa Única**. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, SBRC 2003, 11., 2003, Natal. Porto Alegre: Sociedade Brasileira de Computação, 2003. v. 1., p. 455-470.

FENNER, W. **Internet Group Management Protocol, Version 2**: RFC 2236. California: IETF, 1997. http://www.ietf.org/rfc/rfc2236.txt

FLOYD, S.; FALL, K. Promoting the use of end-to-end congestion control in the Internet. **IEEE/ACM Transactions on Networking**, New York, v.7, n.4, p.458–472, Aug. 1999.

GEVROS et al. "Congestion control mechanisms and the best effort service model". **IEEE Network**, New York, v.15, n.3, May/June 2001.

HUBERT, B. et al. Linux **Advanced Routing & Traffic Control HOWTO**. Disponível em <http://lartc.org/howto/>. Acessado em: fev. 2009.

INJONG, R.; LISONG, X. Limitations of equation-based congestion control. **IEEE/ACM Transactions on Networking**, New York, v.35, n.4, p.49–60, Oct. 2007.

IPFW. Disponível em: <http://www.freebsd.org/doc/en/books/handbook/firewalls-ipfw.html>. Acesso em: abr. 2007.

JIANG, L.; SHIVKUMAR, K. **ORMCC**: A Simples and Effective Single-Rate Multicast Congestion Control Scheme. 2003. Disponível em: <http://citeseer.ist.psu.edu/624716.html>. Acesso em: jun. 2009.

KALAIARASAN, C.; SELVAN, S. A New Approach for Multicast Congestion Control using Asymmetric Paths. In: INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND COMMUNICATIONS, ADCOM, 14., Feb. 2006. **Proceedings**… India: IEEE, Feb. 2006 p.389-392.

KAMMOUN, W.; YOUSSEF, H. Improving the performance of End-to-End single rate Multicast Congestion Control. Computers and Communications, 2008. ISCC 2008. **IEEE Symposium**, p.1128-1132, 2008.

KROB, A. et al. ALMTF: adaptive layered multicast tcp-friendly. In: WEBMEDIA, 2007. **Proceedings**. . . [S.l.: s.n.], 2007, p.9–16.

KUZNETSOV, A. **CBQ: Manual Reference Pages**. Disponível em: < http://www.squarebox.co.uk/cgi-squarebox/manServer/tc-cbq-details.8>. Acesso em maio 2008.

KWON, G.; BYERS, J. W. Smooth Multirate Multicast Congestion Control. In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, IEEE INFOCOM, 22., 2003, San Francisco, CA. **Proceedings**. . . Piscataway: IEEE, 2003. v.2, n.1, p.1022–1032.

LEGOUT, A.; BIERSACK, E. PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes. In: INTERNACIONAL CONFERENCE ON MEASUREMENT AND MODELING OF COMPUTER SYSTEM, SIGMETRICS, 2000, Santa Clara, California. **Proceedings**… New York: ACM, 2000. p. 13-22.

LI, B.; LIU, J. Multirate Video Multicast over the Internet: an overview. **IEEE Network**, New York, USA, v.17, n.1, p.24–29, 2003.

LI, J. et al. Generalized multicast congestion control. **Elsevier Computer Networks**, [S.l.], v.51, n.6, p.1421–1443, Apr. 2007.

LI, J.; YUKSEL, M.; KALYANARAMAN, S. Explicit Rate Multicast Congestion Control. **Computer Networks**, [S.l.], v. 50, n. 15, p. 2614-2640, Oct. 2006.

LI, Y.; MUNRO, A; KALESHI, D. Multi-rate Congestion Control over IP Multicast. In: INTERNACIONAL CONFERENCE ON NETWORKING, ICN, 4., Apr. 2005, **Proceedings**… France: Springer, 2005, p.1012-1022.

LIU, J. et al. Adaptive Video Multicast over the Internet. **IEEE Multimedia**, [S.l.], v.10, n.1, p.22–33, Mar. 2003. doi:10.1109/MMUL.2003.1167919.

LIU, J. et al. Adaptive Video Multicast over the Internet. **IEEE Multimedia**, [S.l.], v.10, n.1, p.22–33, Mar. 2003. doi:10.1109/MMUL.2003.1167919.

MACCANNE, S. et al. Receiver driven layered multicast. In: ACM SIGCOMM, 1996, Stanford, California, USA. **Proceedings**. . . New York: ACM, 1996. p.117–130.

NETEM. Disponível em: < http://www.linuxfoundation.org/en/Net:Netem>. Acesso em ago. 2007.

NIST. Disponível em: <http://www.antd.nist.gov/nistnet>. Acesso em: mar. 2008.

NONNENMACHER, J.; BIERSACK, E. Scalable Feedback for Large Groups. **IEEE/ACM Transactions on Networking**, pp. 375–386, June 1999.

