Projeto de Controladores Ótimos para Gerenciamento Ativo de Filas

Michele Mara de Araújo Espíndula Lima

Tese de Doutorado

Instituto de Computação
Universidade Estadual de Campinas

Projeto de Controladores Ótimos para Gerenciamento Ativo de Filas

Michele Mara de Araújo Espíndula Lima[1]

Novembro de 2005

Banca Examinadora:

- Nelson Luis Saldanha da Fonseca
  IC - UNICAMP (Orientador)
- Edmundo Albuquerque de Souza e Silva
  LAND - UFRJ
- Edmundo Roberto Mauro Madeira
  IC - UNICAMP
- Islene Calciolari Garcia
  IC - UNICAMP
- José Ferreira de Rezende
  GTA - UFRJ

[1]**Este trabalho foi suportado em parte pelo CNPQ através do processo 141935/2001-4**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Bibliotecário: Maria Júlia Milani Rodrigues – CRB8a / 2116

Lima, Michele Mara de Araújo Espíndula

L628p Projeto de controladores ótimos para gerenciamento ativo de filas / Michele Mara de Araújo Espíndula Lima -- Campinas, [S.P. :s.n.], 2005.

Orientadores : Nelson Luis Saldanha da Fonseca; José Claudio Geromel

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. TCP/IP (Protocolo de rede de computação). 2. Teoria do controle. 3. Redes de computadores . I. Fonseca, Nelson Luis Saldanha. II. Geromel, José Claudio. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: Design of optimal active queue management controllers

Palavras-chave em inglês (Keywords): 1. TCP/IP (Computer network protocol); 2. Control theory. 3. Computer networks

Área de concentração: Redes de Computadores

Titulação: Doutora em Ciência da Computação

Banca examinadora: Prof. Dr. Nelson Luis Saldanha da Fonseca (IC-UNICAMP)
Prof. Dr. Edmundo Albuquerque de Souza e Silva (LAND-UFRJ)
Prof. Dr. Edmundo Roberto Mauro Madeira (IC-UNICAMP)
Prof. Dr. José Ferreira de Rezende (GTA-UFRJ)
Prof. Dra. Islene Calciolari Garcia (IC-UNICAMP)

Data da defesa: 17/11/2005

TERMO DE APROVAÇÃO

Tese de doutorado defendida por Michele Mara de Araújo Espíndula Lima e aprovada pela Banca Examinadora em 17 de novembro de 2005.

Nelson Luis Saldanha da Fonseca
IC - UNICAMP (Orientador)

Edmundo Albuquerque de Souza e Silva
LAND - UFRJ

Edmundo Roberto Mauro Madeira
IC - UNICAMP

Islene Calciolari Garcia
IC - UNICAMP

José Ferreira de Rezende
GTA - UFRJ

Projeto de Controladores Ótimos para Gerenciamento Ativo de Filas

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Michele Mara de Araújo Espíndula Lima e aprovada pela Banca Examinadora.

Campinas, 17 de novembro de 2005.

Nelson Luis Saldanha da Fonseca
IC - UNICAMP (Orientador)

José Claudio Geromel (Co-orientador)
FEEC - UNICAMP

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação.

Resumo

A ocorrência de congestionamento degrada o desempenho das redes de computadores. Dentre as conseqüências negativas da sua ocorrência cita-se a diminuição da vazão, a perda de pacotes, e o aumento do atraso. Para prevenir e controlar o congestionamento, o protocolo *Transmission Control Protocol* (TCP) varia a taxa de transmissão de dados de acordo com o nível de congestionamento existente. As políticas de Gerenciamento Ativo de Filas, do Inglês *Active Queue Management* (AQM), monitoram o nível de ocupação das filas, afim de notificar o congestionamento incipiente aos nós emissores. Esta notificação é realizada através da marcação ou do descarte de pacotes.

O sistema de controle de congestionamento em redes TCP/IP, pode ser visto como um sistema de controle por retroalimentação, no qual, a taxa de transmissão dos nós fontes é ajustada de acordo com o nível de ocupação da fila. Os controladores para o gerenciamento ativo de filas determinam o valor da probabilidade de descarte ou de marcação, buscando a maximização da vazão e a minimização das perdas, garantindo, assim, a estabilidade do tamanho da fila independentemente das variações das condições da rede.

Nesta tese, são utilizadas técnicas da teoria de controle ótimo para definir uma política ótima de gerenciamento ativo de filas, denominada H2-AQM. A principal característica da H2-AQM é o uso de controladores não racionais, superando-se, assim, a dificuldade de se incorporar no projeto do controlador a garantia de estabilidade em relação ao atraso da retroalimentação. Outrossim, a estabilidade e os objetivos de desempenho do sistema são completamente expressos e solucionados através de desigualdades matriciais lineares, permitindo que os parâmetros do controlador possam ser calculados através da solução de um problema convexo simples.

Diferentes controladores operando no mesmo ponto de equilíbrio definem diferentes caminhos entre um ponto qualquer de operação do sistema e o ponto de equilíbrio. Por outro lado, o caminho percorrido para atingir a estabilidade depende dos objetivos usados para projetar o controlador. Nesta tese, é discutida, também, a escolha dos objetivos do projeto de um controlador ótimo para o gerenciamento ativo de filas. Os desempenhos dos diferentes controladores são avaliados e a eficácia do controlador que apresentou o melhor desempenho foi comparado com o desempenho das políticas RED e PI-AQM.

Abstract

Congestion is one of the most significant problems in networking. When congestion occurs, the network performance degrades, leading to throughput decrease, delay increase and packet losses. In order to avoid congestion the Transmission Control Protocol (TCP) changes its transmission rate according to the level of congestion. AQM policies notify incipient congestion to TCP source by marking or dropping packets.

In TCP/ICP networks, congestion control system can be viewed as a feedback control system in which the transmission rate of the sources are adjusted according to the level of congestion inferred by the queue occupancy. Controllers are responsible for determining the appropriate value of the dropping/marking probability values that stabilizes the queue size regardless of the network condition.

In this thesis, optimal control theory is used to conceive an optimal AQM policy, called H2-AQM. The novelty of the proposed approach lies in the use of non-rational controllers that overcomes the difficulty of incorporating guarantees of the stability with respect to the delayed part of the system in the controller design. Furthermore, in the proposed approach stability and performance objectives are completely expressed as Linear Matrix Inequalities (LMIs), thus requiring the solution of a single convex problem for the computation of the controller parameters.

Different controllers define different pathes for taking the system state to a target point of equilibrium. Moreover, the path depends on the objectives established for the design of the controller. In this thesis, a discussion on the design of AQM optimal controllers for optimal performance is also presented. The performance produced by different optimal controllers was investigated. The efficacy of the controller which presented the best performance was, then, compared to the performance of both RED and PI-AQM policies.

Dedicatória

"O tempo é muito lento para os que esperam,
muito rápido para os que tem medo,
muito longo para os que lamentam,
muito curto para os que festejam.
Mas para os que amam o tempo é a eternidade."
(William Shakespeare)

Aos dois grandes amores da minha vida:
meu marido Humberto e Ana Carolina, minha filha.

Agradecimentos

A Deus, pelo dom da vida, pela força diária que fez possível a conclusão desta caminhada, bem como por todas as oportunidades e realizações.

A meus pais, Espíndula e Genalda e aos meus irmãos: Pierre, William, Arttur e Kamilla, pelo amor incondicional, e que apesar da distância sempre se fazem presente, me apoiando e me incentivando. Esta conquista também é de vocês!

A Humberto, meu marido, pelo amor, pelo carinho, pela amizade, pelo apoio, pelo incentivo, e pela compreensão por todos os momentos que foram tirados da nossa convivência para que este trabalho fosse concluído, bem como por ter me dado o tesouro mais precioso, a nossa filha Ana Carolina.

Ao meu orientador Prof. Nelson Fonseca, pela orientação, paciência, confiança, apoio e pelos valiosos direcionamentos, os quais tornaram possível a realização deste trabalho. Pelo exemplo de profissional/pessoa demonstrado, e principalmente pela amizade destes últimos anos de convivência te agradeço.

Ao Prof. Geromel, pelas fundamentais e preciosas contribuições neste trabalho.

Ao Eduardo, não só por sua amizade, mas também e principalmente pela colaboração no desenvolvimento deste trabalho.

A todos os amigos que conquistei neste período, em especial gostaria de citar: Rodolfo, Sandro, Aninha, Guilherme, Borin, Juliana Freitag, Nahri, Islene, Guilherme, Desire, Juliana e Norton, pelos momentos divertidos e felizes que compartilhamos juntos durante o almoço e depois na sala do café.

Aos funcionários e amigos do IC/UNICAMP, em especial, a Vera, Daniel e Dona Mercedes, por seu carinho e dedicação.

Aos professores do Instituto de Computação da UNICAMP.

Aos professores do Colegiado de Informática da UNIOESTE - Universidade Estadual do Oeste do Paraná, que colaboraram para o meu afastamento para doutoramento, em especial as Professores e Amigos Chicão, Josué, Adriana e Sarajane.

Ao Povo Brasileiro que através do CNPq financiou parte deste trabalho.

A todos os que de alguma forma contribuíram para que a conclusão desta etapa fosse possível.

Siglas e Abreviaturas

ACK	ACKnowledgement
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
ARED	Adaptive Random Early Drop
AVQ	Adaptive Virtual Queue
BER	Bit Error Rate
CE	Congestion Experienced
CWND	Congestion Window
CWR	Congestion Window Reduced
DF	Don't Fragment bit
DRED	Dynamic RED
DS	Differential Service
DSACK	Duplicate SACK
ECN	Explicit Congestion Control
ECT	ECN-Capable Transport
E-RED	Exponential-RED
FPQ	Flow Proportional Queuing
FRED	Flow Random Early Drop
FTP	File Transfer Protocol
HSTCP	High Speed TCP
H-TCP:	TCP for high-speed and long-distance networks
HTTP	HyperText Transfer Protocol
I	Integral Controller
IETF	The Internet Engineering Task Force
IP	Internet Protocol
LMI	Linear Matrix Inequalities
LQR	Linear Quadratic Regulators

LRED	Loss Ratio based RED
MMEP	Média Móvel Exponencial Ponderada
MSS	Maximum Segment Size
NS	Network Simulator
P	Proportional Controller
PD	Proportional-Derivative Controller
PD-AQM	Proportional Derivative AQM
PD-RED	Proportional Derivative RED
PI	Proportional-Integral Controller
PI-AQM	Proportional Integrator AQM
PID	Proportional-Integral-Derivative Controller
PIP	Proportional Integral and Positional feedback compensation
QCF	Queue control function
QoS	Quality of Service
RED	Random Early Detection
REM	Random Exponential Marking
RFC	Request for comments
RIO	Red with In/Out Bit
RTO	Retransmission Timeout
RTT	Round-Trip-Time
SACK	Selective ACKnowledgment
SFB	Stochastic Fair Blue
SMSS	Sender Maximum Segment Size
SMVSAQM	Sliding Mode Variable Structure Control AQM
SRED	Stabilized RED
STCP	Scalable TCP
Svegas	Stabilized Vegas
TCP	Transmission Control Protocol
TCP BIC	TCP - Binary Increase Congestion Control
TCPW	TCP Westwood
TOS	Type Of Service
UDP	User Datagram Protocol
VRC	Virtual Rate Control
VS-AQM	Variable Structure AQM
XCP	eXplicit Control Protocol

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A Internet apesar de ter sido inicialmente desenvolvida para fins militares e acadêmicos, tem se tornado um veículo vital de comunicação. O grande pivô deste desenvolvimento foi o IP (*Internet Protocol*), um protocolo robusto e flexível que permite que redes com diferentes tecnologias de enlaces possam ser interconectadas. Entretanto, estas vantagens são propiciadas ao custo de um serviço de entrega não confiável, que não fornece quaisquer garantias, ou seja, o tráfego é processado o mais rapidamente e da melhor forma possível, de acordo com as condições da rede. Deste modo, o serviço tradicional oferecido pela Internet ficou conhecido como melhor-esforço (*best-effort*). Em outras palavras, no modelo de serviço melhor-esforço, apenas um único tipo de serviço é oferecido. Este serviço tem sua qualidade degradada sob congestionamento, cujas conseqüências são o desperdício de recursos, a diminuição da vazão e o aumento do atraso na entrega da informação.

Por muito tempo, este serviço foi adequado para as aplicações de dados e para a demanda existente até então. Entretanto, a crescente importância da Internet, sua rápida transformação em uma infra-estrutura comercial e a demanda por novos serviços e aplicações fizeram com que o modelo de serviço melhor-esforço se tornasse inadequado. Assim, novos modelos de serviço, nos quais as necessidades das aplicações podem ser especificadas vem sendo propostos. A qualidade de transporte oferecida ao fluxo de pacotes gerados por uma aplicação é normalmente denominada Qualidade de Serviço, ou QoS (*Quality of Service*). As arquiteturas propostas para dar suporte a QoS, essencialmente prevêem uma degradação suave ou nenhuma degradação de desempenho para alguns fluxos na presença de congestionamento [1, 2].

Evitar a ocorrência de congestionamento e controlá-lo são de capital importância para a operação da rede. Em redes que oferecem o serviço melhor-esforço, o congestionamento pode degradar ainda mais os baixos níveis de qualidade de serviço. Em redes com suporte a QoS, o congestionamento pode inviabilizar o compromisso de atendimento dos requisitos de qualidade de serviço das aplicações. Assim sendo, a prevenção e o controle

de congestionamento, o que permite uma degradação suave do desempenho das redes, são fatores essenciais para garantir a otimização de recursos da rede, bem como prover desempenho previsível ou garantido às aplicações.

Nagle em meados dos anos 80, na fase inicial da Internet, foi um dos primeiros a verificar que os roteadores são vulneráveis a um fenômeno que foi denominado como "Colapso de Congestionamento" [3]. Tal fenômeno pode ocorrer quando o tráfego é intenso e a rede está congestionada, levando ao descarte de pacotes nos roteadores e conseqüentemente à retransmissões, aumentando assim o RTT (*Round Trip Time*) a tal ponto que as máquinas envolvidas acabam derrubando suas conexões.

Pode-se dizer que o congestionamento é um dos problemas mais importantes no gerenciamento das redes de computadores. Quando ocorre congestionamento, tem-se a diminuição da vazão e o aumento do atraso fim-a-fim do tráfego. Além disto, os recursos utilizados pelos pacotes que foram descartados são desperdiçados e quando estes pacotes são retransmitidos, novos recursos precisam ser alocados.

O congestionamento ocorre quando há insuficiência de recursos para acomodar a carga presente na rede ou quando ocorre um desbalanceamento do tráfego nos nós da rede, sendo um subconjunto de recursos sobrecarregado com a carga a ele atribuído, enquanto que outro conjunto de recursos é subutilizado [4]. Tal desbalanceamento pode ser corrigido utilizando técnicas de Engenharia de Tráfego [5], enquanto que a insuficiência de recursos pode ser amenizada por mecanismos que fazem a prevenção e o controle do congestionamento.

A primeira solução proposta para tratar do problema de congestionamento foi apresentada para redes TCP/IP, por Jacobson [6]. Para controlar o congestionamento o TCP (*Transmission Control Protocol*) utiliza um mecanismo de janela deslizante, que adeqüa a sua taxa de transmissão de acordo com a estimativa de banda passante disponível. Este mecanismo faz com que os TCPs emissores diminuam sua taxa de transmissão de pacotes na ocorrência de congestionamento. Por este motivo, o TCP é comumente chamado de protocolo adaptativo ou bem comportado.

O mecanismo de controle de congestionamento da implementação padrão do TCP, denominada TCP Reno, é composto de quatro algoritmos: *Slow Start*, *Congestion Avoidance*, *Fast Retransmit* e *Fast Recovery* [7]. Os dois primeiros algoritmos são utilizados pelo TCP emissor para controlar a quantidade de dados que está sendo injetada na rede.

Os dois últimos algoritmos fazem parte do mecanismo de recuperação de perdas do TCP Reno, o qual não é eficiente quando existem múltiplas perdas numa mesma janela. Diversas variações do protocolo TCP Reno foram propostas pelo IETF (*The Internet Engineering Task Force*) para aprimorar o seu mecanismo de recuperação de perdas: o *TCP SACK*, que ameniza os problemas do TCP Reno provendo informações de quais e quantos pacotes foram recebidos corretamente pelo TCP receptor; o *TCP NewReno*, que

faz o aprimoramento através da avaliação e diferenciação dos tipos de reconhecimentos recebidos; e o *TCP Limited Transmit*, desenvolvido para melhorar o desempenho do TCP Reno para conexões com tamanho pequeno de janela.

Apesar da eficácia dos mecanismos de controle de congestionamento TCP, eles não são suficientes para prover serviços de boa qualidade em todas as situações, como também não impedem que o congestionamento ocorra, dado que outros protocolos, como o UDP, são mal-comportados, ou seja, não diminuem a taxa de transmissão na presença de congestionamento. Desta forma, novos mecanismos de controle de congestionamento, foram desenvolvidos para se obter controle de congestionamento fim-a-fim.

Os mecanismos de controle de congestionamento implementados nos roteadores monitoram a fila de saída, podendo, assim, detectar o congestionamento em forma incipiente. Além disto, como possuem a visão de qual fluxo está contribuindo para o congestionamento, facilitam a tomada de decisões sobre qual fluxo notificar o congestionamento. Dentre os mecanismos de prevenção e controle de congestionamento, pode-se destacar os algoritmos de gerenciamento de filas, que controlam o tamanho da fila de pacotes através do descarte ou marcação de pacotes quando necessário. A idéia por trás do Gerenciamento Ativo de Filas ou AQM (*Active Queue Management*) é a notificação antecipada do congestionamento incipiente, de forma a permitir que os TCP emissores possam reduzir sua taxa de transmissão antes que as filas transbordem, evitando assim, a degradação do desempenho do TCP.

O algoritmo *Random Early Detection, RED* [8], é o algoritmo de AQM recomendado pelo IETF para a Internet. RED estima o tamanho médio da fila, que é comparado com dois limiares min_{th} e max_{th}. Se o valor estiver abaixo de min_{th} o algoritmo está na zona normal de operação e nenhum pacote é descartado/marcado. Caso o valor esteja entre os dois limiares, cada pacote que chega é descartado/marcado com uma probabilidade p_a, que cresce linearmente com o tamanho médio da fila estimado. Se o valor estiver acima de max_{th}, todos os pacotes que chegam são descartados. Determinar os parâmetros de RED é um desafio, dado que quando seus valores não são corretamente definidos, seu desempenho cai significativamente.

Com o intuito de superar as dificuldades de determinar corretamente os parâmetros de RED, vários estudos baseados em heurísticas e simulações vêm sendo desenvolvidos. Contudo, tais estudos não garantem que um ponto de equilíbrio seja atingido, nem tampouco garantem a estabilidade do tamanho da fila. Por outro lado, investigações vêm sendo conduzidas para derivar configurações para RED de uma forma mais sistemática.

Para se projetar e desenvolver políticas de AQM que garantam estabilidade em torno de um ponto de equilíbrio, algumas políticas baseadas em Otimização e Teoria de Controle tem sido propostas.

Nas políticas de AQM baseadas em otimização, define-se uma função de utilidade de

fluxos agregados que caracterizam o sistema de controle de fluxo, como o objetivo de maximizar a taxa de transmissão e garantir o compartilhamento igualitário dos recursos entre as fontes. Busca-se, então, caracterizar quais são as condições de equilíbrio que podem ser obtidas deste sistema, de acordo com as condições da rede.

Nas políticas de AQM baseadas em Teoria de Controle, o controle de congestionamento é visto como um sistema de controle de retroalimentação, onde a taxa de transmissão dos nós fontes deve ser ajustada de acordo com o estado de congestionamento, que é determinado pela ocupação da fila. Desta forma, os controladores são responsáveis por determinar qual a probabilidade de descarte/marcação adequada que estabiliza o tamanho da fila independentemente das variações das condições da rede.

1.1 Principais Contribuições desta Tese

Nesta tese, são utilizadas técnicas da teoria de controle ótimo para abordar um dos maiores problemas de controle relacionados a Internet: o sistema de controle de congestionamento. Este sistema é tratado como um sistema de controle de retroalimentação, onde a taxa de transmissão dos nós fontes deve ser ajustada de acordo com o estado de congestionamento na rede, inferido pelos roteadores através da monitoração do nível de suas filas e retornado para os nós fontes através da descarte/marcação de pacotes. Desta forma, os controladores são responsáveis por determinar qual a probabilidade de descarte/marcação adequada, que maximize a vazão e minimize as perdas, e que ainda garanta a estabilidade do tamanho da fila independentemente das variações das condições da rede. Dentre as contribuições principais desta tese, pode-se destacar:

- Uma revisão bibliográfica sobre os mecanismos de controle de congestionamento do TCP Reno, a implementação padrão do TCP, bem como um levantamento e discussão de suas deficiências e das propostas apresentadas para aprimorá-lo;

- Apresentação do estado da arte das pesquisas relacionadas ao desenvolvimento de políticas de AQM baseadas em abordagens analíticas, com ênfase nos trabalhos que utilizam as ferramentas da Teoria da Otimização e Teoria de Controle;

- Projeto de um controlador AQM ótimo que utiliza uma abordagem não-racional para obtenção dos controladores para o sistema de congestionamento. Além disto, a estabilidade e os objetivos de desempenho do sistema de congestionamento são expressos e solucionados através de LMIs (*Linear Matrix Inequalities*). Na derivação do controlador, foi utilizada uma planta que representa a dinâmica do sistema de congestionamento em detalhes, garantindo a sua estabilidade independente das condições de rede;

- Discussão e análise dos objetivos de projeto de um controlador ótimo para AQM que obtenha o melhor desempenho;

- Investigação do melhor ponto de equilíbrio para o sistema de controle de congestionamento. Com a utilização deste ponto de equilíbrio, foi demonstrado através de simulações que não apenas H2-AQM supera PI-AQM e RED em atingir os objetivos esperados para uma política de AQM, como o controlador PI-AQM derivado com este ponto de equilíbrio apresenta um desempenho superior ao PI-AQM original;

- Verificação de que o aprimoramento do sistema de controle de congestionamento depende da garantia da estabilidade do tamanho da fila, bem como da janela do TCP;

- Apresentação de uma variação do controlador H2-AQM, denominada H2-TAQM, derivado utilizando um modelo estendido da dinâmica do TCP [9], que inclui o seu mecanismo de temporização e a avaliação da eficácia do mecanismo *Limited Transmit* quando é utilizada uma política de AQM baseada em controle ótimo, como o H2-TAQM.

