UNIVERSIDADE DO VALE DO RIO DOS SINOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
APLICADA - PIPCA

Avaliação do Protocolo Multicast PePcc para Transmissões Confiáveis na Internet

por

ROBERTO PERADOTTO

Dissertação submetida a avaliação,
como requisito parcial para a obtenção do grau de
Mestre em Ciência da Computação

Prof Dr. Antônio Marinho Pilla Barcellos
Orientador

São Leopoldo, junho de 2003

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Peradotto, Roberto

Avaliação do Protocolo Multicast PePcc para Transmissões Confiáveis na Internet / por Roberto Peradotto. — São Leopoldo: Centro de Ciências Exatas e Tecnológicas da UNISINOS, 2003.

92 f.: il.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos. Centro de Ciências Exatas e Tecnológicas Programa Interdisciplinar de Pós-Graduação em Computação Aplicada - PIPCA, São Leopoldo, BR–RS, 2003. Orientador: Barcellos, Antônio Marinho Pilla.

1. controle de congestionamento. 2. multicast. 3. Internet. I. Barcellos, Antônio Marinho Pilla. II. Título.

"Dedico esta dissertação a minha esposa Margot e a minha filha Brenda "

Agradecimentos

Agradeço sinceramente a todas pessoas que contribuíram para a realização desta dissertação, tanto pelo apoio técnico, quanto por criarem as condições necessárias para que ele se realizasse.

Ao meu professor orientador, Dr. Marinho Pilla Barcellos, por toda sua motivação e compartilhamento da sua experiência em pesquisa e redes de computadores.

Aos professores do PIPCA por toda oportunidade de aprendizado.

Ao grupo de pesquisa em redes do PIPCA, por tudo que aprendi durante as nossas reuniões e na convivência acadêmica, principalmente a André Detsch, cujos trabalhos anteriores serviram de base para esta pesquisa.

À minha esposa Margot, pelo apoio incondicional, e por ter me emprestado um pouco da sua força para que eu pudesse concluir este projeto.

À minha filha Brenda, por si só, o maior incentivo e por todo tempo que era seu e que me cedeu.

À minha irmã Isa e meu cunhado Pita, por terem sido um porto seguro na tempestade, me permitindo navegar até aqui, e ao meu sobrinho e afilhado Bruno pelo tempo que também me cedeu.

À minha sogra Scarlet, por todo apoio e incentivo.

Finalmente, gostaria de agradecer a CAPES, que forneceu o apoio financeiro necessário.

Sumário

Lista de Figuras

Lista de Abreviaturas

AIMD	Additive Increase and Multiplicative Decrease
ARQ	Automatic Repeat Request
CBR	Constant Bit Rate
CBQ	Class-based Queuing
DV	Distance Vector
GPL	General Public License
IP	Internet Protocol
FLID-DL	Fair Layered Increase/Decrease with Dynamic Layering
IETF	Internet Engineering Task Force
LPR	Linear Proporcional Response
LTRC	Loss Tolerant Rate Controller for Reliable Multicast
LTS	Layered Transmission Scheme
LVMR	Layered Video Multicast with Retransmission
NS	NS Network Simulator
NCA	Nominee-Based Congestion Avoidance
MLDA	Multicast Lost-delay Based Adaption Algorithm
MTCP	Multicast TCP
PeP	Periodic Polling
PePcc	Periodic Polling with Congestion Control
PGMCC	Pragmatic General Multicast Congestion
PLM	Packet-Pair Receiver-Driven Cumulative Layered Multicast Protocol
PRMP	Polling-based Reliable Multicast Protocol
RAINBOW	ReliAble multicast by INdividual Bandwidth adaptation using windOW

RED	Random Early Detection
RLA	Random Listening Algorithm
RLC	Receiver-Driven Layered Congestion Control
RLM	Receiver-Driven Layered Multicast
RP	Rendez-vous Point
RTO	Retransmission Timeout
RTT	Round Trip Time
TCP	Transmission Control Protocol
TEAR	TCP Emulation at Receivers
TFRP	TCP-friendly Transport Protocol
TRAM	Tree-based Reliable Multicast Protocol

TITLE: "EVALUATION OF THE PEPCC PROTOCOL FOR RELIABLE MULTICAST TRANSMISSION IN THE INTERNET"

Abstract

Applications as the World-Wide Web, electronic mail and file transfer are the main source of traffic in communication networks nowadays. The IP protocol, which is the fundamental technology of the Internet, does not support resource reservation. The fair division of the network capacity among the competing flows is achieved with the help of mechanisms that should be present in the protocols executed in the end hosts. These are known as end-to-end congestion control. The traffic in the Internet is predominantly composed of TCP flows, which is built in with congestion control. For the Internet to operate properly, new transport or application protocols have to include congestion control mechanisms that are 'friendly' to TCP. In other words, these protocols should not take more resources than their "fair share".

In the end of the 90s, IP multicast allowed new applications in the Internet, such as groupware applications, distributed databases, video-conference systems, etc. Many of these applications depend on multicast protocols for efficient communication, because it allows a single copy of the data to be sent.

However, unfortunately, the use of multicast protocols is not as common as expected. The main reason is the Internet providers reluctance in enabling multicast in their network routers (they already support it). Such reluctance steems, among other reasons, from the fact that there is no appropriate congestion control mechanisms available for multicast protocols. This is an active area of research, and the subject of this dissertation. The focus of the work lies in evaluating, through an extensive set of simulation experiments proposed in the literature, a multicast protocol based on window and polling (named PePcc), incremented with a congestion control mechanism. This protocol is used predominantly for reliable transfer of data, similarly to TCP. The dissertation analyzes the behavior of the PePcc protocol in multiple scenarios found in the Internet, from those typical to the extreme ones.

Keywords: congestion control, multicast, Internet.

Resumo

Aplicações como a World-Wide Web, correio eletrônico e transferência de arquivos são a principal fonte de tráfego em redes de comunicação entre computadores hoje em dia. A tecnologia IP, base da Internet, não dispõe de reservas de recursos, e a divisão justa da capacidade da rede entre os fluxos que competem se dá através de mecanismos que devem estar presentes nos protocolos executados nas estações fim (*hosts*). Tal é denominado controle de congestionamento fim a fim. O tráfego na Internet é predominantemente composto por fluxos TCP, que é dotado de controle de congestionamento. Para o bom funcionamento da Internet, novos protocolos de transporte ou de aplicação devem incluir mecanismos de controle de congestionamento que sejam "amigáveis" ao TCP, ou seja, que não ocupem mais recursos do que deveriam.

O surgimento de IP multicast viabilizou, no final nos anos 90, novas aplicações na Internet, como aplicações de trabalho em grupo, bancos de dados distribuídos, vídeo-conferência, etc. Muitas dessas aplicações dependem de protocolos multicast para comunicação eficiente na Internet, pois essa tecnologia permite que apenas uma cópia dos dados seja enviada.

Entretanto, infelizmente, o uso de protocolos multicast não é tão comum quanto poderia se imaginar. A principal razão para isso é a relutância dos provedores Internet em habilitar multicast nos equipamentos, pois esses já a suportam. Tal relutância advém do fato de não existirem mecanismos de controle de congestionamento adequados para protocolos multicast. Esta é uma área ativa de pesquisa, e assunto desta dissertação. O foco do trabalho reside em avaliar, através de um extenso conjunto de experimentos de simulação propostos na literatura, o PePcc, um protocolo multicast baseado em janela e *polling*, acrescido de um mecanismo de controle de congestionamento. Este tipo de protocolo é utilizado predominantemente para transferência confiável de dados, similarmente ao TCP. A dissertação analisa o comportamento do protocolo mediante situações tantos típicas como extremas, que podem ser encontradas na Internet.

Palavras-chave: controle de congestionamento, multicast, Internet.

Capítulo 1

Introdução

A maioria das aplicações em uso na Internet está baseada na *comunicação ponto-a-ponto* e no protocolo de transporte TCP (*Transmission Control Protocol*) ([49]). O funcionamento "saudável" da Internet depende da implementação apropriada de mecanismos de controle de congestionamento em cada uma das estações que se comunicam através dela. Novos mecanismos de controle de congestionamento precisam ser "amigáveis" ao esquema adotado no TCP e seus algoritmos para controle de congestionamento.

Aplicações para *comunicação multiponto*, tais como ferramentas de teleconferência e serviços de disseminação de informação, têm se tornado cada vez mais populares. A solução natural para aplicações multiponto seria utilizar uma variação do protocolo TCP como transporte. Entretanto, esta abordagem implica usar uma coleção de fluxos ponto-a-ponto (*multi-unicast*) e transmitir várias cópias da mesma informação, o que é inadmissível em termos de largura de banda e escalabilidade.

Essa limitação tem motivado o desenvolvimento de protocolos de transporte multicast confiáveis visando uma distribuição eficiente e escalável de dados a múltiplos pontos. Tais protocolos são projetados tendo IP multicast ([13]) como camada subjacente para com isso evitar o envio de dados repetidos sobre os mesmos segmento de rede. Adicionalmente, esses protocolos são ditos "escaláveis" (*scalable*) quando possuem algum esquema para evitar ou diminuir o risco de "implosão de reconhecimento" (*feedback implosion*).

Esses protocolos de multicast confiável são ditos de transporte, pois sua função é transportar dados entre um origem e seus destinos (ou seja, "fim-a-fim"), sem considerar aspectos de roteamento. Existem exemplos de protocolos multicast confiáveis e escaláveis propostos na literatura; a ênfase desses protocolos é o tratamento da implosão de reconhecimento. Para que os protocolos multicast sejam utilizados seguramente na Internet, é imperativo que eles incorporem mecanismos para tratar congestionamento. Portanto, mais recentemente, estão sendo investigados mecanismos para controle de congestionamento em multicast, particularmente em protocolos para transmissão confiável, ao mesmo tempo que novos mecanismos de controle de congestionamento têm sido propostos e avaliados para o TCP, como por exemplo TCP Vegas ([29]) e *notificação explícita de congestionamento* ([19]).

Esta dissertação avalia o protocolo **PePcc** - *Periodic Polling with Congestion Control*,

combinação do protocolo para transmissão multicast confiável PeP - *Periodic Polling* ([4]) com o mecanismo de controle de congestionamento inicialmente concebido para o mesmo ([14]).

Ela está organizada da seguinte forma. O Capítulo 2 apresenta uma revisão bibliográfica sobre os três principais assuntos objetos desse trabalho: controle de congestionamento (unicast), protocolos de transporte multicast confiáveis e escaláveis; e a fusão dos dois assuntos anteriores, o estado da arte em controle de congestionamento em protocolos multicast confiáveis e escaláveis. No Capítulo 3 é descrito o protocolo PePcc e o seu mecanismo de controle de congestionamento. O Capítulo 4 discute a metodologia de avaliação por simulação, incluindo ferramenta de simulação, procedimentos adotados, a configuração que foi empregada nos experimentos, premissas assumidas e métricas colhidas. O conjunto de experimentos realizados foi baseado em trabalhos na literatura. O Capítulo 5 apresenta e comenta sobre os resultados obtidos. Por fim, considerações finais, limitações e trabalhos futuros aparecem no Capítulo 6.

Capítulo 2

Controle de Congestionamento em Protocolos Multicast

Este capítulo oferece uma revisão bibliográfica sobre os assuntos a serem tratados nessa dissertação, ou seja, controle de congestionamento, protocolos de transporte multicast confiáveis e escaláveis, e por fim controle de congestionamento em protocolos multicast confiável e escalável.

2.1 Controle de Congestionamento

Com a evolução das tecnologias de rede, a maior causa de perda de pacotes na Internet se deve ao congestionamento que ocorre em roteadores sobrecarregados ([29]). A função básica de um roteador é encaminhar pacotes que são recebidos por enlaces de entrada e repassados a um enlace da saída, de acordo com tabelas de roteamento que são usualmente construídas através de algoritmos de roteamento distribuídos (no caso de multicast, exemplos são DVMRP ([24]) e PIM ([24])). Quando o somatório dos fluxos que chegam a um roteador e devem ser redirecionados a um determinado enlace de saída excede a capacidade do mesmo, a fila de saída associada a esse enlace no roteador tende a encher e, mais cedo ou mais tarde, causar o descarte de pacotes por estouro de *buffer*.

Controle de congestionamento é um desafio na Internet, devido ao limitado grau de observação e controle da rede (particularmente, nas "pontas"). A principal forma de controle de congestionamento possível na Internet é o controle fim-a-fim do tráfego do usuário na camada de transporte. Este controle deve ser exercido utilizando apenas a observação limitada da rede que pode ser feita localmente, baseado no seu próprio desempenho ([20]).

Perdas de pacotes são geralmente resultado da sobrecarga dos *buffers* dos roteadores quando a rede torna-se congestionada. Em protocolos com controle de erro baseado em ARQ (*Automatic Repeat reQuest*) ([22]), como é o caso da maioria dos protocolos atuais (incluindo-se aí o TCP), perdas são recuperadas através da retransmissão de pacotes. A retransmissão trata um sintoma do congestionamento da rede, mas não trata sua causa: um conjunto de remetentes tentando enviar dados a uma taxa muito alta. Para tratar a causa, são necessários mecanismos para diminuir a taxa de envio desses remetentes, de forma igualitária, quando ocorre congestionamento

na rede ([29]).

Controle de congestionamento está relacionado à alocação de recursos na rede de forma que a mesma possa operar com nível de desempenho aceitável quando a demanda exceder ou estiver próxima de superar os recursos da rede. Estes recursos incluem a largura de banda dos enlaces, tamanho dos *buffers* (memória) e a capacidade de processamento dos nodos intermediários. Sem mecanismos de controle o *throughput* pode ser reduzido consideravelmente quando a rede estiver sobrecarregada ([26]).

Em termos de Internet, controle de congestionamento está fortemente associado ao TCP e suas diferentes versões. A seguir, será abordado o TCP e os algoritmos de controle de congestionamento que fundamentam sua operação.

2.2 TCP - *Transport Control Protocol*

TCP é o protocolo de transporte ponto-a-ponto dominante na Internet[1]. TCP e seus algoritmos de controle de congestionamento têm sido instrumentos na prevenção de colapso e em manter as taxas de perda de pacote "baixas" na Internet. A estabilidade corrente da Internet depende de seu controle de congestionamento fim-a-fim, baseado em um algoritmo de aumento aditivo e diminuição multiplicativa (denominado AIMD, ou *Additive Increase and Multiplicative Decrease* ([18])).

Existem dois esquemas para controlar o envio, aplicáveis a controle de fluxo e controle de congestionamento: baseado em controle de taxa de envio (*rate-based*) e baseado em janela (*window-based*). O TCP trabalha com mecanismos baseados em janela: a taxa de envio é controlada em função de uma "janela de congestionamento" (*congestion window*). Essa "janela" é na realidade apenas um valor, como será discutido mais tarde, que é usado para limitar a extremidade direita da janela. Ou seja, efetivamente "encolhe" a janela e restringe a quantidade de dados que o remetente pode enviar. O remetente TCP usualmente transmite uma janela de pacotes por *round trip time* (RTT). Assim, TCP ajusta o tamanho da sua janela no remetente (através da janela de congestionamento e da quantidade de espaço reportada pelo receptor) para refletir as condições da rede da seguinte forma ([36]):

- cada vez que uma janela de pacotes é transportada com sucesso, o TCP remetente aumenta o tamanho da janela em 1 pacote;
- cada vez que o TCP detecta que a rede descartou um pacote, ele corta a janela pela metade.

E, no caso das versões de TCP que empregam retransmissão rápida (*fast retransmit*),

- o remetente pode detectar a perda de pacotes rapidamente utilizando retransmissão rápida, contanto que a janela seja maior do que 3 pacotes;

[1] apenas para constar, não existe protocolo de transporte multiponto dominante na Internet, e sim uma pletora de protocolos.

- quando retransmissão rápida falha, TCP volta para um temporizador de retransmissão[2] conservador de 1 segundo ou mais.

Os principais elementos de controle de congestionamento do TCP são considerados em detalhe a seguir.

2.2.1 Janela de Congestionamento

O mecanismo de controle de congestionamento do TCP mantém em cada lado da conexão uma variável chamada *cwnd* (de *congestion window*). Esta variável impõe uma limitação adicional de quanto tráfego que uma estação pode enviar em uma conexão, regulando o tempo no qual os pacotes de dados[3] são enviados para a rede ([29]).

TCP envia uma janela de pacotes antes de esperar por um ACK. Cada vez que TCP recebe um ACK, ele envia outro pacote de dados. Este procedimento tende a manter a janela de pacotes em movimento e a rede ocupada. Em geral, o tamanho correto da janela é o produto da largura de banda disponível e o atraso de propagação de 1/2 RTT ([36]).

2.2.2 *Slow-start*

No início de uma transmissão, os *buffers* do receptor se encontram vazios, e em função disso a janela anunciada pelo receptor indica que o remetente pode enviar uma quantidade de dados potencialmente grande (uma janela inteira de dados). O remetente ainda não possui informações sobre a rede, e a transmissão em rajada de uma janela inteira pode ocasionar um transbordo nos *buffers* em roteadores.

Slow-start (ou início lento) é um algoritmo utilizado para evitar que um remetente envie uma janela inteira antes de fazer uma investigação da capacidade disponível na rede. Portanto, ao invés de enviar uma janela inteira, o remetente envia apenas um pacote e espera pelo ACK correspondente; quando o ACK é recebido, o tamanho da janela é dobrado e por conseqüência dois pacotes são enviados. Durante o *slow-start*, o tamanho da janela inicia em 1 mas cresce exponencialmente.

A fase de *slow-start* acaba quando ou uma perda é detectada ou quando a janela atinge um valor limite (dito *threshold*). Nesse momento, o TCP remetente passa à fase de *congestion avoidance*. Resumindo, o algoritmo de *slow-start* do TCP faz o seguinte ([29]):

- inicialmente *cwnd* é ajustada para 1 pacote;
- se o ACK chega antes de expirar o temporizador, incrementa *cwnd* com 1 pacote e envia 2 pacotes;
- se retornarem os dois ACKs, incrementa *cwnd* com 2 pacotes e envia 4 pacotes;

[2] denominado RTO, ou *retransmission timeout,* esse valor indica o período em que o remetente deve esperar por um ACK antes de assumir que o pacote de dados ou seu ACK foram perdidos.

[3] na realidade, o termo mais correto, segundo a terminologia TCP, seria "segmento".

- continua neste crescimento exponencial até que *cwnd* exceda o *threshold*, quando então passa para a fase de *congestion avoidance*.

2.2.3 *Congestion Avoidance*

Quando TCP está em modo *congestion avoidance*, ele está procurando por um tamanho razoável de janela. A meta é incrementar a janela lentamente, até que ocorram perdas ou que o limite máximo de janela seja atingido. Note que a capacidade da rede varia com o tempo, e a largura de banda da rede pode aumentar. *Congestion avoidance* funciona da seguinte forma, assumindo que o valor de *cwnd* é maior do que o valor de *threshold* ([29]):

- a cada ACK recebido incrementa *cwnd* em $1/cwnd$;
- quando (ou caso) expirar o temporizador sem ter recebido um ACK (indica perda do pacote), *threshold* é ajustado para metade do valor da *cwnd* e *cwnd* é reinicializada para 1 pacote;
- começa novamente o processo de *slow-start* ([29]).

2.2.4 *Retransmissão Rápida* e *Recuperação Rápida*

O esquema de *slow-start* e *congestion avoidance* visto anteriormente pertence ao esquema de controle de congestionamento do TCP chamado versão Tahoe. Uma variante dele é o esquema da versão TCP Reno ([15]), que inclui os mecanismos de retransmissão rápida *(fast-retransmit)* e recuperação rápida *(fast-recovery)*.

Retransmissão rápida retransmite o pacote perdido antes que o temporizador do pacote expire: se o remetente recebe três ACKs para o mesmo pacote, ele considera isto como uma indicação de que o pacote seguinte ao pacote que teve três ACKs foi perdido (pois a cada vez que o receptor recebe um pacote fora de ordem, ele envia um ACK respectivo ao último pacote em ordem recebido com sucesso). Recuperação rápida suprime a fase de *slow-start* após um retransmissão rápida ([29]).

TCP tradicionalmente espera pelo menos 1 segundo antes de expirar o temporizador e retransmitir um pacote perdido. Isto faz com que a perda de pacotes tenha um grande impacto na eficiência. TCP utiliza retransmissão rápida para recuperar a perda de um pacote em um RTT ao invés de 1 segundo ([36]).

Retransmissão rápida e recuperação rápida funcionam da seguinte forma:

- quando um pacote é recebido fora de ordem, TCP imediatamente gera um ACK duplo;
- o propósito deste ACK duplo é fazer com que a outra ponta da conexão saiba que um pacote foi recebido fora de ordem, e lhe informar qual número de seqüência é esperado;
- como TCP não sabe se o ACK duplo é causado por um pacote perdido ou é apenas uma reordenação de pacotes, ele espera até que três ACKs duplicados sejam recebidos;

- é assumido que na reordenação de pacotes, haverá no máximo apenas um ou dois ACKs duplicados antes que o pacote reordenado seja processado;
- se três ou mais ACKs duplicados são recebidos, é uma forte indicação que um pacote tenha sido perdido;
- a retransmissão do que parece ser o pacote perdido é executada, sem esperar que o temporizador de retransmissão expire;
- após isto, no lugar de *slow-start*, é executado *congestion avoidance* (este é o algoritmo de recuperação rápida, [45]).

2.3 Protocolos de Transporte Multicast Confiável

Multicast é um paradigma de comunicação onde uma mensagem é enviada a um grupo de receptores identificados por um endereço único ([4]). Protocolos de transporte multicast confiáveis e escaláveis podem ser classificados em dois tipos principais ([40]): *orientados a remetente* e *orientados a receptor*. A seguir é apresentada a definição de cada um destes tipos.

2.3.1 Protocolos Multicast Confiáveis Orientados a Remetente

Um protocolo multicast confiável orientado a remetente requer que o remetente receba ACKs de todos receptores, antes de ele estar capacitado a liberar a memória para o dado associado com os ACKs. O remetente deve conhecer a constituição do conjunto de receptores, e este esquema sofre com o problema da implosão de reconhecimentos (*feedback implosion*). O emissor mantém os pacotes na memória até que cada nodo receptor tenha enviado um ACK informando que recebeu o dado. Após expirar o tempo que o remetente estipulou como limite para receber os ACKs, o remetente reenvia o pacote. Além de controle através da expiração do tempo, o remetente pode detectar perdas de pacotes através de NACKs. Um receptor pode, ao detectar ele mesmo uma provável perda de pacote, notificar o remetente sobre a perda através de um pacote de NACK. A principal limitação dos protocolos orientados a remetente não é o uso de ACKs, mas a necessidade do remetente de processar todos ACKs e conhecer o conjunto de receptores ([32]).

Os protocolos orientados a remetente apresentam as seguintes vantagens ([39]):

- o remetente pode controlar quem se une ao grupo;
- o remetente pode conhecer os membros do grupo;
- as soluções baseadas neste modelo são mais simples.

2.3.2 Protocolos Multicast Confiáveis Orientados a Receptor

O problema da implosão de ACKs é causado pelo grande volume de ACKS, levando à perda de pacotes e aumento de custo de rede e latência. Uma solução para isto é o uso de uma

abordagem orientada a receptor, de maior escalabilidade: o peso do processamento de ACKs é deslocado para os receptores, e os pacotes de NACKs são usados apenas sob demanda, como requisição de retransmissão. Eliminando ACKs obrigatórios para cada pacote, o custo de rede é reduzido assim como também o risco de implosão. Entretanto, estes protocolos não estão livres de implosão: quando as mesmas perdas ocorrem para muitos receptores, um grande número de retornos negativos pode causar uma implosão de NACKs ([2]).

Protocolos multicast confiáveis orientados a receptor colocam a responsabilidade de assegurar confiabilidade da entrega dos pacotes a cada receptor. O aspecto crítico destes protocolos é que ACKs não são usados. O receptor envia um NACK de volta para o remetente quando uma retransmissão é necessária, detectada por erro, um salto na seqüência de números usados ou pela expiração do tempo estipulado para receber o pacote. O remetente recebe NACKs dos receptores apenas quando pacotes são perdidos e não quando eles são entregues; por isto, o remetente está desabilitado a determinar quando ele pode descartar seguramente dados da memória. Não há um mecanismo explícito em protocolos orientados a receptor para o remetente descartar dados da memória. Entretanto, seus mecanismos de retransmissão são escaláveis e eficientes ([32]). Algumas características importantes de protocolos multicast confiáveis orientados a receptor são ([3]):

- retirar do remetente qualquer informação de estado sobre os receptores;
- substituir ACKS e temporizadores por NACKS como principal mecanismo de detecção de perdas no remetente;
- o remetente não conhece a composição do grupo destinatário; seu papel é enviar pacotes a determinado grupo e atender solicitações de retransmissão de dados que sejam recebidas;
- para assegurar confiabilidade a receptores, o remetente deve manter na sua memória cópias dos pacotes enviados, por tempo indeterminado. Isto porque ele não pode estabelecer um tempo de espera por NACKS, por desconhecer o RTT entre si mesmo e cada um dos receptores.

Neste tipo de protocolo, o remetente continua a transmitir novos pacotes de dados até que ele receba um NACK de um receptor. Quando isto ocorre, o remetente retransmite o pacote requisitado pelo receptor; é função do receptor checar por pacotes perdidos. Se ele decide que não recebeu um pacote particular, ele transmite um NACK para o remetente. De forma a resguardar-se contra a perda de um NACK ou a subseqüente retransmissão de um pacote, o receptor usa um temporizador de modo análogo aos utilizados pelos remetentes em protocolos orientados a remetente ([40]).

Como vantagem de protocolos orientados a receptor podem ser citados:

- não há um único ponto de processamento centralizado no remetente, que inclui, exclui e mantém a lista de todos receptores;
- pode haver grupos que são tão grandes que seria impraticável manter o controle de todos seus membros em apenas um lugar.

2.4 Controle de Congestionamento Multicast

A principal questão em controle de congestionamento multicast é desenvolver mecanismos para determinar a taxa de transmissão do remetente, uma vez que os receptores podem operar em velocidades diferentes e a rede pode ter congestionamentos em diferentes partes em diferentes tempos ([38]). A detecção de perda de pacote no remetente, segundo um dos mecanismos de controle de erro acima descritos, reflete diversas condições de congestionamento em várias partes da rede, e tem que ser levada em conta quando tomada a decisão de uma taxa de envio única.

Adicionalmente, conexões multicast não deveriam estar habilitadas a apoderar-se injustamente de uma grande parte da largura de banda, pois elas poderiam enfraquecer injustamente conexões unicast. Por outro lado, uma taxa de sessão multicast não deve ser estrangulada inteiramente caso sua parte na largura de banda seja drasticamente reduzida. Isto desencorajaria o desenvolvimento e uso da tecnologia multicast ([5]).

A *Internet Engineering Task Force* (IETF) requer que qualquer controle de congestionamento padrão, unicast ou multicast, seja amigável ao TCP ([12]). Ou seja, que ele compartilhe a largura de banda de forma **justa** (*fair*) com os fluxos enviados através do TCP ([11, 51]). Seus mecanismos de controle de congestionamento têm sido um fator crítico da robustez da Internet ([17]). Um fluxo que atue como TCP deve responder à indicação de congestionamento reduzindo drasticamente sua taxa de transmissão, aumentando-a lentamente durante o estado estável (princípio AIMD, conforme visto anteriormente). Este mecanismo propicia um compartilhamento justo de um enlace congestionado ([41]). Diferentemente de uma conexão utilizando TCP, um protocolo multicast tem que fazer ajustes em múltiplos caminhos da rede. Tentar obter esta justiça é um grande desafio no desenvolvimento do controle de congestionamento para multicast confiável. Existem duas maneiras propostas para atingir este objetivo ([12]): "emulador de TCP" ou "controle baseado em equação". Um emulador de TCP funciona da seguinte forma:

- um receptor representativo é selecionado do grupo multicast;
- este receptor e o remetente executam um algoritmo semelhante ao do TCP;
- os mesmos atuam como um par remetente/receptor unicast, enquanto que o restante do grupo apenas recebe passivamente;
- o principal objetivo é selecionar o representante correto, aquele que está utilizando os recursos mais congestionados da rede;
- se o ponto de congestionamento se desloca, é necessário eleger um novo representante.