PADHYE, J. et al. Modeling TCP Reno performance: a simple model and its empirical validation. **IEEE/ACM Transactions on Networking**, New York, v.8, n.2, Apr. 2000.

PADHYE, J. et al. Modeling TCP throughput: a simple model and its empirical validation. In: ACM Special Interest Group on Communications, **ACM SIGCOMM**, 1998, Vancouver, Canada, 1998. New York: ACM, v.28, n.4, 1998.

PAPADIMITRIOU, P; TSAOUSSIDIS, V. Real-Time Video Transport in Heterogeneous IP Networks. In: Mobile Multimedia: Communication Engineering Perspective, Editors I. K. Ibrahim and D. Taniar, pp. 117-135. Nova Publishers, 2006, ISBN: 1-60021-207-7.

PAPAZIS, K. et al. A New Adaptative Layered Multicast Protocol. In: INTERNACIONAL CONFERENCE HIGH SPEED NETWORK AND MULTIMIDIA COMMUNICATIONS, HSNMC, 2004, Toulouse. **Proceedings**… France: IEEE, 2004. p. 381-389.

PAYNTER, E. HTB User Manual. Disponível em: < http://luxik.cdi.cz/~devik/qos/htb/userg.pdf>. Acesso em abr. 2008.

PERES, M. R et al. An Auto-Configurable Hybrid Approach to Multicast Congestion Control. In: CONFERENCE GLOBAL TELECOMMUNICATIONS, GLOBECOM, 2., 2005, St. Louis, MI. **Proceedings**… New York: IEEE, 2005. v.2., p. 657-661.

PUANGPRONPITAG, S. et al. Performance Evaluation of Layered Multicast. Congestion Control Protocols: FLID-DL vs. PLM. **Proceedings**... SPECTS 2003, Montreal, 2003.

REDIGOLO, F. et al. **PlanetLab - Ambiente para Desenvolvimento e Pesquisa de Aplicações Distribuídas.** In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, SBRC 2007, 25., 2007, Belém. PA: Sociedade Brasileira de Computação, 2007.

RHEE, I.; OZDEMIR, V.; YI, Y. **TEAR**: TCP emulation at receivers flow control for multimedia streaming. North Carolina: NCSU, Department of Computer Science, 2000. (Technical Report).

Rizzo, L. Dummynet Home Page. Disponível em: < http://www.dummynet.com/>. Acesso em: maio 2007.

RIZZO, L. Fast group management in IGMP. In: HIGH PERFORMANCE PROTOCOL ARCHITECTURES (Hipparch Workshop), 1998, London, UK. **Proceedings**… London: UCL, 1998.

ROESLER, V. **SAM**: um sistema adaptativo para transmissão e recepção de sinais multimídia em redes de computadores. 2003. Tese (Doutorado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

SEADA, K.; HELMY, A. Fairness evaluation experiments for multicast congestion control protocols. In: **Proceedings** of IEEE Globecom'02, Nov. 2002, v.3, p. 2614-2618, Nov. 2002.

SILVA, C. L. J. **ProCon** - Prognóstico de Congestionamento de Tráfego de Redes usando Wavelets. 2004. 169f. Tese (Doutorado em Ciência da Computação) - Universidade Federal de Pernambuco, Recife.

TIRUMALA et al. **Iperf - The TCP/UDP bandwidth measurement tool**. Disponível em: <http://www.dast.nlanr.net/Projects/Iperf>. Acessado em: Jan. 2008.

TSAO et al. Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes on Fairness, Aggressiveness, and Responsiveness, **IEEE Network**, v.21, n.6, p.6-15, 2007.

VICISANO, L. et al. TCP-like congestion control for layered multicast data transfer. In: IEEE INFOCOM, 1998, San Francisco, California, USA. **Proceedings**. . . New York: IEEE, 1998.

WEI, S.; SHIN, K. G. **TCP performance under aggregate fair queuing**. In.: GLOBAL TELECOMUNICATIONS CONFERENCE, 2004. GLOBECOM '04. [S.l.]: IEEE, v.3, p.1308-1313, 2004.

WIDMER, J. et al. A Survey on TCP-Friendly Congestion Control. **IEEE Network**, [S.l.], v.15, n.3, p.28–37, May 2001.

WIDMER, J.; HANDLEY, M. Extending equation-based congestion control to multicast applications. In: CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATIONS, 2001. **Proceedings**. New York: ACM, 2001. p.275–285. doi:10.1145/383059.383081.

WU, L.; SHARMA, R.; SMITH, B. Thin Streams: an architecture for multicasting layered video. In: WORKSHOP ON NETWORK AND OPERATING SYSTEMS SUPPORT FOR DIGITAL AUDIO AND VIDEO, 1997, Washington, USA. **Proceedings**... New York: IEEE, 1997.

YANG, R.; SIMON, S. Internet Multicast Congestion Control: A Survey. In **Proceedings** of ICT 2000, Acapulco, Mexico, May 2000.