1.2 Descrição Geral e Organização desta Tese

Esta tese está organizada da seguinte forma: no Capítulo 2, são apresentados os mecanismos de controle de congestionamento próprio do protocolo TCP. São introduzidos os algoritmos que fazem parte de sua implementação padrão, o TCP Reno. Além disto, são discutidas suas principais deficiências e é feito um levantamento de propostas apresentadas para aprimorá-lo.

No Capítulo 3, é apresentado o mecanismo de controle de congestionamento presente nos roteadores, o gerenciamento ativo de filas ou AQM. Neste capítulo, a política de AQM recomendada pelo IETF para a Internet, RED, é apresentada, e é feita uma análise das suas vantagens e deficiências. Posteriormente, são apresentados alguns dos algoritmos AQM que foram propostos para tentar solucionar as desvantagens de RED. Em uma outra seção deste capítulo é apresentada uma revisão bibliográfica sobre os trabalhos que possuem uma abordagem semelhante a que foi utilizada no desenvolvimento deste trabalho. São revisados alguns dos desenvolvimentos mais recentes e significativos de projeto de mecanismos AQM estáveis, focando nos trabalhos que utilizam ferramentas de Otimização e Teoria de Controle. Além disto, é apresentado o mecanismo denominado *Explicit Congestion Notification* (ECN), que possibilita dissociar a notificação de congestionamento do descarte de pacotes.

O Capítulo 4, visa apresentar ao leitor os conceitos básicos da área de modelagem matemática e teoria de controle necessários para a compreensão do trabalho desenvolvido.

No Capítulo 5, o sistema de controle de congestionamento é apresentado como um problema de controle. Neste capítulo, o modelo simplificado da dinâmica do sistema de controle de congestionamento, que modela a variação da janela TCP em função da variação do tamanho da fila é apresentado [9]. Este modelo foi utilizado em [10], para derivar o *Proportional Integrator AQM* (PI-AQM), que utiliza um controlador de mesmo nome.

Posteriormente, no Capítulo 6, é apresentado o desenvolvimento do projeto de um controlador ótimo para AQM, denominado H2-AQM, que utiliza o mesmo modelo que foi usado para derivar o controlador PI-AQM. No entanto, a planta utilizada no desenvolvimento do controlador H2-AQM representa a dinâmica do sistema em detalhes, garantindo a sua estabilidade independente das condições de rede. Uma outra diferença é que H2-AQM usa técnicas de controle ótimo ao invés de controladores clássicos.

A principal característica do controlador é a nova abordagem de se utilizar controladores não racionais para o sistema. Outrossim, a estabilidade e os objetivos de desempenho do sistema de AQM são completamente expressos e solucionados através de desigualdades matriciais lineares ou LMIs (*Linear Matrix Inequalities*), o que significa que os parâmetros do controlador podem ser calculados através da solução de um simples problema convexo.

Neste capítulo, também são discutidos quais devem ser os objetivos de projeto de um controlador ótimo para AQM, de forma a obter o melhor desempenho. Na obtenção dos controladores foi investigado qual o melhor ponto de equilíbrio para o sistema de controle de congestionamento. Além disto, é feita uma análise da robustez do sistema, utilizando-se técnicas de controle robusto.

No Capítulo 7, os controladores obtidos para os diferentes objetivos foram comparados entre si em um ambiente dinâmico de rede com o intuito de verificar qual deles apresenta o melhor desempenho quando o tráfego principal é de longa ou de curta duração. O controlador, cujo objetivo apresentou o melhor resultado, é comparado com RED e PI-AQM. A eficácia deste controlador é verificada em um ambiente dinâmico de rede, fazendo-se simulações exaustivas utilizando o simulador de redes NS. A carga da rede foi variada e foi inserido tráfego de ruído com o intuito de verificar a robustez desta política sob diferentes condições de rede. Um gerador de tráfego foi utilizado para gerar tráfegos específicos FTP e WEB baseados nos modelos das distribuições estatísticas que os descrevem.

No Capítulo 8, é apresentada uma extensão do controlador H2-AQM, denominada H2-TAQM. A diferença entre ambos é que H2-TAQM foi derivado utilizando um modelo estendido da dinâmica do TCP [9], que inclui o seu mecanismo de temporização. Em [11], foi apresentada uma comparação entre diferentes versões do TCP utilizando RED como

a política de AQM, e foi verificado que o uso em conjunto do algoritmo *Limited Transmit* com TCP Reno, TCP NewReno e TCP Sack pode melhorar o desempenho do TCP na recuperação de perdas. Entretanto, a extensão destes benefícios quando políticas de AQM mais eficientes são utilizadas ainda não foi verificado. Desta forma, neste capítulo, a política de AQM obtida é utilizada para avaliar a eficácia do mecanismo *Limited Transmit* quando é utilizada uma política de AQM mais eficiente.

No Capítulo 9, são apresentadas as considerações finais e os trabalhos futuros.

1.3 Publicações Realizadas pela Autora

Esta seção apresenta as publicações decorrentes dos resultados obtidos nesta tese.

1.3.1 Publicações Realizadas pela Autora Relacionadas a esta Tese

Relatório Técnico

- [12] Michele M. A. E. Lima, J. C. Geromel, and N. L. S. Fonseca, "H2-AQM - an optimal AQM controller", State University of Campinas, Institute of Computing, Campinas, Brazil, Tech. Rep. IC-03-09, April 2003.

Conferências Nacionais

- [13] Michele M. de A. Lima, N. L. S. da Fonseca, J. C. Geromel, "Eficácia do uso conjunto do mecanismo Limited Transmit com diferentes versões do protocolo TCP", Anais da conferência XXII Simpósio Brasileiro de Redes de Computadores, pp. 395-408, Gramado-RS, Maio, 2004.

- [14] Michele M. de A. Lima, N. L. S. da Fonseca, J. C. Geromel, "Uma Avaliação da Eficácia do Controlador Ótimo H2-AQM para o Gerenciamento Ativo de Filas", Anais do XXII Simpósio Brasileiro de Redes de Computadores, pp. 423-436, Gramado-RS, Maio, 2004.

- [15] Michele M. de A. Lima, N. L. S. da Fonseca, J. C. Geromel, "Um Controlador Ótimo para o Gerenciamento Ativo de Filas", Anais do 20º Simpósio Brasileiro de Telecomunicações, pp 471-477, Rio de Janeiro-RJ, Outubro, 2003.

Conferências Internacionais

- [16] Michele M. de A. Lima, N. L. S. da Fonseca, J. C. Geromel, "Design Objetives of Optimal Active Queue Management Controllers", Proceedings of IEEE Global Telecommunications Conference, Dallas EUA, December, 2004.

- [17] Michele M. de A. Lima, N. L. S. da Fonseca, J. C. Geromel, "On the Efficacy of TCP Limited Transmit Under Active Queue Management", Proceedings of IEEE International Conference on Communications, pp 1083-1088, Paris, June, 2004.

- [18] Michele M. de A. Lima, N. L. S. da Fonseca, J. C. Geromel, "An Optimal Active Queue Management Controller", Proceedings of IEEE International Conference on Communications, pp 2261-2266, Paris, June, 2004.

- [11] Michele M. de A. Lima, N. L. S. da Fonseca, J. F. Rezende. "On the Performance of TCP Loss Recovery Mechanisms". Proceedings of IEEE International Conference on Communications, pp 1812-1816, Anchorage, Alaska, May, 2003.

Periódicos

- [19] Michele M. de A. Lima, N. L. S. da Fonseca, J. C. Geromel, "Design of Optimal Active Queue Management Controllers", Submetido para Computer Networks Journal, 2005.

1.3.2 Outras Publicações Relacionadas

Durante o período de desenvolvimento da tese foram produzidos também os seguintes trabalhos:

Capítulos de Livros

- [20] J. F. Rezende, M. M. de A. Lima, and N. L. S. da Fonseca, "Mobility over Transport Control Protocol/Internet Protocol (TCP/IP)", in The Handbook of Ad hoc Wireless Networks, M. Ilyas, Ed. CRC Press, pp 19.1-19.15, 2002.

- [21] M. M. de A. Lima and N. L. S. da Fonseca, "Controle de tráfego Internet", em Mini- cursos SBRC2002, 2002, p. 187-249.

Capítulo 2

O Protocolo Transmission Control Protocol (TCP)

O *Transmission Control Protocol* (TCP) é o protocolo de transporte dominante, atualmente, na Internet. Dados de 1998 informam que o TCP era o responsável por cerca de 95% do total dos *bytes* e 90% dos segmentos enviados e por 80% dos fluxos gerados [22]. Em 2000, os dados indicam basicamente a mesma proporção, sendo o TCP responsável por 91% dos *bytes* e 83% dos segmentos gerados [23]. Dados mais recentes informam que o TCP ainda é o principal protocolo de transporte, sendo responsável por $83\% \pm 11\%$ do total dos *bytes*, $75\% \pm 12\%$ dos segmentos enviados e por $56\% \pm 15\%$ dos fluxos gerados [24]. Tais resultados indicam uma contínua estabilidade no tráfego gerado pelo TCP ao longo do tempo, apesar do crescimento do número de aplicações que utilizam o UDP nos últimos anos. Por este motivo, não se pode falar em controle de congestionamento sem estudar como funcionam os mecanismos de controle de congestionamento TCP.

Para controlar o congestionamento, o TCP utiliza um mecanismo de janela deslizante, que adeqüa a sua taxa de transmissão, de acordo com a estimativa de banda passante disponível. Tal adequação é governada pelo recebimento de reconhecimentos (*ACKnowledgements, ACK*) enviados pelo receptor.

A implementação padrão do TCP, denominada TCP Reno, possui um mecanismo de controle de congestionamento composto de quatro algoritmos: *Slow Start*, *Congestion Avoidance*, *Fast Recovery* e *Fast Retransmit* [7]. Os dois primeiros algoritmos são utilizados pelo TCP emissor para controlar a quantidade de dados que está sendo injetada na rede, evitando, assim, o congestionamento, ou seja, evitando injetar na rede uma carga maior do que esta pode absorver. Os dois últimos são utilizados pelo TCP para se recuperar de perdas de segmentos.

Apesar da sua relativa eficiência, o mecanismo de controle de congestionamento do TCP apresenta uma série de problemas. O mecanismo de recuperação de perdas não é

eficiente quando existem múltiplas perdas numa mesma janela. Além disto, a estratégia utilizada para governar o comportamento da sua janela de transmissão é inerentemente oscilatória, o que faz com que o uso da banda passante disponível também oscile significantemente. Desta forma, tem-se períodos em que recursos são subutilizados e outros em que ocorre o congestionamento. Este problema é ainda mais grave em redes com grande produto banda-atraso. Um outro problema a ser destacado é o fato de que fluxos TCP com grande RTT são prejudicados em detrimento dos que possuem RTT pequeno, não conseguindo a sua porção justa da banda passante. Finalmente, o TCP só detecta o congestionamento através de perdas. Conseqüentemente, o TCP interpreta de forma errada a ocorrência de uma perda devido a erros de transmissão, reduzindo sua taxa de transmissão.

Diversas variações do protocolo TCP Reno foram propostas pelo IETF para aprimorar o seu mecanismo de recuperação de perdas. Congestionamento intenso e longos períodos de perda impactam diferentemente o desempenho destas variações. O *TCP SACK* ameniza os problemas do TCP Reno provendo informações de quais e quantos segmentos foram recebidos corretamente pelo TCP receptor. O *TCP NewReno* faz o aprimoramento do mecanismo de recuperação de perdas do TCP Reno através da avaliação e diferenciação dos tipos de reconhecimentos recebidos. O *TCP Limited Transmit* foi desenvolvido para melhorar o desempenho do TCP Reno para conexões com tamanho pequeno de janela. Além destas variações existem outras propostas cujo objetivo principal é além de minimizar os problemas de recuperação de perdas do TCP, melhorar o seu desempenho em redes com elevado produto banda-atraso e garantir a estabilidade da janela, e conseqüentemente da taxa de transmissão.

Neste capítulo, são apresentados inicialmente os conceitos, mecanismos e procedimentos utilizados pelo TCP para fornecer o serviço de transferência confiável de dados. São introduzidos, posteriormente, os algoritmos que compõem o mecanismo de controle de congestionamento da implementação padrão do TCP, o TCP Reno. Os problemas relacionados a esta implementação do TCP são discutidos e são apresentadas as propostas definidas pelo IETF que visam melhorar o seu desempenho.

2.1 Transferência Confiável de Dados

O protocolo TCP, inicialmente definido em [25], fornece um serviço de entrega de segmentos confiável e orientado à conexão. O serviço de transferência confiável de dados é garantido através da implementação de mecanismos de recuperação de dados perdidos ou danificados. Além disto, o TCP faz o controle de fluxo entre os nós finais da conexão e previne o congestionamento na rede.

Antes de iniciar a troca de dados, os dois nós finais de uma aplicação que utiliza o TCP,

como protocolo de transporte, precisam estabelecer uma conexão. Em cada cabeçalho de segmentos TCP existem informações relativas ao segmento enviado e também sobre o estado do emissor, necessárias para garantir a transferência confiável dos dados, bem como para fazer o controle de fluxo e de congestionamento. Os campos que compõem o cabeçalho TCP são apresentados na Figura 2.1 e são descritos a seguir:

Porta de origem: número da porta da aplicação de origem;

Porta de destino: número da porta da aplicação de destino;

Número de seqüência: número de seqüência que identifica o primeiro *byte* dentro do fluxo de dados do segmento enviado do TCP emissor para o TCP receptor;

Número de reconhecimento: número de seqüência do próximo *byte* que o emissor do reconhecimento espera receber;

Tamanho do cabeçalho: informa o tamanho do cabeçalho;

Janela do receptor: indica a quantidade de *bytes* que o receptor pode receber;

Checksum: campo utilizado para verificar se o segmento não contém erros. Deve ser calculado pelo emissor, e deve cobrir todo o segmento;

Ptr para dados urgentes: válido apenas se o "flag" URG estiver ligado. Usado para permitir ao emissor enviar dados de emergência;

Opções: campo de tamanho variável que é utilizado pra negociar o tamanho máximo do segmento (MSS) ou opções de fatores de escala para a janela de transmissão;

Dados: contém os dados passados pela camada de aplicação;

URG: "flag" que informa que o ponteiro para dados urgentes é válido;

ACK: "flag" que informa que o número de ACK é válido;

PSH: "flag" que informa ao receptor que deve passar os dados recebidos para a aplicação o mais rápido possível;

SYN, FIN e RSH: flags utilizados na gestão de conexão. O SYN sincroniza os números de seqüência para iniciar uma conexão. O FIN indica que o emissor está parando de enviar dados. O RSH reinicializa a conexão.

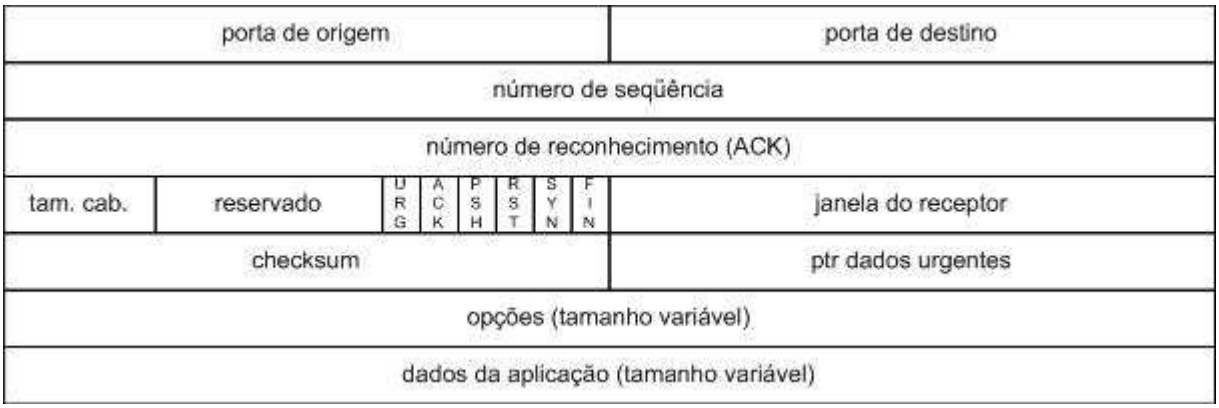

Figura 2.1: Cabeçalho TCP

Para fazer o controle de fluxo e prevenir o congestionamento, o TCP utiliza um mecanismo de janela deslizante, que adeqüa a sua taxa de transmissão de acordo com a estimativa de banda passante disponível e a capacidade de recepção do receptor. Apesar das conexões TCP serem bidirecionais, por questões de simplicidade, na discussão a seguir sobre o mecanismo de janela deslizante utilizado pelo TCP, apenas as funções da comunicação em uma direção serão apresentadas. O emissor será tratado como o nó final que envia segmentos e o receptor como o nó final que os recebe e que envia confirmações para o emissor do recebimento destes segmentos. Dentre as informações contidas nos campos do cabeçalho TCP, apenas três campos vão ser aqui abordados: o número de seqüência, $NumSeq$; o número de reconhecimento, ACK ($ACKnowledgement$); e a janela do receptor, $RcvWnd$. O detalhamento sobre os demais campos podem ser encontrados em [25].

Para cada segmento enviado é atribuído um número de seqüência, $NumSeq$, de forma a permitir que o receptor possa reordenar os segmentos, caso eles tenham sido recebidos fora de ordem, e para solicitar a retransmissão dos segmentos não recebidos.

O número de reconhecimento ou ACK é utilizado pelo receptor para informar ao emissor que dados foram recebidos; este campo indica que o receptor recebeu corretamente todos os *bytes* até o número $ACK - 1$, e que espera receber o segmento com número de seqüência ACK. O receptor confirma o recebimento de cada segmento recebido, entretanto, se segmentos forem recebidos fora de ordem, o receptor continua enviando o ACK com o valor do número de seqüência do segmento que espera receber. Para o emissor, o recebimento de três ou mais destes $ACKs$ duplicados, é utilizado como uma indicação de que o segmento esperado pelo receptor e informado no ACK, foi perdido. Os segmentos recebidos fora de ordem ficam armazenados no *buffer* do receptor até que os segmentos perdidos sejam recebidos, quando então são repassados para a camada de aplicação. Apesar do receptor utilizar reconhecimento cumulativo, ou seja, um ACK indicar o re-

cebimento correto de mais de um segmento, para fins de simplificação, na discussão que segue, assumir-se-á que cada segmento é reconhecido por ACKs distintos.

O TCP receptor possui um *buffer* interno, no qual os segmentos são temporariamente armazenados e processados até que possam ser repassados para a aplicação na ordem correta. A capacidade deste *buffer* será aqui denotada por $RcvBuff$. O valor da janela do receptor, $RcvWnd$, informa a quantidade de espaço disponível no seu *buffer* para armazenar os segmentos recebidos. Este valor é utilizado pelo emissor para fazer o controle de fluxo, ou seja, não enviar mais dados do que o receptor tem condições de receber. Seja o último *byte* recebido pelo receptor denotado por $RcvLastByteRcvd$, e seja o último *byte* processado pela aplicação no receptor denotado por $RcvLastByteRead$, o valor da janela $RcvWnd$, fica então determinada por:

$$RcvWnd = RcvBuff - (RcvLastByteRcvd - RcvLastByteRead) \tag{2.1}$$

Seja $SndNextByteSent$ o valor do número de seqüência do próximo segmento a ser enviado pelo emissor e seja $SndLastByteAcked$ o valor do último número de seqüência reconhecido pelo receptor, $PacketsOut$, o número de segmentos em trânsito, ou seja, segmentos que foram enviados, mas nenhum ACK correspondente foi recebido, é dado por: $PacketsOut = SndNextByteSent - SndLastByteAcked$. O emissor, por receber informações relativas a capacidade de armazenamento do receptor, pode determinar a quantidade de dados que pode enviar sem sobrecarregá-lo, ou seja, o valor de $PacketsOut$ não pode exceder o valor de $RcvWnd$.

Um outro limitante para a taxa de transmissão do emissor é a capacidade da rede, a qual é determinada pela quantidade de dados que pode transmitir em um dado momento, medida em *bytes* por segundo. Entretanto, a rede não pode informar ao emissor sobre sua disponibilidade de recursos. Desta forma, o emissor possui uma variável denominada janela de congestionamento, $SndCwnd$, a qual é usada para limitar a quantidade de dados enviada pelo emissor, baseada em uma estimativa da quantidade de banda passante disponível. Assim, o valor da janela é determinado dinamicamente, de forma a prevenir a ocorrência de congestionamento. A cada ACK não duplicado recebido o seu valor deve ser aumentado e será diminuído quando perdas ocorrerem. Na seção seguinte, serão apresentados os algoritmos utilizados pelo TCP para realizar o controle de congestionamento, ou seja, controlar a variação da janela de congestionamento, $SndCwnd$.

As Figuras 2.2 e 2.3 apresentam os mecanismos de janela deslizante utilizados, respectivamente, pelo receptor para controlar a quantidade de dados recebidos, e pelo emissor para controlar a quantidade de dados enviados. Pode-se verificar na Figura 2.3, que a quantidade de dados que o emissor pode transmitir dentro de uma mesma janela em um dado momento, denominado $SndUseableWnd$, é dado por:

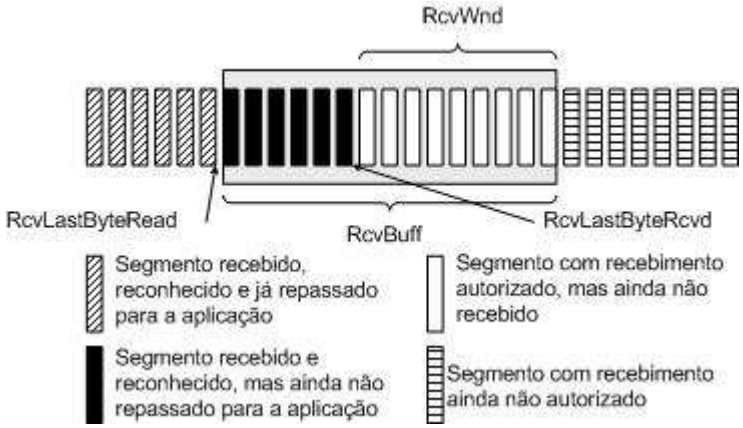

Figura 2.2: Mecanismo de Janela Deslizante utilizado pelo TCP Receptor

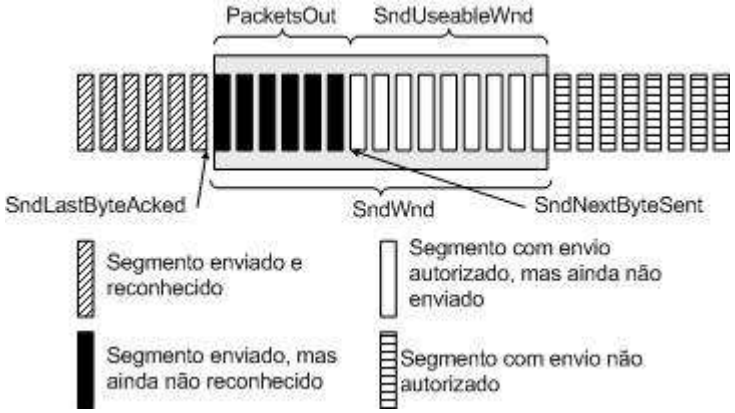

Figura 2.3: Mecanismo de Janela Deslizante utilizado pelo TCP Emissor

$$\begin{aligned} SndUseableWnd &= SndWnd - PacketsOut; \\ &= SndWnd + SndLastByteAcked - SndNextByteSent; \end{aligned} \tag{2.2}$$

A cada segmento enviado o TCP emissor inicializa um temporizador, que é utilizado para detecção de perdas. Caso não seja recebido o reconhecimento para o segmento enviado dentro de um intervalo, denominado intervalo de temporização, diz-se que aconteceu um evento de *timeout*, ou seja, o intervalo de temporização expirou. O segmento em questão é considerado perdido, sendo, então, retransmitido.