A outra forma de tentar compartilhar a largura de banda de forma justa com TCP é utilizar um controle baseado na fórmula TCP ([12]):

- coleta informações sobre a taxa de perdas de pacotes e o RTT;
- quando estas informações são consideradas confiáveis, são inseridas na fórmula TCP;

- o *throughput* derivado é então utilizado como taxa para regular a transmissão;
- utiliza controle baseado em taxa (*rate-based*), em contraste ao controle baseado em janela utilizado pelo TCP.

Neste esquema, a questão é como medir a taxa de perdas de pacotes e o RTT. É proposto que a taxa de perda e o RTT corretos sejam de um dado "receptor representativo", que utilize os recursos mais congestionados, como no emulador de TCP. A questão passa a ser como manter o foco dinamicamente no representante correto.

Os esquemas de controle de congestionamento geralmente atuam sobre a taxa de envio de dados do remetente, como forma de adaptar a taxa de envio à capacidade disponível na rede. Os esquemas de controle de congestionamento multicast podem ser divididos em dois grandes grupos, em função da taxa de envio: "taxa-única" (*single-rate*) e multi-taxa (*multi-rate*). Os esquemas de taxa única funcionam da seguinte forma ([43]):

- todos receptores recebem a mesma taxa de dados e o remetente adapta-se ao receptor mais lento;
- a maioria dos protocolos multicast de taxa única confiáveis tentam implementar um serviço como TCP sobre multicast;
- existem limitações na presença de grupos grandes e heterogêneos:
  - um único receptor lento pode diminuir a taxa de todo grupo;
  - problema do "drop-to-zero", isto é, a estimativa de taxa de perda é muito maior do que a taxa de perda atual de cada receptor.

Já os esquemas multi-taxa possuem as seguintes características ([43]):

- são baseados na habilidade de gerar o mesmo dado em diferentes taxas sobre múltiplas *streams* no remetente ou como um processo de filtragem feito por elementos intermediários, como por exemplo, roteadores;
- os receptores tentam escutar uma ou mais *streams*, de acordo com sua capacidade;
- a vantagem é que receptores com diferentes velocidades podem ser servidos por taxas mais próximas de suas necessidades, não ficando presos a taxa do mais lento do grupo;
- esta flexibilidade é paga em termos de custos de código e alguma ineficiência no uso da largura de banda.

Segundo [9], controle de congestionamento multi-taxa é necessário para transmissões escaláveis para grandes audiências, por evitar o problema de estabelecer uma única taxa, ajustada de acordo com a demanda do receptor de menor taxa.

Um problema a ser enfrentado em esquemas de controle de congestionamento é o problema da "perda em múltiplos caminhos". Este problema surge porque **um mesmo pacote** transmitido pode ser perdido em um ou mais dos muitos caminhos de uma árvore multicast. Conseqüentemente, se a taxa de transmissão de um remetente multicast for regulada de acordo com a indicação de perdas dos receptores, a taxa pode ser reduzida excessivamente, conforme aumenta o número de perdas nos caminhos ([5]). O problema da perda em múltiplos caminhos reduz o compartilhamento de banda de sessões multicast, competindo com sessões unicast:

- um único pacote perdido pode afetar múltiplos receptores: como o remetente pode receber mais de uma indicação de perda por pacote, se ele reduzir sua taxa para cada indicação de perda, ele estaria supercompensando uma única perda;
- supondo que um remetente multicast reduza sua taxa em resposta à indicação de perda de todos seus receptores, mas reaja a não mais do que uma indicação de perda por pacote transmitido. Porém, um pacote transmitido pode ser perdido independentemente em um dos múltiplos caminhos da árvore. Como o número de caminhos aumenta, a probabilidade que o remetente receba pelo menos uma indicação de perda para cada pacote enviado, também aumenta.

Existem várias propostas para algoritmos de controle de congestionamento para transporte multicast. Os mesmos serão apresentados nas seções a seguir.

2.4.1 Protocolos Multicast Baseados em Taxa com Taxa Única

LTRC - *The Loss Tolerant Rate Controller for Reliable Multicast*

LTRC [35] é um mecanismo de controle de congestionamento multicast de taxa única baseado em taxa. Controla o envio de dados do remetente de acordo com o relatório de perdas enviadas pelo receptor. O esquema localiza as perdas de pacotes nos receptores e as comunica para o remetente. Estes relatórios são armazenados como histórico. O remetente toma suas decisões de mudança de taxa baseado nos novos relatórios de perdas e no histórico armazenado. Desta forma é possível perceber quando ocorre uma perda independente, quando não é necessária uma mudança drástica de taxa de envio, mas apenas um pequena mudança em curto prazo para manter o equilíbrio. O receptor envia relatórios de perdas periódicos para o remetente. Juntamente com este relatório, ele envia para o remetente o valor médio de perdas ocorridas no espaço de tempo entre dois relatórios. Para reportar as perdas para o remetente, cada receptor deve manter uma média de perdas. Sempre que um pacote de dados é recebido, a média de perdas é atualizada, dividindo o número de pacotes perdidos pelo número de pacotes esperados. Quando o receptor envia reconhecimentos negativos para o remetente, juntamente é reportada esta taxa média de perdas. A taxa de envio de dados do remetente sofrerá um pequeno ajuste no caso de uma perda, a taxa será ajustada num valor maior apenas na ocorrência de perdas mais persistentes.

TRAM - *The Tree-based Reliable Multicast Protocol*

O TRAM [12] é um protocolo multicast de taxa única baseado em taxa onde a taxa de envio do remetente é ajustada dinamicamente baseada no reconhecimento de congestionamento dos receptores. O remetente e os receptores de uma sessão multicast TRAM interagem dinamicamente formando uma árvore, composta pelo remetente na raiz e por grupos de reparo organizados de forma hierárquica. Os grupos de reparo são formados por um receptor que atua como líder do grupo e os demais receptores, membros do grupo, que são filiados a seu líder. Exceto o remetente, cada líder de grupo de reparo é membro de algum outro grupo de reparo. O líder do grupo recebe relatórios de perdas e de pacotes recebidos com sucesso pelos seus membros filiados.

Quando o líder de um grupo de reparo recebe uma mensagem enviada pelo remetente, ele a armazena. Quando ele recebe um relatório de perdas de um de seus membros ele verifica se possui este pacote armazenado, se sim, o transmite para o receptor do seu grupo que solicitou, se não, faz um relatório de perda para o seu líder de grupo, e assim sucessivamente até chegar ao próprio remetente. Cada líder de reparo é responsável por garantir que o dado seja recebido por todos os membros do seu grupo. Ele mantém o dado em memória até que todos seus membros tenham recebido o dado e confirmado com reconhecimentos positivos.

A árvore existente é dinamicamente atualizada, possibilitando um membro encontrar seu melhor líder de reparo. Membros e líderes de grupo de reparo estão continuamente se monitorando. Se algum membro não responder ele pode ser excluído do grupo. No caso do próprio líder não responder ele poderá ser substituído por um outro líder ativo no seu grupo.

Os membros da árvore reportam periodicamente estatísticas e relatório de congestionamento para seu líder de reparo. As estatísticas incluem informações que auxiliam na formação da árvore e os relatórios de congestionamento permitem ao remetente adaptar sua taxa de dados de acordo com as condições da rede.

Tear - *TCP Emulation at Receivers*

TEAR [42] é um esquema de controle de congestionamento multicast de taxa única baseado em taxa em que os receptores estimam sua própria taxa de recepção emulando TCP. Os receptores enviam periodicamente para o remetente, reconhecimento de suas taxas estimadas. Com base nestes relatórios, o remetente seleciona o receptor que esteja mais estrangulado e ajusta a sua taxa de envio de dados para a reportada por este receptor.

TEARS estima o *throughput* de uma conexão TCP apenas observando o processo de chegada de pacotes no receptor. Para isto ele mantém uma janela de congestionamento como TCP, mas localizada no receptor ao invés de estar localizada no remetente, e atualiza ela de acordo como o mesmo algoritmo TCP, porém baseado na chegada de pacotes de dados e não na chegada de reconhecimento. A sessão de transmissão é particionada em períodos de tempos consecutivos, chamadas rodadas *(rounds)*.

Em uma rodada podem chegar aproximadamente o número de pacotes que cabem em uma janela de congestionamento. Diferentemente de TCP, que identifica uma rodada quando o remetente recebe um reconhecimento do receptor, indicando a recepção de pacotes que estão na janela de congestionamento corrente. Em TEAR, a rodada é identificada quando o receptor re-

cebe pacotes de dados do remetente. Desta forma, a duração de uma rodada em TEAR depende do tempo entre as chegadas de pacotes pertencentes a uma mesma janela de congestionamento, que por sua vez está vinculada à taxa de transmissão do remetente. Em TCP, a janela de congestionamento é atualizada na recepção de um reconhecimento do receptor (indicando a recepção de pacotes). Portanto, em TCP uma rodada é o tempo entre o envio do pacote e a recepção do reconhecimento referente a este pacote, ou seja um RTT. Para lidar com esta diferença entre o valor que uma rodada TCP teria e o valor da rodada de TEAR, TEAR atribui um tempo de RTT fictício para cada rodada. Ao fim de cada rodada, o receptor armazena o tamanho corrente da janela de congestionamento e o tempo de RTT utilizado. A taxa de transmissão que uma conexãoTCP utilizaria poderia ser obtida dividindo a janela de congestionamento pelo RTT, mas isto provocaria as mesmas flutuações de taxa características de TCP. Para tornar os ajustes de taxa mais suaves, são utilizadas médias dos valores armazenados no histórico após cada rodada. Estas médias são utilizadas para estimar uma taxa amigável ao TCP.

2.4.2 Protocolos Multicast Baseados em Janela com Taxa Única

PRMP - *Polling-based Reliable Multicast Protocol*

O PRMP [1] é um protocolo de taxa única baseado em janela e *polling* onde o remetente detecta perdas através de reconhecimento enviado pelos receptores. O remetente mantém armazenada uma cópia do pacote de dados enviado , até obter reconhecimentos de todos receptores para o pacote. Quando o remetente detecta uma perda, ela é recuperada por retransmissão. O remetente mantém um conjunto de janelas de envio, uma para cada receptor.

Para evitar implosão de ACKs, não é permitido que cada receptor retorne um pacote de ACK positivo, ao invés disto, é utilizado um mecanismo de *polling*. O remetente planeja, conforme sua disponibilidade de processar o reconhecimento dos receptores num determinado período de tempo, que um subconjunto dos receptores envie reconhecimento. Os receptores que foram nomeados, enviam para o remetente ACKs positivos para os pacotes que tenham recebidos até então e ACKS negativos para os pacotes supostamente perdidos.

De forma a obter homogeneidade na taxa de chegada dos reconhecimentos, o mecanismo controla os tempos de chegada dos reconhecimentos com antecedência. Para fazer isto, o tempo de uma sessão é dividido em épocas de tamanho fixo. Cada época comporta receber uma determinada quantidade de reconhecimentos. O remetente estima o tempo de chegada da resposta utilizando o RTT entre ele e cada receptor.

O planejamento de um *poll* é feito de forma que uma requisições de *polling* seja enviada para um receptor e que a sua resposta seja prevista para ser recebida numa época que comporte a sua resposta, pois cada época possui um número limitado de resposta que podem ser esperadas. Se uma determinada época não comporta mais respostas, o tempo de envio da requisição de *polling* é atrasada.

PeP - *Periodic Polling*

O PeP representa uma simplificação do PRMP, visto acima. Como o PRMP, emprega janelas deslizantes para controle de erro, de fluxo e de congestionamento, e controla a quantidade de reconhecimentos através de *polls*. O PeP utiliza um intervalo entre dois *polls* consecutivos para limitar a sua transmissão e consequentemente a quantidade de ACKs. A transmissão de *polls* é baseada em um esquema periódico: a cada determinado tempo, o remetente verifica se é necessário enviar um *poll* para algum receptor. Este tempo entre cada verificação é passado por parâmetro. O protocolo é detalhado no Capítulo 3.

RLA - *Random Listening Algorithm*

RLA [49] é um algoritmo para controle de congestionamento multicast de taxa única baseado em janela que reduz a janela do remetente, após receber um sinal de congestionamento, com a probabilidade de 1/n, onde n é o número de receptores que estão reportando perdas freqüentes. Devido a isto, o remetente reduz sua janela, em média, a cada n sinais de congestionamento. No caso de todos receptores do grupo multicast, estarem experimentando a mesma média de congestionamento, o remetente age como se estivesse escutando apenas um receptor, o representante do grupo multicast. Com o objetivo de obter uma reação moderada aos sinais de congestionamento durante uma sessão, o esquema escuta os sinais de congestionamento de todos receptores e reage de forma aleatória a estes sinais.

Para detectar perdas, os receptores multicast utilizam reconhecimentos seletivos da mesma forma que receptores TCP. Uma perda pode ser detectada pelo remetente quando este identifica um número de seqüência de ACK descontínuo ou quando expira o temporizador associado a recepção de reconhecimento para um determinado pacote. De forma a evitar retransmitir um pacote devido a número de seqüência descontinuo, o remetente só considera um determinado pacote perdido quando for recebido um ACK para um pacote com um número de seqüência pelo menos três vezes maior que o dele.

No momento que uma perda é detectada, começa o período de congestionamento. Uma vez tendo detectado o congestionamento de um determinado receptor, é verificado se esta perda é rara para o receptor. Se for uma perda rara, o receptor não será considerado com problemas. Se esta perda não for uma perda rara, poderão ser tomadas duas ações distintas. Se já faz um tempo relativamente longo que a janela de congestionamento não é reduzida pela metade, então ela será reduzida pela metade neste caso. No caso de ter transcorrido um tempo relativamente curto desde a última vez que a janela de congestionamento foi reduzida pela metade, será gerado um número aleatório. Se este número for menor ou igual a um limite estipulado dinamicamente, a janela de congestionamento será reduzida pela metade, senão, a janela permanece com o mesmo tamanho. Este mecanismo tem o objetivo de proteger o sistema, amortecendo a aleatoriedade e impedindo que a janela de congestionamento seja incrementada durante um tempo muito longo.

LPR - *Linear Proporcional Response*

O LPR [6] é um esquema de controle de congestionamento multicast de taxa única baseado em janela que utiliza um filtro para indicações de perdas. O remetente identifica e responde a indicações de perdas de apenas um receptor, aquele que apresenta os maiores sinais de congestionamento do grupo multicast. O filtro localizado no remetente conhece a probabilidade de perda de cada receptor.

Quando o remetente recebe a indicação de perda de algum receptor, ele a passa através do filtro, que leva em conta que a probabilidade da perda seja proporcional a probabilidade de perda do receptor. O filtro de perdas recebe indicação de perdas de vários receptores, e as submete a um processo de filtragem. A indicação de perda que permanecer será então fornecida como valor de entrada para o algoritmo que calcula o ajuste da taxa de transmissão de dados do remetente.

Os filtros de indicação de perda auxiliam a evitar que uma sessão multicast reduza a sua taxa de resposta para cada indicação de perda que ela recebe, fazendo com que sua taxa seja totalmente estrangulada. O filtro, despreza indicações de perdas isoladas ou para um pacote que já tenha recebido a indicação anteriormente, evitando que a taxa seja diminuída mais de uma vez pelo mesmo motivo. Os filtros também auxiliam a alocar a largura de banda correta para uma sessão multicast, se concentrando nas indicações de perdas das rotas congestionadas da árvore multicast e desprezando as indicações de perdas das outras rotas da árvore.

Quando um filtro recebe uma indicação de perda de um determinado receptor, ele calcula para que a resposta para a indicação de perda seja proporcional a probabilidade de perda estimada pelo receptor. O filtro observa a indicação de perdas de muitos receptores, principalmente os que reportam um nível mais alto de congestionamento, isto auxilia o esquema a tomar medidas mais corretas em relação às mudanças de condições da rede.

MTCP - *Multicast TCP*

MTCP [41] é um protocolo de controle de congestionamento para multicast confiável de taxa única baseado em janela e em árvore lógica multi-nível, onde a raiz é o remetente e os outros nodos da árvore são os receptores. Quando os receptores recebem pacotes de dados, eles enviam reconhecimentos para seus pais na árvores. Alguns nodos internos da árvore são designados como agentes remetentes (SAs), sua responsabilidade é lidar com o reconhecimento gerado pelos seus filhos e retransmitir pacotes perdidos para seus filhos.

Após receber pacotes de dados que o remetente transmitiu para todos os receptores, os SAs armazenam estes pacotes e associam um temporizador para cada pacote. O remetente também associa um temporizador para cada pacote enviado. No caso do receptor ter recebido o pacote, ele envia um ACK para seu pai, caso contrário envia um NACK. No momento que um SA ou o remetente, recebe ACK de todos seus filhos, ele descarta o pacote que ele tinha armazenado. Se o temporizador mantido pelo remetente e pelos SAs expirar antes de ter recebido todos ACKs, ou se o remetente ou um SA tenha recebido um NACK, ele retransmite o pacote para o filho que solicitou.

Cada SA monitora o nível de congestionamento de seus filhos mantendo uma janela de congestionamento, utilizando os mecanismos de controle *slow-start* e *congestion avoidance* do

TCP-Vegas, baseado no reconhecimento dos seus filhos. Esta informação do nível de congestionamento dos seus filhos é incluído quando o SA envia um ACK para seu pai.

O remetente regula sua taxa de transmissão de dados baseado nestes resumos que lhe são reportados. O remetente obtém destes resumos a largura de banda disponível do enlace mais estrangulado, permitindo ao remetente regular sua taxa de acordo com esta indicação. Para evitar que o remetente diminua sua taxa cada vez que ocorra uma perda independente, a janela de congestionamento é reduzida apenas quando a perda é acompanhada de indicações de congestionamento, como temporizadores de retransmissão ou vários NACKS consecutivos duplicados.

NCA - *Nominee-Based Congestion Avoidance*

O NCA [28] é esquema de controle de congestionamento multicast de taxa única baseado em janela e em nomeação que utiliza um algoritmo de ajuste de taxa baseado em remetente que regula a taxa de transmissão de acordo com a indicação de perda de pacotes de um único receptor do grupo multicast, o nomeado.

Outro algoritmo é utilizado pelo remetente para selecionar o seu receptor nomeado. O nomeado é selecionado via reconhecimento do nível de congestionamento dos receptores. Cada receptor calcula seu nível de congestionamento baseado na observação de suas perdas de pacotes e no RTT entre ele e o remetente. Este valor calculado é reportado periodicamente para o remetente. O remetente escolhe o receptor que tenha o maior valor de perdas e RTT mais alto como o pior receptor do grupo multicast, o nomeado. O remetente ajusta sua taxa em resposta às perdas deste único receptor, ignorando o nível de perda de todos os outros receptores.

O esquema agrega os relatórios de congestionamento periódicos enviados pelos receptores, em nodos intermediários da árvore multicast chamados de servidores de reparação. Cada receptor envia seu relatório de perdas para o servidor de reparação acima mais próximo. Cada servidor de reparação seleciona o maior valor de congestionamento enviado pelos receptores e envia este valor para o servidor de reparação acima e assim sucessivamente até o remetente. Este mecanismo auxilia na prevenção da implosão de reconhecimento no remetente.

Quando o remetente identifica o pior receptor, o nomeado, ele envia uma mensagem para este receptor solicitando que o receptor envie reconhecimentos para cada pacote recebido. Se o remetente receber ACKs para três pacotes que foram transmitidos depois de determinado pacote ou expirar o temporizador atribuído a este pacote, o pacote é dado como perdido e o remetente executa o *fast recovery*. Indicações repetidas de perda para uma mesma janela são ignoradas.

PGMCC - *Pragmatic General Multicast Congestion*

O pgmcc [43] é um esquema de controle de congestionamento multicast baseado em taxa única e janela, que busca obter escalabilidade, estabilidade e resposta rápida às variações das condições da rede, utilizando um controle baseado em janela como TCP e em ACKs positivos, que executam entre o remetente e um representante de grupo, o reconhecedor (*acker*). O procedimento de seleção do reconhecedor busca ser veloz e gerar baixa sobrecarga, considerando que selecionar o

reconhecedor correto é uma tarefa complexa, pois o este pode estar constantemente trocando de posição.

Uma vez que o reconhecedor (o receptor mais lento do grupo multicast) tenha sido escolhido, o remetente transmite os dados de acordo com a capacidade do reconhecedor, desde que esta taxa não seja superior a que uma conexão TCP utilizaria nas mesmas condições.

Sempre que um receptor envia um NACK para o remetente devido à perda de pacotes ou pacotes que chegam fora de ordem, ele inclui no NACK um relatório reportando sua taxa de perdas. De acordo com estes relatórios, o remetente seleciona um representante do grupo, o reconhecedor, que será aquele receptor que tiver o pior *throughput*. Após o reconhecedor ter sido escolhido, um esquema de controle de congestionamento baseado em janela similar ao controle de congestionamento do TCP, roda entre o remetente e o reconhecedor. O reconhecedor passará então a enviar reconhecimentos positivos para cada pacote de dados recebido.

Para selecionar o reconhecedor, o remetente utiliza, além dos relatórios que os receptores enviam embutidos nos NACKs, as medidas de RTT entre o remetente e os receptores. A escolha de um único remetente enviando reconhecimentos positivos para cada pacote de dados recebidos e a utilização de um esquema de supressão de NACKs via aleatoriedade, beneficiam a escalabilidade do esquema, mas não desobrigam o remetente de ter um constante conhecimento da situação da rede e do estado dos receptores.

2.4.3 Protocolos Multicast Baseados em Taxa com Múltiplas Taxas

RLM - *Receiver-Driven Layered Multicast*

RLM [34] é um protocolo de taxa múltipla baseado em taxa orientado a receptor, onde o remetente não tem uma função ativa no protocolo, simplesmente transmite cada camada do seu sinal em um grupo multicast separado. Os receptores multicast procuram se adaptar a heterogeneidade e as variações da capacidade da rede. Cada receptor executa o mecanismo principal do protocolo, agregando ou abandonando grupos conforme a necessidade. Na ocorrência de congestionamento o receptor descarta uma camada, na ausência de congestionamento, agrega uma camada. Camadas são adicionadas até que ocorra congestionamento. Quando ocorre o congestionamento o receptor volta para um ponto de operação anterior ao ponto onde foi detectado gargalo.

O remetente determina se seu nível de assinatura está muito alto observando se ele causa congestionamento, que é detectado pela perda de pacotes. Para determinar se o nível de assinatura está muito baixo, não existe um indicativo semelhante. Para resolver isto, o esquema adiciona camadas em tempos pré-definidos. Se esta assinatura espontânea da próxima camada na hierarquia causa congestionamento, o receptor descarta esta camada. Por outro lado, se esta adesão de uma nova camada não causa congestionamento, o receptor a mantém.

A adição de camadas em tempos pré-definidos não deve ser feita de forma independente pelos receptores, pois o ato da adição provoca um congestionamento passageiro que geraria um ruído nas medições, fazendo que os experimentos interfiram uns nos outros. Para evitar isto, antes de adicionar uma camada, o receptor notifica todo grupo, identificando a camada experimental. Desta forma receptores aprendem da experiência de outros receptores. Na avaliação de cada receptor se o experimento teve sucesso ou não, são levadas em consideração as condições da rede

entre ele e o remetente.

TopoSense

TopoSense [25] é algoritmo de taxa múltipla baseado em taxa para controle de congestionamento que utiliza camadas de *streams* e pressupõe a existência de um agente controlador centralizado que conhece a estrutura de toda árvore. Os receptores são informados por este agente de quais camadas eles deveriam assinar. O objetivo deste esquema é explorar a utilização da topologia da árvore para ajustar o congestionamento na rede, em detrimento da escalabilidade. O esquema utiliza agentes controladores distribuídos em diferentes locais da árvore multicast. Cada agente controlador verifica regularmente a topologia da árvore no seu domínio e as camadas multicast que por ali trafegam. O algoritmo toma sua decisões baseado na topologia da sessão multicast e nas taxas de perda de pacotes.

O algoritmo utiliza regras de congestionamento para rotular cada nodo da árvore como congestionado ou não congestionado. Um nodo está congestionado se seu pai está congestionado ou se seus filhos tem uma taxa de perda superior a determinado limite. Após ter determinado quais os nodos estão congestionados e quais não, o algoritmo, baseado no histórico de adesões de camadas e do estado de congestionamento de cada nodo, planeja as adesões futuras de cada nodo. Um determinado receptor só poderá assinar novas camadas quando seu histórico demonstrar que ele não está congestionado por dois intervalos consecutivos.

Após isto, o agente controlador envia para cada receptor uma mensagem sugerindo quais camadas ele deve aderir. Se ao agregar uma nova camada ocorrer congestionamento, será atribuído um temporizador para o receptor. Enquanto não expirar este temporizador, o receptor estará impossibilitado de agregar novas camadas.

LVMR - *Layered Video Multicast with Retransmission*

O LVMR [33] é um sistema para distribuição de vídeo de taxa múltipla baseado em taxa que utiliza codificação em camadas e um mecanismo de controle de taxa hierárquico. O esquema separa *streams* de código de vídeo em duas ou mais camadas. A primeira camada é uma camada básica, seguida de uma ou mais camadas. É possível o receptor receber apenas a camada básica, que fornece um nível básico de qualidade de vídeo. Após assinar a camada básica, um receptor pode assinar as demais camadas que fornecem uma melhoria na qualidade de vídeo. Um receptor pode assinar uma, duas ou mais camadas, dependendo da sua capacidade e das condições da rede. No caso do receptor experimentar congestionamento ele descarta uma ou mais camadas de forma a reduzir o congestionamento.

O esquema divide o grupo multicast em vários domínios, em cada domínio um nodo é nomeado como agente intermediário. Os domínios são divididos em sub-redes, e em cada sub-rede um nodo é nomeado como agente sub-rede. A função de ambos agentes é agregar e repassar informação para seus filhos. Um agente intermediário no nível mais baixo passa a informação do seu domínio para o próximo agente intermediário do nível mais alto, e assim por diante até que atinja o remetente. Da mesma forma, existe um fluxo de informação do agente intermediário

mais alto para os agentes intermediários do próximo nível mais baixo, e assim por diante até que atinjam todos agentes sub-rede.

Na ocorrência de congestionamento, os receptores procuram por outros receptores que também estejam congestionados na mesma sub-rede e nivelam o número de camadas agregadas pelo número de camadas do receptor que estiver assinando menos camadas. Só após este nivelamento, o receptor com menor número de camadas agregadas começa a descartar camadas. Quando a maioria das sub-redes que se reportam a um agente intermediário estiverem congestionadas, ele obriga que os seus membros descartem camadas.

PLM - *Packet-Pair Receiver-Driven Cumulative Layered Multicast Protocol*

PLM [31] é um protocolo de taxa múltipla baseado em taxa e em camadas cumulativas utilizando um par de pacotes como meio de determinar a largura de banda disponível e poder então decidir o número de camadas que um receptor deve agregar. O remetente possui a função de enviar dados através de camadas e transmitir para cada camada pacotes em pares. Cada camada gerada pelo remetente tem o mesmo conteúdo, mas com qualidade diferente.

O sinal é codificado em uma camada básica e uma ou mais camadas otimizadas. Quanto mais camadas o receptor agregar, maior a qualidade dos sinais. A largura de banda disponível para o receptor é inferida através do par de pacotes enviados pelo remetente, medindo o espaço entre os pacotes do par e o tamanho dos pacotes. Este método, diferente de TCP, não é baseado na medida de perdas de pacotes.