O valor do intervalo de temporização deve ser superior ao tempo necessário para enviar o segmento e receber o reconhecimento correspondente, denominado por RTT (*Round Trip Time*). Desta forma, a correta determinação do intervalo de temporização depende

de uma estimativa adequada do RTT. Caso o valor do intervalo seja menor que o RTT, retransmissões desnecessárias irão ocorrer. Do contrário, o TCP levaria um tempo adicional para descobrir a ocorrência de uma perda e fazer a retransmissão, conseqüentemente, teria-se a introdução de atrasos significativos e desnecessários na transferência de dados para a aplicação.

Seja $SndSampleRTT$, o valor do intervalo decorrido entre o envio de um segmento e o recebimento do ACK correspondente. Como o estado da rede é dinâmico, os valores obtidos para segmentos distintos pode variar. Desta forma, para determinar o valor do intervalo de temporização, o TCP emissor mantém uma estimativa do valor típico de RTT, denominado $SndEstimatedRTT$, a qual é atualizada cada vez que um ACK é recebido, obtendo-se assim, uma nova amostra de $SndSampleRTT$. A variável $SndVarRTT$ mede a variação do RTT. O cálculo da estimativa do RTT, bem como da sua variação é baseado em uma média móvel exponencial ponderada (MMEP). No caso do cálculo de $SndEstimatedRTT$, a idéia é atribuir um peso maior aos valores das amostras de $SndSampleRTT$ obtidas mais recentemente, com o intuito de fazer com que o seu valor reflita com mais exatidão o estado atual da rede. Para o cálculo de $SndVarRTT$, a idéia é verificar o quanto $SndSampleRTT$ se desvia do valor de $SndEstimatedRTT$. Os valores utilizados para α e β são $\alpha = 1/8$ e $\beta = 1/4$ como recomendado em [6, 26]. Os valores de $SndEstimatedRTT$ e $SndVarRTT$ são dados por:

$$SndEstimatedRTT = (1-\alpha) * SndEstimatedRTT + \alpha * SndSampleRTT; \quad (2.3)$$

$$SndVarRTT = (1-\beta) * SndVarRTT + \beta * |SndEstimatedRTT - SndSampleRTT|; \quad (2.4)$$

Uma vez obtido o valor da estimativa do RTT, e de sua variação o valor do intervalo de temporização, denotado por $SndRTO$ (*Sender Retransmit TimeOut interval*), pode ser calculado. Se o valor de $SndVarRTT$ for alto, implica que há grande variação nos valores obtidos para os RTTs, logo o valor de $SndRTO$ deve ser bem maior que o valor de $SndEstimatedRTT$. Quando ocorre pouca variação, o valor obtido para $SndEstimatedRTT$, reflete realmente o estado atual da rede, e, assim, o valor de $SndRTO$ deverá ser apenas um pouco maior que o valor de $SndEstimatedRTT$. Desta forma, o valor de $SndRTO$ é determinado por [26]:

$$SndRTO = SndEstimatedRTT + 4 * SndVarRTT; \quad (2.5)$$

2.2 Mecanismo de Controle de Congestionamento do TCP Reno

Nesta seção são apresentados os mecanismos utilizados pelo TCP Reno, a implementação padrão do TCP, para controlar a forma como sua janela de congestionamento deve variar. O objetivo é adequar a taxa de transmissão ao limite permitido pela janela anunciada pelo receptor, bem como as condições da rede, prevenindo, assim, a ocorrência de congestionamento.

O mecanismo de controle de congestionamento do TCP Reno é composto de quatro algoritmos: *Slow Start*, *Congestion Avoidance*, *Fast Recovery* e *Fast Retransmit* [7]. Os dois primeiros algoritmos são utilizados para controlar a quantidade de dados que está sendo injetada na rede, enquanto que os dois últimos fazem parte do mecanismo de recuperação de perdas do TCP. Apesar de serem quatro algoritmos distintos, na prática os algoritmos são implementados como se fossem apenas dois: *Slow Start* e *Congestion Avoidance*, e *Fast Recovery* e *Fast Retransmit*, e serão assim abordados nas subseções seguintes.

2.2.1 Slow Start e Congestion Avoidance

No início de uma transmissão, o TCP emissor tem que lidar com condições desconhecidas da rede, o que requer que o algoritmo de *Slow Start* seja utilizado para lentamente estimar a capacidade disponível, evitando assim o congestionamento, ou seja, evitando injetar na rede mais carga do que esta pode absorver.

O TCP assume como indicação de congestionamento eminente a ocorrência de perda de segmentos. Como mencionado na Seção 2.1, o TCP emissor pode detectar a perda de segmentos de duas formas: quando são recebidos ACK's duplicados para o mesmo segmento enviado; ou quando ocorre um *timeout*.

Durante a fase de *Slow Start*, o TCP emissor incrementa sua janela de congestionamento para cada ACK recebido que reconhece dados novos. Desta forma, após serem recebidos todos os ACKs correspondentes aos segmentos enviados, dobra-se o número de segmentos que podem ser enviados. Quando a janela de congestionamento atinge um limiar ou quando um congestionamento é detectado, o algoritmo *Slow Start* é finalizado e o algoritmo de *Congestion Avoidance* é iniciado, reduzindo a janela de congestionamento para diminuir a taxa de transmissão. Mesmo com o algoritmo *Slow Start* pode ocorrer congestionamento na rede, devido ao fato de que vários segmentos de conexões distintas podem chegar ao mesmo tempo em um roteador cuja capacidade é menor que o total de segmentos que chegam.

Para implementar estes dois algoritmos, três novas variáveis são acrescentadas às

variáveis de estado das conexões TCP: *SndCwnd*, denominada de janela de congestionamento, que limita a quantidade de dados que o emissor pode transmitir na rede depois de receber um reconhecimento (ACK); *RcvWnd*, o tamanho da janela de recepção do receptor; e *SndSSThresh - slow start threshold*, variável usada para determinar qual algoritmo *Slow Start* ou *Congestion Avoidance* está controlando a transmissão de segmentos.

Seja $SndWnd$, a janela de transmissão do emissor, *SndSMSS* o tamanho do maior segmento que o emissor pode transmitir e $SndIW$, o valor inicial da janela de transmissão. Quando uma nova conexão é iniciada, o valor de *SndCwnd* é inicializado com o valor de $SndIW$ (Equação 2.6). O valor inicial de *SSThresh* deve ser um valor alto e geralmente é utilizado o valor de $RcvWnd$.

$$SndIW = min(4 * SndSMSS, max(2 * SndSMSS, 4380bytes)); \tag{2.6}$$

De modo a realizar simultaneamente o controle de fluxo e o controle de congestionamento, a janela de congestionamento do receptor, *SndCwnd*, juntamente com a janela de recepção do receptor, $RcvWnd$, são utilizadas para determinar o valor da janela de transmissão do emissor, $SndWnd$. Desta forma, a quantidade máxima de dados que o emissor pode transmitir em um dado momento, sem esperar pelo reconhecimento é dada por:

$$SndWnd = min(SndCongWnd, RcvWnd) \tag{2.7}$$

Quando um ACK é recebido, *SndCwnd* é incrementada. O incremento depende de qual algoritmo está sendo usado. Se $SndCwnd \leq SSThresh$, o algoritmo usado é o *Slow Start*; se $SndCwnd > SSThresh$, o algoritmo usado é o *Congestion Avoidance*. É importante destacar que a escolha do valor do limiar, *SSThresh*, que é uma estimativa do valor do ponto operacional de equilíbrio, é fundamental para o desempenho destes algoritmos [27].

Na fase de *Slow Start*, a cada ACK recebido, a janela *SndCwnd* é incrementada de um segmento, como apresentado na Equação (2.8). Pode-se verificar que o crescimento é exponencial durante a fase deste algoritmo, já que o envio de dois segmentos e o recebimento dos seus respectivos ACK's, faz com que quatro segmentos possam ser enviados, e depois com o recebimento dos seus ACK's, mais oito, e assim por diante, até que o valor de *SndCwnd* atinja o valor de *SSThresh*, ou que perdas aconteçam.

$$SndCwnd = SndCwnd + 1; \tag{2.8}$$

Uma vez que o TCP detecta a perda de segmentos, ou que o limiar *SndSSThresh* tenha sido atingido, o algoritmo *Congestion Avoidance* entra em cena. Para este algoritmo, o incremento da janela é de um segmento a cada RTT (Equação 2.9), provendo um

crescimento linear para a janela de congestionamento nesta fase.

$$SndCwnd = SndCwnd + SndSMSS * SndSMSS / SndCwnd \tag{2.9}$$

Caso uma perda tenha sido detectada, o valor do limiar *SSThresh* deve ser reajustado, de forma a garantir que a mudança da fase de crescimento exponencial da janela para a fase linear aconteça mais cedo (Equação 2.9).

$$SndSSThresh = max(PacketsOut/2, 2 * SndSMSS) \tag{2.10}$$

Caso a perda tenha sido detectada através da ocorrência de um *timeout*, o valor da janela $SndCwnd$ deve ser de apenas um segmento, independentemente do valor de $SndIW$ (Equação 2.11). Depois que o segmento perdido for retransmitido, o algoritmo de *Slow Start* volta a governar o incremento da janela, até que o novo valor de *SndSSThresh* seja atingido, quando então, o algoritmo de *Congestion Avoidance* entra em ação, e assim, o ciclo continua.

$$SndCwnd = SndSMSS \tag{2.11}$$

2.2.2 Fast Retransmit e Fast Recovery

Como apresentado na Seção 2.1, quando um segmento é recebido fora de ordem, o TCP receptor deve enviar imediatamente um ACK informando ao emissor o valor do número de seqüência do segmento que espera receber. Para o TCP emissor este ACK será uma confirmação duplicada.

O TCP emissor ao receber ACK's duplicados não tem como saber se o que gerou a duplicação foi o recebimento fora de ordem ou se houve perda de segmentos. Assim, o TCP emissor aguarda por um pequeno número de ACK's para tomar uma decisão. Se apenas um ou dois forem recebidos, isto indica que o receptor recebeu os segmentos fora de ordem e que já conseguiu ordená-los. Se três ou mais ACK's duplicados forem recebidos, é uma forte indicação de que houve de fato a perda do segmento.

Em meados de 1990, Jacobson propôs alterações ao mecanismo de prevenção e de controle de congestionamento TCP [28]. Nesta proposta, o TCP emissor ao receber três ACK's duplicados, ou seja quatro ACK's idênticos, o TCP deve retransmitir o segmento que aparentemente foi perdido, sem esperar no entanto que expire o intervalo de temporização, ou seja, sem que ocorra um *timeout*. Este algoritmo é denominado *Fast Retransmit* e foi implementado na versão 4.3BSD Tahoe TCP.

Depois que o TCP emissor retransmitir o segmento aparentemente perdido, o algoritmo *Fast Recovery* passa a governar a transmissão de novos segmentos até que um ACK não

duplicado seja recebido. Este algoritmo faz com que o algoritmo *Congestion Avoidance* seja executado ao invés do *Slow Start*. Esta modificação permite que se consiga uma alta vazão em congestionamentos moderados [7].

O motivo pelo qual o algoritmo *Congestion Avoidance* e não *Slow Start* é executado é que o recebimento de ACK's duplicados indica não apenas que um segmento foi perdido, mas também que segmentos estão trafegando na rede, ou seja, como o receptor só pode gerar ACK's duplicados quando um segmento é recebido, isto significa que este segmento passou pela rede e está no *buffer* do receptor. Como existem dados fluindo entre o emissor e o receptor, o TCP pode continuar a transmitir segmentos, só que a uma taxa um pouco menor.

Os algoritmos *Fast Retransmit* e *Fast Recovery* são implementados juntos, seguindo os passos apresentados a seguir:

1. Quando o terceiro ACK duplicado é recebido, o valor do limiar *SndSSThresh* deve ser ajustado de acordo com a Equação (2.10). Depois o segmento perdido e indicado nos ACKs recebidos deve ser retransmitido.

2. O valor da janela de congestionamento *SndCwnd* deve ser alterado de acordo com a Equação (2.12). Esta alteração faz com que a janela de congestionamento seja atualizado com o número de segmentos que deixou a rede e que estão armazenados no receptor, ou seja, a janela é incrementada de três segmentos.

$$SndCwnd = SndSSThresh + 3 * SndSMSS; \tag{2.12}$$

3. A cada vez que um ACK duplicado adicional for recebido, *SndCwnd* deve ser incrementado de um segmento, ou seja, pelo valor de $SMSS$. Este incremento faz com que a janela de congestionamento seja atualizada para refletir que um segmento deixou a rede. Depois disto, um novo segmento deve ser transmitido se permitido pelo novo valor da janela, *SndCwnd*.

4. Finalmente, quando for recebido um ACK que confirma o recebimento de um novo segmento pelo receptor, o valor de *SndCwnd* deve ser modificado para o valor de *SndSSThresh*, que foi definido no passo 1. Este ACK faz o reconhecimento do segmento que foi retransmitido no passo 1, além de ser uma confirmação de que todos os segmentos intermediários que foram enviados no período entre o envio do segmento que foi perdido e o recebimento dos três ACK's duplicados.

A versão de TCP que implementa os dois algoritmos apresentados anteriormente, além dos algoritmos *Fast Restransmit* e *Fast Recovery* ficou conhecida como TCP-Reno, por ter sido implementada na versão 4.3BSD Reno.

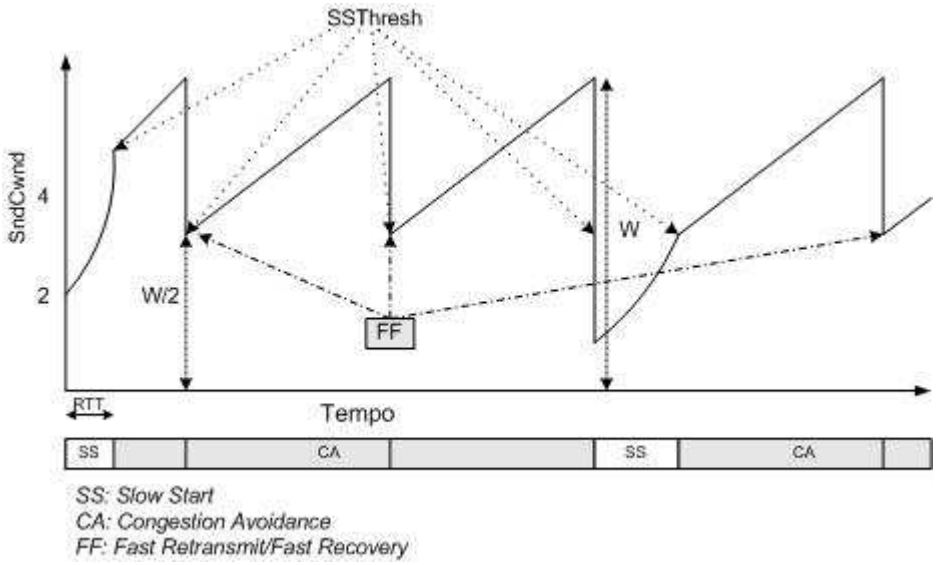

Figura 2.4: Comportamento da Janela de Congestionamento do TCP

Na Figura 2.4, é apresentado o comportamento da janela de congestionamento do TCP, *SndCwnd*, quando cada um dos quatro algoritmos é utilizado. Pode-se verificar seu crescimento exponencial durante a fase de *Slow Start*, até que se atinja o valor de *SndSSThresh*, quando, então, passa a ter um crescimento linear. Quando ocorre uma perda, o valor de *SndSSThresh* é reduzido para a metade do valor corrente da janela *SndCwnd*. Caso a perda tenha sido detectada pelo recebimento de ACKs duplicados os algoritmos de *Fast Retransmit* e *Fast Recovery* entram em ação para retransmitir o pacote perdido, e ativar o algoritmo de *Congestion Avoidance* para governar o crescimento da janela. Caso a perda tenha sido detectada pela ocorrência de *timeout*, o valor da janela passa a ser igual a um segmento, e o algoritmo de *Slow Start* passa a governar o crescimento da janela. Pode-se verificar que na fase estável do TCP, a janela varia entre W e $\frac{W}{2}$, onde W depende da capacidade da rede e do número de conexões ativas [29].

2.3 Deficiências do TCP Reno

Apesar do TCP Reno ter melhorado o desempenho do mecanismo de recuperação de perdas do TCP-Tahoe, quando o congestionamento é moderado, o TCP-Reno não pode se recuperar de forma eficiente quando existem múltiplas perdas em uma mesma janela. Além deste, o TCP Reno também apresenta uma série de outros problemas.

A estratégia AIMD utilizada para governar o comportamento da sua janela de transmissão é inerentemente oscilatória, o que faz com que o uso da banda passante disponível também oscile de forma significante. Um outro problema é que a taxa de transmissão é

inversamente proporcional a raiz quadrada da probabilidade de perda, o que implica em dizer que para se obter altas taxas de transmissão, a taxa de perda tem que ser extremamente baixa. Os dois problemas anteriores são ainda mais grave em redes com grande produto banda-atraso.

Um outro problema a ser destacado é o fato de que fluxos TCP, com grande RTT são prejudicados em detrimento dos que possuem RTT pequeno, não conseguindo a sua porção justa da banda passante. Finalmente, o TCP "tem que criar" perdas para detectar o estado de congestionamento da rede. Conseqüentemente, o TCP interpreta de forma errada a ocorrência de uma perda devido a erros de transmissão, reduzindo sua taxa de transmissão.

2.3.1 Recuperação de Múltiplas perdas em uma mesma Janela

O TCP fonte utiliza os ACKs enviados pelo TCP receptor para estimar a quantidade de banda passante disponível, adequando o valor da sua janela ao estado de congestionamento da rede. O TCP infere a existência de congestionamento quando tem seus pacotes descartados pelos roteadores.

Como apresentado na Seção 2.2.2, o TCP emissor pode detectar que perdas aconteceram quando o temporizador expira ou quando são recebidos ACKs duplicados. Para recuperar-se de perdas através da ativação dos algoritmos de *Fast Retransmit* e *Fast Recovery*, o TCP Reno aguarda o recebimento de três ACKs duplicados para um mesmo pacote. Caso o número necessário de ACKs não seja recebido, estes algoritmos não poderão ser ativados. Esta situação geralmente ocorre por duas razões: a janela de congestionamento é pequena ou um grande número de segmentos foram perdidos em uma mesma janela.

Apesar do TCP Reno ter aprimorado o mecanismo de recuperação de perdas do TCP Tahoe através da inclusão do algoritmo de *Fast Recovery*, ele não se recupera de forma eficiente quando existem múltiplas perdas numa mesma janela. A perda de múltiplos segmentos pode ter um efeito extremamente danoso na vazão TCP e na latência na transferência dos arquivos. Em [30], são feitas simulações que mostram que quando duas perdas acontecem, o TCP Reno consegue recuperar-se das perdas, entretanto os algoritmos de *Fast Retransmit* e *Fast Recovery* são ativados em seqüência, reduzindo desnecessariamente o valor da janela de congestionamento duas vezes, e conseqüentemente a taxa de transmissão é drasticamente reduzida. Quando três ou mais segmentos são perdidos, apenas o primeiro segmento é recuperado pelo algoritmo *Fast Retransmit*. Para recuperar-se dos demais, o TCP Reno falha em ativar o algoritmo de *Fast Recovery*, e basicamente é forçado a esperar pela ocorrência de um *timeout* para descobrir que os demais segmentos foram perdidos, e a retransmissão só é feita através do uso do algoritmo *Slow Start* [27, 30, 31]. Com isto o TCP Reno só se recupera de múltiplas perdas após um longo período, o que

aumenta o atraso da transmissão, dado que depois da ocorrência de um *timeout* a janela de transmissão é reduzida drasticamente para um segmento e o TCP emissor entra na fase de *Slow Start* e o retorno para o tamanho original só irá ocorrer após um considerável intervalo de tempo. [7].

Quando o congestionamento é intenso, rajadas de perdas de segmentos podem acontecer, aumentando a probabilidade de ocorrência de *timeouts* e conseqüentemente degradando o desempenho do TCP. Medidas obtidas a partir de um servidor Web congestionado indicam que aproximadamente 56% das retransmissões de segmentos ocorrem apenas após a expiração do intervalo de temporização, enquanto que os outros 44% ocorrem depois da ativação do algoritmo de *Fast Retransmit* [32]. Além disto, foi verificado em [33], que 85% das retransmissões feitas após a ocorrência do *timeout* poderiam ser evitadas através da ativação correta do algoritmo de *Fast Retransmit* [33].

Na Seção 2.4 são apresentadas as propostas do IETF definidas para aprimorar o mecanismo de recuperação de perdas do TCP Reno.

2.3.2 Necessidade de Detectar Perdas para Determinar o Estado da Rede

No início do estabelecimento de uma conexão, o TCP receptor informa ao emissor qual deve ser sua taxa máxima de transmissão. Um outro limitante para a taxa de transmissão do emissor é a capacidade da rede. Entretanto, a rede não pode informar ao emissor sobre sua disponibilidade de recursos. Desta forma, o emissor deve estimar a quantidade de banda passante disponível. Para tanto, o TCP Reno utiliza um mecanismo reativo de controle de congestionamento, no qual precisa gerar perdas para inferir o estado da rede, para então readequar sua taxa de transmissão [34].

O mecanismo de *Slow Start* apresentado do TCP Reno dobra o valor da sua janela de congestionamento a cada RTT até que ocorram perdas, quando então a janela é reduzida a metade. No momento em que a janela atinge um valor maior que a capacidade da rede, ocorrem perdas que podem chegar a ordem de cerca da metade do valor da janela de congestionamento. Mesmo quando o algoritmo de *Congestion Avoidance* é utilizado, a janela de congestionamento também é sempre incrementada, inclusive quando a taxa de transmissão adequada é atingida. Desta forma, o TCP Reno precisa criar perdas para determinar que está transmitindo a uma taxa excessiva.

O TCP Vegas [35,36] utiliza um mecanismo proativo para detectar e corrigir o congestionamento, mesmo quando este é incipiente. O TCP Vegas detecta o congestionamento através da avaliação da taxa de vazão. A idéia principal por traz do mecanismo de congestionamento do TCP Vegas é que o número de *bytes* em trânsito é diretamente proporcional a vazão esperada, ou seja, quando a rede não está congestionada a diferença entre a vazão

medida e a esperada é mínima, mas quando está congestionada, a vazão medida é bem menor que a esperada, o que indica que a janela de congestionamento está acima do que deveria estar. Desta forma, ao contrário do TCP Reno que sempre incrementa sua janela de congestionamento, podendo, assim, gerar perdas desnecessárias, o TCP Vegas para de incrementar sua janela quando um ponto de equilíbrio é atingido.

2.3.3 Penalização de Conexões com RTTs longos

O TCP fonte utiliza os ACKs enviados pelo TCP receptor para estimar a quantidade de banda passante disponível. O tempo decorrido entre o envio do segmento e recebimento do reconhecimento correspondente, é denominado por RTT (*Round Trip Time*). Desta forma, o TCP emissor só pode alterar o valor de sua janela de transmissão a cada RTT. Isto significa que fluxos TCP que estão sujeitos a longos RTTs [1] vão demorar mais a receber os ACKs e conseqüentemente, irá diminuir a taxa de crescimento de sua janela. Como resultado, tem-se que fluxos com RTT curto obtêm uma vazão maior do que os que têm RTT longo. Além disto, os fluxos com RTT longo irão ter dificuldade em obter a sua porção justa de banda passante, já que os fluxos com RTT curto irão aumentar sua taxa de transmissão mais rapidamente [34]. Este fato ainda é mais grave em redes com grande produto banda-atraso. A dependência da vazão em relação ao RTT foi verificada em [37,38], onde foi apresentado que a vazão média de uma conexão TCP é inversamente proporcional a RTT^{α}, onde $1 \leq \alpha \leq 2$.

Um outro problema enfrentado pelos fluxos TCP com longos RTTs é que a duração da fase inicial de *Slow Start*, que deveria ser rápida e transitória, é desnecessariamente prolongada. Como nesta fase o TCP emissor está fazendo uma sondagem para verificar qual deve ser a taxa de transmissão adequada, a quantidade de banda passante utilizada pelo emissor geralmente está aquém do que seria a sua porção justa. Desta forma, tem-se o aumento da latência das transferências e a diminuição do desempenho destes fluxos, principalmente os que são de curta duração.

Como já apresentado anteriormente, o TCP Reno não se recupera eficiente, quando múltiplas perdas ocorrem em uma mesma janela, levando um tempo significativo para fazer as retransmissões e fazer com que o valor da janela de congestionamento volte ao valor anterior a ocorrência das perdas. Se a conexão TCP apresentar ainda longos RTTs, seu mecanismo de recuperação de perdas terá seu desempenho prejudicado, dado que levará um tempo maior para descobrir que perdas aconteceram, bem como para recuperar-se delas.

[1] Longos RTTs podem ocorrer por várias razões, entre elas pode-se citar: elevado tempo de propagação característico do enlace utilizado; atraso enfrentado nas filas dos roteadores; grande número de enlaces intermediários entre o nó emissor e o nó receptor. RTTs longos tornaram-se comuns na Internet, principalmente depois da introdução dos enlaces de satélite.

O TCP BIC -*Binary Increase Congestion Control*, apresentado em [39], teve como um dos seus objetivos de projeto a redução do problema de penalização dos fluxos com maiores RTTs.

2.3.4 Lei do Inverso da Raíz Quadrada

Dada uma probabilidade de descarte/marcação p, o tamanho de um segmento $SndMSS$, e considerando-se um RTT basicamente constante, pode-se determinar de forma aproximada e simplificada o valor máximo para a taxa de transmissão de uma fonte TCP, na sua fase de estabilização por [40]:

$$SndBw \leq \frac{SndMSS}{RTT}\frac{Const}{\sqrt{p}}; \tag{2.13}$$

Como a taxa de transmissão $SndBw = \frac{SndWndSndMSS}{RTT}$, tem-se que:

$$SndWnd = \frac{Const}{\sqrt{p}}; \tag{2.14}$$

Como o TCP garante a eqüidade entre as conexões ativas, tem-se que, no equilíbrio, todas as conexões ativas sobre um enlace de capacidade C terão valor de janela igual a $SndWnd = \frac{C}{N}$, onde N é o número de conexões ativas. Como a capacidade do enlace é um valor constante, pode-se reescrever a Equação (2.14) da seguinte forma:

$$p = N^2 Const; \tag{2.15}$$

O valor da constante $Const$ agrupa termos que são tipicamente constantes e que é conseqüência da implementação TCP (Reno, NewReno, Sack), da estratégia de reconhecimento utilizada (atrasado ou não-atrasado) e pelo modelo de perdas utilizado (aleatórias ou periódicas).

A Equação (2.14) ficou conhecida como "Lei do Inverso da Raíz Quadrada" (*Inverse Square Root Law*). Esta lei determina que a vazão TCP é inversamente proporcional a raíz quadrada da probabilidade de descarte/marcação p, que por sua vez aumenta com o quadrado do número de conexões ativas. Em [41] é apresentada um detalhamento da sua derivação e em [40] é feita sua validação através da comparação da vazão TCP obtida em simulação com a obtida utilizando-se este modelo.

Esta lei foi utilizada em [42], para cunhar e definir o termo "*TCP-friendly*". Um fluxo é dito *TCP-friendly* se a sua taxa de transmissão não excede a taxa de transmissão de uma conexão TCP dada pela Equação (2.14), que segue a estratégia AIMD (*Additive Increase Multiplicative Decrease*) para governar sua janela, e que se encontra nas mesmas condições.

Pode-se verificar pela Equação (2.14), que para se obter um valor alto para a janela de congestionamento, são requeridos valores extremamente baixos para a probabilidade de descarte/marcação p. Como exemplo, considere uma conexão TCP cujo valor da janela de recepção seja extremamente alto, ou seja, a janela de transmissão, $SndWnd$ será igual a janela de congestionamento, $SndCwnd$. Considere que o tamanho dos segmentos seja igual a $SndMSS = 1500$ *bytes*, e o RTT seja basicamente constante e igual $RTT = 100ms$. Para se atingir uma vazão de 10Gbps, durante a fase de estabilização, o valor da janela deve ser de cerca de $SndCwnd = 83.333$ segmentos, e a taxa de perda deve ser de no máximo um pacote a cada um dos 5.000.000.000 pacotes enviados durante cerca de uma hora. O valor da probabilidade de descarte, na ordem de $p = 10^{-10}$ corresponde a uma taxa de erro de na ordem de no máximo 10^{-14}, valor este irreal para as redes atuais e inferior ao limite teórico de taxa de erro [43].

Algumas propostas de modificação do TCP Reno que conseguem obter maiores valores para a janela de transmissão com taxas de perdas mais realistas foram apresentadas. Em [43, 44] os autores propõem um protocolo, denominado HSTCP (*High Speed* TCP), que muda a dependência da janela de $1/\sqrt{p}$ para $1/p^{0,8}$. O STCP - *Scalable TCP* [45] e o FAST TCP [46] conseguiram obter tamanhos de janela ainda maiores, fazendo com que a dependência de suas janelas de transmissão seja de $1/p$.

Em [47] é feita a análise do comportamento do TCP Westwood - TCPW, e é verificado que para o mesmo valor de probabilidade de descarte p, a razão entre o valor da janela de congestionamento para o TCPW, denominada aqui por $SndWnd_{TCPW}$, e a janela do TCP Reno é dada por: $\frac{SndWnd_{TCPW}}{SndWnd} \cong \frac{1}{\sqrt{2p}}$ Isto implica que para os mesmos valores de p, o TCPW obterá maiores taxas de transmissão, e esta diferença acentua-se especialmente para valores pequenos de p, quando então, o TCPW obterá valores de janela bem maiores que o TCP Reno.

No TCP BIC, a taxa de transmissão é proporcional a $1/p^d$, onde $1/2 \leq d \leq 1$. Desta forma, ele é *TCP-friendly* quando a janela é pequena, e permite uma maior taxa de transmissão em ambientes com grande produto banda-atraso, onde é necessário grandes valores de janela.

2.3.5 Comportamento Oscilatório da Janela de Transmissão

A estratégia AIMD (*Additive Increase, Multiplicative Decrease*) utilizada pelo TCP Reno para governar o comportamento da sua janela de transmissão, é inerentemente oscilatória. O sistema não converge para um ponto de equilíbrio, e fica oscilando em torno deste. Isto significa que o TCP fonte oscila consideravelmente a sua taxa de transmissão, não estabilizando-a no valor da taxa de transmissão ideal, que é igual a sua porção justa de banda passante [48, 49]. Isto faz com que o uso da banda passante disponível também

oscile de forma significante. Desta forma, quando a carga na rede é pequena, tem-se períodos em que recursos são subutilizados e quando a carga na rede intensifica, tem-se a ocorrência do congestionamento e suas conseqüências [34].

Segundo [46], um algoritmo de controle de congestionamento pode ser desenvolvido em dois níveis. No nível macroscópico ou de fluxo, o projeto objetiva a alta utilização dos recursos, pequenos atrasos de fila, baixa taxa de perda, justiça e estabilidade. No nível de pacote, os objetivos definidos para o nível de fluxo devem ser implementados de acordo com as limitações impostas pelo controle fim-a-fim. No desenvolvimento do TCP Reno não houve a definição inicial dos objetivos a nível de fluxo, ou seja, o protocolo foi implementado diretamente no nível de pacote. Desta forma, as propriedades resultantes do nível de fluxos, tais como justiça, estabilidade, e equilíbrio entre o tamanho da janela e a probabilidade de descarte tiveram que ser verificados a posteriori.

A causa principal do comportamento oscilatório do TCP Reno está no seu projeto tanto em nível de fluxo quanto em nível de pacote [46]. Em nível de pacote, o incremento linear de um pacote por RTT é muito lento, enquanto que o decréscimo multiplicativo por cada evento de perda é muito drástico. Além disto, como o TCP utiliza um mecanismo binário como sinal de congestionamento (perdas de pacotes), a oscilação é inevitável, principalmente quando a carga, caracterizada pelo número de conexões ativas, é muito menor quando comparada com o produto banda-atraso. Em nível de fluxo, significa que para se obter um valor alto para a janela de congestionamento, são necessários valores extremamente baixos para a probabilidade de descarte/marcação, como apresentado na seção anterior. Além disto, a dinâmica é instável, o que faz com que as oscilações só possam ser reduzidas através da determinação precisa da probabilidade de descarte/marcação bem como de um projeto estável para a dinâmica dos fluxos. Finalmente, com o aumento de redes com grandes produto banda-atraso, o TCP Reno tornar-se-á o gargalo do desempenho destas redes [50, 51].

Vários protocolos tiveram seu projeto guiados pelos objetivos em nível de fluxo, e depois foram implementados em nível de pacote. Desta forma, as propriedades em nível de fluxo, em especial o equilíbrio entre o tamanho da janela e a probabilidade de descarte são garantidos. Dentre estes protocolos podem ser citados: o TCP Vegas [35, 36] e a sua variação Svegas - *Stabilized Vegas* [52]; o HSTCP - *High Speed* TCP [43, 44]; o STCP - *Scalable TCP* [45]; o FAST TCP [46] e o TCP Westwood+ apresentado em [53].

2.3.6 Baixo Desempenho em Redes com Grande Produto Banda-Atraso

Na Seção 2.3.3, foi apresentado o fato de que fluxos com RTT longo são penalizados em relação aos fluxos com RTT curto. Fluxos com RTT longo têm dificuldade em obter a sua

porção justa de banda passante, e aumentam sua taxa de transmissão mais lentamente que os fluxos com RTT curto.

Na Seção 2.3.4, foi apresentado que para se obter um valor alto para a janela de congestionamento, são requeridos valores irreais para a probabilidade de descarte/marcação, o que coloca uma grave limitação para os valores que as janelas de congestionamento podem atingir em ambientes reais de rede com grande produto banda-atraso [54].

Na Seção 2.3.5, a estratégia utilizada para governar o variação da janela do TCP Reno, é inerentemente oscilatória, tendo-se períodos em que recursos são subutilizados e outros em que ocorre o congestionamento.

Estes três problemas juntos fazem com que o TCP Reno seja o gargalo para o desempenho de redes com grande produto banda-atraso. Desta forma, tem-se que nestes tipos de rede, a utilização eficiente dos recursos fica seriamente comprometida, tornando-se difícil atingir grandes valores de janela. Além disto, a forma de crescimento da janela é muito lenta e o decréscimo é muito drástico, fazendo com que recursos fiquem subutilizados.

Vários protocolos foram desenvolvidos com o objetivo de aprimorar o desempenho do TCP para redes com grande produto banda-atraso. Dentre eles pode-se citar: o HSTCP - *High Speed* TCP [43, 44]; o STCP - *Scalable* TCP [45]; o FAST TCP [46], o TCP Westwood [55] e a sua versão estável, o TCP Westwood+ [53]; o SQRT [56]; o XCP -*eXplicit Control Protocol* [57] e o H-TCP: *TCP for high-speed and long-distance networks* [58].

2.3.7 Adoção de Perdas de Pacotes como Indicativo de Congestionamento

Apesar do TCP ter sido desenvolvido de forma a ser independente da tecnologia utilizada na camada de enlace, seu desempenho pode ser severamente degradado na presença de enlaces sem fio e nós móveis.

O TCP utiliza a ocorrência de perdas de pacotes como um indicativo de congestionamento; presunção esta adequada em redes cabeadas, mas não em redes sem fio onde a maioria das perdas ocorre devido a erros de transmissão.

Nos enlaces sem fio, as transmissões são sujeitas à altas taxas de erros ou BER (*Bit Error Rate*), que corrompem pacotes, resultando posteriormente no seu descarte. O TCP, entretanto, interpreta a perda destes segmentos apenas como uma indicação de congestionamento, reduzindo desnecessariamente sua taxa de transmissão [20].

Nas redes móveis, desconexões ocorrem com grande freqüência, seja devido ao bloqueio do sinal por obstáculos físicos ou devido às constantes movimentações dos nós móveis. Quando uma desconexão ocorre, pacotes ou ACKs podem ser perdidos, o que levará o TCP emissor a decrementar desnecessariamente sua janela de congestionamento. Além disto,

estas desconexões freqüentes geram *timeouts* em série. Como o TCP usa o mecanismo de *exponencial back-off* para retransmitir os segmentos, o problema de *timeouts* em série pode levar ao fim de uma conexão ou a sua inatividade, mesmo quando o nó móvel voltar a se reconectar [20].

Em [20], são discutidos os problemas relacionados a redes móveis e o protocolo TCP, e são apresentadas algumas das propostas existentes para superar estes problemas e melhorar o desempenho do TCP. Basicamente tais propostas diferem pela ciência ou não da existência de enlaces sem fio e de nós móveis.

2.4 Propostas para aprimorar o mecanismo de recuperação de perdas do TCP Reno

Esta seção descreve de forma suscinta as propostas do IETF apresentadas para superar as limitações do TCP Reno em se recuperar de forma efetiva quando ocorrem múltiplas perdas em uma mesma janela. O *TCP SACK* ameniza os problemas do TCP Reno provendo informações de quais e quantos pacotes foram recebidos corretamente pelo TCP receptor. O *TCP NewReno* faz o aprimoramento através da avaliação e diferenciação dos tipos de reconhecimentos recebidos. Finalmente o *TCP Limited Transmit* foi desenvolvido para melhorar o desempenho do TCP Reno para conexões com tamanho pequeno de janela.

2.4.1 Sack TCP

Uma das razões pela qual o TCP Reno apresenta problemas de desempenho quando múltiplos segmentos são perdidos em uma mesma janela é o fato do TCP emissor só poder obter informações sobre um determinado segmento quando um ACK é recebido. Se este ACK não reconhece todos os segmentos já enviados, o TCP emissor não tem como saber quantos, nem quais segmentos foram perdidos.

Para resolver este problema, foi proposto em [59] a utilização do SACK (*Selective Acknowledgment*). Com SACK, o receptor pode informar ao emissor quais segmentos foram recebidos corretamente, possibilitando ao emissor retransmitir apenas os segmentos que foram perdidos de fato.

SACK utiliza duas opções TCP: *SACK-permitted*, opção que só deve ser enviada em um segmento SYN, como um indicativo que a opção *SACK* pode ser utilizada quando a conexão for estabelecida; e a opção *SACK*, que deve ser enviada pelo TCP receptor para enviar reconhecimento seletivo para o TCP emissor dos segmentos que foram recebidos, caso tenha havido autorização prévia através da opção *SACK-permitted*.

A opção *SACK* possui uma lista de blocos em seqüência, correspondendo aos segmentos armazenados no *buffer* do receptor. Estes blocos são definidos na opção *SACK* como inteiros de 32 *bits*. Um segmento TCP com a opção *SACK* que especifica n blocos, vai ocupar $8 * n + 2$ *bytes*, onde os 2 *bytes* que são somados correspondem aos campos *kind* e *length* da opção *SACK*. O limite para o número máximo de blocos que podem ser enviados em uma opção *SACK*, é de 4 blocos, caso outras opções TCP também não estejam sendo usadas.

A seguir, são descritos o comportamento do receptor e do emissor dentro do contexto de SACK.

Comportamento do Receptor

Se o receptor receber no segmento SYN da conexão a opção *SACK-permitted* e possuir suporte a SACK, então ele deve responder com a opção *SACK* inclusa em todos os ACK's que não reconhecem o segmento com o maior número de seqüência armazenado no receptor. Caso não receba esta opção o receptor não deve enviar a opção *SACK* em seus ACK's.

Ao enviar uma opção *SACK*, o receptor deve respeitar as seguintes regras:

- O primeiro bloco da opção *SACK* deve especificar o bloco de dados contendo o segmento que disparou este ACK;
- O receptor pode incluir tantos blocos quanto possível;
- A opção *SACK* pode ser preenchida com os blocos *SACK* que foram enviados recentemente e que não são subconjuntos de um bloco *SACK* da opção que está sendo enviada. Isto permite que o receptor informe ao emissor a existência de blocos não contíguos que estão armazenados no receptor.

Como há uma limitação no número de blocos que uma opção *SACK* pode enviar, é muito importante que sempre seja informado em um bloco, o segmento que foi recebido mais recentemente para que o emissor possa ter informações do estado da rede e do estado do *buffer* do receptor [59].

Um outro ponto importante a ser destacado, é que o receptor pode, se necessário, descartar um segmento armazenado em seu *buffer* que já foi enviado em um bloco da opção *SACK*.

Na Tabela 2.1, é apresentado um exemplo da utilização de SACK. Este exemplo é uma combinação dos exemplos contidos em [59]. Assumindo-se que a borda esquerda da janela tem o valor 5000 e que os dados são transmitidos em uma rajada de 8 segmentos, cada um contendo 500 *bytes*. Supondo que o segundo e o sexto segmentos tenham sido perdidos e

Segmento	ACK	Primeiro Bloco	Segundo Bloco
5000	5500		
5500 (perdido)			
6000	5500	6000 - 6500	
6500	5500	6000 - 7000	
7000	5500	6000 - 7500	
7500 (perdido)			
8000	5500	8000 - 8500	6000 - 7500
8500	5500	8000 - 9000	6000-7500

Tabela 2.1: Exemplo de Utilização da opção SACK

os demais tenham sido recebidos com sucesso, os ACK's gerados são os apresentados na Tabela 2.1.

Comportamento do Emissor

Quando o emissor recebe um ACK que contém a opção *SACK*, ele pode armazenar estas informações para tomar decisões futuras sobre a retransmissão de um segmento.

O emissor deve acrescentar um *flag*, *SACKed*, a cada um dos segmentos armazenados em sua fila de segmentos transmitidos e ainda não reconhecidos. Este *flag* indica se este segmento foi recebido em um bloco da opção *SACK*.

Quando um ACK com a opção *SACK* é recebido, o emissor vai ligar os *flags* dos segmentos contidos nos blocos. Quando uma retransmissão for feita, o emissor não irá enviar os segmentos que estiverem com o *flag* ligado. Todo segmento que não estiver com o *flag* ligado e que for menor que o maior segmento que esteja com o *flag* ligado pode ser retransmitido.

No exemplo apresentado na Tabela 2.1, o TCP emissor ao receber estes ACK's saberá que deve retransmitir apenas os segmentos 2 e 6.

Se o *timeout* expirar e ainda restarem segmentos na fila do emissor com os *flags* ligados, então, significa que o receptor descartou estes segmentos do seu *buffer*. O emissor deve então desligar os *flags* de todos os segmentos, e retransmitir o segmento da borda esquerda da janela depois do *timeout*, mesmo que este segmento estivesse com o seu *flag* ligado. Os segmentos não são descartados da fila até que a borda esquerda da janela avance [59].

DSACK - Duplicate SACK

A opção *SACK* foi especificada para permitir que o TCP receptor pudesse informar ao TCP emissor o recebimento de segmentos fora de ordem. Entretanto, o TCP receptor ao receber segmentos duplicados não pode informar ao TCP emissor este fato utilizando a

opção *SACK*.

Para que a opção *SACK* possa também informar o recebimento de segmentos duplicados, foi apresentada em [60] uma extensão à esta opção que ficou denominada DSACK (*Duplicate-SACK*). Tal extensão não necessita de uma negociação separada entre um emissor e um receptor TCP que já tenham negociado a opção *SACK*. Isto significa que o TCP receptor pode enviar blocos *DSACK*, mesmo que o TCP emissor não tenha suporte à esta extensão. Neste caso, o TCP emissor simplesmente ignora os blocos *DSACK* enviados.