Cada receptor adere apenas à camada básica e espera seu primeiro par de pacotes, se após um determinado período o receptor não receber nenhum pacote, é por que o receptor não tem capacidade de receber a camada básica, não podendo aderir à sessão. Caso contrário, o receptor continua adicionando camadas até atingir a largura de banda disponível. No caso de exceder a capacidade, descarta camadas. Se após um determinado período o receptor deixa de receber pacotes, ele deve ir descartando camadas até voltar a recebê-los ou tendo descartado todas camadas, deve abandonar a sessão. Após uma camada ter sido descartada, o receptor entra num período de espera, durante o qual nenhuma ação é tomada, para evitar reagir mais de uma vez para a mesma perda.

RLC - *Receiver-Driven Layered Congestion Control*

RLC [48] é um esquema de controle de congestionamento de taxa múltipla baseado em taxa e em camadas, que desloca todas decisões de controle de congestionamento para os receptores. Utiliza um esquema para transmissão de dados hierárquico em camadas, onde os receptores podem aderir a um ou mais grupos multicast para receber dados em uma taxa que corresponda aproximadamente à sua largura de banda para o remetente. Os receptores estimam a situação da rede através da medição da taxa de perdas de pacotes. Cada receptor agrega ou abandona camadas dependendo dos sinais de congestionamento recebidos, procurando emular o comportamento do TCP. Para um receptor poder agregar uma camada ele deve esperar que ocorra um ponto de sincronismo, como forma de manter os receptores em sincronia. As decisões são baseadas em um histórico acumulado entre cada ponto de sincronismo.

Periodicamente o remetente emite rajadas curtas de pacotes, aumentando a taxa de transmissão em todas as camadas, fazendo com que cada camada tenha a taxa da camada superior em condições normais. Se houverem perdas de pacotes, os receptores interpretam como um sinal, não para descartarem camadas, mas sim para não aumentar o nível da assinatura. Após uma perda o receptor diminui o nível de assinatura e não reage a outras perdas por um determinado tempo, considerando que o roteador esteja congestionado.

O receptor somente pode aumentar seu nível de assinatura nos pontos de sincronismo, quando não ocorrem perdas durante a rajada, mas pode descartar camadas a qualquer momento sempre que houverem perdas durante a transmissão normal.

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

FLID-DL [9] é um mecanismo de controle de congestionamento para multicast de taxa múltipla baseado em taxa orientado a receptor, no qual os receptores adicionam camadas para manter ou aumentar a sua taxa de recepção e abandonam camadas na presença de congestionamento. Os receptores aumentam ou diminuem suas taxas baseados nas condições de congestionamento de forma que a média da largura de banda consumida seja semelhante a um fluxo TCP.

Não há reconhecimento dos receptores para o remetente e receptores diferentes podem agregar diferentes números de camadas dependendo das diferentes condições de rede nas rotas entre os servidores e os receptores. O remetente envia pacotes em taxas diferentes nas diferentes camadas multicast, porém estas camadas são dinâmicas, isto é, alteram sua taxa de transmissão no decorrer do tempo.

O remetente, ao longo do tempo, diminui a taxa de envio de cada camada. Se o receptor não agregar nenhuma outra camada, sua taxa diminuirá rapidamente. Para os receptores manterem uma determinada taxa de recepção, eles devem periodicamente agregar novas camadas. Para os receptores aumentarem sua taxa de recepção eles devem agregar camadas além das necessárias para manter a taxa constante. Este mecanismo procura evitar que a lentidão das operações de abandono de camada afetem as repostas para congestionamento.

LTS - *Layered Transmission Scheme*

LTS [47] é um esquema de controle de congestionamento de taxa múltipla baseado em taxa e remetente no qual os receptores agregam camadas até atingir uma largura de banda que uma conexão TCP equivalente utilizaria entre o receptor e o remetente. O receptor adere ao grupo e assina a camada básica. A seguir o receptor mede ou estima os parâmetros necessários para calcular a largura de banda que uma conexão TCP utilizaria nas mesmas condições. Os parâmetros são tamanho máximo de pacote utilizado na conexão, o RTT e a taxa média de perda de pacotes.

Tendo feito este calculo, o receptor adere a um número de grupos até que atinja a largura de banda estimada para o TCP. Desta forma, a taxa na qual os dados fluem entre o remetente e qualquer um dos receptores é equivalente a que uma conexão TCP utilizaria.

A cada camada que o receptor assina, ele estima novamente os parâmetros e calcula a conexão equivalente, observando o impacto da adesão dos novos grupos no congestionamento da

rede. Se a taxa calculada for menor do que a taxa atual, o receptor descarta uma camada. No caso da taxa resultante ficar abaixo de uma taxa mínima que o remetente estipulou, o esquema envia uma mensagem para o receptor para que este abandone a sessão.

TFRP - *TCP-friendly Transport Protocol*

O TFRP [46] é um esquema de taxa múltipla baseado em taxa e controle de erro para transmissão multicast de vídeo, que aplica FEC (*forward error corretction*). FEC hierárquico utiliza *streams* redundantes que pertencem a grupos multicast diferentes. Quando um receptor assina mais grupos ele está adquirindo mais proteção para os pacotes que está recebendo. Cada camada de FEC é transmitida para um endereço multicast diferente, de forma que os receptores adaptem seus níveis de assinatura de acordo com suas necessidades, baseado nas estatísticas de recepções anteriores. Para diminuir o tráfego, as camadas FEC são fornecidas apenas para as camadas de dados mais importantes.

O remetente de vídeo envia pacotes de vídeo em várias camadas e constrói camadas adicionais de FEC. Os receptores que estejam experimentando perdas e possuem pouca tolerância ao atraso, assinam várias camadas de dados, conforme a largura de banda disponível. Já receptores que estejam experimentando perdas mas tem maior tolerância de atraso, podem assinar camadas FEC para poder reparar os pacotes perdidos a custo de tempo extra para fazer o reparo. Para reduzir a complexidade do receptor o esquema considera que, como a primeira camada de dados é a mais importante, apenas ela seja protegida por FEC.

MLDA - *Multicast Lost-delay Based Adaption Algorithm*

O MLDA [44] é um algoritmo de taxa múltipla baseado em taxa e transmissão de dados em camadas onde agentes multimídia ajustam sua taxa de transmissão de acordo com o estado de congestionamento da rede. O número de camadas transmitidas é determinado de forma dinâmica de acordo com o reconhecimento gerado pelos receptores. Os receptores estimam a taxa de perda de pacotes, o RTT e o tamanho máximo de pacote na rota entre eles e o remetente. Com estes valores calculam a largura de banda compatível com a que uma conexão TCP utilizaria sob as mesmas condições.

Com base nestas informações recebidas periodicamente pelos agentes multimídia, o remetente ajusta sua taxa de transmissão e o tamanho das diferentes camadas que estão sendo transmitidas, de acordo com a largura de banda disponível. Periodicamente o remetente faz um *poll* solicitando reconhecimento dos receptores, em intervalos regulares distribuídos uniformemente. O remetente envia para o receptor neste relatório o número de camadas que ele está transmitindo, a taxa de envio de cada camada e o seu endereço.

No relatório enviado pelo receptor para o remetente consta a largura de banda amigável ao TCP estimada por ele. O remetente utiliza os relatórios recebidos entre dois *polls* para fazer seus ajustes de taxa.

2.4.4 Protocolo Multicast Baseados em Janela com Múltiplas Taxas

Rainbow - *ReliAble multicast by INdividual Bandwidth adaptation using windOW*

O Rainbow [51] é um esquema de transporte multicast de taxa múltipla baseado em janela onde cada receptor mantém sua própria janela de congestionamento, executando *slow-start* e *congestion avoidance* para obter equivalência com TCP. O receptor aumenta a janela quando todos pacotes são recebidos e diminui pela metade quando ocorre uma perda, procurando um comportamento semelhante como se houvesse uma conexão TCP entre o remetente e o receptor.

Os receptores requerem individualmente a transmissão de cada pacote de dados, que são marcadas com um rótulo que indica a sua posição na janela de congestionamento. Roteadores intermediários armazenam informações sobre as requisições recebidas, e agregam múltiplas requisições com o mesmo rótulo vindas de receptores diferentes. O roteador que estiver mais próximo do remetente entrega a requisição para ele. O remetente responde a cada requisição, que é transmitida em direção inversa a das requisições, fazendo com que o roteador exclua a informação sobre a requisição quando o pacote é transmitido para o receptor.

Capítulo 3

Protocolo PeP e Mecanismo de Controle de Congestionamento

Este capítulo descreve o protocolo para transmissão multicast confiável PeP - *Periodic Polling*, proposto originalmente em [4], e explora o mecanismo de controle de congestionamento inicialmente concebido para o mesmo em ([14]).

3.1 Protocolos Multicast Confiáveis baseados em *Polling*

Protocolos multicast confiáveis em nível de transporte, conforme visto na Seção 2.3 visam à transmissão de dados de uma origem para um grupo de receptores, acrescentando algum grau de confiabilidade em relação ao que é oferecido pelo IP multicast subjacente. Os protocolos se encontram divididos como visto na Seção 2.4 . Neste trabalho, analisa-se o protocolo descrito em [4], protocolo de taxa única, baseado em janela, cujo objetivo é a transmissão confiável de dados.

A técnica de *polling* em protocolos multicast confiáveis tem sido usada de forma a aumentar a escalabilidade de tais protocolos ([2, 23]), evitando o problema da implosão de reconhecimento. Basicamente, a técnica consiste na gerência por parte do remetente, da taxa de reconhecimento gerada pelos receptores. O remetente mantém o controle do andamento da transmissão, o que permite uma maior eficiência na entrega dos dados.

O protocolo avaliado, segundo a descrição em [4], inclui controle de erro e de congestionamento, mas não controle de fluxo: presume-se que os receptores consomem pacotes imediatamente após seu recebimento. As Seções 3.1.1 e 3.1.2 apresentam o funcionamento do mecanismo de controle de erro, enquanto a Seção 3.1.3 apresenta as características do protocolo.

3.1.1 Controle de erro para protocolos multicast baseados em *polling*

O protocolo de *polling* utilizado em [4] segue o estilo de detecção e recuperação de perdas de pacotes baseado em janela deslizante e temporizadores. O remetente envia pacotes de dados via multicast, e guarda o estado de cada receptor em uma *janela de transmissão*. Cada receptor

mantém o estado dos pacotes já recebidos em uma *janela de recepção*. A janela de recepção avança conforme os pacotes são recebidos em ordem. Uma janela de transmissão avança quando o pacote mais à esquerda teve seu recebimento confirmado (através de um pacote de reconhecimento) pelo receptor correspondente. Além das N janelas, o remetente computa uma *janela de transmissão global*, que consolida informações sobre todos os receptores. A janela de transmissão global está atrelada à janela de transmissão mais atrasada, de forma que seu avanço só ocorre quando o recebimento do pacote mais à esquerda tiver sido confirmado por todos os receptores. O remetente necessita esperar por confirmações positivas de todos os receptores em função da semântica do serviço prestado pelo protocolo à aplicação; a aplicação pode, em teoria, configurar o protocolo de forma a remover do grupo um receptor muito atrasado.

Conforme indicado acima, para atualizar uma janela de transmissão, o remetente necessita obter reconhecimento do receptor correspondente. Para tal, ele transmite um *poll* (pacote POLL) que requisita uma resposta (pacote RESPONSE) do receptor. POLLs podem ser explícitos em pacotes de controle ou embutidos em pacotes de dados. Em função de um ou mais POLL*s*, cada receptor vai, mais cedo ou mais tarde, enviar um RESPONSE ao remetente. Ao recebê-lo, o remetente atualiza o estado interno relativo ao receptor em questão (modificando a janela de transmissão de acordo com ACKs e NACKs na resposta, conforme descrito posteriormente).

A atualização do estado interno do remetente vai ser influenciada por dois fatores principais: a **freqüência** na qual as respostas são recebidas de um determinado receptor, e a **quantidade** de informações que cada resposta contém. A freqüência de solicitação de reconhecimento é dada pelo protocolo em questão. Em um extremo, uma resposta pode ser solicitada apenas após todos os pacotes terem sido enviados; no outro extremo, uma resposta será solicitada para cada pacote de dados enviado. A quantidade de informação contida em uma resposta, da mesma forma, pode variar consideravelmente, referenciando de um pacote a uma janela inteira.

A relação acima representa um *perde-ganha*: protocolos de *polling* reduzem a quantidade de pacotes de reconhecimento para evitar implosão, mas em contrapartida, aumentam o montante de informação por pacote, demandando mais processamento. Entretanto, sabe-se empiricamente que em geral receber e processar dois pacotes requer tipicamente mais tempo do que receber e processar um pacote maior. Implosões podem ser evitadas a partir do momento em que o aumento do processamento por pacote é relativamente pequeno frente a grande redução no volume de pacotes de respostas gerados.

3.1.2 Sistema de janelas e detecção de perda

A seguir, descreve-se em maior detalhe o mecanismo de janela implementado pelo protocolo de *polling* considerado. O remetente mantém uma janela de transmissão (sw_i) para cada receptor (R_i), que por sua vez mantém o estado dos pacotes de dados recebidos em uma janela de recepção (rw_i). Desta forma, cada janela sw_i representa uma janela rw_i correspondente, sendo a primeira atualizada de acordo com informações recebidas de R_i através de respostas (pacotes RESPONSE). Uma resposta contém uma cópia de rw_i no momento em que a ela foi enviada. Janelas tem tamanho ws pacotes, e são compostas de:

- $w[1..dp]$: vetor de bits que representa o estado de todos os pacotes entre 1 e dp (número

total de pacotes) – na prática, possui ws entradas;

- le: número de seqüência do pacote mais à esquerda da janela, ou seja, o primeiro bit 0;
- re: número de seqüência do pacote mais à direita da janela, dado por $le + ws - 1$;
- hr: maior número de seqüência de pacote registrado em w até o momento.

Quando um RESPONSE é recebido pelo remetente, sw_i é atualizada em função da cópia de rw_i embutida. Este processo permite ao remetente detectar NACKS em sw_i. Uma entrada $w[seq]$ é interpretada como ACK, NACK ou estado indefinido de acordo com as seguintes regras:

- ACK seq: $seq < le \vee w[seq]$
- NACK seq: $seq \leq hr \wedge \sim w[seq]$
- estado indefinido: $seq > hr$

Além da detecção da perda de pacotes de dados descrita acima, é necessário que o remetente trate perdas de pacotes RESPONSE e POLL. O remetente mantém uma tabela de RTTs atualizada dinamicamente, de onde pode ser obtida uma estimativa do tempo que expira o temporizador de retransmissão para cada receptor (denominada rto_i, de *retransmission timeout*). A forma de fazer estas estimativas foi baseada no TCP. O valor rto_{max} é definido como o maior timeout de retransmissão no conjunto de rto_i's.

3.1.3 PeP - *Periodic Polling*

A idéia básica do protocolo é que exista um intervalo de transmissão entre POLLs de maneira a evitar a implosão de reconhecimentos. O envio de POLLs, como indicado pelo nome, está baseado em um esquema periódico: a cada tempo T, o remetente verifica se é necessário enviar um POLL a algum receptor. Caso positivo, um POLL é enviado para o receptor. POLLS podem ser enviados também junto a pacotes de dados (*piggybacked*).

O protocolo possui duas partes independentes: uma que envia pacotes de dados de acordo com o que é permitido pela janela (ou seja, $seq \leq re$), e outra que envia POLLs periodicamente.

Ao enviar um POLL, o remetente (re-)programa um temporizador para limitar a espera de uma resposta gerada por **esse** POLL (apenas um temporizador por receptor). O remetente controla a necessidade de enviar POLLs a receptores através de um vetor V, cuja i-ésima entrada indica (caso marcada) se um POLL deve ser enviado ao receptor R_i. Uma entrada $V[i]$ é marcada sempre que for (re-)transmitido um pacote de dados a R_i, ou expirar o temporizador relativo a R_i. Para enviar POLLs, o remetente percorre V de maneira circular procurando por uma entrada marcada. Caso encontrada (p.ex., i), o remetente envia um POLL ao receptor R_i e desmarca $V[i]$.

Quando o temporizador expira, a posição em V referente ao receptor cujo temporizador expirou é marcada, de forma que seja, posteriormente, enviado um POLL para o receptor. Devido à existência de apenas um temporizador por receptor, o mecanismo de controle de respostas pendentes não é preciso. Para remover essa imprecisão, seria necessário manter um temporizador

por resposta pendente, o que seria demasiado custoso em termos de processamento e estado no remetente, além de tornar o protocolo mais complexo.

A perda de pacotes é detectada através de NACKs identificados na janela recebida em respostas. Perdas são recuperadas através de retransmissões com multicast. Uma vez encontrado um NACK com seqüência *seq*, o processo de retransmissão é iniciado de maneira a enviar o pacote de dados *seq*. Esse processo consiste na criação de um temporizador associado à *seq* e programado para expirar em rto_{max}; o que ocorre apenas caso ainda não exista em andamento um temporizador associado à *seq*. Tal temporizador é necessário para suprimir retransmissões desnecessárias; ele permite a chegada de mais respostas referentes à *seq*.

O tempo T é um parâmetro do protocolo que é configurado de forma a resultar em uma taxa de respostas adequada. Um tempo excessivamente curto levará a implosões; por outro lado, um tempo muito longo poderá levar ao "sufocamento" do remetente: travamento da janela de transmissão devido à ausência de reconhecimento.

3.2 Mecanismo de Controle de Congestionamento

A presente dissertação avalia o desempenho do PePcc, protocolo multicast confiável PeP acrescido de mecanismo de controle de congestionamento. Esta seção resume o mecanismo de controle de congestionamento adotado (apresentado em [14]), proposto considerando os modelos de protocolos multicast desenvolvidos em [4].

Nos protocolos com *polling*, uma vez que o reconhecimento por parte dos receptores contém o estado da janela de recepção, uma forma direta de detectar sinais de congestionamento é através da identificação, por parte do remetente, de NACKs nestas respostas, como em ([1]). São requisitos básicos deste mecanismo ([14]):

- protocolo deve apresentar um comportamento amigável ao TCP;
- não apresentar o problema de *loss path multiplicity* ([5]);
- regular o fluxo de transmissão utilizando parâmetros que não levem a uma taxa demasiadamente baixa, como apontado em ([21]) .

A seguir, são apresentadas as etapas do mecanismo: detecção de congestionamento, ajuste da janela de congestionamento e obtenção da taxa de transmissão.

3.2.1 Detecção de Congestionamento

O processo de detecção de congestionamento é realizado pelos receptores. É considerado sinal de congestionamento a detecção de uma perda de pacote de dados. A perda de um pacote de dados é detectada através de um NACK na janela de recepção (Seção 3.1.2). Sempre que um pacote é recebido o receptor verifica, através do número de seqüência do pacote e do estado da janela de recepção, se a atualização da janela criou uma nova lacuna entre pacotes recebidos

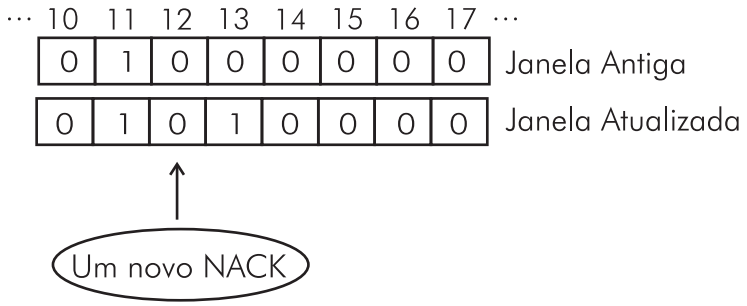


FIGURA 3.1 – Exemplo de detecção de um novo NACK após recebimento do pacote com número de seqüência 13

(Figura 3.1). Se este for o caso, considera-se que o evento da recepção do pacote indicou ao receptor que existe um provável congestionamento na rede.

Cada receptor armazena o tempo de detecção do último sinal de congestionamento . Esta informação é repassada ao transmissor a cada resposta enviada, através de um campo denominado `lastCongestionTime`, para que este possa ajustar o tamanho da janela de congestionamento.

3.2.2 Ajuste da Janela de Congestionamento

Assim como no TCP ou em qualquer mecanismo de controle de congestionamento baseado em janela, o ajuste da taxa de transmissão se dá através da variação do tamanho da janela de congestionamento. No esquema verificado, este ajuste é realizado de forma independente para cada receptor do grupo. Isso se dá através da adição de um atributo `cwnd_` (de *congestion window*) em cada janela de transmissão presente no transmissor. Assim como no TCP, este valor de janela de congestionamento serve para limitar o número de pacotes que estão transitando pela rede.

A variação deste valor é realizada também de forma similar ao TCP, estabelecendo dois períodos diferentes quando do aumento da janela, equivalentes ao *slow-start* e *congestion avoidance*. A transição de um período para outro, bem como o ajuste do limite (*threshold*) envolvido, é realizado da mesma maneira que o TCP (Seção 2.2), com a diferença de que *slow-start* é adotado apenas no início da transmissão.

O aumento da janela de congestionamento referente a um receptor é realizado quando o pacote que tem seu recebimento confirmado, ou seja, sempre que uma resposta traz informações de ACKs referentes a pacotes com recebimento até então não confirmado pelo receptor (Figura 3.2). Quando isso acontece, a janela de congestionamento (`cwnd_`) tem seu tamanho aumentado *de forma proporcional ao número de novas confirmações.* Este ajuste pode ser realizado de duas maneiras:

- quando em *slow-start*, cada nova confirmação de recebimento de pacote de dados acarreta no aumento de um *slot* na janela de congestionamento, ou seja, `cwnd_ = cwnd_ + 1;`

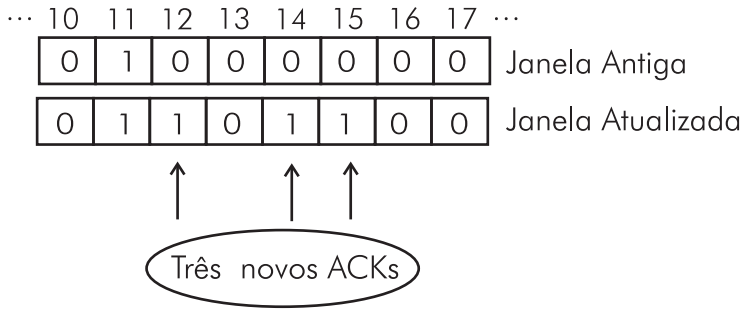


FIGURA 3.2 – Exemplo da ocorrência de novos ACKs após recebimento de reconhecimento

- quando em *congestion avoidance*, cada nova confirmação provoca o aumento correspondente a uma fração de *slot* inversamente proporcional ao tamanho atual da janela de congestionamento, ou seja, `cwnd_ = cwnd_ + 1/cwnd_`.

Já a redução do tamanho da janela de congestionamento é realizada de acordo com o valor de `lastCongestionTime_` nas respostas recebidas. Em um primeiro momento, a janela de congestionamento é reduzida quando o `lastCongestionTime_` recebido do receptor assume um valor válido, ou seja o primeiro sinal de congestionamento foi detectado. A partir daí, a janela só é reduzida quando um valor de `lastCongestionTime_`recebido é maior do que o último valor de `lastCongestionTime_`que levou a um decréscimo na janela acrescido da estimativa de RTT até o receptor em questão. Ou seja, só são considerados os sinais de congestionamento detectados no receptor quando estes estão espaçados por pelo menos um RTT. Este intervalo entre sinais de congestionamento considerados é exigido para que sinais de um congestionamento já tratado (através da redução da janela de congestionamento) levem a uma **nova** redução da taxa de transmissão. Cada sinal de congestionamento considerado leva a uma redução pela metade da janela de congestionamento, ou seja, `cwnd = cwnd/2`.

3.2.3 Obtenção da Taxa de Transmissão

A taxa de transmissão de protocolos com algoritmos de controle de congestionamento baseados em janela se dá através da limitação do número de pacotes enviados e ainda sem recebimento confirmado. No caso do mecanismo aqui avaliado, existe um tamanho de janela de congestionamento para cada receptor. A taxa de transmissão de um protocolo de taxa única deve obedecer aos requisitos de amigabilidade ao TCP para o receptor gargalo. Entretanto, não se pode simplesmente utilizar o mínimo dentre os tamanhos de janela de congestionamento e aplicá-lo sobre a janela de transmissão global, sob pena de reduzir demasiadamente a taxa no caso de o receptor com menor tamanho de janela não corresponder ao receptor com maior RTT.

Como forma de evitar este problema sem comprometer a amigabilidade ao TCP, o mecanismo estudado calcula, para cada receptor, qual o pacote de maior seqüência que pode ser transmitido considerando o estado da janela de transmissão e o tamanho da janela de conges-

tionamento individual (este valor é denominado `highestTransmittable`). A partir dos valores computados para todos os receptores, é obtido o valor mínimo (mínimo `highestTransmittable`), que limita a transmissão de novos pacotes. Desta forma, mesmo no caso de haver dois receptores, um com altos valores para o RTT e o tamanho da janela de congestionamento grande (R_1) e outro com valores reduzidos de RTT e tamanho da janela (R_2), o comportamento do protocolo não fica prejudicado, pois à medida em que as confirmações de recebimento de R_2 chegam, a transmissão pode avançar mesmo sem as confirmações de recebimento vindas de R_1, pois este possui um valor de `cwnd_` maior.

3.3 Características

O mecanismo tem três características essenciais, conforme a seguir. Primeiro, é amigável ao TCP. O algoritmo imita o comportamento do mecanismo de controle de congestionamento TCP, ajustando os valores de janela de congestionamento para cada receptor utilizando as mesmas fórmulas empregadas no mesmo. Como a taxa de transmissão é regida pelo valor mínimo de `highestTransmittable` (conforme definição apresentada na seção anterior), os requisitos de amigabilidade ao TCP para uma sessão multicast ficam garantidos.

Segundo, o mecanismo não apresenta o problema de *loss path multiplicity* ([5]). Uma vez que existe uma janela de congestionamento para cada receptor, perdas independentes de diferentes receptores não participam do cômputo de um valor comum, eliminando a possibilidade de haver esse problema.

Terceiro, o mecanismo regula o fluxo de transmissão utilizando parâmetros que não levem a uma taxa demasiadamente baixa, como apontado em ([21]). O problema é evitado utilizando-se o valor mínimo dentre os valores de `highestTransmittable` para cada receptor (e não diretamente o mínimo entre as janelas de congestionamento).

3.4 Limitações

Protocolos multicast, particularmente para transmissão confiável de dados, tipicamente buscam reduzir o montante de reconhecimento gerado pelos receptores como forma de evitar problemas de implosão e permitindo uma maior escalabilidade. Como conseqüência, o espaçamento entre respostas vindas de cada receptor pode aumentar muito se comparado ao de uma transmissão TCP, onde cada pacote enviado gera uma resposta por parte do receptor. Este aspecto pode influenciar negativamente o comportamento do mecanismo de controle de congestionamento multicast empregado, posto que sua *responsividade* é reduzida, ou seja, o tempo que o protocolo leva para detectar e tratar uma situação de congestionamento aumenta.

Uma vez que protocolos de *polling* também controlam o montante de reconhecimento oriundo dos receptores, o problema apontado acima também pode ocorrer no esquema de controle de congestionamento aqui estudado. A partir do fato de que a taxa de transmissão de um protocolo de taxa única deve ser adaptada pelo receptor com as piores condições de rede, recai sobre o reconhecimento recebido dos receptores mais lentos o ajuste adequado da taxa. A partir destas

considerações, uma forma de amenizar o problema da responsividade é estabelecer uma maior freqüência de reconhecimento por parte dos receptores gargalo, como acontece no PGMCC ([43]). Esta solução pode ser adaptada para o modelo de *polling* aqui utilizado: a partir do estado das janelas de transmissão e do RTT dos receptores (informações já disponíveis no remetente), pode-se determinar os receptores "mais lentos" e, ajustando o mecanismo de requisição de reconhecimento, reduzir o espaçamento entre respostas enviadas por estes receptores.

Capítulo 4

Metodologia de Avaliação

Este capítulo descreve a metodologia empregada na avaliação do mecanismo de controle de congestionamento para protocolos multicast confiáveis, alvo de estudo desta dissertação, tomando como base para avaliação o protocolo PeP (visto na Seção 3.1). Conforme já mencionado, o protocolo resultante é denominado PePcc. A metodologia está baseada na execução de um conjunto de experimentos de simulação, configurados conforme proposto em [10].