Quando o TCP receptor receber um segmento duplicado, ele deve enviar um ACK com a opção *SACK*, onde o primeiro bloco da opção *SACK*, deve ser um bloco *DSACK* que especifica qual foi o segmento duplicado que disparou o envio do ACK.

As regras para utilização da extensão DSACK pelo TCP receptor, são as seguintes:

- Um bloco *DSACK* só deve ser utilizado para informar o recebimento de um segmento duplicado pelo receptor dentre os segmentos recebidos mais recentemente;
- Cada segmento duplicado é informado em no máximo um bloco DSACK, a não ser que o receptor receba dois segmentos duplicados idênticos;
- O limite inferior de um bloco *DSACK* especifica o primeiro número de seqüência do segmento duplicado e o limite superior especifica o número de seqüência imediatamente seguinte ao último número de seqüência do segmento;
- Se o segmento duplicado fizer parte de um bloco maior de dados já armazenado no *buffer* do receptor, então o próximo bloco *SACK* deve conter este bloco maior;
- Outros blocos *SACK* como especificados em [59], também podem ser enviados depois dos blocos já descritos anteriormente.

Na Tabela 2.2 é apresentado um exemplo da utilização da extensão DSACK. Este exemplo foi retirado de [60]. Neste exemplo, o receptor recebeu um segmento fora de ordem, já que o terceiro segmento enviado pelo emissor foi perdido, o que levou ao envio de um ACK com a opção *SACK*. Só que este ACK, como também os ACK's anteriores também foram perdidos, fazendo com que o emissor retransmita o primeiro segmento. O TCP receptor então envia um ACK com um bloco *DSACK*, informando o recebimento de um segmento duplicado, juntamente com um bloco *SACK*.

Em [30], são discutidas possíveis utilizações de SACK na tomada de decisão em controle de congestionamento. Por exemplo, em redes sem fio, as perdas de segmentos não são apenas por congestionamento, mas também é freqüente a perda decorrente da corrupção do conteúdo do pacote. Com SACK, a retransmissão seletiva pode melhorar o desempenho destas redes.

Segmento Enviado	Segmento Recebido	ACK Enviado (Incluindo blocos SACK)
3000-3499	3000-3499	3500 (ACK perdido)
3500-3999	3500-3999	4000 (ACK perdido)
4000-4499	(perdido)	
4500-4999	4500-4999	4000, SACK=4500-5000 (ACK perdido)
3000-3499	3000-3499	4000, DSACK=3000-3500, SACK=4500-5000

Tabela 2.2: Exemplo de utilização da extensão DSACK

Um outro ponto importante é que com SACK a decisão de quando retransmitir um segmento muda para qual segmento deve ser retransmitido. Isto abre um leque de alternativas para aumentar a eficiência de algoritmos de controle de congestionamento.

Com SACK, o TCP emissor pode ter informações mais precisas para verificar que decidiu erroneamente que um segmento havia sido descartado, e desfazer esta decisão, voltando a janela para seu valor original. Com a extensão DSACK, o TCP emissor também pode verificar quando está retransmitindo desnecessariamente segmentos que já estão presentes no emissor.

2.4.2 TCP NewReno

Um dos grandes problemas com o TCP Reno na ausência de SACK é que o TCP emissor possui pouca informação sobre quais foram os segmentos perdidos, o que o impossibilita de tomar decisões adequadas durante a fase de *Fast Retransmit* [31].

Quando o TCP emissor recebe três ACK's duplicados, a fase *Fast Retransmit* é iniciada, retransmitindo o segmento perdido. Um ACK recebido depois da retransmissão pode fazer um reconhecimento total, ou pode ser apenas um reconhecimento parcial, ou seja, este ACK pode estar reconhecendo alguns dos segmentos enviados mas não todos.

Nos trabalhos apresentados em [27, 61], foi verificado que um reconhecimento parcial seria um forte indicativo de que mais de um segmento foi perdido, e sugeriu que o segmento indicado neste ACK deveria ser retransmitido como forma de recuperar mais de um segmento perdido, sem que se tivesse que esperar pela ocorrência de um *timeout*.

As alterações sugeridas em [27] ficaram conhecidas como TCP NewReno. Três anos após, algumas destas alterações foram propostas como um RFC com *status* experimental [62], e, posteriormente, foi publicado como padrão [31]. Mesmo quando SACK é utilizado, este algoritmo é recomendado, já que SACK só pode ser utilizado por conexões TCP onde o emissor e o receptor possuem suporte a SACK.

A versão modificada dos algoritmos *Fast Retransmit* e *Fast Recovery*, denominada de TCP NewReno, foi apresentada em [31]. Basicamente são feitas três modificações

em relação ao algoritmos apresentado na Seção 2.2.2. É acrescentada uma variável as já existentes, denominada *SndRecover*, cujo valor inicial é o número de seqüência inicial. Além disto, quando um ACK que reconhece novos dados é recebido após a retransmissão do segmento perdido, é verificado se ele é um reconhecimento total ou parcial, e mediante tal verificação uma decisão é tomada. Finalmente, é feita uma verificação, no Passo 1 e 5 descritos abaixo, para evitar que sejam feitas retransmissões desnecessárias, através da ativação de múltiplas fases de *Fast Retransmit*. Estas modificações fazem com que a fase de *Fast Recovery* seja iniciada com o recebimento dos três ACK's duplicados e termine quando ocorrer um *timeout* ou quando é recebido um ACK que reconheça todos os segmentos enviados, inclusive os que estavam em trânsito quando a fase de *Fast Recovery* foi inicializada. Os passos do algoritmo são descritos a seguir:

1. Quando o terceiro ACK duplicado é recebido, e o emissor ainda não se encontra na fase de *Fast Recovery*, deve ser verificado se o reconhecimento cumulativo recebido no ACK reconhece os segmentos com número de seqüência maiores que o valor armazenado na variável $SndRecover$. Se este for o caso, o valor da variável $SndSSTresh$ deve ser dado pela Equação (2.10), o valor do maior número de seqüência transmitido deve ser armazenado na variável $SndRecover$ e deve-se ir para o Passo 2. Caso contrário, deve-se ir para o Passo 4, ou seja, os algoritmos de *Fast Retransmit* e *Fast Recovery* não devem ser acionados novamente, nem tampouco a variável *SndSSThresh* deve ser alterada;

2. Neste passo, o algoritmo de *Fast Retransmit* deve ser acionado, retransmitindo o segmento perdido e alterando o valor da janela $SndCwnd$ de acordo com a Equação (2.12), de modo a atualizar a janela com o número de segmentos que deixou a rede e que estão armazenados no receptor;

3. A cada vez que um ACK duplicado adicional for recebido, $SndCwnd$ deve ser incrementado pelo valor de $SMSS$. Este incremento faz com que a janela de congestionamento seja atualizada para refletir que um segmento deixou a rede e foi recebido no destino. Depois disto, um novo segmento deve ser transmitido se permitido pelo novo valor de $SndCwnd$;

4. Quando for recebido um ACK que reconhece novos dados, é verificado se ele é um reconhecimento parcial ou total. Provavelmente, este ACK foi gerado devido a retransmissão realizada no Passo 2.

   Reconhecimento Total: o ACK recebido reconhece todos os segmentos já enviados, incluindo o que foi enviado com valor de seqüência $SndRecover$, ou seja, este ACK está reconhecendo todos os segmentos que foram enviados no período

entre a transmissão original do segmento que foi perdido e o recebimento do terceiro ACK duplicado. O valor da variável *SndCwnd* deve ser modificado de acordo com a Equação (2.16) e deve-se sair da fase de *Fast Recovery*. O valor mínimo é utilizado para impedir que seja enviada uma rajada de segmentos;

$$SndCwnd = min(SndSSThresh, PacketsOut + SndSMSS) \quad (2.16)$$

Reconhecimento Parcial: neste caso, o ACK recebido faz apenas um reconhecimento de apenas uma parte dos segmentos enviados, e mais segmentos devem ter sido perdidos. Desta forma, o valor de *SndCwnd* deve ser atualizado de acordo com o número de segmentos reconhecidos. O primeiro segmento que não foi reconhecido deve ser retransmitido e *SndCwnd* deve ser incrementado pelo valor de $SMSS$, de modo a refletir que um segmento deixou a rede, como no Passo 3. Depois disto, um novo segmento deve ser transmitido se permitido pelo novo valor de *SndCwnd*. A fase de *Fast Recovery* não deve ser abandonada, ou seja, caso outros ACKs duplicados sejam recebidos, o Passo 3 deve ser repetido. Caso o primeiro ACK parcial recebido durante a fase de *Fast Recovery* o temporizador deve ser inicializado;

5. Caso aconteça um *timeout*, o valor do maior número de seqüência transmitido deve ser armazenado na variável *SndRecover*, e deve-se sair da fase de *Fast Recovery*.

A verificação feita no Passo 1 e a inclusão do Passo 5 tem como principal objetivo evitar que sejam feitas retransmissões desnecessárias, através da ativação de múltiplas fases de *Fast Retransmit*. Este problema pode acontecer por exemplo, quando são retransmitidos três segmentos consecutivos após a ocorrência de um *timeout*, e serem recebidos três ACKs duplicados, cujo valor de ACK não cobre o número de seqüência armazenado na variável *SndRecover*. Neste caso, estes ACKs duplicados indicam apenas que retransmissões desnecessárias foram feitas e não são uma nova indicação de congestionamento.

2.4.3 TCP Limited Transmit

Dado que tanto o TCP SACK quanto o TCP NewReno necessitam receber três reconhecimentos duplicados para iniciar o algoritmo de *Fast Retransmit*, eles apresentam um desempenho insatisfatório quando as janelas de congestionamento são pequenas [63], uma vez que as perdas geralmente só são detectadas depois da ocorrência da expiração do intervalo de temporização. De uma maneira geral, apenas 4% das retransmissões ocorridas após a ocorrência da expiração do intervalo de temporização (RTO´s - *Retransmission TimeOut*) poderiam ser eventualmente evitadas pelo uso da opção SACK [63].

Quando a duração da conexão é curta, o TCP emissor não pode sondar precisamente a banda disponível, devido a pequena quantidade de dados que tem para enviar, o que ocorre tipicamente em transmissões de seções WEB. Resultados indicam que apenas 10% das conexões onde ocorrem retransmissões após a ocorrência da expiração do intervalo de temporização tem janelas de congestionamento maiores que 10 segmentos [33].

Para aprimorar a eficiência do TCP na presença de janelas de congestionamento pequenas, o mecanismo *Limited Transmit* foi proposto em uma RFC com *status experimental* [63]. No algoritmo *Limited Transmit*, um novo segmento pode ser transmitido quando dois reconhecimentos duplicados forem recebidos desde que a janela do receptor permita que seja enviado este segmento, e que a quantidade de segmentos em trânsito seja menor ou igual ao valor da janela de congestionamento mais dois segmentos.

A idéia principal é fazer com que mais reconhecimentos duplicados sejam recebidos e conseqüentemente possibilitar que o algoritmo *Fast Retransmit* possa ser iniciado. Com esta pequena alteração, 25% das retransmissões ocorridas devido a expiração do intervalo de temporização podem ser evitadas.

2.4.4 Avaliação das Propostas

Uma avaliação do TCP *Limited Transmit* é apresentada em [64]. Neste artigo é demonstrado que o uso deste algoritmo reduz a latência para a transmissão dos arquivos, já que reduz a ocorrência de *timeouts*. Apesar de ter sido desenvolvido com o objetivo otimizar o mecanismo de recuperação de perdas do TCP para conexões TCP de curta duração. Neste artigo é apresentado que conexões TCP de longa duração também se beneficiam, já que a redução significativa na ocorrência de *timeouts* faz com que a diminuição na latência seja ainda mais acentuada.

Em [11], foi verificado que o uso em conjunto do algoritmo *Limited Transmit* com TCP Reno (*reno-lr*), TCP NewReno (*newreno-lt*) e TCP Sack (*sack-lt*), pode melhorar o desempenho do TCP na recuperação de perdas. As três versões do TCP foram comparadas em ambiente dinâmico de rede fazendo-se simulações exaustivas utilizando o simulador de redes NS [65]. Um gerador de tráfego foi utilizado para gerar tráfegos específicos FTP e WEB baseados nos modelos das distribuições estatísticas que os descrevem. A carga foi variada para cada tipo de tráfego com o intuito de verificar a eficiência das variações TCP em diferentes estágios de congestionamento. Neste artigo, também foi verificado que conexões TCP de curta, bem como de longa duração beneficiam-se com o uso do TCP *Limited Transmit*.

No algoritmo *Fast Retransmit* apresentado em [31], apenas um único pacote é retransmitido depois que um ACK parcial é recebido. A idéia é utilizar um mecanismo conservativo para minimizar retransmissões desnecessárias de pacotes, retransmitindo-se

um único segmento perdido por RTT. Desta forma, quando vários segmentos são perdidos em uma única janela, o TCP emissor irá se recuperar das perdas apenas após um atraso considerável. Por este motivo, esperava-se que o TCP que apresentasse o melhor desempenho seria o que fizesse uso em conjunto do algoritmo de *Limited Transmit* com o TCP SACK, *sack-lt*. Entretanto, o que apresentou o melhor desempenho, reduzindo o número de ocorrências de *timeout* e com maiores valores de *goodput* foi o *newreno-lt* (uso em conjunto do algoritmo de *Limited Transmit* com o TCP NewReno).

Uma justificativa para o NewReno ter um melhor desempenho é que existem evidências de que as informações retornadas na opção SACK não estão sendo utilizadas pelo TCP emissor para fazer as retransmissões dos segmentos perdidos [66]. Desta forma, mesmo quando há suporte a SACK, o TCP emissor terá o mesmo comportamento do TCP Reno quando múltiplas perdas acontecem, e assim, um comportamento inferior ao TCP NewReno. Uma possível é o fato de que quando SACK foi definido em [59], não foram especificados mecanismos de controle de congestionamento específicos para serem utilizados em conjunto com SACK. Assim, apesar de terem suporte a SACK, algumas implementações não usam as informações contidas nos blocos SACK para decidir quais segmentos devem ser retransmitidos.

Em [67] é apresentada o TCP FACK, uma proposta que utiliza os blocos SACK para fazer uma estimativa mais precisa sobre a quantidade de dados que está em trânsito, e assim, determinar mais corretamente a taxa de transmissão mais adequada com o estado da rede. Em [68], é apresentada uma proposta de um mecanismo conservativo de recuperação de perdas para o TCP. A idéia é substituir o algoritmo de *Fast Recorevey* proposto em [7] por um algoritmo que utiliza a opção SACK para determinar quais e quantos segmentos devem ser retransmitidos na ocorrência de múltiplas perdas.

Capítulo 3

Gerenciamento de Filas em Redes TCP/IP

Os mecanismos de controle de congestionamento do protocolo não são suficientes para prover serviços de boa qualidade em todas as situações, como também não impedem que o congestionamento ocorra, dado que outros protocolos não diminuem a taxa de transmissão na presença de congestionamento. Desta forma, mecanismos adicionais são necessários para se obter controle de congestionamento fim-a-fim.

Os mecanismos de controle de congestionamento nos roteadores monitoram o tamanho da fila, podendo detectar o congestionamento incipiente e tomar decisões sobre a notificação do congestionamento e o descarte de pacotes. Dentre os mecanismos de controle de congestionamento presentes nos roteadores estão os algoritmos de gerenciamento de filas, que descartam pacotes quando necessário.

Neste capítulo, são discutidas políticas de gerenciamento de filas em redes TCP/IP. É apresentada, primeiramente, a mais simples das políticas denominada *tail drop*, e são levantados seus principais problemas. Na seção 3.2, discorre-se sobre a necessidade de utilização de um novo mecanismo de gerenciamento de filas denominado gerenciamento ativo de filas ou AQM. Na seção 3.3 são discutidos os requisitos que as políticas de gerenciamento ativos de filas devem atender. Na seção 3.4 é apresentada RED, que é a política padrão de AQM utilizada na Internet. Na Seção 3.5, são descritas algumas políticas de AQM propostas para aprimorar RED: FRED, ARED, FPQ, BLUE e sua variação SFB. Na Seção 3.6, são introduzidas algumas políticas analíticas de AQM, desenvolvidas utilizando-se a Teoria da Otimização e a Teoria do Controle. Na Seção 3.7, o mecanismo *Explicit Congestion Notification* (ECN), que permite dissociar a notificação do congestionamento do descarte de pacotes é introduzido.

3.1 Tail Drop

A forma mais simples de se fazer o gerenciamento do tamanho das filas dos roteadores é conhecida como *tail drop*. Nesta técnica, existe um valor máximo para o tamanho da fila, em termos de número de pacotes. O roteador aceita pacotes até que o valor máximo de pacotes na fila seja atingido, depois disto os pacotes seguintes serão descartados, ou seja, pacotes são aceitos enquanto existir espaço na fila [69].

Esta técnica foi utilizada por muitos anos na Internet por ser muito simples, no entanto, apresenta algumas deficiências:

Monopólio: *tail drop* permite que uma simples conexão ou um conjunto mínimo de fluxos monopolize o espaço na fila, enquanto outras conexões têm seus pacotes descartados por falta de espaço. Um outro problema relacionado à justiça, é que *tail drop* não faz descarte de pacotes de um tráfego proporcional a sua utilização da fila. Desta forma, um tráfego pode ter seus pacotes descartados mesmo sem ter utilizado nenhum recurso da fila. Além disto, como apenas alguns fluxos monopolizam a fila, os fluxos que transmitem em rajada serão prejudicados, já que encontrarão a fila cheia e terão todos os seus pacotes descartados, e portanto, sua taxa de perda será maior;

Filas Cheias: *tail drop* permite a ocupação total da fila por muito tempo, dado que sinaliza congestionamento apenas quando a fila está cheia. Quando a fila está cheia, pacotes de tráfego em rajada são descartados podendo causar injustiças, como descrito anteriormente. Além disto, o problema de filas cheias favorece a ocorrência do fenômeno de sincronização global, que ocorre quando diversos fluxos TCP tem pacotes perdidos, e assim, reduzem simultaneamente sua taxa de transmissão.

3.2 Gerenciamento Ativo de Filas - AQM

Com o crescimento da Internet e conseqüentemente do seu tráfego, os problemas inerentes à política de gerenciamento de filas *tail drop* tornaram-se mais críticos.

Duas alternativas foram apresentadas para resolver a questão de monopólio da política *tail drop*: descarte aleatório de um pacote ou descarte do primeiro pacote da fila [69]. Em ambas as propostas, um pacote é removido da fila para dar espaço a um pacote que chega, quando a fila está cheia. Com o descarte aleatório, um pacote é escolhido para ser o removido da fila de forma randômica, o que tem um custo elevado ($O(n)$), onde n é o tamanho da fila). Estas políticas, no entanto, não resolvem os problemas de *tail drop*.

O principal objetivo de se utilizar filas com *buffers* em redes é o de se permitir que dados enviados na forma de rajadas possam ser absorvidos temporariamente para serem depois transmitidos, aumentando, assim, a vazão. Caso as filas estejam sempre cheias

ou "quase" cheias, não haverá espaço suficiente para absorver as rajadas. Aumentar o tamanho das filas também não é uma alternativa ideal, dado que aumentaria o atraso. Desta forma, o tamanho máximo das filas deve refletir o tamanho da rajada máxima que se deseja absorver. Deve-se, também, manter um tamanho médio de fila baixo para diminuir o atraso fim-a-fim dos pacotes. Flutuações no tamanho da fila devem ser permitidas para absorver os fluxos que transmitem em rajada e também para não penalizar os congestionamentos passageiros.

Além do problema de se manter o tamanho médio da fila baixo para lidar com tráfego em rajada, é necessário, também, minimizar o número de pacotes descartados. Quando a política de gerenciamento de filas utilizada é *tail drop*, o emissor só reduz a sua taxa de transmissão de pacotes quando é notificado de congestionamento através do descarte de pacotes, que só ocorre quando a fila está cheia, ou seja, quando o roteador já está sem recursos. O intervalo de tempo entre o momento do descarte do pacote pelo roteador até o momento que o emissor descobre que seu pacote foi descartado é considerável e o emissor durante este período continua transmitindo pacotes que também serão descartados, dado o congestionamento existente. Assim, são necessários mecanismos que evitem o máximo possível o descarte de pacotes, dado que estes consomem recursos da rede. O emissor precisa, conseqüentemente, ser notificado de que deve diminuir a sua taxa de transmissão antes que haja o transbordo da fila.

Para se manter um tamanho médio da fila baixo, pacotes devem ser descartados de forma aleatória antes que a fila fique cheia, notificando os nós finais sobre a ocorrência de congestionamento, facilitando, então, que estes evitem a formação de congestionamento e, conseqüentemente, o transbordo das filas. Este mecanismo de gerenciamento de filas permite que os roteadores controlem quando e como devem descartar pacotes, tentando solucionar os problemas de monopólio e de filas cheias apresentados por *tail drop*. Tal mecanismo recebe a denominação de gerenciamento ativo de filas, ou *Active Queue Management (AQM)*.

O propósito de AQM é a notificação do congestionamento incipiente, de forma a permitir que os emissores TCP possam reduzir sua taxa de transmissão antes que as filas transbordem, evitando assim, a degradação do desempenho.

3.3 Requisitos para Políticas de AQM

Nesta seção, os requisitos principais das políticas de Gerenciamento Ativo de Filas são destacadas. Uma das formas mais comuns de combate ao congestionamento é fazer o descarte de pacotes quando necessário. O descarte de pacotes deve ser feito de acordo com alguma política definida para prevenir ou combater o congestionamento. Tais políticas devem fazer o descarte seletivo de pacotes baseados em alguns requisitos são apresentados

a seguir [70].

Quando uma perda é detectada por uma conexão TCP, a sua taxa de transmissão é diminuída no mínimo pela metade. Se em uma política de AQM, um pacote for descartado de cada conexão TCP ativa, tem-se a redução simultânea das suas taxas de transmissão, e como conseqüência, uma baixa utilização da rede. Tal fenômeno é denominado sincronização global [8]. A ocorrência deste fenômeno, apesar de ser raro, acarreta uma degradação da rede e como tal deve ser evitado, através do descarte de pacotes de forma seletiva e limitando o número de fluxos afetados por este descarte.

Uma boa política de AQM tem que garantir que conexões que compartilham as mesmas condições de rede tenham uma taxa de perda de pacotes proporcional a sua utilização da fila de pacotes. Além disto, quando múltiplas perdas em uma conexão forem inevitáveis, a política de AQM deve prevenir que tais perdas sejam em rajada de forma a permitir a ativação dos mecanismos de recuperação de perdas do TCP, evitando, assim, as danosas conseqüências da ocorrência de um *timeout*. Estas políticas também têm que garantir que fluxos que estejam se comportando de forma errada, ou seja, que estejam utilizando mais recursos do que deveriam utilizar, sejam penalizados tendo uma taxa de descarte de pacotes maior que os fluxos "bem comportados". Um outro ponto importante relacionado à justiça, diz respeito ao tipo de tráfego. Fluxos que transmitem em rajada não devem ser penalizados pois enviam muitos dados, em um espaço de tempo curto.

Para que se tenha o maior aproveitamento possível dos recursos da rede, pacotes que já utilizaram uma grande quantidade de recursos em comparação com outros pacotes, ou que já percorreram um longo caminho devem ter prioridade sobre outros pacotes, o que significa que só devem ser descartados em último caso. Este requisito apesar de desejável, é um dos mais difíceis de se obter, devido a dificuldade em se mensurar a quantidade de recursos que um determinado pacote já consumiu, ou o caminho percorrido por este pacote.

Outros dois objetivos de desempenho para políticas de AQM são apresentadas em [10]: eficiente utilização da fila e a garantia de baixo retardo e de baixa variação no retardo. Utilizar a fila de forma eficiente significa evitar que existam períodos alternados de transbordo e de ociosidade do servidor. O primeiro caso resulta na perda de pacotes, retransmissões desnecessárias, como também penaliza tráfegos em rajada, enquanto o segundo caso leva à subutilização do enlace. Para que se tenham baixos valores de retardo deve-se ter tamanhos pequenos de fila, o que, por outro lado, pode gerar a subutilização do enlace. Além disto, é desejável que grandes variações no tamanho da fila sejam evitados para prevenir variações no retardo (*jitter*), o que é prejudicial para algumas aplicações de tempo real.

A principal função dos roteadores é enviar pacotes para a rota adequada, o que deve ser realizado o mais rápido possível. Sendo assim, um algoritmo que representa uma política

de AQM deve ser o mais simples e eficiente possível para não degradar o desempenho da rede. Isto implica que o número de parâmetros, os critérios, a dinâmica do algoritmo e as tarefas a serem realizadas devem ser levadas em consideração na definição do algoritmo.

A política de AQM deve ser definida de tal forma que o aumento do tráfego e do número de conexões não afete o desempenho provido pelas políticas de descarte de pacotes. Isto implica em dizer que a política adotada deve permitir o crescimento do sistema sem impor limitações.

Finalmente, a política de AQM deve ser robusta, ou seja, definida de tal forma que o sistema deve manter-se estável e com o seu desempenho esperado independente das condições da rede, que incluem variações no RTT (*Round Trip Time*) e no tráfego.

Em resumo, a política de AQM deve prevenir e controlar o congestionamento, ser escalável, simples, justa, evitar o fenômeno de sincronização global e ainda determinar o valor ideal da probabilidade de descarte/marcação que estabilize o sistema e que maximize as taxas de transmissão e minimize o tamanho da fila sujeito às condições da rede, de forma a evitar perdas de pacotes desnecessárias.

Apesar de todos os requisitos descritos anteriormente serem importantes, quais deles devem ser utilizados no projeto de uma política de AQM é uma questão que precisa ser respondida para que uma política efetiva possa ser definida. A resposta para esta questão irá depender basicamente do tipo de tráfego e das necessidades das aplicações que se deseja atender.

3.4 RED - Random Early Detection

Nesta seção, a política de AQM recomendada pelo IETF para a Internet, é apresentada, e é feita uma análise das suas vantagens e deficiências.

O algoritmo *Random Early Detection* RED [8] foi desenvolvido com os seguintes objetivos: prevenção de congestionamento e um baixo tamanho médio das filas, evitar o problema da sincronização global, prevenir a penalização de tráfego em rajadas e manter um limite máximo para o tamanho médio da fila

RED faz a prevenção do congestionamento através da comparação do tamanho médio da fila com dois limiares, e de acordo com o resultado desta comparação uma decisão é tomada. Caso o congestionamento incipiente seja detectado, uma notificação de congestionamento é gerada através da marcação ou descarte de um pacote. O algoritmo é apresentado na subseção seguinte. É importante destacar que RED especifica que uma notificação de congestionamento deve ser feita para o nó que está causando o congestionamento, mas tal notificação pode ser feita através do descarte de pacotes ou da marcação dos pacotes caso o protocolo de transporte tenha suporte para isto. O nó ao receber a notificação de congestionamento deve agir da mesma forma tenha sido seu pacote descartado

ou marcado.

Para evitar o problema da sincronização global e prevenir a penalização dos tráfegos em rajada, RED só notifica uma fração aleatoriamente selecionada dos usuários. A probabilidade de marcação ou descarte de um pacote de uma determinada conexão é proporcional a utilização da banda desta conexão.

Um dos objetivos dos mecanismos AQM é controlar o tamanho médio das filas mesmo quando não há cooperação de protocolos da camada de transporte, ou quando o nó não reduz o tráfego quando há indicação de congestionamento, de modo a não penalizar tráfegos em rajada e também não produzir longos atrasos. RED consegue fazer isto através da marcação ou do descarte dos pacotes que chegam quando o tamanho médio da fila excede um valor máximo.

O algoritmo RED consiste de duas etapas. Na primeira, é feita a estimativa do tamanho médio da fila, e na segunda é tomada a decisão de marcar ou descartar os pacotes que chegam.

O cálculo do tamanho médio das filas é feito utilizando-se uma fórmula de média ponderada móvel (*weighted moving average*), ou seja, um filtro passa baixa é utilizado no cálculo do tamanho médio da fila para filtrar do cálculo da média a ocorrência de congestionamento passageiro. O valor obtido para o tamanho médio da fila é utilizado na segunda fase do algoritmo para decidir o que fazer com os pacotes.

O valor do tamanho médio da fila é comparado com dois limiares min_{th} e max_{th}. Estes limiares são utilizados para definir três zonas. Se o valor estiver abaixo de min_{th} o algoritmo está na zona normal de operação e nenhum pacote é marcado ou descartado, e todos os pacotes que chegam são aceitos. Caso o valor esteja entre os dois limiares, o algoritmo está na zona de prevenção de congestionamento e cada pacote que chega é marcado ou descartado com uma probabilidade p_a. Se o valor estiver acima de max_{th}, o algoritmo está na zona de controle de congestionamento e todos os pacotes que chegam são descartados. A probabilidade de um pacote de uma determinada conexão ser marcado durante a zona de prevenção de congestionamento é proporcional a utilização da banda por esta conexão. A Figura 3.1 apresenta o comportamento do algoritmo RED, através das suas três zonas de atuação definidas a partir dos limiares min_{th}, max_{th}, e da probabilidade max_p.

A probabilidade p_a cresce de acordo com dois fatores: um contador que é incrementado a cada vez que um pacote chega no roteador e é armazenado na fila, e que é reinicializado quando um pacote é descartado; e uma probabilidade intermediária p_b, cujo valor máximo, max_p, é atingido quando o valor do tamanho médio da fila é igual a max_{th}. Quanto maior o crescimento do tamanho médio da fila, maior é a probabilidade de descarte p_a. Se o tamanho médio da fila é mantido constante, todos os pacotes que chegam têm a mesma probabilidade de descarte. Desta forma, RED descarta pacotes em proporção a

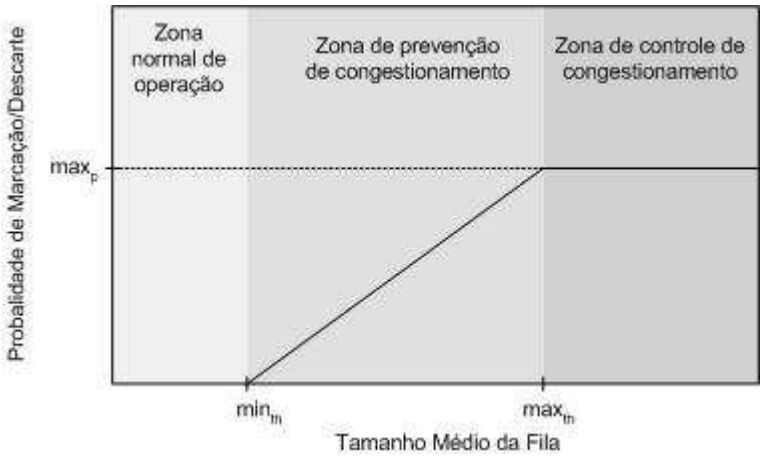

Figura 3.1: Comportamento do algoritmo RED

utilização de banda da conexão. O valor *default* para o limite superior da probabilidade p é $max_p = 0.1$.

O cálculo do tamanho médio da fila utiliza uma fórmula de média ponderada móvel com peso, W_q, para filtrar do cálculo congestionamentos passageiros. No algoritmo apresentado a seguir, pode-se verificar que o cálculo do tamanho médio da fila leva em consideração o período em que a fila ficou vazia, através da estimativa do número de pacotes que foram transmitidos pelo roteador durante o período em que a fila ficou vazia (*idle period*).

O valor de W_q determina a forma como o algoritmo irá responder as mudanças ao tamanho da fila. Se W_q for muito alto, então a média não conseguirá filtrar os congestionamentos passageiros. Se ao contrário, o valor escolhido for muito baixo, a reação ao congestionamento é muito lenta. A escolha do valor correto depende do valor de min_{th} e da quantidade de pacotes em rajada que se deseja suportar [70]. Desta forma, dado a quantidade de tráfego em rajada que se deseja suportar em número de pacotes, P, e min_{th}, W_q pode ser calculado da seguinte forma: $P + 1 + \frac{(1-W_q)^{(P+1)}-1}{W_q} < min_{th}$. Isto significa que se W_q for escolhido de forma adequada, o roteador pode aceitar uma rajada de até P pacotes. Por exemplo, se $min_{th} = 5$ pacotes e $P = 50$ pacotes, o valor de W_q deve ser $W_q \leq 0.0042$. Um valor recomendado para o peso da fila é $W_q \leq 0.002$ [71].

A seguir são descritas as variáveis, os parâmetros fixos e as funções utilizadas no algoritmo RED. O Algoritmo 3.1 apresenta em detalhes os passos de RED.

Parâmetros Fixos:

W_q: peso da fila;
min_{th}: limiar mínimo;
max_{th}: limiar máximo;

max_p: valor máximo de p_b quando $Q_{avg} = max_{th}$;

Demais Variáveis e Funções:
Q_{avg}: tamanho médio da fila;
Q_l: tamanho atual da fila;
p_a: probabilidade atual de marcar um pacote;
p_b: função linear do tamanho médio da fila, que varia entre 0 e max_{th};
$Time$: valor atual do tempo;
Q_{time}: período de tempo em que a fila esteve vazia;
$Count$: número de pacotes que chegaram depois o último pacote foi marcado;
Num_{Pkg}: estimativa do número de pacotes transmitidos durante o período em que a fila esteve vazia;
$f(t)$: uma função linear no tempo t.

3.4.1 Avaliação RED

Desde que foi apresentado no artigo [8], várias pesquisas foram realizadas com o propósito de se sugerir valores para os seus parâmetros [8, 71–73], propor modificações [74, 75], recomendar o seu uso [42, 69, 76] e fazer avaliações [77–82].

Dentre as propostas para os seus parâmetros, uma que merece destaque é a apresentada em [73,83]. Esta proposta faz uma recomendação para que seja utilizada a versão "*gentle*" de RED. Nesta versão, o valor da probabilidade de descarte p_a varia linearmente de 0 até max_p, quando o tamanho médio da fila varia de min_{th} até max_{th} e varia linearmente de max_p até 1, quando o tamanho médio da fila varia de max_{th} até $2 * max_{th}$. Esta opção é recomendada para que se tenha um melhor comportamento de RED, fazendo com que fique mais robusto em relação aos valores definidos para os parâmetros max_{th} e max_p. A Figura 3.2 apresenta o comportamento de RED com a opção "*gentle*".

A seguir são discutidas as vantagens de RED que fizeram com que fosse a política de AQM mais indicada para ser implementada nos roteadores como também os seus pontos fracos ou desvantagens que vem sendo discutidos em vários artigos.

Vantagens

1. *Eficiente para evitar congestionamento*: Se W_q for inicializado com um valor adequado, RED garante um limite no tamanho médio da fila, mesmo quando não existe cooperação entre as fontes. Isto faz com que rajadas e congestionamentos passageiros sejam suportados;

Algoritmo 3.1 Random Early Detection - RED

ProcedimentoRED($W_q, min_{th}, max_{th}, max_p$)
// Inicialização
$Q_{avg} \Leftarrow 0$;
$Count \Leftarrow -1$;
Para cada pacote que chega **faça**
// Cálculo do tamanho médio da fila Q_{avg}
Se a fila não estiver vazia **então**
$Q_{avg} \Leftarrow Q_{avg} * (1 - W_q) + W_q * Q_l$;
Senão
$Num_{Pkg} \Leftarrow f(Time - Q_{time})$;
$Q_{avg} \Leftarrow Q_{avg} * (1 - W_q)^{Num_{Pkg}}$;
Se $min_{th} \leq Q_{avg} < max_{th}$ **então**
$Count \Leftarrow Count + 1$;
// Cálculo da probabilidade p_a
$p_b \Leftarrow max_p * (Q_{avg} - min_{th})/(max_{th} - min_{th})$;
$p_a \Leftarrow p_b/(1 - Count.p_b)$;
// Com probabilidade p_a
Marca o pacote que chegou;
$Count \Leftarrow 0$;
Senão
Se $max_{th} \geq Q_{avg}$ **então**
Marca o pacote que chegou;
$Count \Leftarrow 0$;
Senão $Count \Leftarrow -1$;
FimPara
Quando a fila ficar vazia **faça**
$Q_{time} \Leftarrow Time$;
FimProcedimentoRED

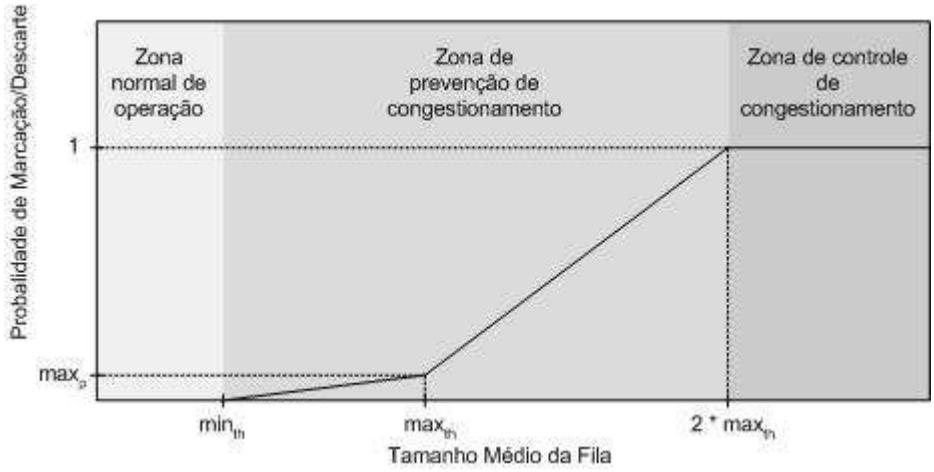

Figura 3.2: Comportamento do algoritmo RED com a opção "*gentle*"

2. *Previne a sincronização global*: Em RED, a taxa de descarte de pacotes cresce com o aumento do congestionamento e a probabilidade de descarte de um pacote que chega de um determinado fluxo é proporcional ao uso do enlace por este fluxo;

3. *Boa razão vazão/retardo*: como RED consegue controlar o tamanho médio da fila, enquanto acomoda congestionamentos passageiros, garante alta vazão e um baixo atraso em redes de alta velocidade onde as conexões TCP utilizam janelas grandes;

4. *Reage bem ao tráfego em rajada*: RED garante um limite para o tamanho médio da fila o que faz com que consiga suportar tráfegos em rajada;

5. *Identifica conexões mal comportadas*: como os pacotes que são descartados durante o congestionamento são escolhidos de forma aleatória, e como a probabilidade de descarte de um pacote que chega de um determinado fluxo é proporcional ao uso do enlace por este fluxo, pode-se identificar quais são os fluxos que tiveram mais pacotes descartados, que são justamente os mais mal comportados;

6. *Usa operações simples e pouca memória*: RED utiliza apenas variáveis globais, não armazena informações para cada fluxo, o que faz com que ele gaste pouca memória. Além disto, o algoritmo utiliza operações simples, o que faz com que ele também seja rápido;

7. *Pode ser implementado gradualmente*: RED é simples, não requer nenhuma modificação aos protocolos de transporte e não exige que todos os roteadores o implementem, o que faz com que ele possa ser implementado de forma gradual. Atualmente a grande maioria dos roteadores comerciais já o implementa.

Problemas e Desvantagens

1. *Injustiça*: Um dos grandes problemas com RED é a injustiça. A seguir são apresentados as deficiências de RED que acarretam este problema:

   - *Penaliza conexões TCP com janelas pequenas*: O grau de tolerância a perda de pacotes depende basicamente do tamanho da janela. Enquanto conexões com uma janela grande se recuperam de múltiplas perdas em apenas um RTT, uma conexão com uma janela pequena terá que esperar por um *timeout* relativamente grande de um segundo ou mais para se recuperar de apenas uma perda. Como RED escolhe um pacote para ser descartado de forma aleatória, e este pacote pode ser de uma conexão que possui uma janela pequena, tal conexão será muito mais penalizada do que seria uma outra conexão com uma janela maior [75].
   - *Voltado para tráfegos adaptativos*: O TCP é sensível ao congestionamento, ou seja, ele diminui a sua taxa de transmissão na presença de congestionamento. Em compensação se existir um tráfego não adaptativo, ou seja, um tráfego que envia a uma taxa alta independente da existência de perdas, este tráfego fará com que aumente a taxa de descarte em todas as conexões, fazendo com que as adaptativas reduzam sua taxa de transmissão enquanto a não-adaptativa conseguirá obter uma maior banda [75];
   - *Realiza descarte não proporcional*: Quando um pacote de uma nova conexão é aceito, a probabilidade de descarte dos futuros pacotes a serem transmitidos pelas outras conexões que já estavam ativas aumenta, mesmo que estas conexões consumam menos banda. Este fato pode causar temporariamente o descarte não proporcional mesmo entre tráfegos idênticos.

     Em RED, a taxa de descarte de pacotes de um pacote que chega de um determinado fluxo é proporcional ao uso do enlace por este fluxo. Isto faz com que todos os fluxos obtenham a mesma taxa de descarte. Desta forma, uma conexão que esteja utilizando menos banda do que teria direito, estaria sendo penalizada.

2. *Dificuldade em se determinar corretamente os limiares min_{th} e max_{th}*: Os valores adequados dependem do valor do tamanho médio da fila que se deseja manter, e este por sua vez, depende do tipo de tráfego. Como dificilmente se tem a priori as características do tráfego com que se vai trabalhar, fica quase impossível determinar quais são os valores ótimos dos limiares. Se o tráfego for em rajadas o valor de min_{th}, deve ser alto o suficiente para manter uma alta taxa de utilização da rede. O valor de max_{th} deve ser determinado de acordo com o valor médio máximo de

atraso que deve ser permitido nos roteadores. Mais uma vez, determinar este valor depende do tipo de aplicação que estará sendo usada na rede. Por exemplo, se o valor de max_{th} for muito alto, aplicações interativas, extremamente sensíveis a atrasos serão bastante penalizadas. Empiricamente foi comprovado que o valor de max_{th} deve ser igual a pelo menos três vezes o valor de min_{th} [71]. Além isto, a diferença entre max_{th} e min_{th} deve ser maior que o incremento típico do cálculo da média em um RTT para que RED funcione mais eficientemente, como também para evitar a sincronização global. Esta é a argumentação mais usada contra a utilização de RED, já que uma má escolha dos parâmetros pode levar a um comportamento inadequado de RED.

Um outro problema que decorre da má definição dos limiares é que RED pode ter um comportamento similar ou até pior que *tail drop*. É por este motivo que [77] e [81] desaconselham o desenvolvimento e utilização de RED.

Em [78] é feito um estudo sobre os efeitos de se utilizar RED em enlaces que transportam apenas tráfegos Web, e observaram que mais uma vez devido a dificuldade de se determinar corretamente os limiares, RED não oferece vantagens sobre *tail drop* em relação ao tempo de resposta para os usuários.

3. *Grande quantidade de fluxos ativos*: Quando uma grande quantidade de fluxos estão ativos, o tráfego agregado é basicamente em rajadas o que degrada o comportamento de RED, dado que o tamanho médio da fila varia rapidamente antes que RED possa reagir. Dependendo da agressividade destas rajadas e da quantidade de *buffers* disponíveis no enlace, podem ocorrer várias perdas, levando a uma eventual baixa utilização do enlace [29]. Para minimizar este problema, deve-se utilizar uma quantidade adequada de *buffers* que é de duas vezes o produto banda-retardo(*bandwith*delay*). Esta solução já vem sendo implementada em vários roteadores comerciais. O problema é que quando este produto é muito alto, a utilização do dobro deste produto em número de *buffers* pode aumentar o atrasso fim-a-fim como também a variação do retardo (*jitter*), prejudicando assim aplicações interativas. Além disto, roteadores que possuem poucos recursos não podem adotar tal solução [29].

   Além deste, um outro problema pode ser causado quando o número de fluxos ativos é muito grande. Uma das vantagens anunciadas de RED, é que previne o fenômeno de sincronização global, porém esta não é uma verdade absoluta. A probabilidade de descarte de RED é uma função linear do tamanho médio da fila, Q_{avg}, e como este valor varia muito quando há um grande número de fluxos ou conexões ativas, a probabilidade de descarte varia consideravelmente em um curto espaço de tempo fazendo com que RED falhe em marcar os pacotes de forma aleatória, não sendo

tão eficiente nestes casos em prevenir o fenômeno de sincronização global.

4. *Ponto operacional ideal*: Pode-se afirmar que RED só atinge um ponto operacional "ideal" quando possui *buffers* suficientes e quando seus limiares estão corretamente definidos.