O capítulo discute, na Seção 4.1, questões metodológicas importantes no processo de simulação em redes e como elas foram tratadas; a Seção 4.2 oferece uma rápida revisão do VINT ns, ferramenta usada na condução dos experimentos. A Seção 4.3 descreve os experimentos realizados e a implementação dos mesmos no ns.

4.1 Considerações Metodológicas

Simulação é uma prática bastante adotada na avaliação de protocolos em redes de computadores. Simulação é flexível, pois permite que o grau de abstração do modelo seja ajustado de acordo com o problema tratado.

Entretanto, simulação é quase uma arte, é muito fácil desenvolver um experimento de simulação que produza resultados incorretos, incompatíveis com a realidade. De forma genérica, existem várias causas para que ocorram erros em simulações ([27]), sendo as mesmas discutidas a seguir:

- **nível de detalhe inapropriado**: um modelo com pequeno grau de detalhe tende a produzir um resultado pobre; entretanto, à medida que aumenta o número de detalhes, aumenta a necessidade de conhecimento dos parâmetros de entrada, nem sempre conhecidos ou genéricos;
- **linguagem de programação imprópria**: linguagens específicas para simulação requerem menor tempo de desenvolvimento do modelo e facilitam a obtenção de dados de saída; por outro lado, linguagens de propósito geral são mais portáveis e permitem escrever o código de forma a obter uma melhora nos tempos de execução;

- **modelos não validados**: modelos de simulação costumam ser programas grandes, sujeitos a vários *bugs* e erros de programação;
- **modelos inválidos**: mesmo não havendo erros de programação, o resultado pode ser inapropriado devido a considerações incorretas sobre o sistema;
- **erro em lidar com as condições iniciais**: a parte inicial de uma simulação geralmente não é representativa do sistema num estado estabilizado, e portanto a parte inicial deve ser descartada;
- **simulações muito curtas**: neste caso, as condições iniciais acabam por ter um peso muito grande, deturpando o resultado do experimento;
- **gerador de números aleatórios deficiente**: é mais seguro utilizar uma solução consagrada do que desenvolver uma própria;
- **seleção imprópria de sementes**: a semente para a geração de números aleatórios deve ser cuidadosamente escolhida para manter a independência entre os números gerados.

Portanto, para que um experimento sobre uma estratégia de controle de congestionamento multicast gere resultados compatíveis com a realidade, parâmetros de simulação devem ser cuidadosamente ajustados. Os mesmos são enumerados a seguir, seguindo a proposta apresentada em [10]. Inicialmente são tratados os princípios que se aplicam a estudos de controle de congestionamento em geral, para então continuar com princípios específicos para multicast.

4.1.1 Tamanho de fila correto

Controle de congestionamento é diretamente afetado pela capacidade da rede, ou seja, dos nós entre uma origem e seu(s) destino(s). O tamanho mínimo de fila para qual controle de congestionamento se comporte de acordo com o padrão de "serra"[1] esperado é aquele no qual o roteador gargalo tem um tamanho de fila igual ao produto entre largura de banda e atraso da conexão[2]. O produto nesse caso resulta em uma quantidade medida em bits, mas o tamanho das filas de pacotes das estações e roteadores são especificadas em número de pacotes, independente do tamanho de cada pacote transmitido. Como pode ser visto na Seção 4.2, essa é uma limitação do simulador ns. Dependendo, o número de bits pode ser tão pequeno que resulte mesmo em menos de um pacote. Para roteadores *drop-tail*, é recomendado avaliar neste ajuste, e com um tamanho de fila quatro vezes maior.

4.1.2 Tipo de TCP e ajustes

Existem diversas versões de TCP, sendo as mesmas diferenciadas pelos seus mecanismos de controle de erro e de congestionamento (vide Seção 2.2). De acordo com [10], em simulações de

[1] *sawtooth*.

[2] esse produto se refere à conexão, e não ao *link* gargalo. Ou seja, que engloba **todos** os roteadores entre o origem e o destino.

controle de congestionamento na Internet se deve empregar ou TCP SACK ([15]), pela robustez em relação a múltiplas perdas por janela, ou TCP Reno, por estar se tornando a versão mais popular, respectivamente. Para os experimentos aqui relatados, adotou-se o TCP Reno.

O TCP realiza o processamento de eventos de temporização de maneira periódica, o que é ditado pelo seu relógio. Portanto, a granularidade do relógio do TCP deve ser apropriada. Nos experimentos realizados, assumiu-se os valores padrão presentes na implementação de TCP do ns.

A janela de recepção TCP deve ser grande suficiente para evitar qualquer impacto nos algoritmos de controle de congestionamento, caso contrário o mecanismo de controle de fluxo entraria em ação de forma a frear o transmissor. Tal se daria através de uma redução no tamanho da janela anunciada pelo receptor. No lado do transmissor, um tamanho de janela reduzida, combinado à ocorrência de perdas de pacotes, também teria impacto nos resultados. Portanto, o tamanho de janela, seguindo a recomendação em [10], foi configurada com um valor bastante alto: 10.000 pacotes.

Para permitir a estabilização do sistema (da rede, no caso), como indicado na Seção 4.1, e evitar uma influência da ativação inicial do mecanismo de *slow-start* nos resultados, medidas foram coletadas sempre após um intervalo de tempo suficiente (15 segundos) a partir do estabelecimento da conexão TCP.

4.1.3 Largura de banda do gargalo

A Internet apresenta uma diversidade bastante grande de tecnologias e capacidades. Um mecanismo de controle de congestionamento, tanto em comunicação unicast como multicast, deve operar corretamente para qualquer que seja a largura de banda presente no enlace gargalo. Em uma simulação, é fundamental que as estratégias de controle de congestionamento sejam avaliadas em um intervalo grande de largura de bandas no gargalo.

Nesta dissertação, exceto quando explicitamente mencionado, foram realizados experimentos com as seguintes larguras de banda, conforme sugerido em [10]:

- 200Kbits/s;
- 500Kbits/s;
- 1Mbits/s;
- 8Mbits/s; e
- 64Mbits/s.

4.1.4 Aleatoriedade

Simuladores orientados a evento tendem a simular efeitos que podem não estar presentes no mundo real. Para evitá-los, se deve utilizar aleatoriedade nas simulações. Outras técnicas possíveis são o início dos fluxos dentro de uma margem aleatória, o uso de roteadores RED (*Random Early* Detection) na rede, e a inclusão de um atraso aleatório na transmissão de cada pacote

(disponível em apenas uma das implementações de TCP no ns). A maioria dos experimentos nesta dissertação está bem definida (em [10]) e não prevê a inclusão de tráfego adicional. Nos experimentos realizados, a única fonte de aleatoriedade presente na ferramenta ns são as perdas provocadas em enlaces com taxas de perdas maior do que zero.

4.1.5 Políticas de fila

Genericamente, o modelo simulado de roteador pode variar bastante quanto ao grau de detalhe considerado, tanto em termos de encaminhamento de pacotes (política das filas, tempo de processamento, contenção, acesso à tabela de roteamento, etc.) como em termos de algoritmos de roteamento (quanto à exatidão do modelo de algoritmo como o algoritmo de roteamento em si). Os dois tipos mais comuns de roteadores são, de acordo com sua política de filas, *drop-tail* e RED. Portanto, é aconselhável avaliar mecanismos de controle de congestionamento com roteadores *drop-tail,* RED e uma mistura de ambos. Neste trabalho, para manter a quantidade de experimentos e resultados apresentados compatível com uma dissertação, foram considerados apenas roteadores *drop-tail*. Este tipo foi escolhido dado seu grande emprego atualmente.

4.1.6 Configuração do roteamento de protocolo multicast

A avaliação de um protocolo de transporte ou de aplicação é influenciada pelas opções adotadas nas camadas subjacentes. Visando obter resultados mais genéricos possíveis, as opções deveriam corresponder a tecnologias e esquemas largamente adotados na Internet.

Particularmente no caso deste trabalho, a escolha do algoritmo de roteamento multicast pode exercer influência significativa nos resultados. Dependendo do protocolo adotado: haverá mais ou menos tráfego de controle, gerado pelo próprio protocolo de roteamento; pacotes seguirão rotas diferentes; estações levarão mais ou menos tempo para entrar e sair de grupos com comportamento dinâmico.

A ferramenta ns, empregada nesta simulação, fornece um número de opções de roteamento multicast possíveis, mas nenhuma delas reproduz completamente o comportamento destes protocolos em redes reais ([10]). Particularmente problemático é a falha do ns para simular controle de tráfego corretamente. Tal como sugerido nesta referência, os experimentos empregaram roteamento PIM-SM[3] centralizado, que utiliza um roteador intermediário como "ponto de encontro" entre a origem e o destino. Este roteador é chamado de *rendez-vous point* (RP).

4.2 Simulador VINT ns

A ferramenta ns é um simulador livremente distribuído (GPL, *General Public License*) cujo foco é modelar protocolos de rede. O simulador pode ser utilizado para avaliações de: comportamento do TCP, políticas de filas dos roteadores, transporte multicast, multimídia, redes sem-fio, respostas dos protocolos a variações da topologia e protocolos em nível de aplicação ([8]).

O ns é um simulador orientado a eventos: o tempo avança de forma discreta, de acordo

[3] *Protocol-Independent Multicast - Sparse Mode.*

com a ordem programada dos eventos. Um evento é escalonado para o futuro, de forma que uma rotina seja "disparada" quando o momento chegar. Eventos são criados em função da recepção de pacotes, bem como de estouros de temporizadores.

Para descrever uma simulação, o usuário prepara um *script* que define a topologia e configura os elementos/protocolos que tomam parte no experimento. A seguir serão descritos os principais componentes do ns conforme seu manual e documentação *on line* ([16]). Os mesmos são necessários ao entendimento dos experimentos, mas o leitor familiar com o ns pode proceder direto à Seção 4.3.

4.2.1 Conexão OTcl

O ns é um simulador orientado a objetos escrito nas linguagens C++ (compilada) e OTcl (um dialeto de Tcl, interpretada). Existem duas hierarquias de classes, uma em C++ e outra em OTcl. A justificativa em empregar duas linguagens é, segundo os autores do ns, a eficiência, a facilidade de codificar algoritmos e a possibilidade de manipular bytes, pacotes e cabeçalhos presentes em C++, aliados a uma interface que permita a variação de parâmetros, as alterações das configurações e a troca rápida de cenários como a fornecida pelo OTcl.

Há uma conexão entre as classes principais de C++ e OTcl, sendo que os objetos criados na hierarquia de classes interpretada são espelhados na hierarquia de classes compilada. A interface OTcl oferece comandos simples interpretados e métodos para comandos mais complexos. Também tem a capacidade de acessar variáveis C++ , como se fossem variáveis da instância OTcl.

Embora essa dualidade de linguagens traga benefícios em termos de desempenho, ela complica sobremaneira o entendimento do código do ns (que é suficientemente grande e complexo por si só). O uso de uma linguagem apenas, moderna (tal como Java), representa um grande avanço em termos de técnicas de simulação. O Simmcast ([37]) segue essa abordagem, porém não foi adotado porque não contém suporte a experimentos com controle de congestionamento.

4.2.2 Nodos e transmissão de pacotes

Para criar uma topologia no ns, é necessário primeiro criar os nodos e depois conectá-los através de enlaces. Os *agentes* são vinculados aos nodos, que podem ser nodos unicast ou multicast. Cada nodo recebe um número único de identificação ao ser criado. Existem métodos para:

- retornar o ponto de entrada do nodo;
- designar o primeiro elemento que irá manipular pacotes que chegam no nodo;
- excluir todos agentes do nodo;
- instalar "classificadores" de pacotes necessários para converter o nodo unicast em multicast;
- endereçar número de porta;
- retornar o número do nodo;

- retornar o agente designado para uma determinada porta;
- retornar o número da próxima porta disponível;
- adicionar um agente à lista de agentes;
- listar seus vizinhos adjacentes;
- adicionar vizinhos.

4.2.3 Enlaces

Enlaces conectam nodos de forma unidirecional ou bidirecional. São compostos por uma seqüência de conectores, que ao receberem um pacote, executam algumas funções e encaminham o pacote para seu vizinho ou o descartam. Dentre as funções executadas pelos conectores, estão a de rotular pacotes com o identificador de interface de chegada, modelar o atraso do enlace e as características de largura de banda e decrementar o TTL em cada pacote recebido. É possível especificar as características do enlace: a largura de banda, a latência, o tipo de fila utilizada, a orientação do enlace e a posição da fila.

4.2.4 Gerenciamento de filas e o escalonamento de pacotes

As filas são elementos onde pacotes podem ser conservados (ou descartados). O escalonamento de pacotes trata quais pacotes devem ser aproveitados ou descartados. O gerenciador de *buffer* regula a ocupação de uma fila, que pode ser do tipo *drop-tail* (FIFO), RED ou CBQ (*Class-Based Queueing*). As filas podem ser bloqueadas ou desbloqueadas pelos seus vizinhos: a fila é bloqueada quando um pacote está em trânsito entre ela e seu vizinho e permanecerá bloqueada enquanto o enlace estiver ocupado. É possível configurar o tamanho máximo em pacotes de todas filas ou para um determinado enlace.

4.2.5 Agentes

Agentes são pontos finais em que pacotes das camadas de rede são construídos ou consumidos. Os agentes particularmente relevantes a esse trabalho são descritos a seguir:

- **TCP.** Remetente TCP. O agente TCP possui os seguintes parâmetros de configuração: limite superior da janela em pacotes, limite superior da janela de congestionamento, tamanho inicial da janela de congestionamento no *slow-start*, média aleatória que atrasa cada saída de pacote, e o tamanho em bytes de todos pacotes. O agente atualiza as seguintes variáveis de estado: último ACK visto pelo receptor, RTT estimado e número de seqüência corrente transmitido. Existem as seguintes versões de TCP no ns: *Tahoe*, *Reno*, *NewReno*, *Sack1*, *Vegas*, *Fack*.
- **TCPSink.** Receptor TCP. Este agente recebe pacotes de dados do emissor TCP e envia pacotes de ACK. É possível configurar o tamanho em bytes utilizado para todos pacotes de ACKs.

- **UDP.** Agente UDP básico.
- **LossMonitor.** Receptor que monitora perdas. Implementa um controle de tráfego e estatísticas sobre dados recebidos, atualizando uma série de variáveis de estado: número de pacotes perdidos, número de pacotes recebidos, número de bytes recebidos, tempo no qual o último pacote foi recebido e número de seqüência esperado para o próximo pacote.
- **CtrMcast/Encap.** Encapsulador "multicast centralizado".
- **CtrMcast/Decap.** Desencapsulador "multicast centralizado".
- **Null.** Agente para descartar pacotes.

Os principais parâmetros para configurar os agentes são: identificação do fluxo, prioridade, endereço do agente, endereço da porta do agente, endereço de destino do agente, TTL. No *script* Tcl, uma vez que um novo agente é criado e é definido seu tipo, ele deve ser vinculado a um nodo. A partir deste momento é possível vincular uma aplicação ao agente e estabelecer uma conexão entre o agente remetente e o agente receptor.

4.2.6 Suporte

O ns possui suporte matemático, de monitoração e animação, descritos abaixo:

- **suporte matemático**: são funções para geração de números aleatórios;
- **suporte a monitoração**: inserido na topologia da rede e vinculados às filas, registra estatísticas de chegadas e tempos; estes registros são armazenados em um arquivo designado no *script* de simulação;
- **suporte a animação**: é feito através da ferramenta NAM, software escrito em Tcl/Tk; para sua utilização é necessário designar um arquivo NAM no *script* da simulação, onde serão armazenadas as informações da topologia e o registro do movimento dos pacotes.

4.2.7 Roteamento unicast

O roteamento unicast especifica a estratégia ou o protocolo de roteamento. A estratégia de roteamento é definida pelo mecanismo para computar as rotas, que pode ser: estático, de sessão ou dinâmico. Já o protocolo de roteamento executa um algoritmo específico. Os três tipos são:

- **roteamento unicast estático**: computa a rota de forma estática, através do mecanismo de roteamento de rota padrão, utilizando o algoritmo de Dijkstra. Ele executa uma vez no inicio da simulação, e neste momento é atribuído um determinado custo para cada enlace, que permanece estático durante toda simulação;
- **roteamento unicast de sessão**: é idêntica ao roteamento estático, executa o algoritmo de Dijkstra e estabelece custos dos enlaces da topologia, porém executa o algoritmo novamente para recalcular as rotas, sempre que a topologia for alterada durante a simulação;

- **roteamento unicast dinâmico**: neste caso, as rotas são estabelecidas baseadas na informação de mensagens que os nodos trocam entre si; ele implementa o roteamento DV *(Distance Vector)*.

4.2.8 Roteamento multicast

O roteamento multicast disponível no ns pode ser do tipo centralizado, modo denso ou árvore compartilhada:

- **centralizado**: utiliza um ponto central que recebe todos pacotes endereçados ao grupo e os distribui de acordo com uma árvore multicast tradicional. Para cada grupo, entre a origem e o destino existe um roteador intermediário chamado de *rendez-vous point* (RP).
- **modo denso**: projetado para operar num ambiente onde os membros do grupo estão densamente distribuídos. Pode ser executado em PIM-DM ou DVMRP;
- **árvore compartilhada**: o remetente envia o pacote a ser distribuído para um roteador núcleo. O núcleo redistribui uma cópia do pacote para cada destinatário, através da árvore multicast, cuja raiz é o próprio núcleo. Existe uma única árvore para cada grupo, independente do número de remetentes.

4.3 Experimentos

Esta seção descreve o conjunto de experimentos realizados como parte da dissertação, enfatizando a preparação dos mesmos (os resultados são vistos e comentados no capítulo seguinte). O conjunto de experimentos está baseado em [10].

Os experimentos tipicamente avaliam um ponto específico do mecanismo de controle de congestionamento, como escalabilidade do mecanismo, comportamento em configurações heterogêneas e justiça na utilização da largura de banda.

Todas considerações metodológicas descritas na Seção 4.1 foram observadas. Exceto quando indicado explicitamente, emprega-se o seguinte conjunto de valores padrão:

- 10ms de latência em todos enlaces;
- 100Mbits/s de largura de banda em enlaces que não o gargalo;
- 0% de perdas aleatórias nos enlaces;
- tamanho de pacote de 1KB (incluindo cabeçalhos).

Os experimentos executados tem duas fontes de perda de pacotes diferentes, dependendo do objetivo da simulação. Em alguns experimentos, há suficiente largura de banda para suportar os fluxos e emprega-se descarte de pacotes aleatório em certos enlaces. Em outros, estuda-se o fator de enfileiramento do sistema, descartando pacotes apenas quando ocorre *overflow* nas filas.

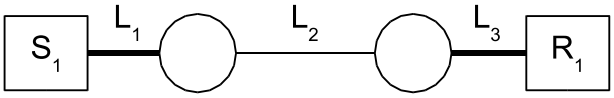


FIGURA 4.1 – Gargalo simples

4.3.1 Verificação de sanidade

O objetivo deste experimento é verificar se o PePcc se comporta minimamente dentro do padrão esperado. Isto é verificado através de uma transmissão ponto a ponto com o protocolo multicast, que inicia sozinho a transmissão mas que depois passa a compartilhar um enlace com um fluxo de taxa constante.

Mais precisamente, há um único par remetente/receptor rodando o PePcc, e eles começam a se comunicar através de uma rota de três pontos com largura de banda no enlace gargalo igual a 1Mbits/s, conforme demonstrado na Figura 4.1. Após 30 segundos, um fluxo CBR (*Constant Bit Rate*) emite tráfego através do enlace gargalo na taxa de 500Kbits/s. Após 60 segundos, o fluxo CBR diminui sua taxa para 250Kbits/s.

4.3.2 Perda correlata

Em protocolos unicast, a comunicação se dá entre um remetente e um receptor apenas; quando um pacote é perdido em um enlace ou em um roteador, o receptor não receberá o pacote e será necessário recuperar essa perda. Em protocolos multicast, um número qualquer de receptores (entre 1 e n, onde n é o tamanho do grupo) pode observar a perda do mesmo pacote. O número de receptores dependerá da posição na árvore onde a perda ocorreu. Quando dois ou mais receptores observam a mesma perda, houve "correlação de perda" ([50]).

Este experimento tem por objetivo verificar o comportamento do PePcc em relação a perdas correlatas. No primeiro experimento, observa-se o comportamento do protocolo mediante perdas correlatas para três receptores; no segundo, para apenas dois receptores, enquanto no terceiro não há correlação de perdas.

Para tal, emprega-se uma topologia em árvore binária com profundidade três: o remetente S_1, na raiz, transmite a três receptores, denominados R_1, R_2 e R_3, conforme ilustrado na Figura 4.2). Variando-se a probabilidade de perda nos enlaces, é possível gerar perdas correlatas e não correlatas.

Tal cenário é obtido com a seguinte configuração (vide Figura 4.3, conforme [10]):

(a) o enlace compartilhado (L_1) experimenta perda aleatória de 10%.

(b) o enlace compartilhado (L_1) experimenta 0% perda de pacote, mas ambos ramos binários (L_2, L_3) do próximo nível experimentam perda de 10% dos pacotes.

(c) os três enlaces *(*L_4, L_5, L_6*)* próximos aos receptores experimentam perda de 10% dos pacotes.

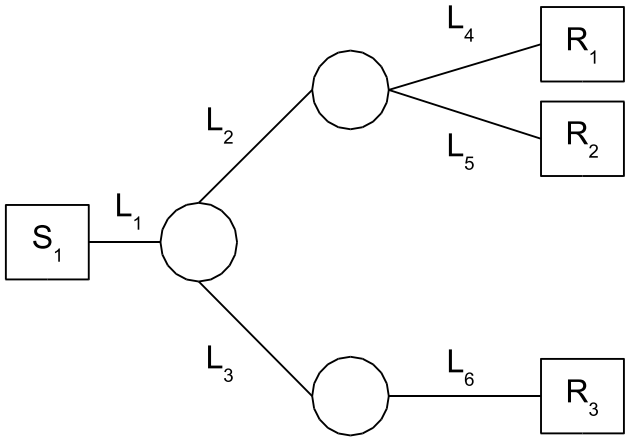
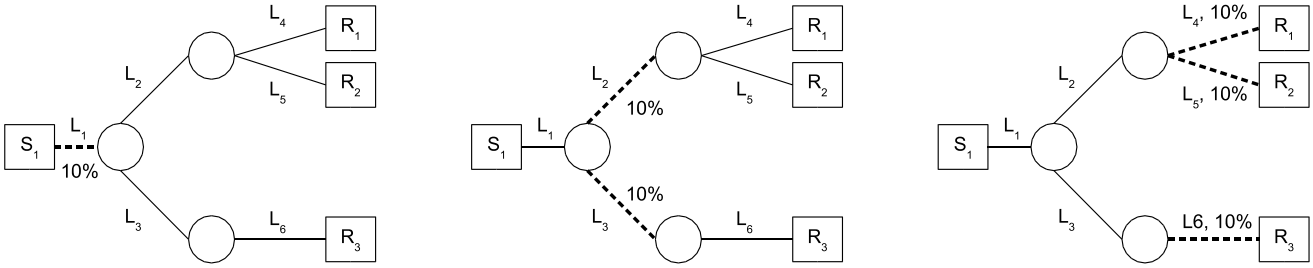


FIGURA 4.2 – Árvore de profundidade 3



(a) perda correlata completa

(b) perda correlata parcial

(c) sem correlação de perda

FIGURA 4.3 – Configurações para experimento com correlação de perda.

4.3.3 Perda heterogênea

Este experimento visa verificar o comportamento do PePcc em situações de perdas heterogêneas de pacotes nos enlaces da árvore multicast. Duas situações de perda heterogênea são examinadas, ambas com uma árvore de profundidade três igual à Figura 4.2:

- na primeira situação, são estipuladas diferentes probabilidades de perdas nos ramos da árvore, fazendo com que receptores R_1, R_2 e R_3 observem probabilidades de perdas iguais a 10%, 10% e 5%, respectivamente;
- na segunda situação, a topologia é alterada fazendo com que as probabilidades de perdas observadas sejam de 10%, 5% e 5%, respectivamente.

As probabilidades de perdas nos enlaces são distribuídas da seguinte maneira:

(a) o enlace compartilhado, L_1, não tem perdas, enquanto L_2 sofre perda de 10% e L_3 perda de 5% dos pacotes;

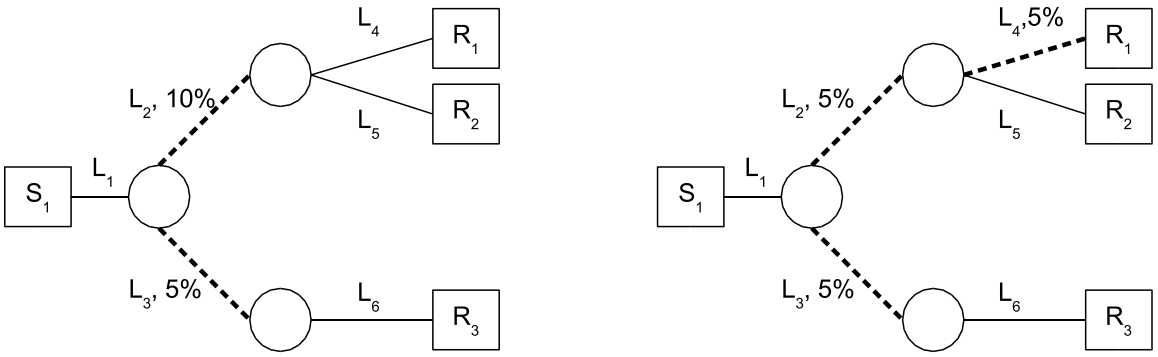
(b) a configuração é alterada para separar o segmento com probabilidade de perda de 10%, L_2, em dois segmentos com probabilidade de perda de 5%, L_2 e L_4, sendo que o segundo deles é utilizado apenas pelo receptor R_1. Com esta alteração, o receptor R_1 tem dois segmentos de perda de 5%, o receptor R_2 tem um único segmento de perda de 5% compartilhado com o receptor R_1, e o receptor R_3 tem segmento de perda não correlata de 5%.

As topologias resultantes são ilustradas nas Figuras 4.4.(a) e 4.4.(b). Em ambos experimentos, um remetente PePcc transmite dados para os três receptores. Os resultados obtidos devem ser comparados com experimentos conduzidos com o TCP: os dois experimentos acima são repetidos, substituindo-se a transmissão multicast por uma conexão TCP para o receptor R_1.

4.3.4 Latência heterogênea

O objetivo deste experimento é observar o comportamento do PePcc na presença de enlaces com tempos de propagação diferentes nas rotas entre o remetente multicast e seus receptores. O primeiro experimento é executado com um remetente PePcc transmitindo para dois receptores, e a seguir repetido substituindo-se o fluxo PePcc por um fluxo TCP na rota de maior latência.

A topologia para este experimento é uma árvore binária com dois receptores, conforme ilustrada na Figura 4.5. O enlace compartilhado, L_1, possui largura de banda de 100Mbits/s e probabilidade de perda de pacotes igual a 10%. Os enlaces independentes, L_2 e L_3, possuem largura de banda e probabilidade de perda idênticas (100Mbits/s e 1% respectivamente), mas latências diferentes em uma ordem de magnitude: L_2 igual a 20ms e L_3 a 200ms.



(a) primeiro caso

(b) segundo caso

FIGURA 4.4 – Topologias para perda heterogênea.

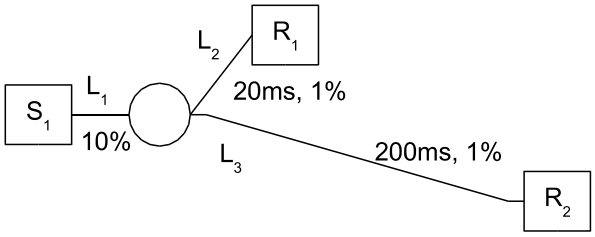


FIGURA 4.5 – Topologia para experimento com latência heterogênea

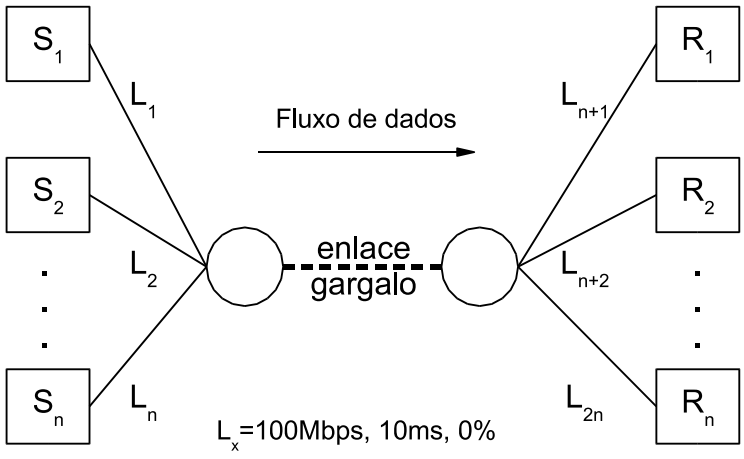


FIGURA 4.6 – Topologia em "estrela dupla"

4.3.5 Justiça simples em TCP

Com este experimento, é verificado o comportamento do PePcc na presença de fluxos TCP. Tal deverá apresentar comportamento justo, do ponto de vista de compartilhamento da largura de banda disponível, com os fluxos TCP que trafegam na mesma rota utilizada pelo fluxo multicast.

A topologia utilizada para a realização deste experimento é uma rede de estrela dupla, conforme representado na Figura 4.6. O enlace gargalo é o único enlace compartilhado. Os remetentes estão dispostos do lado esquerdo do enlace compartilhado e os receptores estão posicionados à direita deste enlace. Todos os enlaces que não o gargalo possuem largura de banda igual a 100Mbits/s, latência de 10ms e probabilidade de perda aleatória 0%.

O experimento consiste em iniciar dois fluxos TCP ($S_1 \rightarrow R_1$ e $S_2 \rightarrow R_2$) e, após 10 segundos, iniciar um fluxo PePcc para dois receptores ($S_3 \rightarrow \{R_3, R_4\}$). O experimento é executado 5 vezes, alterando-se a largura de banda do enlace gargalo para cada um dos valores descritos na Seção 4.1.3.

Por fim, note-se que a topologia em estrela dupla (Figura 4.6) conforme sugerida por [10] para avaliação apresenta em geral um produto entre largura de banda e latência bastante significativo, de no mínimo 250 pacotes. Em cada enlace L_i, para $1 \leq i \leq 2n$, "cabem" 1.000.000 bits, 125.000 bytes, ou 125 pacotes. Considerando-se que há dois enlaces nas extremidades, e tomando-se a faixa de gargalos testados, os produtos resultantes são: 250 pacotes para 200Kbits/s e 500Kbits/s, 251 pacotes para 1Mbits/s, 260 pacotes para 8Mbits/s, e 330 pacotes para 64Mbits/s. Dessa maneira, no melhor caso e com um fluxo, o transmissor poderá enviar 330 pacotes antes que o primeiro deles atinja um receptor. O lado negativo é que o controle de congestionamento do remetente poderá reagir a um congestionamento quando um número potencialmente alto de pacotes já foi enviado.

4.3.6 Justiça em TCP com alta multiplexação

O objetivo deste experimento é verificar o comportamento do PePcc na presença de fluxos TCP do ponto de vista do compartilhamento da largura de banda disponível, quando a quantidade de fluxos TCP e a largura de banda aumentam.

A topologia é idêntica a do experimento da Seção 4.3.5, ilustradas na Figura 4.6. O experimento é executado para três larguras de banda no enlace gargalo; as larguras de banda avaliadas foram escolhidas como as intermediárias (500Kbits/s, 1Mbits/s e 8Mbits/s), em função da proporcionalidade no número de fluxos TCP. O primeiro experimento começa com 3 fluxos TCP, e após 10 segundos, inicia um fluxo multicast para 2 receptores. Os experimentos seguintes são idênticos, porém empregam 4 e 6 fluxos TCP, respectivamente. A relação entre largura de banda e quantidade de fluxos TCP fica assim nos três experimentos:

(a) largura de banda 500Kbits/s com 3 fluxos TCP;

(b) largura de banda 1Mbits/s com 4 fluxos TCP;

(c) largura de banda 8Mbits/s com 6 fluxos TCP.

4.3.7 Justiça entre sessões

Neste experimento é verificado o compartilhamento justo de recursos entre fluxos do PePcc. Existem duas situações distintas; na primeira situação, (a) são executados exclusivamente 2 fluxos PePcc. Na segunda situação, (b) além dos 2 fluxos PePcc, são acrescentados 2 fluxos TCP como ruído.

A topologia é idêntica a do experimento da Seção 4.3.5, ilustrada na Figura 4.6. No experimento (a), dois fluxos PePcc transmitem, cada um para um receptor. O experimento (b) começa com 2 fluxos TCP e, após 10 segundos, 2 fluxos PePcc transmitem, cada um para um receptor. Cada um destes experimentos é executado 5 vezes, uma para cada largura de banda prevista.

4.3.8 Justiça heterogênea

O objetivo deste experimento é verificar se o comportamento do PePcc não é injusto com os fluxos TCP, do ponto de vista do compartilhamento da largura de banda disponível, em situações que o RTT dos fluxos seja heterogêneo, ou seja que, tenham variações na latência.

São executados dois fluxos TCP e um fluxo PePcc para dois receptores. Assim como no experimento anterior, a topologia baseia-se em uma estrela dupla (Figura 4.6), porém com alterações na latência de certos enlaces. Um receptor TCP e um receptor PePcc possuem o enlace de chegada com latência de 10ms, enquanto um receptor TCP e o outro PePcc possuem o enlace de chegada com latência de 100ms. Portanto, haverá dois fluxos TCP, um "longo", $S_1 \rightarrow R_1$, e outro "curto", $S_2 \rightarrow R_2$, além de um fluxo PePcc "longo", $S_2 \rightarrow \{R_3, R_4\}$. A Figura 4.7 mostra o cenário resultante.

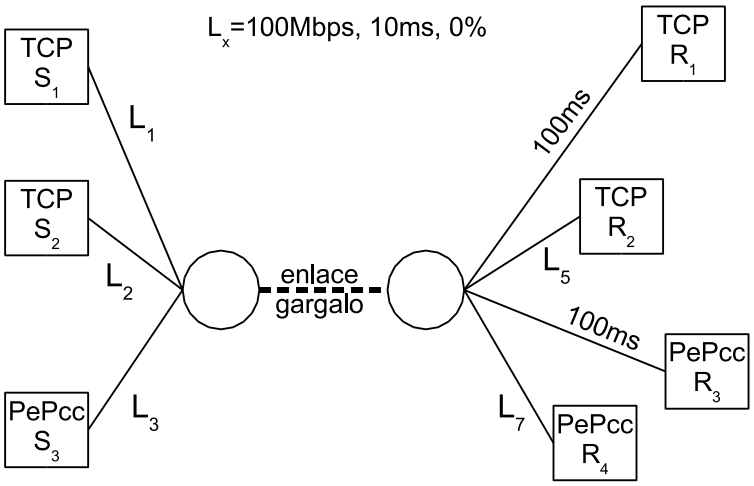


FIGURA 4.7 – Cenário para experimento com justiça heterogênea

4.3.9 Aumento da verificação de sanidade

Tal como o experimento na Seção 4.3.1, o objetivo deste experimento é verificar se o mecanismo do protocolo avaliado se comporta adequadamente perante mudanças na quantidade de largura de banda disponível. Entretanto, neste experimento o tamanho do grupo multicast aumenta para centenas de receptores.

Nessa dissertação, foi escolhida uma topologia que é uma árvore binária de altura 8 e 128 folhas (nodos receptores). Os nodos estão interligados com enlaces padrão, ou seja, de 100Mbits/s, 10ms de latência e 0% de probabilidade de perda. O remetente está diretamente conectado ao nodo raiz da árvore, no enlace gargalo de 1Mbits/s. A execução do PePcc inicia para os 128 receptores através desta árvore. Após 30 segundos, um fluxo CBR emite tráfego através do enlace gargalo na taxa de 500Kbits/s. Após 60 segundos, o fluxo CBR diminui sua taxa para 250Kbits/s.

4.3.10 Curva de perda de taxa

Este experimento tem como objetivo observar a queda de taxa do PePcc perante situações de ocorrência de perdas aleatórias e latência diferentes. A topologia é a mesma do experimento da Seção 4.3.5. Um remetente PePcc transmite para um único receptor através de uma rota de 3 pontos, como na Figura 4.1, com a largura do enlace gargalo ajustada para ser suficientemente grande (100Mbits/s). O experimento é executado para todas as combinações das seguintes probabilidades de perdas e latências, conforme definido em [10]: latências de 10ms e 100ms, e probabilidades de perdas de 3%, 7% e 10%.

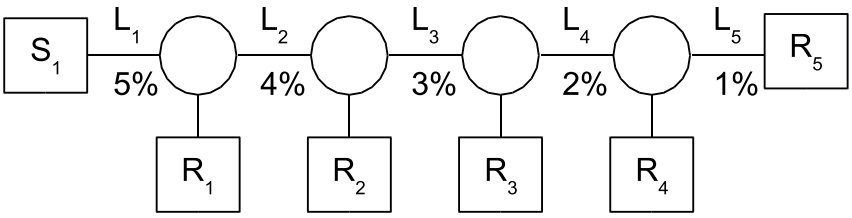


FIGURA 4.8 – Topologia com cascata de gargalos, para experimento de agregação de perda

4.3.11 Agregação correta de perda

O objetivo deste experimento é verificar se o PePcc agrega as perdas de forma correta, em uma situação que os receptores, à medida que se afastam do remetente, vão acumulando probabilidade de perda descendentes nos enlaces que compõe a rota entre eles e o remetente.

Conforme visto na Figura 4.8, há uma árvore com uma seqüência de enlaces de perda diminuindo de 5% para 1% em intervalos de 1%. Um remetente PePcc transmite para 5 receptores, cada um deles localizado imediatamente abaixo de cada um dos enlaces de perda.

4.3.12 Queda até zero (*drop-to-zero*)

Neste experimento, o objetivo é verificar se o PePcc não reduz sua taxa de transmissão a zero caso ocorram perdas em todas as rotas entre origem e receptores. Nestes experimentos, não há enlace gargalo e sim enlaces com perda aleatória. São avaliadas duas situações, conforme explicadas a seguir:

(a) todos receptores sofrem com a mesma probabilidade de perda;

(b) todos receptores sofrem a mesma probabilidade de perda, exceto um deles, que observa uma probabilidade de perda 4 vezes maior.

Os experimentos utilizam uma topologia em estrela, conforme mostrado na Figura 4.9. No experimento (a), o PePcc envia para 50 receptores (R_1 a R_{50}), cada um conectado a um enlace configurado com 5% de probabilidade de perda de pacotes (perda não correlata). No experimento (b), o enlace do receptor R_{50} é configurado com probabilidade de perda igual a 20%.

4.3.13 Perda não correlata

O objetivo deste experimento é verificar se o PePcc reduz adequadamente sua taxa de envio numa situação em que os receptores experimentam uma pequena probabilidade de perda, enquanto uma probabilidade de perda maior é alternada entre todos receptores.

O experimento utiliza uma topologia de estrela, como a da Figura 4.9, porém com um grupo de 20 receptores PePcc. No instante inicial, os enlaces são configurados de forma que 19 receptores experimentem perda aleatória de 1% dos pacotes, enquanto o vigésimo receptor experimenta perda de 5%. A cada 5 segundos, o local do receptor com maior perda é alternado.

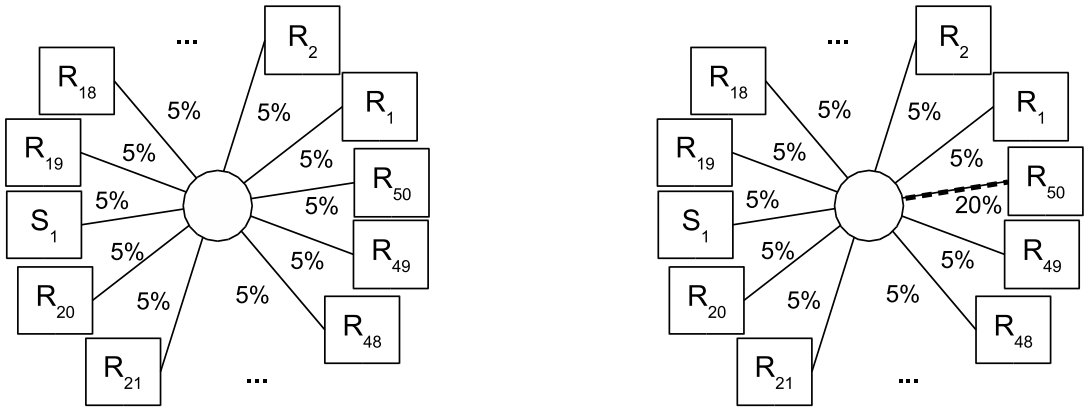


FIGURA 4.9 – Topologias em estrela utilizadas no experimento queda até zero.

Capítulo 5

Resultados de Simulação

Este capítulo se baseia na metodologia de avaliação descrita no capítulo anterior e, através de um conjunto de experimentos de simulação, aborda os resultados obtidos com o protocolo PePcc. A avaliação visa mostrar o comportamento do PePcc ao longo de uma sessão de transferência de dados, em diferentes situações de carga da rede.

Conforme indicado no capítulo anterior (4), nos experimentos com controle de congestionamento há apenas uma métrica de interesse: o *throughput*, ou seja, a taxa com que dados são enviados pelo remetente ou recebidos pelos receptores, dependendo do experimento. A taxa é medida no remetente nas configurações com perda aleatória e sem gargalo (Subseções 4.3.2, 4.3.3, 4.3.4, 4.3.10, 4.3.11, 4.3.12, 4.3.13), pois nestas não faria sentido medir a taxa de recepção após enlaces com taxas de perda não nulas. Assim, em todos os gráficos desta seção, o eixo x mostra o tempo do experimento em segundos, e o eixo y, a taxa observada (em Kbits/s).

O capítulo está organizado em 13 seções, em exata correspondência ao conjunto de experimentos definidos no capítulo anterior (Subseções 4.3.1 a 4.3.13). Apesar de os princípios básicos e objetivos de cada experimento serem relembrados, estima-se que o leitor fará freqüentes referências aos métodos de simulação e topologias lá descritas.

5.1 Verificação de Sanidade

Conforme descrito na Subseção 4.3.1, o primeiro experimento é um teste básico do funcionamento do mecanismo. Ele serve para mostrar que o mecanismo do protocolo se adapta dinamicamente à largura de banda disponível. A largura de banda disponível em um enlace gargalo é variada dinamicamente através da adição de um fluxo de taxa constante. Na topologia com gargalo simples (Figura 4.1 na página 52), um transmissor S_1 e um receptor R_1 se comunicam usando o PePcc através de um conjunto de enlaces, sendo L_2 o enlace gargalo com 1000Kbits/s.

Como comportamento esperado, o fluxo multicast deve se adaptar à largura de banda do enlace gargalo suave e eficientemente, em três momentos: após o início do fluxo CBR, quando ele se reduz, e após sua parada. Um fluxo CBR equivale, de forma geral, a reduzir dinamicamente a quantidade de largura de banda disponível.

O comportamento observado correspondeu à expectativa, conforme pode ser visto na Figura

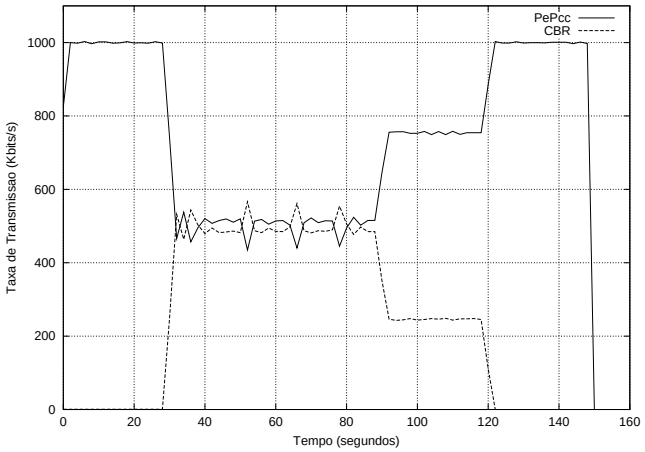


FIGURA 5.1 – Experimento 5.1 que verifica redução dinâmica de largura de banda PePcc (sanidade básico)

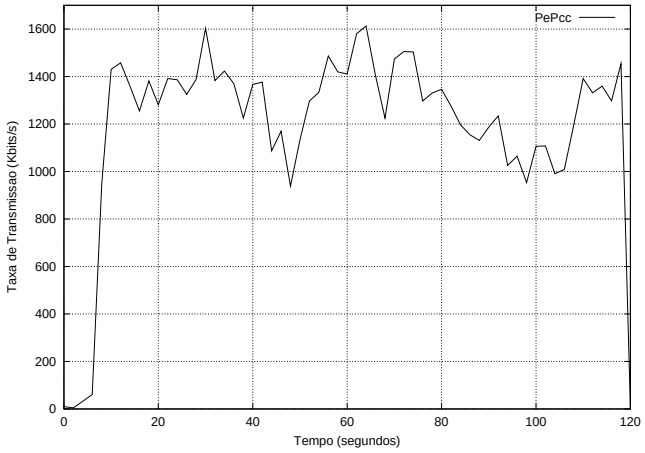
5.1. A largura de banda ocupada pelo PePcc é adaptada rápida e suavemente cada vez que a quantidade de banda disponível é alterada em L_2. Tal adaptação ocorre nos momentos 30s, quando inicia o fluxo constante de 500Kbits/s, e a 90s, quando este baixa para 250Kbits/s. Por fim, note-se que o protocolo aproveita de forma completa e eficiente a largura de banda disponível.

5.2 Perda Correlata

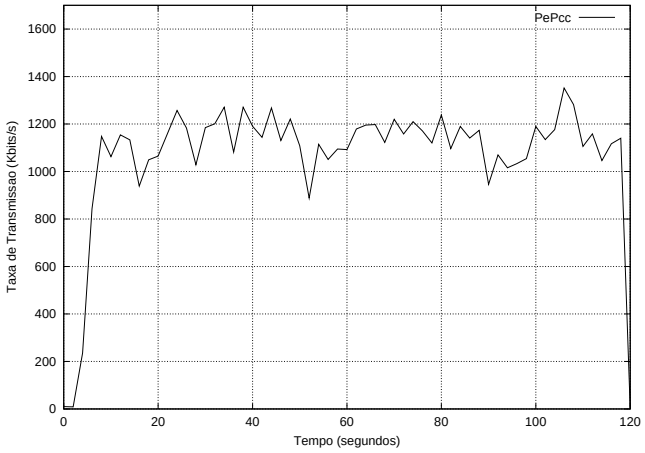
Este experimento serve para averiguar o comportamento do PePcc mediante perdas correlatas, conforme descrito na Subseção 4.3.2. Relembrando, perda correlata ocorre quando um pacote é descartado em um nodo intermediário da árvore de distribuição multicast, fazendo com que dois ou mais receptores deixem de receber o mesmo pacote. Foi empregada uma topologia em árvore binária com profundidade três: o remetente S_1, na raiz, transmite a três receptores, denominados R_1, R_2 e R_3. Os experimentos foram executados em três cenários diferentes, descritos nas Figuras 4.3.(a), 4.3.(b) e 4.3.(c) (na página 53).

Como a probabilidade de perda, do ponto de vista dos receptores, é a mesma nos três experimentos (10%), espera-se que o fluxo multicast acabe por utilizar larguras de banda similares nos três experimentos, com potencial decréscimo entre os casos (a), (b) e (c). Tal se deve à aleatoriedade das perdas nos enlaces, fazendo com que o protocolo tenha que tratar perdas de pacotes diferentes.

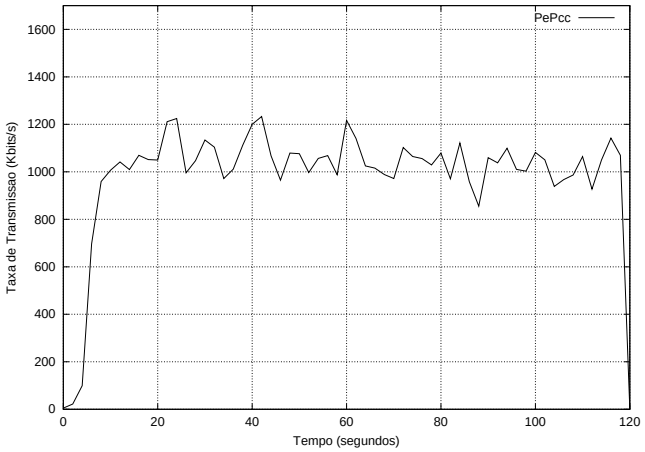
Comparando-se os gráficos nas Figuras 5.2.(a), 5.2.(b) e 5.2.(c), nota-se que as taxas alcançadas nos três cenários é similar, desconsiderando-se os 15 segundos iniciais. Nota-se que **não** há queda de taxa em função de perdas correlatas, ou seja, o tamanho da janela de congestionamento não é reduzido quando as mesmas perdas são observadas por mais de um receptor. Estes resultados são, portanto, satisfatórios.



(a) perda correlata completa



(b) perda correlata parcial



(c) sem correlação de perda

FIGURA 5.2 – Experimento 5.2 sobre reação do PePcc mediante correlação de perdas

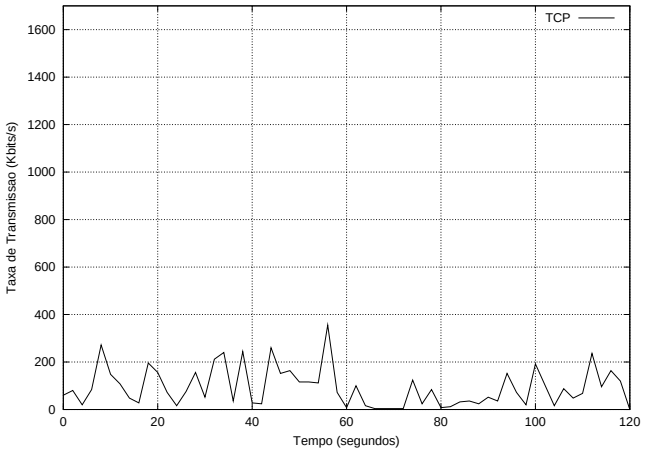


FIGURA 5.3 – Experimento 5.2 sobre reação do TCP mediante correlação de perdas

Adicionalmente, espera-se que em cada um dos experimentos a largura de banda média registrada seja similar a uma conexão TCP ao longo de uma rota entre o remetente e qualquer um dos receptores. O receptor em particular, R_1, R_2 ou R_3, não faz diferença, uma vez que todos observam a mesma probabilidade de perda aleatória de 10%. A avaliação se dá pela execução de um único[1] experimento, substituindo-se o protocolo multicast por um fluxo TCP entre o remetente e o receptor R_1.

Neste quesito, o PePcc apresentou, em todos os três casos, taxas no remetente bastante superiores àquela registrada no TCP (compare as Figuras 5.2 e 5.3). Há duas interpretações possíveis, uma negativa e a outra positiva. Por um lado, observa-se que o PePcc gera taxas mais altas do que o TCP em situações de enlaces com alto índice de perdas aleatórias (10%), o que poderia indicar que o PePcc está transmitindo a uma taxa superior ao que deveria. Por outro, mecanismos de controle de congestionamento tipicamente interpretam perdas de pacotes como sinal de congestionamento; com perdas aleatórias induzidas, não há congestionamento, e a simples redução da taxa não resolveria o problema. Assim, o PePcc é mais "imune" a essas perdas aleatórias, utilizando de maneira mais eficiente a rede.

Por fim, é normal que as taxas de transmissão do PePcc sejam bem superiores: por se tratar de um protocolo multicast, haverá um número significativamente maior de perdas a serem recuperadas pelo remetente (são 3 receptores com perda de 10%). Perdas são tratadas com retransmissões via multicast, conforme visto na Seção 3.1.3. Estas retransmissões são feitas imediatamente, pois não são contabilizadas na janela de congestionamento; entretanto, novas transmissões são contadas na taxa e fazem o consumo de largura de banda aumentar. Pelas razões acima, conclui-se que o PePcc apresenta comportamento compatível com o esperado.

[1] não é necessário executar três experimentos, posto que a única diferença será a localização do enlace com perdas, resultando diferença negligível de desempenho.

5.3 Perda Heterogênea

O objetivo deste experimento é verificar o comportamento do PePcc em situações de perdas de pacotes heterogêneas nos enlaces de uma árvore de profundidade três, conforme descrito na Subseção 4.3.3. Relembrando, perda heterogênea ocorre quando o conjunto de receptores multicast observa taxas de perdas significativamente distintas. Dois cenários de perda heterogênea são avaliados, conforme as Figuras 4.4.(a) e 4.4.(b) (página 55).

O comportamento desejado para ambos os casos acima é que a taxa média da sessão multicast seja adaptada à situação de R_1 (este, nos dois experimentos, é configurado com a maior probabilidade de perda aleatória). Além disso, a taxa deve ser similar a de uma conexão TCP entre S_1 e R_1.

As Figuras 5.4.(a) e 5.4.(b) mostram, lado a lado, as taxas em S_1 para sessões PePcc. Nota-se, primeiramente, que não há diferença significativa de taxa entre os cenários. Como desejado, PePcc adapta a taxa de envio com base no pior receptor, que corresponde a R_1 e R_2 no primeiro cenário e R_1 no segundo.

Em ambos os cenários, o PePcc leva a uma taxa média em torno de 1200Kbits/s. Como citado, a taxa do PePcc deveria se assemelhar àquela verificada com o TCP. Para executar essa avaliação, o experimento foi repetido utilizando-se um fluxo TCP $S_1 \rightarrow R_1$ nos mesmos cenários (Figuras 4.4.(a) e 4.4.(b)). Como demonstram as Figuras 5.4.(c) e 5.4.(d), as taxas dos fluxos TCP oscilam em torno de 100Kbits/s, uma ordem de magnitude abaixo do que foi registrado com PePcc.

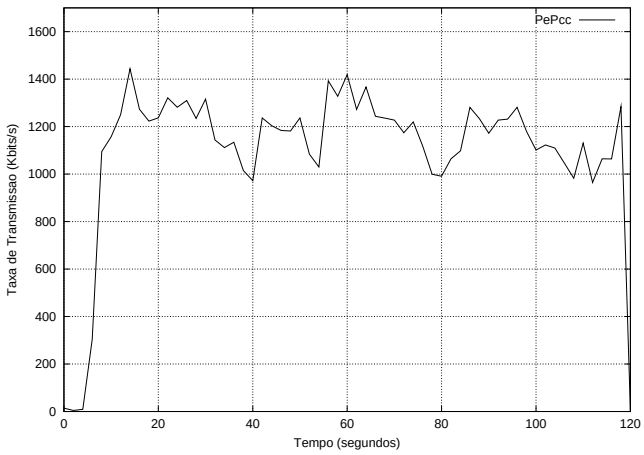
Note-se que as taxas apresentadas pelo PePcc e pelo TCP nesse experimento são **bastante** similares às dos experimentos de perda correlata (na seção anterior). O motivo para as taxas de transmissão do TCP serem menores do que as do TCP já foi discutido naquela seção. Em suma, o PePcc adapta com sucesso a taxa ao pior receptor, porém essa taxa é bem maior do que aquela que seria obtida com TCP.