3.5 Políticas de AQM Propostas para Aprimorar RED

Entre as desvantagens de RED estão a dificuldade em se definir corretamente os seus limiares, a injustiça, a dificuldade em atingir um ponto operacional ideal e os problemas decorrentes de uma grande quantidade de fluxos ativos. Vários algoritmos AQM foram propostos para tentar solucionar estas desvantagens de RED. Nesta seção são apresentados alguns destes algoritmos. ARED - *Adaptive Random Early Drop* e Blue tentam minimizar a dificuldade em se definir corretamente seus limiares. FRED - *Flow Random Early Drop* tem como objetivo reduzir o problema de injustiça de RED. FPQ - *Flow Proportional Queuing* tenta resolver os problemas relacionados com RED quando existe uma grande quantidade de fluxos ativos.

3.5.1 ARED - Adaptive RED

ARED - *Adaptive Random Early Drop* é um algoritmo de gerenciamento ativo de filas que propõe alterações ao algoritmo original RED para tentar reduzir um dos seus principais problemas: a dificuldade em se definir corretamente os seus limiares. Este algoritmo foi inicialmente proposto em, [29, 74] e modificado posteriormente em [84]. A idéia de ARED é inferir quando RED deve ficar mais ou menos agressivo, através da verificação do comportamento do tamanho médio da fila, Q_{avg}.

Quando o número de fluxos ativos é pequeno, RED deve ser mais conservativo, ou seja, o tamanho médio da fila deve oscilar ao redor de max_{th}, levando a diminuição do valor de max_p. Com isto, pretende-se evitar a baixa utilização do enlace.

Quando o número de fluxos ativos é muito grande, RED deve ser mais agressivo, ou seja, o tamanho médio da fila deve oscilar ao redor do limiar min_{th}, o que significa que o valor de max_p deve ser aumentado. Evita-se, portanto, a perda de pacotes e a notificação determinística, que ocorre quando Q_{avg} atinge max_{th} e todos os pacotes são marcados ou descartados com probabilidade max_p. Quando os limiares de RED são mantidos conservativos e existe um grande número de fluxos ativos, não são enviadas notificações suficientes para os emissores; o que faz com que a fila fique permanentemente em transbordo, tornando o comportamento de RED similar a *tail drop*.

Para ajustar RED, ARED varia max_p de dois fatores constantes α e β dependendo de qual limiar é cruzado. Em fase de baixo congestionamento mantém-se a probabilidade

Algoritmo 3.2 Adaptative Random Early Detection

ProcedimentoARED()
Se ($min_{th} \leq Q_{avg} \leq max_{th}$) **então**
Status $\Leftarrow$ Between;
Se ($Q_{avg} < min_{th}$ **e** Status $\neq$ Below) **então**
Status $\Leftarrow$ Below;
$max_p \Leftarrow \frac{max_p}{\alpha}$;
Se ($Q_{avg} > max_{th}$ **e** Status $\neq$ Above) **então**
Status $\Leftarrow$ Above;
$max_p \Leftarrow max_p * \beta$;
FimARED

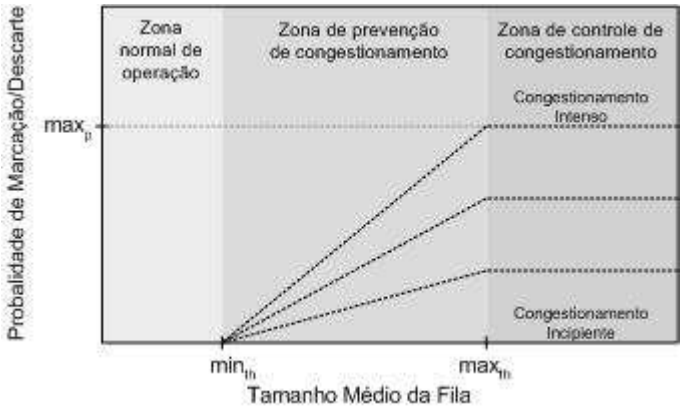

Figura 3.3: Comportamento do algoritmo ARED

de descarte baixa até que Q_{avg} atinja max_{th}. Quando o congestionamento é intenso, a probabilidade de descarte aumenta rapidamente quando Q_{avg} excede min_{th}. Variando o limiar max_p de acordo com a variação do tamanho médio da fila ARED consegue aumentar a vazão e diminuir a perda de pacotes.

O procedimento ARED apresentado no Algoritmo 3.2, deve ser incluído logo em seguida às atualizações de Q_{avg} no Algoritmo 3.1. Variando o limiar max_p de acordo com a variação do tamanho médio da fila, ARED consegue aumentar a vazão e diminuir a perda de pacotes. A Figura 3.3 apresenta o comportamento do algoritmo ARED.

3.5.2 FRED - Flow Random Early Drop

FRED - *Flow Random Early Drop* é um algoritmo de gerenciamento ativo de filas que foi desenvolvido com o objetivo de reduzir um dos problemas de RED: a injustiça. Pode-se dizer que FRED é uma versão modificada de RED, que ao invés de fazer a indicação de congestionamento a fluxos escolhidos de forma aleatória, indica o congestionamento, de forma seletiva, aos fluxos que usam as filas mais intensamente, ou seja, os que possuem mais pacotes enfileirados [75].

FRED mantém o estado dos fluxos ativos, ou seja, dos fluxos que possuem pacotes enfileirados no roteador. Além dos parâmetros utilizados por RED, FRED adiciona os seguintes parâmetros e variáveis: os parâmetros min_q e max_q, que são, respectivamente, os números mínimo e máximo ideais de pacotes que cada fluxo pode enfileirar; a variável global avg_{cq}, que é uma estimativa da média de pacotes no *buffer* por fluxo. Os fluxos com um número de pacotes enfileirados menor que avg_{cq}, são favorecidos, em relação aos demais; as demais variáveis mantidas por fluxo são: $qlen_i$, que informa quantos pacotes enfileirados o fluxo possui e $strike_i$ que informa o número de vezes que uma determinada conexão falhou em responder as indicações de congestionamento.

FRED permite que cada conexão armazene min_q pacotes no *buffer* sem perdas. Um número de pacotes maior que este valor está sujeito a ser descartado de forma aleatória da mesma forma como acontece com RED. FRED utiliza o limiar min_q para decidir quando aceitar um pacote de uma conexão com um fluxo pequeno de forma determinística. Com isto, FRED protege fluxos considerados frágeis ou fluxos que possuem janelas pequenas, fazendo com que pacotes sejam sempre aceitos de conexões que possuem menos do que min_q pacotes armazenados no *buffer* e quando o tamanho médio do *buffer* é menor que max_{th}. FRED consegue, também, atender novas conexões, mesmo quando há congestionamento.

Quando o número de conexões ativas é pequeno, ou seja, $N << min_{th}/min_q$, FRED permite que cada conexão armazene no *buffer* até min_q pacotes, sem que hajam descartes. Para não causar o mesmo problema de injustiça de RED, ou seja, impor a mesma taxa de perda de pacotes para todas as conexões com mais de min_q pacotes armazenados, FRED ajusta dinamicamente o valor de min_q para o valor de avg_{cq}, quando o sistema está operando com um número pequeno de conexões ativas. Desta forma, FRED consegue gerenciar fluxos heterogêneos.

Para impedir que fluxos mal-comportados ou não-adaptativos mantenham o monopólio dos recursos da rede, FRED não permite que mais que max_q pacotes de um mesmo fluxo sejam armazenadas no buffer. FRED armazena nas variáveis de cada fluxo, $strike_i$, o número de vezes que cada fluxo tentou exceder max_q pacotes armazenados no *buffer*. Os fluxos que tem valores de $strike_i$ alto são penalizados, não podendo armazenar na fila mais do que avg_{cq} pacotes. Com isto, tráfegos adaptativos podem continuar transmitindo

rajadas e ainda conseguem prevenir que tráfegos não-adaptativos monopolizem os espaços nos *buffers*.

Ao contrário de RED, FRED calcula o tamanho médio da fila não apenas na chegada de um pacote como também na sua saída. A perda desta variação poderia levar a um erro de cálculo no tamanho médio da fila e acarretar uma baixa taxa de utilização da banda devido a descartes desnecessários.

FRED garante uma maior justiça que RED e também é escalável com o número de conexões ativas. Protege as conexões com janelas pequenas e consegue lidar com fluxos não adaptativos. O preço destas vantagens é uma maior complexidade computacional. Além disto, FRED demanda a utilização de mecanismos mais eficientes de recuperação de perdas tais como TCP NewReno e TCP Sack [70].

O Algoritmo 3.3 apresenta em detalhes as operações realizadas por FRED.

Constantes:
$W_q = 0.002$: peso da fila;
$min_{th} = min(buffer_size/4, RTT)$: limiar mínimo;
$max_{th} = 2 * min_{th}$: limiar máximo;
$max_p = 0.02$: valor máximo de p_b quando $Q_{avg} = max_{th}$;
$min_q = 2$ ou 4: O número mínimo de pacotes que cada fluxo pode enfileirar. Possui valor igual a 2 para pequenos *buffers* e 4 para *buffers* maiores;

Variáveis Globais:
Q_{avg}: tamanho médio da fila;
Q_l: tamanho atual da fila;
p_a: probabilidade atual de marcar um pacote;
p_b: função linear do tamanho médio da fila, que varia entre 0 e max_{th};
$Time$: valor atual do tempo;
$Count$: número de pacotes que chegaram depois que o último pacote foi marcado;
avg_{cq}: estimativa da média de pacotes no *buffer* por cada fluxo;
max_q: o número máximo de pacotes que cada fluxo pode enfileirar;

Variáveis para cada Fluxo:
$qlen_i$: número de pacotes enfileirados do fluxo i;
$strike_i$: número de vezes que o fluxo i não respondeu ao congestionamento;

Funções de Mapeamento:
$conn(P)$: retorna o identificador da conexão do pacote P;
$f(t)$: uma função linear no tempo t.

Algoritmo 3.3 Flow Random Early Drop

ProcedimentoFRED( )
Para cada pacote P que chega **faça**
Se o fluxo $i \Leftarrow conn(P)$ não possui tabela de estado **então**
$qlen_i \Leftarrow 0$;
$strike_i \Leftarrow 0$;
Se a fila estiver vazia **então** CalculaQavg();
$max_q \Leftarrow min_{th}$;
Se $Q_{avg} \leq max_{th}$ **então** $max_q \Leftarrow 2$;
// Identificação e gerenciamento dos fluxos não-adaptativos:
Se $(qlen_i \geq max_q)$ **ou** $(Q_{avg} \geq max_{th}$ **e** $qlen_i > 2 * avg_{cq})$
ou $(qlen_i \geq avg_{cq}$ **e** $strike_i > 1)$ **então**
$strike_i \Leftarrow strike_i + 1$;
Descarta pacote P;
Retorna;
// Modo de descarte randômico
Se $min_{th} \leq Q_{avg} < max_{th}$ **então**
$Count \Leftarrow Count + 1$;
// Descarta pacotes apenas de fluxos robustos
Se $(qleni \geq Max(min_q, avg_{cq})$ **então**
// Cálculo da probabilidade p_a
$p_b \Leftarrow max_p * (Q_{avg} - min_{th})/(max_{th} - min_{th})$;
$p_a \Leftarrow p_b/(1 - Count * p_b)$;
Com probabilidade p_a:
Descarta o pacote P;
$Count = 0$;
Retorna;
Senão
Se $Q_{avg} < min_{th}$ **então** $Count \Leftarrow -1$;
Senão
$Count \Leftarrow 0$;
Descarta pacote P;
Retorna;
Se $qlen_i = 0$ **então** $N_{active} \Leftarrow N_{active} + 1$;
CalculaQavg();
Aceita Pacote P;
Para cada pacote que sai P **faça**
CalculaQavg();
Se $qlen_i = 0$ **então**
$N_{active} \Leftarrow N_{active} - 1$;
Remove da tabela os estados correspondentes ao fluxo i;
FimProcedimentoFRED

Algoritmo 3.4 Algoritmo do Cálculo do Tamanho Médio da Fila no Algoritmo FRED

CalculaQavg()
Se a fila não estiver vazia **ou** não for uma saída de pacote **então**
$Q_{avg} \Leftarrow Q_{avg}.(1 - W_q) + W_q.Q_l$;
Senão
$Num_{Pkg} \Leftarrow f(Time - Q_{time})$;
$Q_{avg} \Leftarrow Q_{avg}.(1 - W_q)^{Num_{Pkg}}$;
$Q_{time} \Leftarrow Time$;
Se $N_{active} > 0$ **então**
$avg_{cq} \Leftarrow Q_{avg}/N_{active}$;
Senão
$avg_{cq} \Leftarrow Q_{avg}$;
$avg_{cq} = Max(avg_{cq}, 1)$;
Se $Q_{len} = 0$ **e** for uma saída de pacote **então**
$Q_{time} = Time$;
FimCalculaQavg

3.5.3 Blue e SFB - Stochastic Fair Blue

Um dos problemas de RED é que ele só consegue atingir um ponto operacional ótimo quando possui um número de *buffers* suficiente e quando os seus limiares são devidamente definidos.

Nesta seção, é apresentado um outro algoritmo de gerenciamento ativo de filas que se baseia no histórico da utilização do enlace e na taxa de perda para gerenciar o congestionamento, ao invés de utilizar o tamanho médio da fila como RED. Este algoritmo denomina-se *Blue* e foi apresentado em [29, 85].

Ele utiliza uma probabilidade p_m, para marcar ou descartar pacotes. Se a fila descarta pacotes continuamente devido a transbordo, a probabilidade p_m é incrementada. Quando a fila fica vazia ou o enlace está parado, p_m é decrementada. Além desta probabilidade, *Blue* utiliza dois outros parâmetros: $freeze - time$, que determina o intervalo mínimo entre duas atualizações sucessivas da probabilidade p_m; e $delta$, que determina o valor de incremento ou de decremento de p_m.

O controle de congestionamento pode ser feito com uma quantidade mínima de espaço, reduzindo assim o atraso fim-a-fim. Garante-se, também, que o tamanho da fila mantenha-se mais estável, sem grandes variações, dado que gerencia sua taxa de marcação de forma mais inteligente. A notificação de congestionamento seja por marcação de pacotes ou por descarte, não gera conseqüentemente, períodos de baixa utilização do enlace nem

Algoritmo 3.5 Blue

ProcedimentoBlue()
Quando ocorrer uma perda de pacotes **faça**
Se $((now - last_update) > freeze_time)$ **então**
$p_m \Leftarrow p_m + delta$;
$last_update \Leftarrow now$;
Quando o enlace estiver parado ou a fila estiver vazia **faça**
$p_m \Leftarrow p_m - delta$;
$last_update \Leftarrow now$;
FimBlue

tampouco períodos com grande perda de pacotes. Uma outra vantagem de *Blue* é que a sua probabilidade de descarte p_m mantém-se estável, fazendo com que os pacotes sejam marcados ou descartados de forma aleatória, prevenindo o fenômeno de sincronização global, de forma mais efetiva que RED. Pode-se dizer que *Blue* atende os objetivos de RED sem, no entanto, apresentar todos os seus problemas. O Algoritmo 3.5 apresenta o procedimento *Blue*.

SFB - *Stochastic Fair Blue* é um algoritmo que utiliza *Blue* para proteger os fluxos bem comportados ou fluxos que respondem a notificações de congestionamento dos fluxos não adaptativos ou mal comportados. A idéia é detectar os fluxos não adaptativos e limitar a sua taxa de transmissão de tal forma que os fluxos que respondem a congestionamentos não sejam prejudicados.

SFB identifica e limita os fluxos não-adaptativos baseado em mecanismos de contabilização. Para isto, SFB mantém NxL depósitos que são organizados em L níveis com N depósitos em cada nível. Além disto, são mantidas L funções *hash* independentes, uma para cada nível. Estes depósitos são utilizados para se manter informações da ocupação estatística da fila de pacotes pertencentes a um depósito em particular. Cada função *hash* mapeia um fluxo em um dos N depósitos daquele nível.

Quando um pacote chega, o seu identificador é mapeado para um dos N depósitos em cada um dos L níveis. Cada depósito possui uma probabilidade p_m que é atualizada com a ocupação do depósito. Quando o número de pacotes mapeados em um depósito atinge um determinado limiar, a probabilidade p_m é incrementada. Caso o número de pacote chegue a zero, p_m é decrementada.

A decisão de se marcar um pacote é baseado em p_{min}, que é o valor mínimo de todos os valores p_m dos depósitos para os quais o fluxo foi mapeado. Se p_m é 1, então o fluxo é identificado como pertencente a um fluxo não-adaptativo e assim sua taxa é limitada. O

Algoritmo 3.6 apresenta o procedimento SFB.

SFB apresenta as mesmas vantagens de *Blue*, é escalável, justo e utiliza poucas informações de estado para cada fluxo, mas também possui algumas limitações. Quando o número de fluxos não-adaptativos aumenta, faz com que a probabilidade de que um fluxo bem comportado seja mapeado em depósitos que estejam poluídos com fluxos mal-comportados também aumente, fazendo com que estes fluxos sejam mal classificados. A probabilidade de um fluxo bem comportado ser mal classificado é dada por:

$$p = [1 - (1 - \frac{1}{B})^M]^L \tag{3.1}$$

onde L é o número de níveis, B é o número de depósitos por nível e M é o número de fluxos mal-comportados ou não adaptativos.

Como pode ser visto pela Equação 3.1, quanto maior for o número de depósitos por nível menor será a probabilidade de que um fluxo seja mal classificado. Uma outra alternativa para evitar que fluxos sejam mal classificados é utilizar funções *hash* móveis, ou seja, periodicamente os depósitos são reinicializados e as funções *hash* para cada nível são modificadas. A idéia é fazer com que o algoritmo possa reagir rapidamente à mudanças de comportamento dos fluxos, ou seja, que o algoritmo consiga reclassificar fluxos que eram mal comportados e que passaram a ser adaptativos.

3.5.4 FPQ - Flow Proportional Queuing

FPQ - *Flow Proportional Queuing* é um algoritmo de AQM que tenta resolver os problemas relacionados com RED quando existe uma grande quantidade de fluxos ativos [86]. A idéia por trás de FPQ é manter o tamanho da fila proporcional ao número de conexões ativas, de forma a não prover mais espaço do que o necessário, prevenindo, assim, atrasos desnecessários. Evita, também, prover menos espaço do que o necessário, o que aumentaria as perdas. Além disto, os roteadores devem ter memória proporcional ao número máximo de conexões ativas que podem absorver.

FPQ mantém o tamanho da fila proporcional ao número de fluxos ativos mantendo fixa a taxa de perda e variando o RTT em proporção ao número de conexões ativas. Com isto, os emissores TCP enviam pacotes a uma taxa inversamente proporcional ao número de conexões ativas.

Para se manter uma taxa de perda fixa, algumas considerações precisam ser feitas. Se o número de conexões ativas é baixo, a taxa de perda deve ser reduzida para permitir o aumento dos tamanhos das janelas, garantindo assim uma alta utilização do enlace. Além disto, roteadores que não são "gargalo" não devem impor taxas de perda mínimas.

Mantendo o tamanho da fila proporcional ao número de fluxos ativos, FPQ consegue uma melhora na previsibilidade e uma distribuição mais justa do tempo de transferência.

Algoritmo 3.6 Stochastic Fair Blue

ProcedimentoSFB()
Quando um pacote chegar **faça**
Calcula os valores das funções *hash* $h_0, h_1, ..., h_{L-1}$;
Atualiza os depósitos de cada nível;
Para $i = 1$ até $L - 1$ **faça**
Se $(B[i][h_i].q_{len} > Bin_{len})$ **então**
$B[i][h_i].p_m \Leftarrow B[i][h_i].p_m + delta$;
Descarta o pacote;
Senão
Se $(B[i][h_i].q_{len} = 0)$ **então**
$B[i][h_i].p_m \Leftarrow B[i][h_i].p_m - delta$;
$p_{min} \Leftarrow min(B[0][h_0].p_m..B[L-1][h_{L-1}].p_m)$;
Se $(p_{min} = 1)$ **então**
LimitaTaxa();
Senão
Marca/Descarta o pacote com probabilidade p_{min};
FimSFB

Isto acontece porque FPQ impõe o mesmo atraso para todas as conexões, já que o atraso neste algoritmo é decorrente do atraso nas filas. Em outros algoritmos de AQM o atraso imposto é desproporcional, já que este é devido não apenas à atrasos de enfileiramento, como também à temporização. Assim sendo, uma conexão "sortuda" pode obter um atraso menor que uma outra.

FPQ faz primeiramente, uma estimativa do número ativo de fluxos TCP. Esta estimativa é realizada utilizando-se um vetor de tamanho fixo. Quando um pacote chega, o identificador da sua conexão (IP e número de porta) é mapeado através de uma função *hash* para um valor e a sua posição correspondente no vetor é marcada. Em um intervalo definido, FPQ faz uma limpeza no seu vetor escolhendo aleatoriamente bits do vetor para serem zerados. Com isto, pode-se obter uma estimativa do número de fluxos ativos nos últimos segundos.

FPQ calcula, posteriormente, o tamanho da fila ideal para a estimativa do número de fluxos ativos. Com o valor do tamanho da fila ideal FPQ calcula a taxa de perda que deseja manter. Este valor é o mínimo entre a taxa máxima de perda para se ter a taxa mínima de transmissão de 8 pacotes por fluxo, e o cálculo da taxa de perda que é igual a $l = (\frac{0.87}{w})^2$, onde l é a taxa de perda, $w = \frac{q_t+P}{2}$ é o tamanho da janela e P o produto banda-atraso. Quando a rede possui apenas um roteador gargalo, a taxa de perda calculada é bem próxima do valor da taxa de perda real, mas quando existem vários outros gargalos, estes valores podem ser bem diferentes. Neste caso, FPQ utiliza um procedimento que faz um ajuste na taxa de perda.

Apesar de suas vantagens, FPQ também apresenta deficiências. O atraso imposto pode ser prejudicial para aplicações interativas. Um outro problema apresentado por FPQ é que ele foi desenvolvido apenas para fluxos adaptativos como TCP. Desta forma, se existirem também fluxos não adaptativos, o que é muito comum, o algoritmo não se comporta corretamente.

A seguir são apresentadas as variáveis e as constantes utilizadas por FPQ. O Algoritmo 3.7 apresenta todos os procedimentos do algoritmo FPQ e o Algoritmo 3.8 apresenta o procedimento FPQ.