5.4 Latência Heterogênea

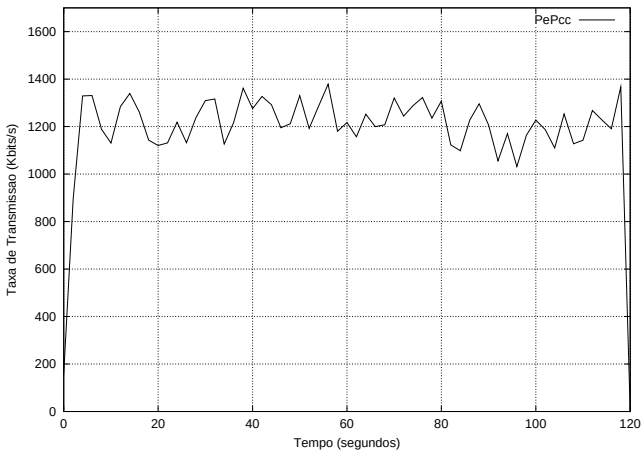
Conforme descrito na Subseção 4.3.4, o objetivo deste experimento é observar o comportamento do PePcc na presença de enlaces com tempos de propagação significativamente diferentes nas rotas entre o remetente multicast e seus receptores. A topologia está ilustrada na Figura 4.5 (na página 55). Neste cenário, além da diferença em latências, enlaces possuem probabilidades de perdas aleatórias substanciais e significativamente diferentes entre si (10% para L_1 e 1% para L_2 e L_3).

O comportamento desejado para o PePcc é que sua taxa média seja aproximadamente igual à taxa de uma sessão TCP. Os resultados obtidos com o PePcc foram comparados com os gerados pelo TCP: o fluxo multicast $S_1 \rightarrow \{R_2, R_3\}$ é substituído por um fluxo TCP $S_1 \rightarrow R_2$.

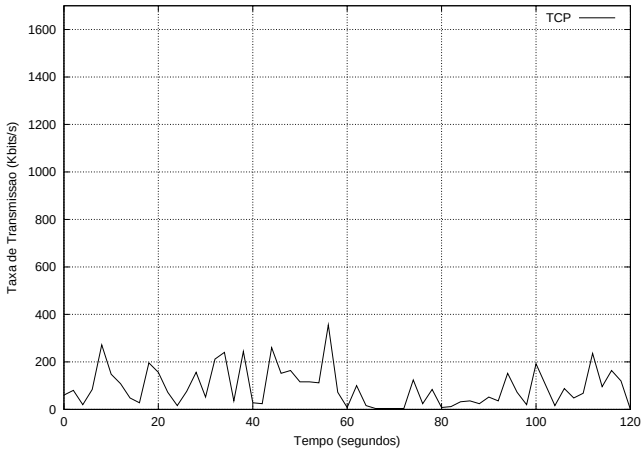
As Figuras 5.5.(a) e 5.5.(b) apresentam as taxas registradas nos receptores para PePcc e TCP. Nota-se, primeiramente, que para ambos PePcc e TCP as taxas médias são bastante baixas, 86 e 35Kbits/s, respectivamente. Isto se deve ao alto número de perdas induzidas nos enlaces, e à grande latência, o que atrasa a "recuperação" da janela de congestionamento. O



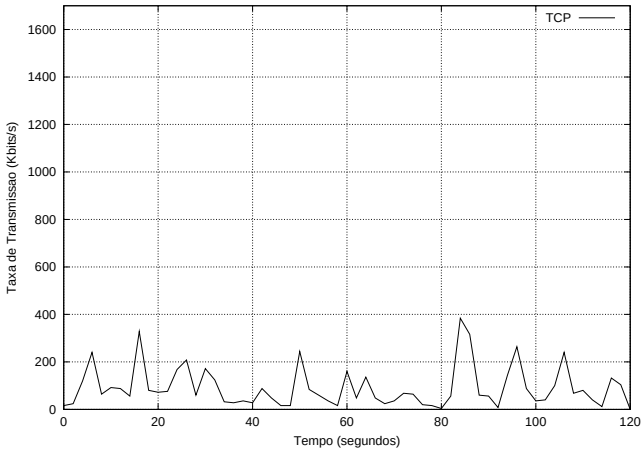
(a) PePcc no primeiro cenário



(b) PePcc no segundo cenário



(c) TCP no primeiro cenário



(d) TCP no segundo cenário

FIGURA 5.4 – Experimento 5.3, que avalia o PePcc e TCP mediante perdas heterogêneas (cenários das Figuras 4.4.(a) e 4.4.(b), respectivamente)

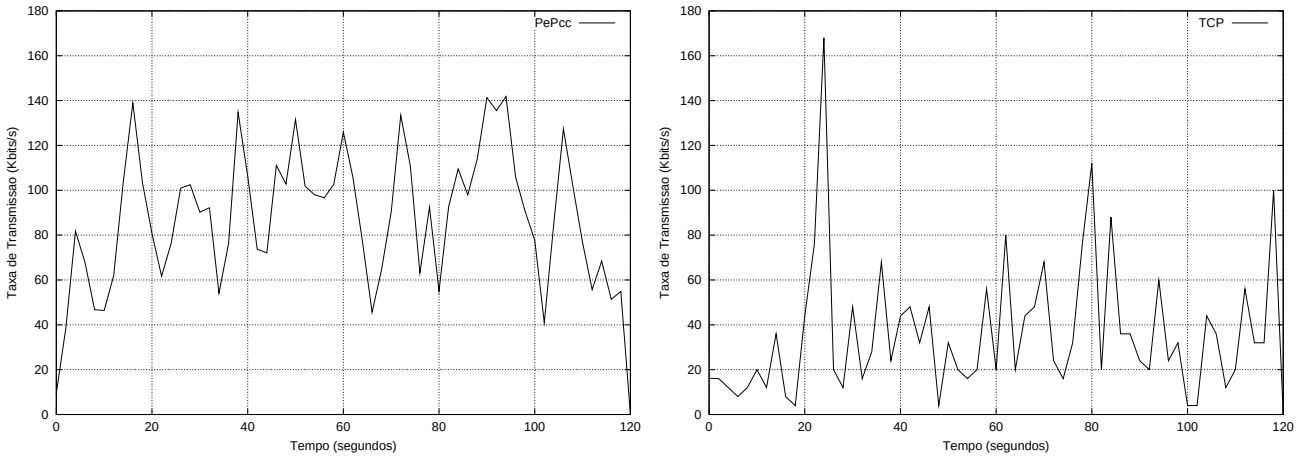


FIGURA 5.5 – Experimento 5.4, que avalia PePcc em relação ao TCP mediante latências heterogêneas (cenário da Figura 4.5)

produto entre largura de banda e latência, para o caso de 100Mbits/s e 210ms, corresponde a aproximadamente 2752 pacotes de 1KB.

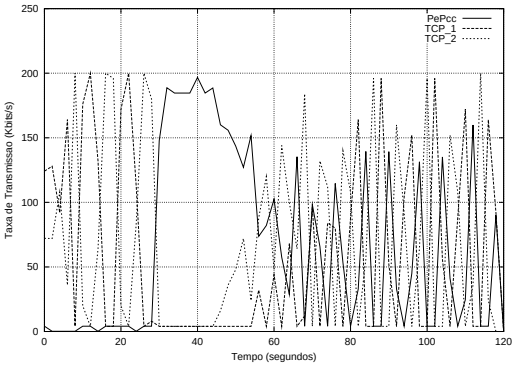
Observa-se que a taxa média registrada para o PePcc é em geral superior à taxa média do TCP. Embora perceptível, a diferença não é significativa se for considerada a capacidade dos enlaces de comunicação, igual a 100Mbits/s. Os 50Kbits/s de diferença podem, como já citado, ser resultado da quantidade de perdas a serem recuperadas pelo transmissor.

5.5 Justiça Simples em TCP

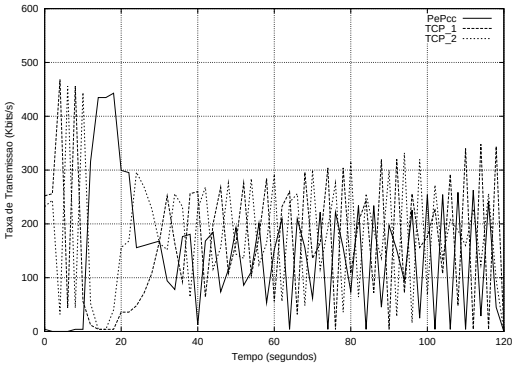
O objetivo deste experimento é verificar o comportamento do PePcc na presença de dois fluxos TCP, para várias larguras de banda no enlace gargalo, tal como descrito na Subseção 4.3.5. O comportamento desejado é que em um tempo relativamente curto após o inicio do fluxo multicast seja alcançado um equilíbrio, de forma que o fluxo PePcc receba aproximadamente 1/3 da largura de banda disponível, independentemente da largura de banda verificada no enlace gargalo.

As Figuras 5.6.(a), 5.6.(b), 5.6.(c), 5.6.(d) e 5.6.(e) apresentam as taxas observadas nos receptores TCP (R_1 e R_2) e PePcc (R_3), para gargalos de 200Kbits/s, 500Kbits/s, 1Mbits/s, 8Mpbs e 64Mbits/s, respectivamente. A escala dos eixos y é configurada em cada gráfico em pouco mais que a largura de banda do gargalo.

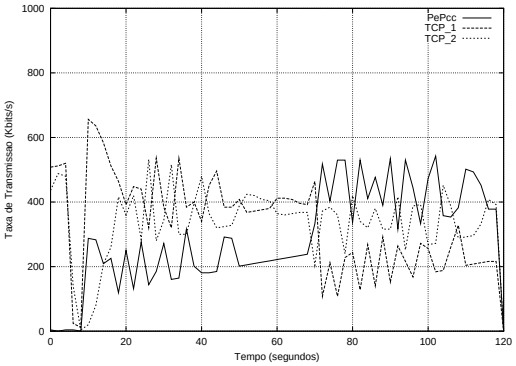
De forma geral, nota-se que a medida que a largura de banda no enlace gargalo aumenta, diminuem as oscilações nas taxas observadas pelo receptores. A razão para isso é que a diferença entre os enlaces antes do gargalo (100Mbits/s) e o gargalo fica cada vez menos significativa. Considerando que a latência é preservada, o aumento da largura de banda aumenta proporcionalmente o produto da latência pela banda, sendo capaz de armazenar um número bem maior



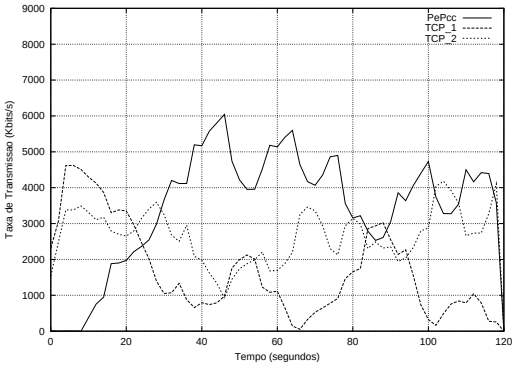
(a) 200Kbps



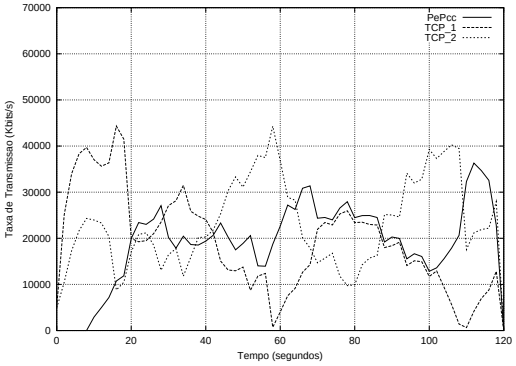
(b) 500Kbps



(c) 1Mbps



(d) 8Mbps



(e) 64Mbps

FIGURA 5.6 – Experimento 5.5 de justiça simples, com 3 fluxos (PePcc e 2 TCPs) para as cinco larguras de banda previstas para o gargalo (topologia da Figura 4.6)

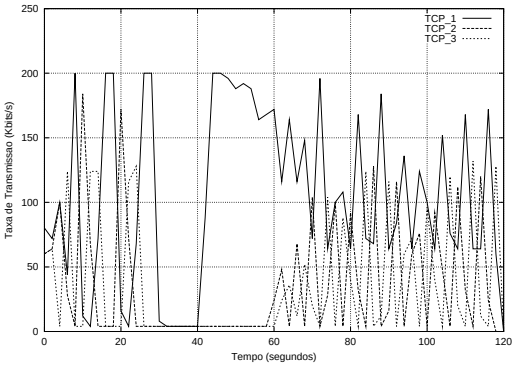
de pacotes em trânsito. Com isso, as filas de pacotes resultam maiores (vide critério para determinação do tamanho das filas na Subseção 4.1.1, e na Seção 4.3 os valores de fato adotados) e as perdas de pacotes demoram bem mais a se refletir na transmissão, levando a oscilações mais suaves. Estas variações agudas em cenários com gargalos extremos (transição de 100Mbits/s ou mais para 200Kbits/s) podem ser pioradas, também, em decorrência da política de fila *drop-tail*, adotada nos experimentos.

Nota-se que apesar das oscilações, para um gargalo de 200Kbits/s (Figura 5.6.(a)), cada um dos fluxos recebe aproximadamente 1/3 da banda, ou seja, 66Kbits/s. Este também é o caso para um gargalo de 500Kbits/s (Figura 5.6.(b)); o PePcc consome um valor bem próximo ao desejado, que é 166Kbits/s ou 1/3 da banda. No cenário com gargalo de 1Mbits/s (Figura 5.6.(c)), entre 20 e 50 segundos de simulação, os fluxos TCP_1 e TCP_2 dividem a largura de banda com pequena, mas notável, vantagem sobre o PePcc; a partir de 50 segundos, os fluxos passam a compartilhar o gargalo com taxas bem próximas dos 333Kbits/s almejados. Para 8Mbits/s, constatou-se que, na média, os fluxos utilizaram a mesma largura de banda, o que foi confirmado pelas taxas médias registradas para cada fluxo: 3,6Mbits/s, 1,63Mbits/s e 2,54Mbits/s para os fluxos PePcc, TCP_1 e TCP_2, respectivamente (a taxa média foi obtida dividindo-se a quantidade de bits enviados pela duração do fluxo, igual a 120 segundos para os TCPs e 110 para o PePcc). A taxa ideal, correspondente a 1/3 da largura de banda, é de 2,6Mbits/s. Algo similar pode ser dito do cenário com gargalo de 64Mbits/s, onde as taxas médias registradas foram de 19,93Mbits/s , 18,20Mbits/s e 22,11Mbits/s para os fluxos PePcc, TCP_1 e TCP_2, respectivamente. A taxa ideal, nesse caso, é de 21,3Mbits/s.

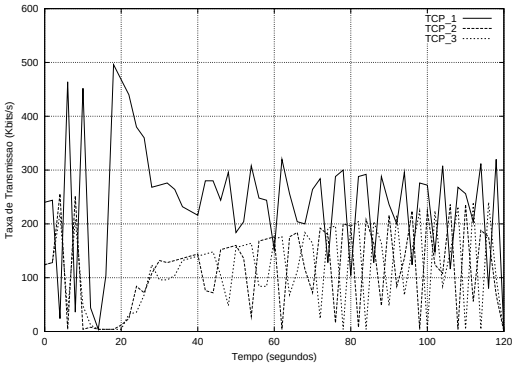
Uma vez demonstrada a divisão equilibrada da largura de banda entre os diferentes fluxos, para criar um termo de comparação, os experimentos foram executados substituindo-se o fluxo multicast por outro fluxo TCP. Isto possibilitou observar como o TCP se comporta sozinho nas mesmas condições. Os resultados são ilustrados nas Figuras 5.7.(a), 5.7.(b), 5.7.(c), 5.7.(d) e 5.7.(e), para os gargalos já definidos.

Confirmou-se que em todos os casos com TCP apenas, o mesmo executa uma alocação justa e eficiente da largura de banda disponível. Adicionalmente, verificou-se que nos gráficos para enlaces de 1Mbits/s, 8Mbits/s e 64Mbits/s as oscilações na taxa são menores do que nos gargalos com 200Kbits/s e 500Kbits/s.

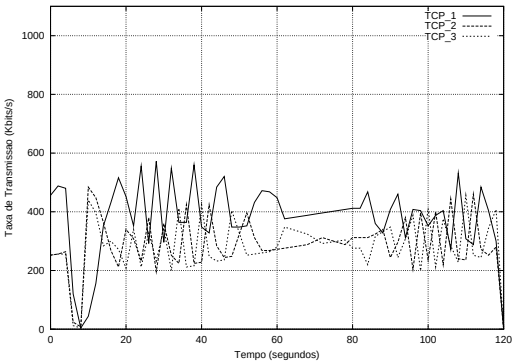
Comparando-se os resultados dos experimentos envolvendo ambos PePcc e TCP com os experimentos envolvendo apenas TCP, nota-se que, quando o gargalo é 8Mbits/s ou 64Mbits/s, com o PePcc e TCP as taxas dos receptores flutuam e passam a maior parte do tempo acima ou abaixo do ponto que determina a alocação justa (equivalente a 1/3 da banda disponível, nesse caso). Isto se deve, possivelmente, a uma perda de "responsividade" a congestionamento por parte do PePcc. Apesar disso, o PePcc apresentou resultados geralmente satisfatórios pois compartilhou adequadamente os enlaces com os fluxos TCP, consumido largura de banda próxima ao 1/3 desejado.



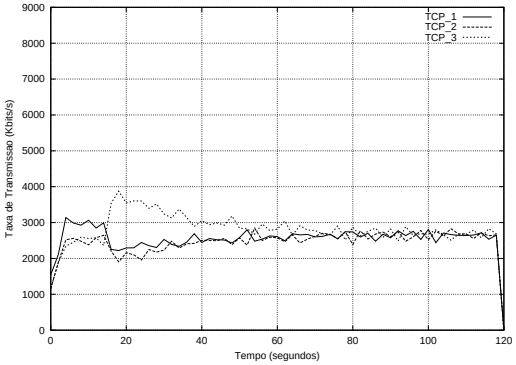
(a) 200Kbps



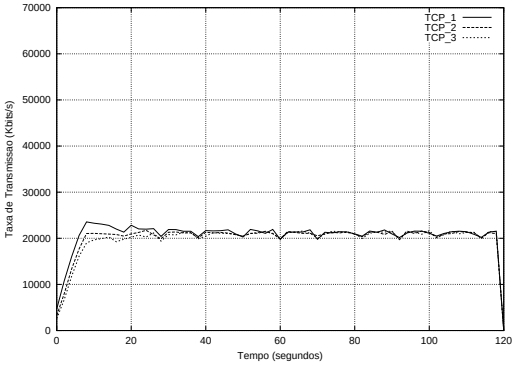
(b) 500Kbps



(c) 1Mbps



(d) 8Mbps



(e) 64Mbps

FIGURA 5.7 – Experimento 5.5 de justiça simples, com 3 fluxos TCP para as cinco larguras de banda previstas para o gargalo, com topologia da Figura 4.6

5.6 Justiça em TCP com Alta Multiplexação

O objetivo deste experimento é verificar o comportamento do PePcc, particularmente quanto à justiça, na presença de múltiplos fluxos TCP (vide Subseção 4.3.6). O comportamento esperado nos três experimentos é que a sessão PePcc continue recebendo sua porção justa (e apenas ela), à medida que o número de fluxos TCP aumenta.

Os gráficos da Figura 5.8 apresentam as taxas registradas nos receptores, o que na topologia empregada (Figura 4.6) representa a alocação de largura de banda no enlace gargalo. Foram escolhidas, arbitrariamente, as larguras de banda e número de fluxos TCP para este experimento de forma a manter certa proporcionalidade entre capacidade e número de fluxos.

As Figuras 5.8.(a), 5.8.(b) e 5.8.(c) apresentam, respectivamente, larguras de banda no gargalo iguais a 500Kbits/s, 1Mbits/s e 8Mbits/s, e 3, 4 e 6 fluxos TCP como tráfego adicional ao fluxo PePcc.

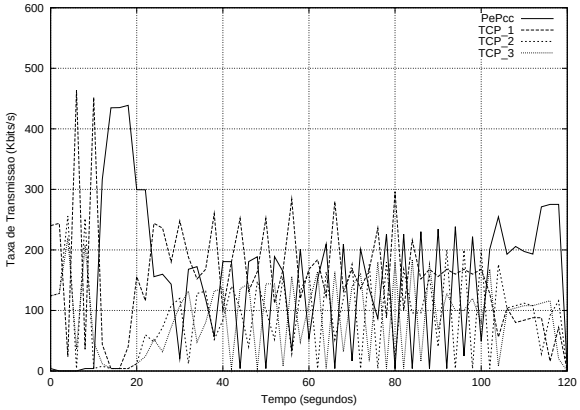
Os comportamentos exibidos pelos fluxos PePcc e TCP correspondem ao esperado; há certa flutuação nas taxas, cuja causa já foi comentada nas seções anteriores. Analisando-se os gráficos nas Figuras 5.8.(a), 5.8.(b) e 5.8.(c), e os arquivos de rastro, obteve-se as seguintes taxas médias registradas para cada fluxo:

- na Figura 5.8.(a) para 4 fluxos e 500Kbits/s de largura de banda no enlace gargalo, os fluxos médios foram de 162Kbits/s, 155Kbits/s, 91Kbits/s e 87Kbits/s para os fluxos PePcc, TCP_1, TCP_2 e TCP_3;
- na Figura 5.8.(b), para 5 fluxos e 1Mbits/s de largura de banda no enlace gargalo, os fluxos médios foram de 280Kbits/s, 140Kbits/s, 180Kbits/s, 160Kbits/s e 160Kbits/s para os fluxos PePcc, TCP_1, TCP_2, TCP_3 e TCP_4, e
- na Figura 5.8.(c), para 7 fluxos e 8Mbits/s de largura de banda no enlace gargalo, os fluxos médios foram de 2.770Kbits/s, 810Kbits/s, 610Kbits/s, 1.220Kbits/s, 600Kbits/s, 850Kbits/s e 840Kbits/s para os fluxos PePcc, TCP_1, TCP_2, TCP_3, TCP_4, TCP_5 e TCP_6.

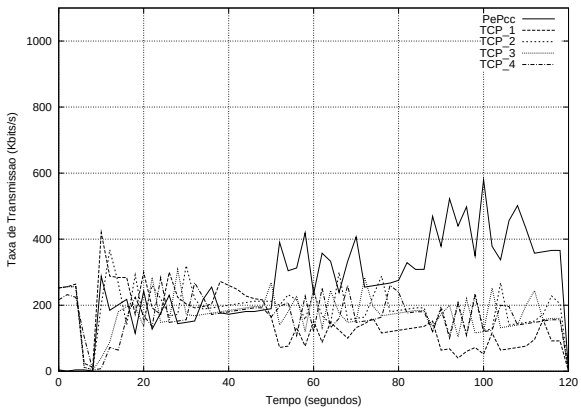
Analisando-se os valores acima, nota-se que há uma alocação adequada de largura de banda entre os fluxos no cenário com gargalo de 500Kbits/s, "razoável" no caso de 1Mbits/s e inapropriada no caso de 8Mbits/s. O PePcc parece consumir maior largura de banda do que deveria; no caso com 8Mbits/s de gargalo, PePcc gera uma taxa (2.770Kbits/s) que é mais que duas vezes maior do que a alocação justa (1,14Mbits/s). Conforme já explicado, esta diferença pode ser atribuída, entre outros fatores, pelo maior número de retransmissões.

Para efeito de comparação, o fluxo PePcc foi substituído por um fluxo TCP e o experimento re-executado. Os resultados são apresentados nas Figuras 5.9.(a), 5.9.(b) e 5.9.(c).

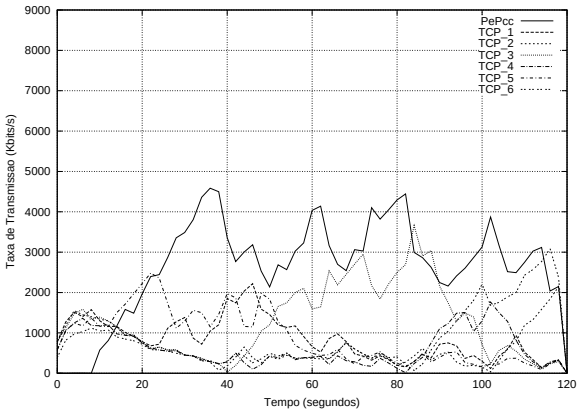
O comportamento apresentado pelo PePcc mediante múltiplos fluxos TCP é proporcionalmente semelhante, porém com menos oscilações, ao observado nas Figuras 5.9, em que só concorrem fluxos TCP.



(a) gargalo de 500Kbps, 1 fluxo PePcc e 3 TCP

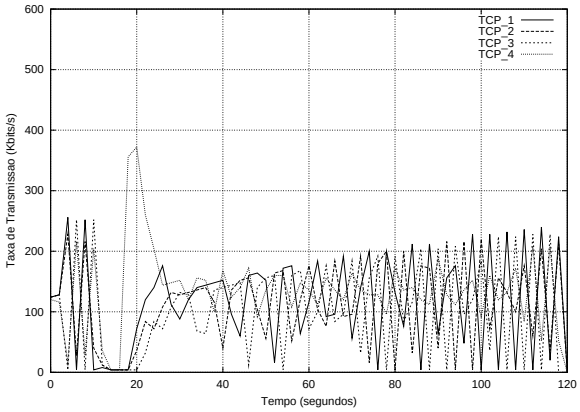


(b) gargalo de 1Mbps, 1 fluxo PePcc e 4 TCP

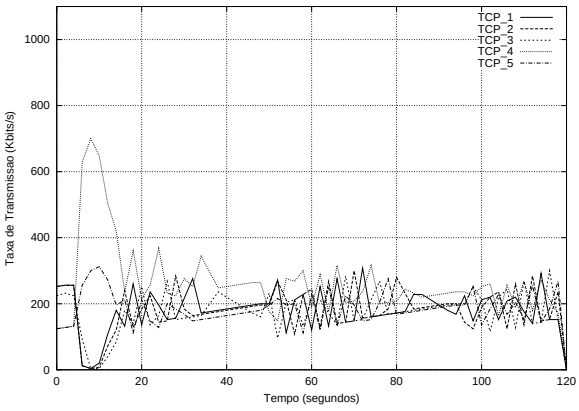


(c) gargalo de 8Mbps, 1 fluxo PePcc e 6 TCP

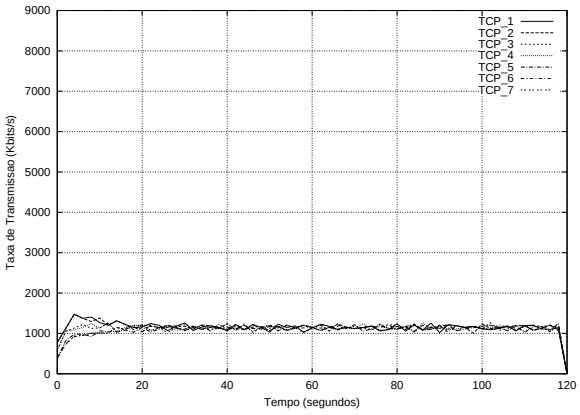
FIGURA 5.8 – Experimento 5.6 de justiça com alta multiplexação (topologia na Figura 4.6)



(a) gargalo de 500Kbps e 4 fluxos TCP



(b) gargalo de 1Mbps e 5 fluxos TCP



(c) gargalo de 8Mbps e 7 fluxos TCP

FIGURA 5.9 – Experimento 5.6 de justiça com alta multiplexação, variando-se o número de fluxos TCP (4 a 7) à medida que a banda do gargalo aumenta (vide Figura 4.6)

5.7 Justiça entre Sessões

O objetivo deste experimento é verificar o compartilhamento justo de recursos entre fluxos de um mesmo PePcc. Conforme descrito na Subseção 4.3.7, o experimento está subdividido em duas partes. Na primeira parte, dois fluxos PePcc transmitem, cada, para um receptor, compartilhando um gargalo (topologia em estrela dupla, conforme Figura 4.6). O comportamento esperado é que a largura de banda disponível seja justamente dividida entre os dois fluxos PePcc, atendendo ao requisito de justiça entre sessões.

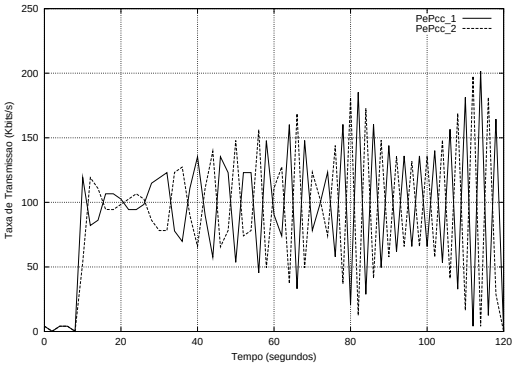
Analisando as Figuras 5.10.(a), 5.10.(b), 5.10.(c), 5.10.(d) e 5.10.(e) e os arquivos de rastro, verifica-se que em geral os dois fluxos PePcc mantiveram um comportamento **justo** em relação um ao outro. Esta observação é sustentada pela análise da taxa média associada aos fluxos PePcc. As larguras de banda no enlace gargalo e as taxas de transmissão médias para os fluxos PePcc_1 e PePcc_2 foram respectivamente:

- na Figura 5.10.(a), para gargalo de 200Kbits/s, 99Kbits/s e 88Kbits/s;
- na Figura 5.10.(b), para gargalo de 500Kbits/s, 246Kbits/s e 221Kbits/s;
- na Figura 5.10.(c), para gargalo de 1Mbits/s, 460Kbits/s e 450Kbits/s;
- na Figura 5.10.(d), para gargalo de 8Mbits/s, 3.880Kbits/s e 3.400Kbits/s e
- na Figura 5.10.(e), para gargalo de 64Mbits/s, 30.140Kbits/s e 24.870Kbits/s.

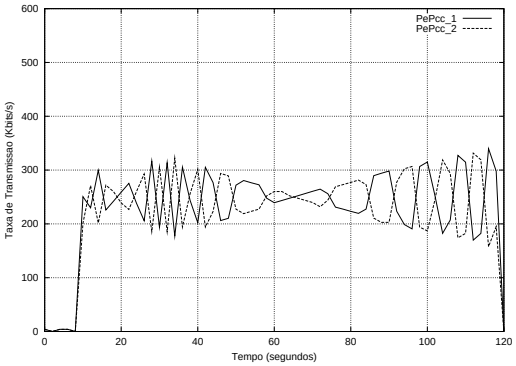
Na segunda parte do experimento, adiciona-se dois fluxos TCP ao cenário anterior (segundo a topologia na Figura 4.6): dois fluxos TCP iniciam e após 10 segundos, dois fluxos PePcc, cada um para um receptor. Neste experimento, é esperado que além dos fluxos PePcc atuarem de maneira justa entre si, também dividam a largura de banda disponível de forma justa com os fluxos TCP.

Analisando os gráficos nas Figuras 5.11.(a), 5.11.(b), 5.11.(c), 5.11.(d) e 5.11.(e), verifica-se que o PePcc apresentou o comportamento desejado, ou seja, os fluxos PePcc mantiveram o comportamento justo entre si e em relação aos fluxos TCP. Tal é confirmado pelas taxas médias de transmissão ao longo da sessão. As taxas de transmissão médias observadas para os fluxos PePcc_1, PePcc_2, TCP_1 e TCP_2 foram respectivamente:

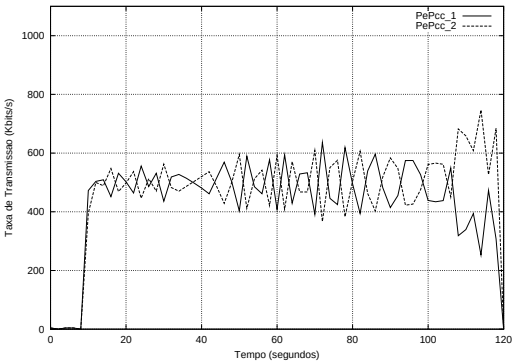
- na Figura 5.11.(a), para gargalo de 200Kbits/s, 49Kbits/s, 48Kbits/s, 49Kbits/s e 56Kbits/s;
- na Figura 5.11.(b), para gargalo de 500Kbits/s, 115Kbits/s, 142Kbits/s, 131Kbits/s e 118Kbits/s;
- na Figura 5.11.(c), para gargalo de 1Mbits/s, 290Kbits/s, 210Kbits/s, 230Kbits/s e 210Kbits/s;
- na Figura 5.11.(d), para gargalo de 8Mbits/s, 2.050Kbits/s, 1.400Kbits/s, 1.160Kbits/s e 3.180Kbits/s, e



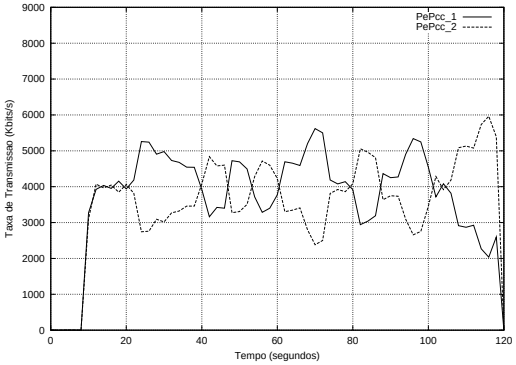
(a) gargalo de 200Kbps



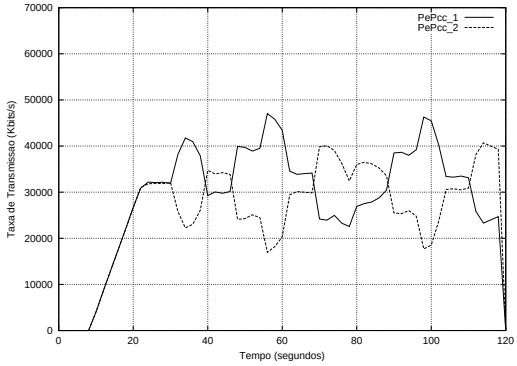
(b) gargalo de 500Kbps



(c) gargalo de 1Mbps

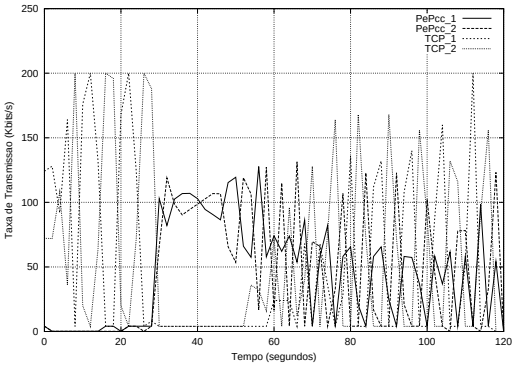


(d) gargalo de 8Mbps

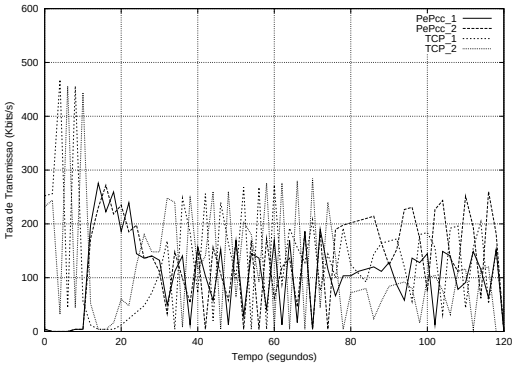


(e) gargalo de 64Mbps

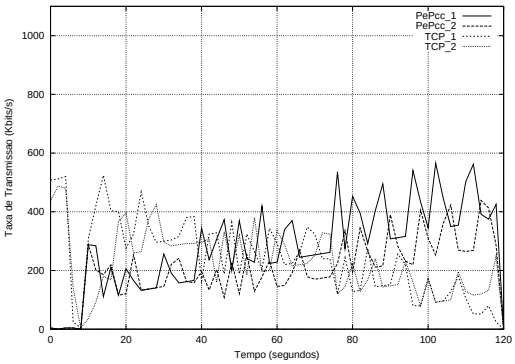
FIGURA 5.10 – Experimento 5.7 - primeira parte, que avalia justiça apenas entre fluxos PePcc



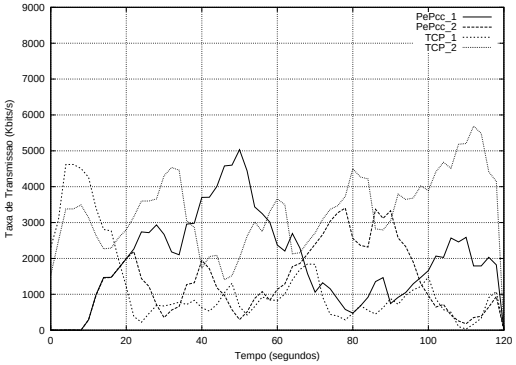
(a) 200Kbps



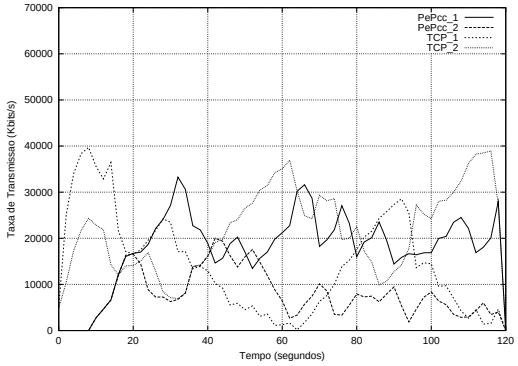
(b) 500Kbps



(c) 1Mbps



(d) 8Mbps



(e) 64Mbps

FIGURA 5.11 – Experimento 5.7 - segunda parte, que avalia justiça entre fluxos TCP e PePcc

- na Figura 5.11.(e), para gargalo de 64Mbits/s, 18.750Kbits/s, 8.150Kbits/s, 13.860Kbits/s e 22.640Kbits/s.

Com o objetivo de observar como o TCP reagiria sozinho nestas condições, este experimento com quatro fluxos foi repetido sem a presença dos fluxos PePcc (foram substituídos por fluxos TCP). As Figuras 5.12.(a), 5.12.(b), 5.12.(c), 5.12.(d) e 5.12.(e) ilustram os resultados.

Primeiramente, os gráficos comprovam que os cenários com enlaces gargalo de capacidade menor tendem a provocar oscilações na taxa (tais oscilações são, portanto, independentes do PePcc). Segundo, nota-se que apenas com fluxos do mesmo protocolo, o TCP oferece uma divisão estável e eficiente da largura de banda do enlace gargalo, para os casos em que esta foi igual ou superior a 1Mbits/s. Há duas formas de se comparar esses resultados: com o cenário contendo apenas fluxos PePcc (gráficos na Figura 5.10), ou com o cenário contendo fluxos PePcc e TCP (gráficos na Figura 5.11). No primeiro caso, assim como o TCP, o PePcc mantém uma alocação justa e eficiente; no segundo, a alocação é justa, porém as taxas registradas nos receptores constantemente oscilam. A menor estabilidade com a presença do PePcc pode ser explicada pela falta de "responsividade" do esquema, em função da restrição de reconhecimento proporcionada pelo esquema de *polling*, e particularmente mediante um elevado produto entre latência e largura de banda (entre 250 e 330 pacotes, conforme explicado na Seção 5.6, na página 4.3.6).

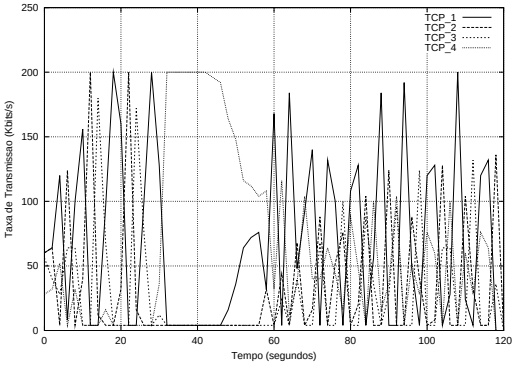
5.8 Justiça Heterogênea

O objetivo deste experimento é verificar se o comportamento do PePcc não é injusto em configurações heterogêneas, conforme descrito na Subseção 4.3.8. Os experimentos foram executados com dois fluxos TCP: TCP_1, "longo", representando $S_1 \rightarrow R_1$, e TCP_2, "curto", $S_2 \rightarrow R_2$, além de um fluxo PePcc "longo", $S_3 \rightarrow \{R_3, R_4\}$ (vide topologia na Figura 4.7, na página 58).

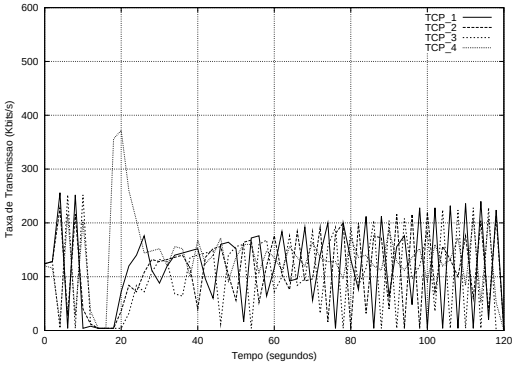
O comportamento desejável é que o fluxo PePcc ajuste sua taxa de envio ao seu receptor mais lento (R_3), uma vez que o esquema empregado é de taxa única. Além disso, a taxa média da sessão PePcc deveria ser semelhante à taxa média obtida pelo fluxo TCP $S_1 \rightarrow R_1$, rota sobre o enlace de maior latência (100ms).

Observando-se as Figuras 5.13.(a), 5.13.(b), 5.13.(c), 5.13.(d) e 5.13.(e), nota-se em geral dois fluxos com taxa "menor", na parte de baixo do gráfico, e um fluxo com taxa "maior", na parte superior do gráfico. A linha da parte de cima representa o fluxo TCP $S_2 \rightarrow R_2$, enquanto as linhas de baixo correspondem aos fluxos TCP $S_1 \rightarrow R_1$ e PePcc $S_3 \rightarrow \{R_3, R_4\}$. Os resultados apresentados pelo PePcc se assemelham em muito ao fluxo TCP; portanto, os resultados obtidos com o PePcc nesse experimento são exatamente os esperados.

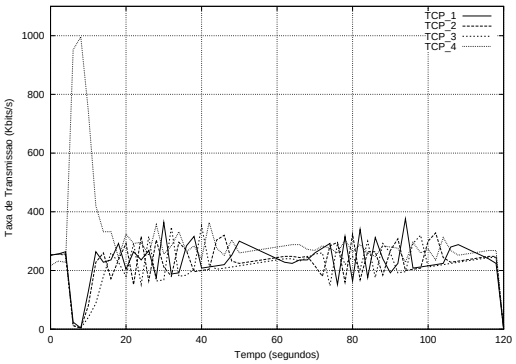
Os fluxos PePcc e TCP_2 possuem valores altos de RTT, devido à alta latência do enlace por eles utilizado. Isto faz com que estes dois fluxos subam lentamente em direção a sua taxa de transmissão ideal. Este comportamento faz com que o fluxo TCP_1, que trafega num enlace de baixa latência, inicie ocupando uma grande fração da largura de banda disponível. A medida que os outros dois fluxos (PePcc e TCP_2) conseguem lentamente elevar sua taxas de envio, o fluxo TCP_1 reduz a sua, num comportamento que tende ao equilíbrio entre os três fluxos.



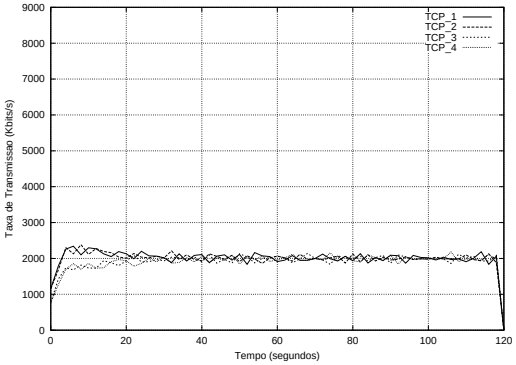
(a) gargalo de 200Kbps



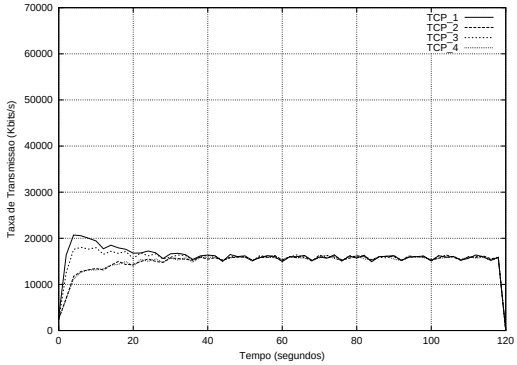
(b) gargalo de 500Kbps



(c) gargalo de 1Mbps

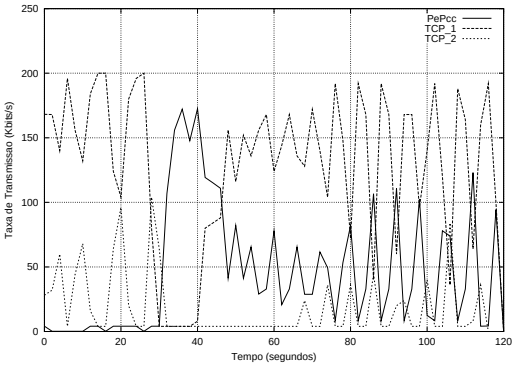


(d) gargalo de 8Mbps

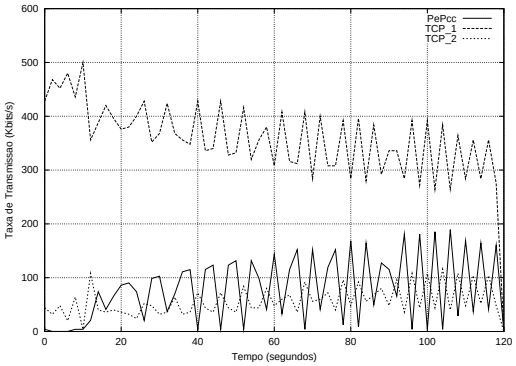


(e) gargalo de 64Mbps

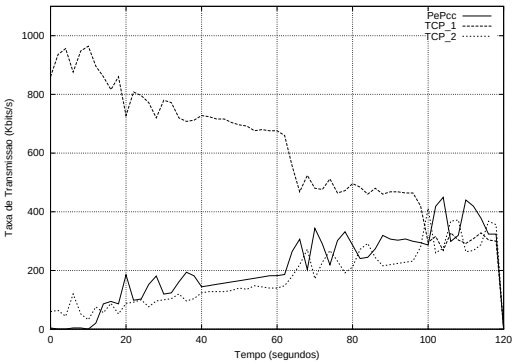
FIGURA 5.12 – Experimento 5.7 - segunda parte, que avalia justiça apenas entre fluxos TCP



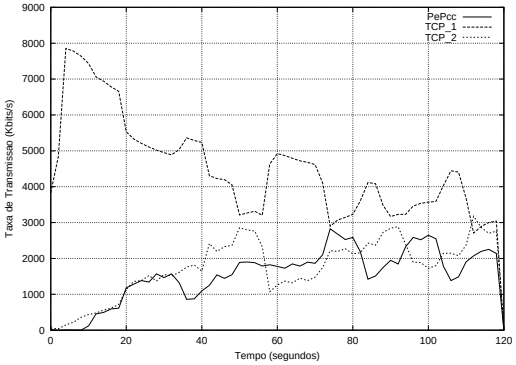
(a) gargalo de 200Kbps



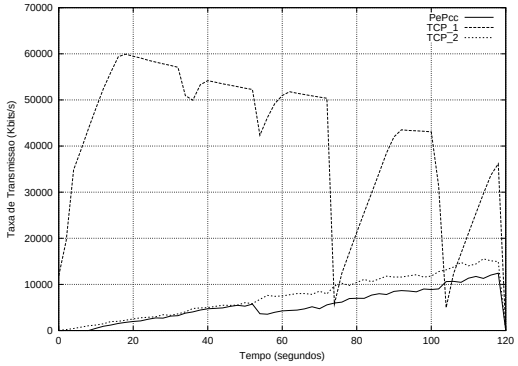
(b) gargalo de 500Kbps



(c) gargalo de 1Mbps



(d) gargalo de 8Mbps



(e) gargalo de 64Mbps

FIGURA 5.13 – Experimento 5.8 de justiça heterogênea para PePcc e TCP

Para se comparar os resultados do PePcc com TCP, os experimentos foram repetidos nas mesmas condições dos experimentos anteriores, porém contando apenas com os dois fluxos TCP. Comparando os gráficos da Figura 5.13 com os gráficos da Figura 5.14, que possui apenas fluxos TCP, observa-se que o comportamento dos fluxos TCP é bastante semelhante, tanto na presença de fluxos PePcc (porém com maior instabilidade) como na ausência destes. A grande diferença é que o fluxo TCP_2 obtém uma fatia de banda disponível ligeiramente maior quando existe o fluxo multicast. Isto é, a queda da taxa de transmissão do TCP_1 devido à presença do PePcc é aproveitada pelo TCP_2 para elevar a sua taxa de transmissão.

5.9 Aumento da Verificação de Sanidade

O objetivo deste experimento é estender o experimento de verificação da sanidade do protocolo, conforme descrito na Subseção 4.3.9. O PePcc inicia uma sessão transmitindo para os 128 receptores, através de um roteador que está situado na raiz de uma árvore multicast de 8 níveis, e cujos folhas são os receptores. Após 30 segundos, um fluxo CBR é inserido entre o remetente e o roteador folha (enlace gargalo), com taxa de 500Kbits/s. Após 60 segundos, o fluxo CBR tem sua taxa diminuída para 250Kbits/s.

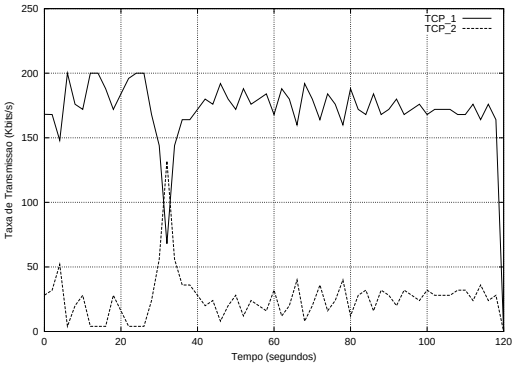
O comportamento desejado é que a taxa média do fluxo PePcc seja semelhante ao obtido no primeiro experimento de verificação de sanidade, visto na Seção 4.3.1.

A Figura 5.15 apresenta a taxa de transmissão no remetente. Observa-se na figura que a taxa de envio do remetente PePcc sobe, até o início um fluxo CBR numa taxa de 500Kbits/s, no instante 30s. Neste momento, o fluxo PeP interrompe sua trajetória ascendente e compartilha a banda disponível com o fluxo CBR, até que este reduz sua taxa para 250 Kbits/s, no momento 90s. O fluxo PePcc então sobe sua taxa para um valor proporcional ao que o fluxo CBR diminuiu. Quando este último encerra sua transmissão, no momento 90s, o PeP novamente volta subir.

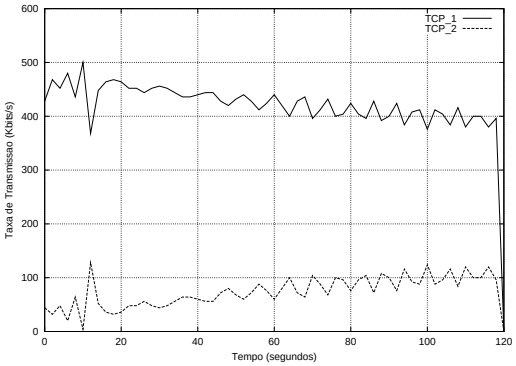
Comparando os resultados da Figura 5.15 com a Figura 5.1 (página 62), nota-se que eles exibem o mesmo padrão de comportamento (correto). Existem algumas diferenças importantes, no entanto: por exemplo, neste experimento, a taxa de transmissão do remetente cresce muito mais lentamente do que no outro; a taxa do PePcc não volta a subir significativamente entre 120s e 150s, quando o fluxo CBR cessa. As razões para essas diferenças são o aumento de latência (RTT passa de 60ms para 120ms) e de receptores (de 1 para 128). Como há mais receptores a consultar via *polling*, e a latência é maior, o protocolo perde em "responsividade". Adicionalmente, o aumento de receptores faz com que aumentem as perdas no enlace engarrafado. Como conseqüência, aumenta a instabilidade dos fluxos e a taxa de transmissão do PePcc sobe mais lentamente.

5.10 Curva de Perda de Taxa

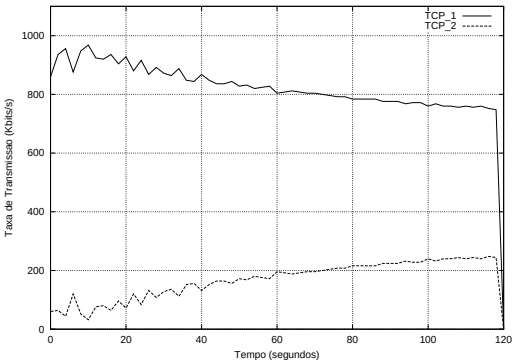
O objetivo deste experimento é observar como se comporta o PePcc, mais precisamente, a queda de seu *throughput* perante seis diferentes combinações de latências e probabilidades de perdas, como descrito na Subseção 4.3.10. Um remetente multicast transmite para um único



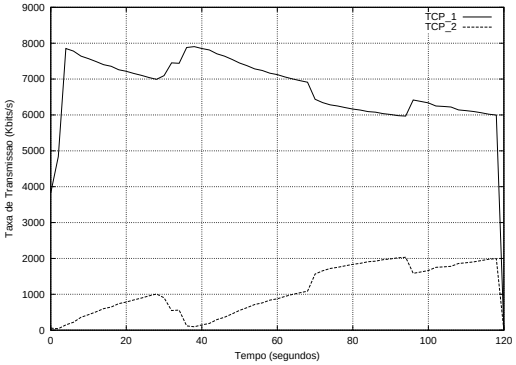
(a) 200Kbps



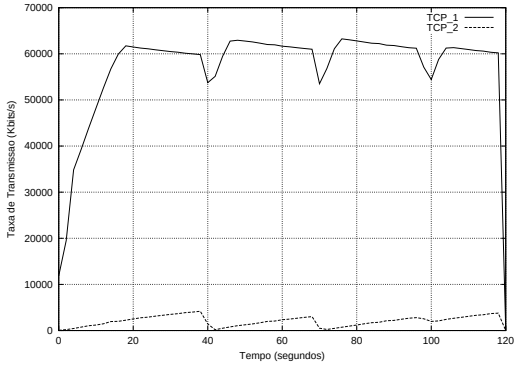
(b) 500Kbps



(c) 1Mbps



(d) 8Mbps



(e) 64Mbps

FIGURA 5.14 – Experimento 5.8 de justiça heterogênea para TCP apenas

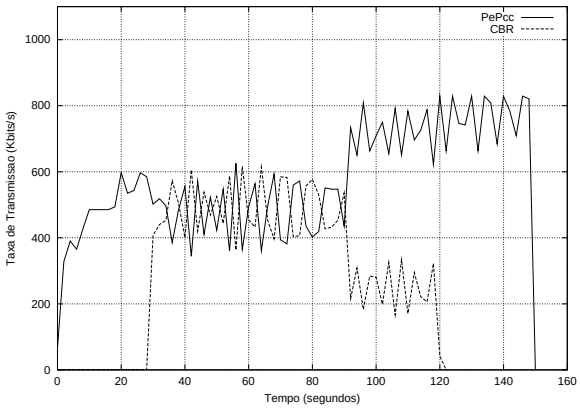


FIGURA 5.15 – Experimento 5.9 de aumento da verificação de sanidade para PePcc e CBR

receptor. Espera-se como comportamento uma queda de taxa que seja proporcional à relação entre a latência e a probabilidade de perda.

Os gráficos na Figura 5.16 apresentam resultados com variações de probabilidade de perda (3%, 7% e 10%) e latência de 10ms, enquanto os gráficos da Figura 5.17 apresentam resultados correspondentes, considerando latência de 100ms.

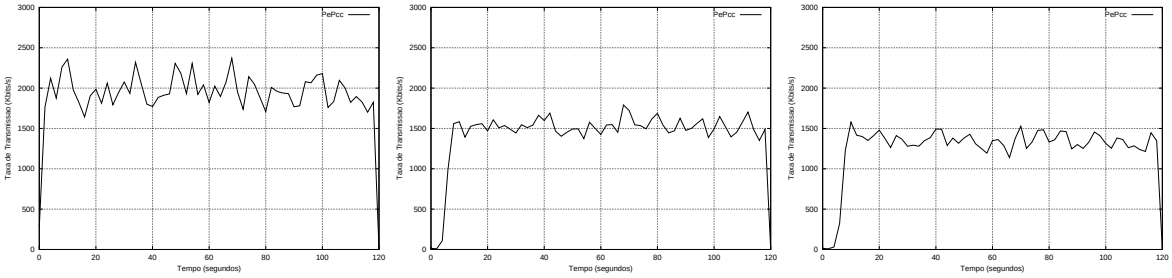
A comparação entre as Figuras 5.16.(a), 5.16.(b) e 5.16.(c) quantifica o impacto das perdas aleatórias na taxa do PePcc. As taxas médias aproximadas são1.892Kbits/s, 1.412Kbits/s e 1.238Kbits/s, para 3%, 7% e 10% de perdas. Comparação idêntica pode ser feita para experimentos com latência de 100ms, nas Figuras 5.17.(a), 5.17.(b) e 5.17.(c): taxas de 306Kbits/s, 179Kbits/s e 151Kbits/s, para probabilidades de perdas 3%, 7% e 10%.

De outra forma, comparando-se as Figuras 5.16.(a) e 5.17.(a), 5.16.(b) e 5.17.(b), e 5.16.(c) e 5.17.(c), isola-se o impacto da latência. Na Figura 5.16.(a), a taxa de transmissão foi de 1892 Kbits/s. Aumentando a latência em 10 vezes, como visto na Figura 5.17.(a), a taxa de transmissão cai para 306Kbits/s, ou seja 6 vezes. Para as probabilidades de perdas de 7 e 10%, a taxa cai 7 e 8 vezes respectivamente, como pode ser visto comparando-se a Figura 5.16.(b) com a Figura 5.17.(b), e a Figura 5.16.(c) com a Figura 5.17.(c).

5.11 Agregação Correta de Perda

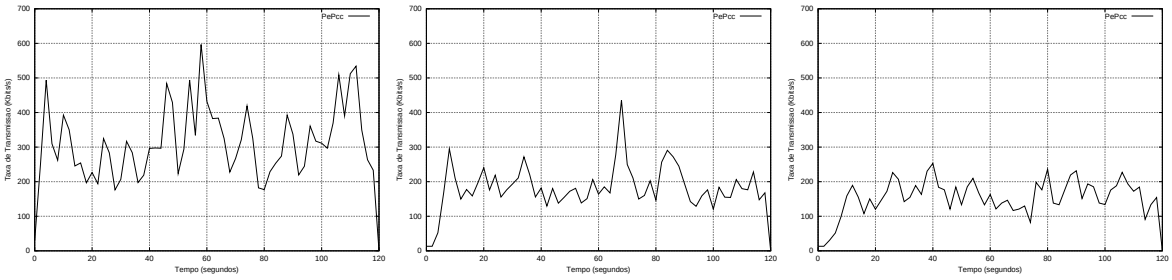
O objetivo deste experimento é verificar se o PePcc agrega as perdas de forma correta, conforme descrito na Subseção 4.3.11. No experimento, um remetente multicast transmite para cinco receptores. O comportamento esperado é que a taxa de envio do remetente seja ajustada para o receptor que mais sofre perdas, ou seja, R_5, cuja rota entre ele e o remetente apresenta probabilidade de perda próxima a 15%.

Na Figura 5.18.(a) pode ser vista a taxa de envio do remetente multicast através dos enlaces com taxas de perdas decrescentes, conforme Figura 4.8. No sentido de comparar os



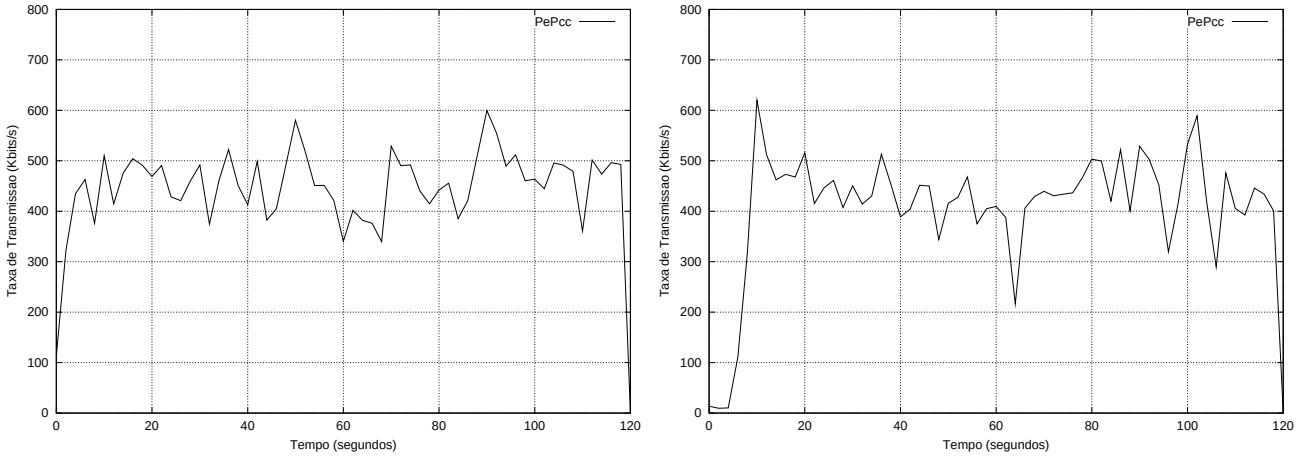
(a) perda de 3 por cento (b) perda de 7 por cento (c) perda de 10 por cento

FIGURA 5.16 – Experimento 5.10 de curva de perda de taxa - latência de 10ms



(a) perda de 3 por cento (b) perda de 7 por cento (c) perda de 10 por cento

FIGURA 5.17 – Experimento 5.10 de curva de perda de taxa - latência de 100ms



(a) perdas distribuídas nos enlaces

(b) perdas concentradas no último enlace

FIGURA 5.18 – Experimento 5.11 de agregação correta de perda

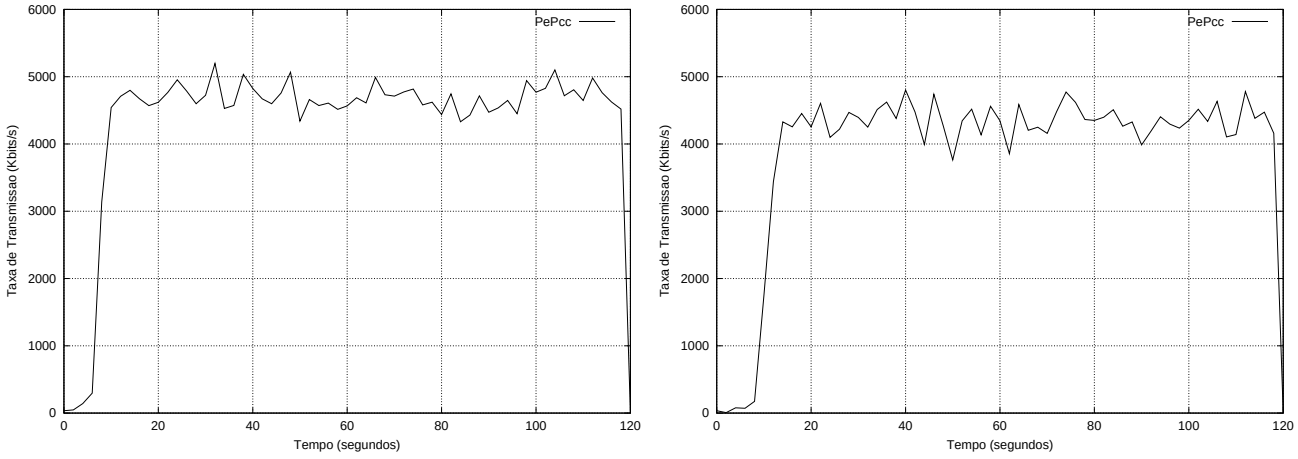
resultados obtidos, o mesmo experimento foi repetido para uma topologia similar: substituindo as perdas dos enlaces L_1, L_2, L_3, L_4, L_5 por um único enlace com perda , L_5, tendo o valor de sua probabilidade de perda representando o somatório das perdas do experimento (a), ou seja 15%. Comparando os gráficos nas Figuras 5.18.(a) e 5.18.(b), nota-se que as taxas de envio são semelhantes, entre 400 e 500Kbits/s. Isto significa que o PePcc obteve os resultados desejados em relação a correlação de perda: ajustou sua taxa de envio para o receptor R_5, que nos dois casos, é o que sofre maiores perdas.

5.12 Queda até Zero[2]

O objetivo deste experimento é verificar se o PePcc não reduz, indevidamente, sua taxa de transmissão até zero quando ocorrem perdas em todas rotas até os receptores, conforme descrito na Subseção 4.3.12. O comportamento desejado, em ambos experimentos, é que a taxa da sessão multicast não caia a zero. Adicionalmente, no segundo experimento (b), o PePcc deve reduzir sua taxa de envio para compensar o receptor com maior perda.

No primeiro experimento, onde um protocolo multicast transmite para 50 receptores, cada um com uma probabilidade de perda não correlata de 5% nos enlaces, pode-se observar na Figura 5.19.(a), que apesar de estarem ocorrendo perdas em todas rotas entre o remetente e os receptores, o PePcc não reduz sua taxa em resposta a todos indicativos de perdas recebidos, o que levaria a sua taxa de transmissão cair a zero. Ao contrario, o PePcc mantém uma janela de congestionamento independente para cada receptor, conforme visto na Subseção 3.2.2 e o

[2] *drop-to-zero.*



(a) todos receptores com mesma taxa de perda

(b) 1 receptor com taxa maior

FIGURA 5.19 – Experimento 5.12 para assegurar que não ocorre a queda até zero, avaliando-se duas topologias em estrela

mecanismo regula o fluxo de transmissão utilizando parâmetros que não levem a uma taxa demasiadamente baixa, como definido na Seção 3.3.

No segundo experimento, onde R_{50} experimenta a probabilidade de perda igual a 20%, observa-se na Figura 5.19.(b) que o protocolo mantém o mesmo comportamento da situação anterior, isto é, não cai sua taxa de transmissão a zero.

Comparando-se os dois experimentos, percebe-se na Figura 5.19.(b) que o remetente PePcc mantém uma taxa de transmissão um pouco menor do que visto na Figura 5.19.(a). Isto ocorre devido ao remetente PePcc, conforme Figura 5.19.(b), reduzir sua taxa de transmissão para a capacidade do receptor que sofre mais perdas, ou seja, aquele que está situado no enlace com probabilidade de perda de 20%.

5.13 Perda Não Correlata

O objetivo deste experimento é verificar se o PePcc reduz adequadamente sua taxa de envio, conforme descrito na Subseção 4.3.13. No experimento, uma sessão multicast transmite para 20 receptores com local de perda do receptor alternando entre os 20 receptores. O comportamento esperado é que o PePcc mantenha sua taxa de envio mais ou menos constante, de acordo com o atual receptor de maior perda, que sempre observará 5%.

Na Figura 5.20, é possível perceber que apesar do receptor com perda de 5% mudar constantemente de local, o mecanismo de controle de congestionamento do PePcc suprime com sucesso novas reações a congestionamento (encurtando a janela de congestionamento). O mecanismo

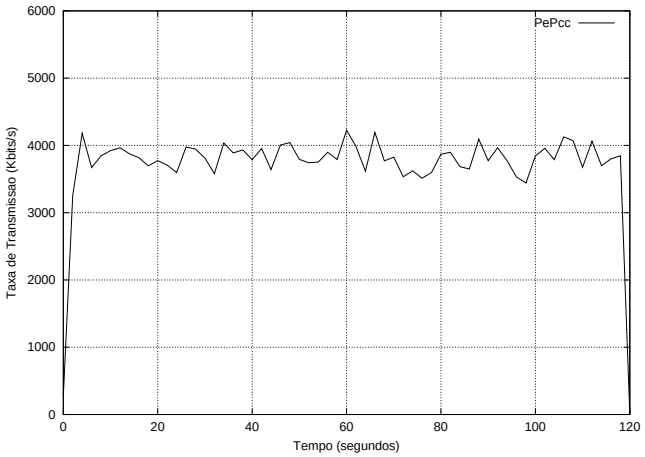


FIGURA 5.20 – Experimento 5.13 para perda não correlata

para isso, explicado na Seção 3.2, suprime novas reações com base no RTT entre o remetente e os receptores, mantendo a taxa entre 3.000 Kbits/s e 3.500 Kbits/s, que é a taxa observada pelo receptor cujo enlace sofre a maior perda.

Capítulo 6

Considerações Finais

Controle de congestionamento é importante para se manter a estabilidade da rede, pois regula a taxa de envio das fontes transmissoras de forma a evitar um colapso (todos pacotes são descartados) em situações de grande demanda. Mecanismos de controle de congestionamento lidam com a sobrecarga nas filas dos roteadores e buscam o compartilhamento justo da largura de banda disponível entre os fluxos nos enlaces da rede. O principal protocolo que incorpora controle de congestionamento é o TCP.

Existem diversas versões de TCP, de acordo com variações no algoritmo de controle de congestionamento. Apesar destes esforços, ainda há problemas a resolver em controle de congestionamento, dada a evolução tecnológica na área de redes de computadores. Particularmente, controle de congestionamento para protocolos multicast oferece um grande desafio, pois os problemas e dificuldades associadas são bem maiores do que com protocolos unicast. Desde 1999, o grupo de pesquisa em redes de computadores do PIPCA tem investigado e desenvolvido protocolos multicast para transmissão confiável em larga escala. Em [4], foi proposto um conjunto de modelos de protocolos multicast cujos mecanismos se baseiam na técnica de *polling*. Em [14], foi investigado o uso de controle de congestionamento explícito em protocolos multicast confiáveis.

Esta dissertação dá continuidade a esse trabalho, a medida que apresenta resultados obtidos através de um extenso conjunto de experimentos com um mecanismo de controle de congestionamento para protocolos multicast, definido, porém não avaliado, em [14]. Para tal, foi realizado um estudo sobre o estado da arte em protocolos multicast e mecanismos de controle de congestionamento em tais protocolos. Este estudo resultou no Capítulo 2, sobre congestionamento e multicast, e Capítulo 3 , sobre o que foi aqui denominado PePcc: o protocolo PeP acrescido de um mecanismo de controle de congestionamento. A seguir, foram estudadas formas de se avaliar mecanismos de controle de congestionamento em protocolos multicast, o que incluiu busca na literatura e o estudo da ferramenta ns. Como base na literatura, particularmente em [10], foram definidos procedimentos para avaliação, resultando em uma metodologia bem definida e claramente apresentada, constante do Capítulo 4. Esta metodologia foi então empregada no desenvolvimento de um conjunto de *scripts* de simulação, que por sua vez foram utilizados na condução de experimentos. Os resultados decorrentes das simulações permitem estimar melhor o comportamento do PePcc na Internet.

A principal contribuição dessa dissertação é a execução de um extenso conjunto de experimentos com o PePcc, permitindo a coleta e análise de resultados. Conclusões interessantes foram tiradas através da análise destes resultados, conforme discutido nas diversas seções do Capítulo 5. Entretanto, ressalte-se que as conclusões não são definitivas, por diversas razões. Primeiro, simulação se baseia em um modelo simplificado da realidade, e dependendo da qualidade do modelo de simulação, está sujeita a distorções nos resultados. No caso dos experimentos realizados embora as questões metodológicas gerais ([27]) e específicas ([10]) tenham sido observadas, o modelo de simulação do ns é limitado; por exemplo, as implementações de TCP, de roteadores e de IP multicast, são simplificadas, e exigem parâmetros globais não realísticos (como janela de congestionamento especificada em pacotes). Segundo, percebeu-se, ao implementá-la, que a metodologia sugerida em [10] apresenta problemas. Por exemplo, diversos experimentos não são suficientemente detalhados, e deixam margem a dúvidas. Terceiro, não existem formas **objetivas** e largamente aceitas de se interpretar os resultados gráficos ilustrados nos experimentos, em função da taxa de transmissão ao longo do tempo. A diferença entre resultados "bons" e "ruins" é uma questão de interpretação, dificultando a análise.

O prosseguimento deste trabalho inclui estender os experimentos para que o conjunto completo de combinações seja avaliado, incluindo melhorias no processo de simulação, que permitam acelerar as simulações e registrar um número maior de informações. Outra expansão possível do trabalho é a avaliação do mecanismo de controle de congestionamento em outros protocolos além do PeP, primeiramente nos quatro outros protocolos definidos em [4] e, após, em outros protocolos encontrados na literatura. Com base nestes resultados, poderão ser propostas melhorias nos protocolos multicast avaliados e no mecanismo de controle de congestionamento.

Bibliografia

[1] M. BARCELLOS. PRMP: A Scaleable Polling-based Reliable Multicast protocol. Ph.D. Thesis, Newcastle University, Newcastle upon Tyne, October 1998.

[2] M. BARCELLOS and P. EZHILCHELVAN. A End-to-End Reliable Multicast Protocol using Polling for Scaleability. In Proc. of IEEE INFOCOM'98, San Francisco, CA, USA, volume 3, pp. 1180-1187, April 1998.

[3] M. BARCELLOS. Redes de Computadores: Multicasting. In Proc. of SBC ERI'2000, Brazil, pp. 57-85, 2000.

[4] M. BARCELLOS e A. DETSCH. Avaliação de Desempenho de Protocolos Multicast com Conhecimento de Grupo Baseados em Polling. In Proc. of SBC SBRC'2002, Búzios, Brazil, pp. 408-423, May 2002.

[5] S. BHATTACHARYYA, D.TOWSLEY, and J.KUROSE. The Loss Path Multiplicity Problem in Multicast Congestion Control. In Proc. of IEEE INFOCOM'99, New York, NY, USA, volume 2, pp. 856-863, March 1999.

[6] S. BHATTACHARYYA, D. TOWSLEY, and J. KUROSE. A Novel Loss Indication Filtering Approach for Multicast Congestion Control. In Computer Communications, volume 24, n. 5-6, pp. 512-524, March 2001.

[7] Z. BRAUDES. Requirements for Multicast Protocols. Network Working Group, Request for Comments:1458, TASC, May, 1993.

[8] L. BRESLAU et alli. Advances in Network Simulation. In IEEE Computer, volume 33, n. 5, pp. 59-67, May 2000.

[9] J. BYERS, M. FRUMIN, G. HORN, M. LUBY, M. MITZENMACHER, A. ROETTER, and W. SHAVER. FLID-DL:Congestion Control for Layered Multicast. In Proc. of NGC'2000, Standford, CA, USA, pp. 71-81, November 2000.

[10] J. BYERS, G. HORN, M. HANDLEY, M. LUBY, W. SHAVER, and L. VICISANO. More Thoughts on Reference Simulations for Reliable Multicast Congestion Control Shemes. Notes from a meeting at Digital Fontain, August 8, 2000.

[11] A. CHAINTREAU, F.BACCELLI, and C. DIOT. Impact of Network Delay Variations on Multicast Sessions with TCP-like Congestion Control. In Proc. of IEEE INFOCOM'2001, Anchorage, Alasca, USA, volume 2, pp. 1133-1142, April, 2001.

[12] D. M. CHIU, M.KADASNKY, J.PROVINO, J.WESLEY, H-P. BISCHOF, and H.ZHU. A Congestion Control Algorithm for Tree-based Reliable Multicast Protocols. In Proc. of IEEE INFOCOM'2002, New York, NY, USA, June 2002.

[13] S. E. DEERING and D.R. CHERITON. Multicast Routing in Datagram Internertworks and Extended LANs. In ACM Trans. on Computer Systems, volume 8, n. 2, pp. 85-110, May 1990.

[14] A. DETSCH. Controle de Congestionamento com Suporte a ECN em Protocolos Multicast. Trabalho de Conclusão, UNISINOS, São Leopoldo, Dezembro 2002.

[15] K. FALL and S. FLOYD. Simulation-based Comparison of Tahoe, Reno, and SACK TCP. In ACM Computer Communications Review, Volume 26, n. 3, pp. 5-21, July 1996.

[16] K. FALL and K.VARADHAN. The ns Manual (formely ns Notes and Documentation), The VINT Project, April, 2002.

[17] S. FLOYD and K. FALL. Promoting the Use of End-to-End Congestion Control in the Internet. In IEEE\slash ACM Transactions on Networking, New York, NY, USA, volume 7, n. 4, pp. 458 - 472, 1999.

[18] S. FLOYD, M. HANDLEY, J. PADHYE, and J. WIDMER. Equation-Based Congestion Control for Unicast Applications. In Proc. of ACM SIGCOMM'2000, Stockholm, Sweden, pp. 43-56, August 2000.

[19] S. FLOYD. TCP and Explicit Congestion Notification. In ACM Computer Comunication Review, volume 24, n. 5, pp. 10-23, October 1994.

[20] S. J. GOLESTANI and S. BHATTACHARYYA. A Class of End-to-End Congestion Control Algorithms for the Internet. In Proc. of IEEE ICNP'98, Augustin, Texas, USA, pp. 137-150, October 1998.

[21] S. J. GOLESTANI and K. K. SANNANI. Fundamental Observations on Multicast Congestion Control in the Internet. In Proc. of IEEE INFOCOM'99, New York, NY, USA, volume 2, pp. 990-1000, March 1999.

[22] G. J. HOLZMANN. Design and Validation of Computer Protocols. Prentice Hall Software Series, Prentice Hall, 1991.

[23] L. HUGHES and M THOMSON. Implosion-Avoidance Protocols for Reliable Group Communications. In Proc. IEEE LCN'94, Minneapolis, Minesota, USA, pp. 218-227, October 1994.

[24] C. HUITEMA. Routing in the Internet. 2nd ed. Prentice Hall, 2000.

[25] S. JAGANNATHAN, K. ALMEROTH, and A. ACHARYA. Topology Sensitive Congestion Control for Real-time Multicast. In Proc. of NOSSDAV'2000, Chapel Hill, NC, USA, June 2000.

[26] R. JAIN. Congestion Control in Computer Networks: Issues and Trends. In IEEE Network Magazine, volume 4, n. 3, pp. 24-30, May 1990.

[27] R. JAIN. The Art of Computer Systems Performance Analysis. John Wiley & Sons, 1991.

[28] S. KASERA et al. Scalable Fair Reliable Multicast Using Active Services. In IEEE Net. (Special Issue on Multicast), volume 14, n. 1, pp. 48-57, Jan./Feb. 2000.

[29] J. F. KUROSE and K. W. ROSS. Computer Networking - A top-down approach featuring the Internet. Addison-Wesley, 1999.

[30] A. LEGOUT, J. NONNENMACHER, and E. W. BIERSACK. Bandwidth Allocation Policies for Unicast ans Multicast Flows. In Proc. of IEEE INFOCOM'99, New York, NY, USA, volume 1, pp. 254-261, March 1999.

[31] A. LEGOUT and E. W. BIERSACK. PLM: Fast Convergence For Cumulative Layered Multicast Transmission Schemes. In Proc. of ACM SIGMETRICS'2000, Santa Clara, CA, USA, pp. 13-22, June 2000.

[32] B. LEVINE and J.J. GARCIA-LUNA-ACEVES. A Comparasion of Reliable Multicast Protocols. In Multimedia Systems, volume 6, n. 5, pp 334-348.

[33] X. LI, M. AMMAR, and S. PAUL. Layered Video Multicast With Retransmission (LVMR): Evaluation of Hierarchical Rate Control. In Proc. of IEEE INFOCOMM'98, San Francisco, CA, USA, volume 3, pp. 1062-1072, March 1998.

[34] S. McCANNE, V. JACOBSON, and M. VETTERLI. Receiver-driven Layered Multicast. In Proc. of ACM SIGCOMM'96, Palo Alto, CA, USA, volume 26, n. 4, pp. 117-130, August 1996.

[35] T. MONTGOMERY. A Loss Tolerant Rate Controller for Reliable Multicast. Techinical Report NASA-IVV-97-011, West Virginia University, August 1997.

[36] R. MORRIS. Scalable TCP Congestion Control. In Proc. of IEEE INFOCOM'2000, Tel Aviv, Israel, volume 3, pp. 1176-1183, March 2000.

[37] H. H. MUHAMMAD, M. P. BARCELLOS, e R. CASAIS. Simulação de Roteamento na Avaliação de Protocolos Multicast e Sistemas Distribuídos em Grupo. In Proc. of SBC, Florianópolis, SC, Brazil, pp. 65-76, July 2002.

[38] S. PAUL. Multicasting on The Internet and Its Applications. Kluwer Academic Publishers, 1998.

[39] R. PERLMAN. Interconections - Briges, Routers, Switches and Internetworking Protocols. 2nd ed, Addison-Wesley, 2001.

[40] S. PINGALI, J. F. KUROSE, and D. TOWSLEY. A Comparasion of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. In IEEE Journal on Selected Areas in Communications, volume 15, n. 3, April 1997.

[41] I. RHEE, N. BALAGURU, and G. ROUSKAS. MTCP: Scalable TCP-Like Congestion Control for Reliable Multicast. In Proc. of IEEE INFOCOM'99, New York, NY, USA, volume 3, pp. 1265-1273, March 1999.

[42] I. RHEE, V. OZDEMIR and Y. YI. TEAR: TCP Emulation at Receivers - Flow Control for Multimedia Streaming. Techinical report, Dept. of Comp. Sci., NCSU, April 2000.

[43] L. RIZZO. pgmcc: A TCP-friendly Single-Rate Multicast Congestion Control Scheme. In Proc. of ACM SIGCOMM'2000, Stockholm, Sweden, pp. 17-28, August 2000.

[44] D. SISALEM and A. Wolisz. MLDA: A TCP-friendly Congestion Control Framework for Heterogeneous Multicast Enviroments. In Proc. of IEEE/IFIP IWQoS'2000, Pittsburgh, PA, USA, June 2000.

[45] W. R. STEVENS. TCP/IP Ilustrated Vol. 1 - The Protocols. Prentice-Hall, 1990.

[46] W. TAN and A. ZAKHOR. Error Control for Video Multicast Using Hierarchical FEC. In Proc. of IEEE ICIP'99, Kobe, Japan, volume 1, pp. 401-405, October 1999.

[47] T. TURLETTI, S. PARISIS, and J. BOLOT. Experiments with a Layered Transmission Scheme over the Internet. Techinical report RR-3296, INRIA, France, Nov. 1997.

[48] L. VICISANO, J. CROWCROFT, and L. RIZZO. TCP-like Congestion Control for Layered Multicast Data Transfer. In Proc. of IEEE INFOCOM'98, San Francisco, CA, USA, volume 3, pp. 996-1003, March 1998.

[49] H. A. WANG and M. SCHWARTZ. Achieving Bounded Fairness for Multicast and TCP Traffic in the Internet. In Proc. of ACM SIGCOMM'98, Vancouver, B.C., Canada, pp.81-92, August 1998.

[50] M. YAJNICK, J. KUROSE, and D. TOSLEY. Packet Loss Correlation in the Mbone Multicast Network. Technical Report 96-32, 1995.

[51] K. YANO and S. McCANNE. A Window-based Congestion Control for Reliable Multicast Based on TCP Dynamics. In Proc. of ACM Multimedia'2000, Los Angels, CA, USA, pp. 249-258, October 2000.