Constantes:

t_{clear}: intervalo em segundos, entre "limpezas" sucessivas no vetor Vet;

v_{max}: tamanho do vetor Vet em número de bits. Este valor deve ser maior que o número de conexões esperadas;

Variáveis:

Q_{avg}: tamanho médio da fila;

q_t: tamanho desejado para a fila;

l_t: taxa de perda desejada para a fila;

l_a: taxa de perda ajustada ou corrigida;

N: estimativa do número de conexões ativas de Vet;
t_{last}: intervalo de tempo em que foi feita a última "limpeza" no vetor Vet;
$Vet[0..v_{max}]$: $Vet[i]$ indica se um pacote de uma conexão com valor *hash* i, chegou nos últimos t_{clear} segundos;

3.6 Políticas de AQM Baseadas em Modelos Analíticos

A principal dificuldade de se utilizar RED é determinar corretamente os seus parâmetros limitantes para que se possa atingir um ponto de operação ideal. Caso tais parâmetros não sejam corretamente definidos, RED pode vir a ter um comportamento similar ou pior a *tail drop*. Ademais, quando uma grande quantidade de fluxos está ativa, o tráfego agregado é em rajadas, o que degrada o comportamento de RED, pois o tamanho médio da fila varia rapidamente, causando grandes variações no seu tamanho e, conseqüentemente, grandes variações de retardo e baixa utilização do enlace. Quando a probabilidade de descarte varia consideravelmente em um curto intervalo de tempo, RED não consegue marcar os pacotes de forma aleatória, não prevenindo o fenômeno de sincronização global.

Com o intuito de superar as dificuldades de determinar corretamente os parâmetros de RED, vários estudos baseados em heurísticas e simulações vêm sendo desenvolvidos [8,29,71–74,84]. No entanto, todas estas políticas não garantem que um ponto de equilíbrio seja atingido, nem tampouco garantem a estabilidade do tamanho da fila, que é de capital importância para evitar grandes variações de retardo e baixa utilização do enlace. Por outro lado, investigações vêm sendo conduzidas para derivar configurações para RED de uma forma mais sistemática [77, 80, 87].

Para se projetar e desenvolver políticas de AQM que garantam estabilidade em torno de um ponto de equilíbrio, algumas políticas baseadas em Otimização e Teoria de Controle tem sido propostas.

Nas políticas de AQM baseadas em Otimização, define-se uma função de utilidade de fluxos agregados que caracterizam o sistema de controle de fluxo, como o objetivo de maximizar a taxa de transmissão e garantir o compartilhamento igualitário dos recursos entre as fontes. Busca-se, então, caracterizar quais são as condições de equilíbrio que podem ser obtidas deste sistema, de acordo com as condições da rede.

Nas políticas de AQM baseadas em Teoria de Controle, o controle de congestionamento é visto como um sistema de controle de retroalimentação, onde a taxa de transmissão dos nós fontes deve ser ajustada de acordo com o estado de congestionamento, que é determinado pela ocupação da fila. Desta forma, os controladores são responsáveis por determinar qual a probabilidade de descarte/marcação adequada que estabiliza o tamanho da fila independentemente das variações das condições da rede.

Algoritmo 3.7 Procedimentos utilizados por FPQ

// Procedimento que estima o número de conexões ativas, nos últimos segundos
ContabilizaConexões(p)
    $h \Leftarrow Hash(p)$;
    Se $(Vet[h] = 0)$ **então**
        $Vet[h] \Leftarrow 1$;
        $N \Leftarrow N + 1$;
    $T \Leftarrow CurrrentTime$;
    $n_{clear} \Leftarrow v_{max} * \frac{t - t_{last}}{t_{clear}}$;
    Se $(n_{clear} > 0)$ **então**
        $t_{last}T$;
        Para ($i = 1$ até $n_{clear} - 1$) **faça**
            $r \Leftarrow random(0..v_{max} - 1)$;
            Se $(Vet[r] = 1)$ **então**
                $Vet[r] \Leftarrow 0$;
                $N \Leftarrow N - 1$;
    Retorna(N);
FimContabilizaConexões

// Procedimento que calcula o tamanho da fila desejado baseado no número de conexões ativas, N, e no produto banda-atraso, P
TamFilaDesejado(N)
    $q_t \Leftarrow max(\frac{P}{2*N-1}, 8 * N)$;
    Retorna(q_t);
FimTamFilaDesejado

// Procedimento que calcula a taxa de perda desejada baseado no número de conexões ativas, N, no produto banda-atraso, P, e no tamanho da fila desejado
TaxaPerdaDesejada(q_t, N)
    $l_t \Leftarrow min((\frac{0.87}{\frac{q_t+P}{N}+1})^2, 0.021)$;
    Retorna(l_t);
FimTaxaPerdaDesejada

// Procedimento que ajusta a taxa de perda quando o tamanho médio da fila diverge do tamanho desejado
AjustaTaxaPerda(l_t, q_t, Q_{avg})
    $l_a \Leftarrow l_t \frac{Q_{avg}}{q_t}$;
    Retorna(l_a);
FimAjustaTaxaPerda

Algoritmo 3.8 Flow Proportional Queuing

```
ProcedimentoFPQ(p)
    N ⇐ ContabilizaConexões(p);
    q_t ⇐ TamFilaDesejado(N);
    l_t ⇐ TaxaPerdaDesejada(q_t, N);
    Q_avg ⇐ CalculaTamMédioFila();
    l_a ⇐ AjustaTaxaPerda(t_t, q_t, Q_avg);
    Se (random() < l_a) então
        Descarta o pacote p;
    Senão
        Enfileira o pacote p;
FimProcedimentoFPQ
```

A partir do trabalho original de Kelly, Maulloo e Tanthe [88], tem crescido o interesse e o número de pesquisas em projetos de mecanismos de controle de congestionamento para a Internet utilizando abordagens analíticas, resultando em um progresso significativo na modelagem matemática destes mecanismos [50,89]. Ferramentas da Teoria da Otimização e da Teoria de Controle têm representado um papel fundamental para o desenvolvimento da Teoria de Controle de Congestionamento para a Internet [90].

Nesta seção são apresentados alguns dos desenvolvimentos mais recentes e significativos de projeto de mecanismos AQM estáveis, focando nos trabalhos que utilizam ferramentas de Otimização e Teoria de Controle.

3.6.1 Trabalhos baseados em Teoria de Otimização

As políticas de AQM desenvolvidas utilizando-se a teoria de otimização geralmente representam o sistema de controle de congestionamento como um problema de otimização, o qual ficou amplamente conhecido como "*Kelly´s System Problem*" [88]. Nesta abordagem, um valor de uma função de utilidade é associado a cada fluxo e a função de utilidade agregada do sistema é maximizada sujeita às restrições do enlace [88]. Os mecanismos de controle de congestionamento assim desenvolvidos tentam obter a solução ótima ou sub-ótima para este problema de maximização [90].

Na abordagem de Teoria de Otimização, as taxas de transmissão são vistas como variáveis primais [88, 91, 92], cujos valores são obtidos com a interação com o TCP, enquanto que as medidas de congestionamento, ou seja, as probabilidades de descarte/marcação, são vistas como variáveis duais e seus valores são obtidos pelas políticas

de AQM [49, 88, 90, 93–97]. Desta forma, o sistema de controle de congestionamento pode ser visto como um algoritmo primal-dual distribuído sobre a Internet, cuja função é maximizar a função de utilidade agregada do sistema sujeita às restrições do enlace.

São considerados algoritmos primais aqueles cuja dinâmica é considerada presente nas fontes, mas são estáticas nos enlaces, e são consideradas algoritmos duais aqueles cuja dinâmica é considerada apenas nos enlaces e não nas fontes. Desta forma, os algoritmos primal-dual a dinâmica é considerada tanto nas fontes quanto nos enlaces [98–100]. Neste caso, a dinâmica das fontes é similar a dinâmica do algoritmo primal, enquanto a dinâmica do enlace é semelhante à apresentada nos algoritmos duais [90].

Uma solução para a formulação primal para o problema de controle de congestionamento utiliza o conceito de fila virtual, cuja capacidade é um pouco menor que o da fila real. A idéia central é descartar pacotes da fila real apenas quando ocorrer o transbordo na fila virtual. Na proposta apresentada em [101], a fila virtual é estática, enquanto que na proposta AVQ - *Adaptive Virtual Queue*, apresentada em [102, 103], a fila virtual é dinâmica, sendo sua capacidade variada de acordo com a taxa de chegada dos fluxos. Além disto, na presença de atraso, os parâmetros de AVQ podem ser escolhidos de forma a garantir a estabilidade local do sistema [90]. No trabalho apresentado em [100], é feita uma extensão de AVQ, na qual são considerados os termos dependentes do atraso presentes no sistema de congestionamento, de forma a garantir uma política estável de AQM baseada em filas virtuais dinâmicas, que suporte atrasos arbitrários. Lakshmikantha, Beck e Srikant [104] investigaram a robustez dos mecanismos AQM baseados em filas reais e virtuais. Utilizando modelos de fluidos simples, eles mostram que mecanismos AQM baseados em filas virtuais são mais robustos a perturbações que os que utilizam filas reais e também têm maior habilidade em manter pequenos atrasos de fila.

A política REM - *Random Exponential Marking*, é apresentada em [105] como uma das soluções para a formulação dual do problema de congestionamento. REM expressa as medidas de congestionamento como um preço, que é calculado em cada enlace utilizando informações locais, e, posteriormente, é retornado para os nós fontes através do descarte/marcação dos pacotes [106].

A política E-RED *Exponential-RED*, é apresentada como uma solução para a formulação primal-dual para o sistema de controle de congestionamento. Esta política determina o valor da probabilidade de descarte/marcação como uma função exponencial do tamanho da fila virtual, cuja capacidade é um pouco menor que o tamanho da fila real [90].

3.6.2 Trabalhos baseados em Teoria de Controle

O sistema de controle de congestionamento pode ser visto como um sistema de controle por retroalimentação, no qual a taxa de transmissão dos nós fontes é ajustada de acordo com o estado de congestionamento na rede, inferido pelos roteadores através da monitoração do nível de suas filas e retornado para os nós fontes através da descarte/marcação de pacotes.

As políticas de AQM baseadas em Teoria de Controle consideram a natureza intrínseca de retroalimentação do sistema de controle de congestionamento. Busca-se compreender a dinâmica do congestionamento, para poder se garantir estabilidade e robustez das condições de equilíbrio das métricas de desempenho desejadas. Os controladores são responsáveis por determinar qual a probabilidade de descarte/marcação adequada, que maximize a vazão e minimize as perdas, e que ainda garanta a estabilidade do sistema independentemente das variações das condições da rede. No caso específico de gerenciamento de filas, a importância da estabilidade está no fato de que sistemas de AQM instáveis produzem grandes variações de retardo e baixa utilização do enlace. Em [107], é apresentada uma uma visão geral sobre a abordagem de Teoria de Controle para o sistema de controle de congestionamento. Segundo os autores, o controle de congestionamento Internet pode ser considerado como o maior sistema artificial de retroalimentação já desenvolvido.

A maioria das políticas de AQM baseadas em Teoria de Controle desenvolvidas utilizam apenas as informações da dinâmica atual do sistema, e assim, desconsideram os termos dependentes do atraso. Algumas exceções são os trabalhos apresentados em [100, 108–110], e a política H2-AQM desenvolvida neste trabalho. É de conhecimento notório que o uso de controle independente do atraso possui limitações de desempenho na presença de grandes atrasos, quando comparado com o controle dependente do atraso [111, 112]. No contexto de redes, isto significa que a utilização da abordagem dependente do atraso ou não racional faz com que os recursos de rede possam ser usados eficientemente [110].

As políticas de AQM desenvolvidas utilizando-se Teoria de Controle utilizam na sua maioria controladores clássicos tais como controladores P (*Proportional*), I (*Integral*), PI (*Proportional-Integral*), PD (*Proportional-Derivative*) ou PID (*Proportional-Integral-Derivative*). Os demais trabalhos utilizam outras ferramentas de controle tais como compensação de retroalimentação e apenas um pequeno número de propostas utilizam técnicas de controle ótimo ou robusto [109, 113–116].

No trabalho apresentado em [117] é realizada uma investigação analítica de RED através da linearização do modelo do sistema de congestionamento que captura o comportamento da variação da janela de TCP em função da variação do tamanho da fila, introduzido em [9] e apresentado na Seção 5.1. Como resultado da análise verificou-se que

RED é um controlador do tipo P, e puderam ser determinadas, de forma sistemática, as configurações adequadas para os parâmetros de RED que levam o sistema à estabilidade.

LRED - *Loss Ratio based RED* é uma política de AQM que utiliza um controlador do tipo P [118]. LRED utiliza a taxa de perda como uma medida complementar ao tamanho da fila para ajustar dinamicamente a probabilidade de descarte/marcação. A idéia desta política é utilizar duas abordagens para o cálculo da probabilidade de descarte/marcação em escalas de tempo distintas. Para pequenas escalas de tempo, o tamanho instantâneo da fila é utilizado para calcular a probabilidade de descarte/marcação quando um novo pacote é recebido, e para escalas de tempo maiores é utilizada a taxa de perda para ajustar dinamicamente a probabilidade de descarte/marcação. Esta combinação possibilita não apenas um tempo de resposta mais rápido para o algoritmo, mas também uma maior robustez.

O objetivo da política *Stabilized RED* (SRED) é estabilizar o tamanho da fila em um valor de referência independentemente da variação do número de conexões ativas. Para tanto, estima o número de conexões ativas e determina o valor apropriado para o valor da probabilidade de descarte/marcação para o número de fluxos estimado [119]. *Dynamic RED* (DRED) foi desenvolvido com a mesma finalidade que SRED, no entanto, DRED não utiliza nenhuma estimativa do número de conexões ativas. Para atingir seu objetivo, DRED adapta a probabilidade de descarte/marcação, $p(t)$, de forma a minimizar o sinal de erro $e(t) = q(t) - q_{ref}$, e assim, manter o tamanho da fila, $q(t)$, tão perto quanto possível do seu valor de referência, q_{ref}. Os valores para os parâmetros do controlador DRED são determinados de forma empírica, através de simulação. Apesar de DRED ser apresentado em [120], como um controlador do tipo P, em [121], é comprovado que DRED é de fato um controlador do tipo I.

A política PI-AQM, *Proportional Integrator* AQM [122], utiliza um controlador do tipo PI, e foi desenvolvida utilizando o mesmo modelo dinâmico para o comportamento do sistema de congestionamento TCP/AQM apresentado na Seção 5.1. Os autores desta proposta são os mesmos do trabalho apresentado em [117], e podem ser considerados como os pioneiros na representação do sistema de controle de congestionamento como um problema de controle e da utilização de técnicas de controle para a obtenção de políticas de AQM que garantam a estabilidade do sistema de congestionamento 6.7. Como o trabalho apresentado nesta tese foi baseado no PI-AQM, esta política será melhor detalhada na Subseção 6.7.

Uma proposta AQM que também utiliza o controlador do tipo PI é apresentada em [123]. A diferença é que neste trabalho o controlador consiste de duas partes: um controlador para a taxa de chegada e um outro controlador para o tamanho da fila. O controlador para a taxa de chegada é do tipo PI, e tem a função de aprimorar o tempo de resposta a variações do tráfego, bem como manter a taxa de chegada em torno da

capacidade do enlace. O controlador para o tamanho da fila é do tipo P, e tem a função de estabilizar o tamanho da fila.

Dentre as políticas de AQM desenvolvidas utilizando-se controladores do tipo PD pode-se destacar os trabalhos [124], [125], [126] e [110].

A política de AQM PD-RED foi apresentada em [124], com o intuito de aprimorar RED, fazendo-se alterações mínimas no seu algoritmo através do uso de um controlador do tipo PD. A alteração proposta é adaptar o parâmetro max_p de RED, que determina o valor máximo para a probabilidade de descarte/marcação, de modo a estabilizar o tamanho da fila.

Em [125] é introduzida a política de AQM *PD-Controller*, desenvolvida utilizando-se um controlador do tipo PD, e cujos objetivos são similares aos da política DRED. Assim como DRED, nesta política os parâmetros do controlador são determinados empiricamente. A diferença é que DRED utiliza o tamanho instantâneo da fila, enquanto que *PD-Controller* utiliza o tamanho médio da fila para determinar a probabilidade de descarte/marcação. Os resultados apresentados em [125] indicarem que *PD-Controller* é estável e robusta em relação a alterações no tráfego. Entretanto, em [126] é discutida a dificuldade em determinar os parâmetros desta política, e é apresentado um método para determinar corretamente seus parâmetros de forma a garantir um controle efetivo e estável.

Em [110] é introduzida uma política de AQM dependente do atraso e que utiliza técnicas de compensação de retroalimentação, denominada PD-AQM (*proportional derivative* AQM) que utiliza uma estrutura de memória de controle para fazer as compensações necessárias decorrentes da dependência do atraso nas medidas de congestionamento obtidas.

A adição de controle do tipo I ou do tipo D pode aumentar de forma significativa o desempenho dos controladores do tipo P. O controle do tipo I elimina o erro em torno do ponto de equilíbrio, enquanto que o controle do tipo D aprimora a estabilidade e determina como uma pequena pertubação na entrada afeta a saída do sistema, ou seja, controla a variação em torno do ponto de equilíbrio [127]. Além disto, os controles P e I atuam apenas sobre os erros passados, não sendo capazes de predizer futuros erros, uma característica que limita o desempenho dos controladores do tipo PI. O controle do tipo D, por ser capaz de predizer erros futuros, e, portanto, tomar medidas corretivas antes que estes erros aconteçam, é utilizado para aprimorar o tempo de resposta. Do ponto de vista do sistema de controle de congestionamento, o uso em conjunto destes tipos de controle tem o intuito de obter políticas de AQM que tenham um mínimo tempo de resposta e garantam a estabilidade em torno do tamanho da fila. O controle do tipo D minimiza o tempo de resposta, enquanto os controles P e I são responsáveis por garantir a estabilidade do sistema e manter o tamanho da fila em torno do valor desejado [109].

Algumas políticas de AQM vem sendo desenvolvidas utilizando-se controladores do tipo PID [108, 109, 121, 127].

O objetivo da política VRC - *Virtual Rate Control* é utilizar um controlador do tipo PID para estabilizar tanto a taxa de chegada dos fluxos bem como o tamanho da fila em torno de seus valores alvo [121]. Para tanto, VRC utiliza a noção de um alvo virtual, apresentado originalmente na política AVQ. A diferença é que o alvo de AVQ é uma fila virtual, enquanto que para VRC o alvo é uma taxa virtual. Nos modelos baseados na taxa de chegada dos fluxos, a idéia é adequar a taxa a um valor de taxa desejado equivalente a capacidade do enlace, compensando a diferença entre a taxa obtida e a desejada. Nos modelos baseados nas taxas de chegada, o tamanho da fila não aparece explicitamente no modelo, enquanto que os modelos baseados nas filas, o tamanho da fila faz explicitamente parte do modelo. No trabalho apresentado em [128], é feito um estudo sobre as políticas de AQM baseadas nas taxa de chegada versus às baseadas no tamanho da fila.

Yellow é uma política de AQM baseada na taxa de chegada dos fluxos, e como tal utiliza a carga da rede ou a utilização do enlace como métrica principal para controlar o congestionamento [129]. Sua principal contribuição é a adoção de uma segunda métrica a ser utilizada ao determinar a probabilidade de descarte/marcação. A diferença entre a taxa e a capacidade do enlace é considerada a primeira métrica. Com o intuito de aprimorar o desempenho do sistema de controle de congestionamento, é introduzido uma função de controle da fila denominado QCF (*queue control function*), o que garante a *Yellow* um bom desempenho mesmo sob mudanças bruscas de tráfego. A utilização das duas métricas garante a *Yellow* um baixo tempo de resposta além de baixos valores de atraso e *jitter*.

Em [127], o modelo do sistema de congestionamento utilizado para analisar RED e para derivar a política PI-AQM (Seção 5.1) é estendido de forma a derivar três plantas distintas: a primeira considera que a rede tenha suporte a ECN, um mecanismo que permite dissociar a notificação do congestionamento do descarte de pacotes apresentado na Seção 3.7; a segunda considera que a rede não tenha suporte a ECN; e a última planta considera a existência de fluxos não adaptativos, ou seja, fluxos que não diminuem sua taxa de transmissão na presença de congestionamento. Estas plantas são analisadas e um controlador do tipo PID é, então, desenvolvido. Os autores também apresentam um mecanismo de controle adaptativo, cujo objetivo é aprimorar a estabilidade e o desempenho do sistema sob mudanças na configuração das plantas.

O trabalho apresentado em [109], combina as vantagens dos controladores P, I e D e ao mesmo tempo limita suas deficiências para obter uma política de AQM. Para projetar o controlador PID e determinar corretamente seus parâmetros foi utilizado o método LQR (*Linear Quadratic Regulators*), um método método de projeto robusto comparado aos demais métodos clássicos.

A política RHA (*Receding Horizon AQM policy*), desenvolvida através de uma abordagem dependente do atraso, faz explicitamente a compensação do atraso nas medidas de congestionamento obtidas utilizando uma estrutura de memória de controle [108]. Posteriormente, em [130] é derivado a versão ótima do RHA que estabiliza o sistema de congestionamento com custo mínimo.

O trabalho apresentado em [115] utiliza a técnica de compensação de retroalimentação para derivar o algoritmo AQM denominado PIP *Proportional Integral based series compensation, and Positional feedback compensation*. Dado que geralmente é difícil de obter o estado transiente ou estável para o comportamento do sistema de controle, elementos de compensação são freqüentemente introduzidos no sistema. O objetivo é, então, escolher parâmetros de compensação adequados de forma a garantir que o sistema atinja o desempenho desejado.

O algoritmo SMVSAQM [114] é baseado na técnica de controle SMVS - *Sliding Mode Variable Structure Control*. A estrutura de controle do SMVSAQM varia durante o processo de controle, de tal forma que o controlador é projetado para conduzir e depois limitar o estado do sistema na vizinhança da função de variação, ou seja, o projeto do controlador é insensível à dinâmica dos parâmetros do sistema de congestionamento.

Outra política de AQM que utiliza a técnica de controle robusto SMVS, é denominada VS-AQM (*Variable Structure AQM*) [116]. A diferença principal em relação ao algoritmo SMVSAQM, é que VS-AQM foi projetada diretamente a partir do modelo não linear do sistema de congestionamento TCP/AQM. Desta forma, obtém-se um controlador mais adequado para o sistema, dado que o controlador não terá que lidar com as eventuais incertezas do modelo decorrentes do processo de linearização.

A Teoria de Controle também pode ser utilizada para analisar as propriedades de estabilidade dos mecanismos de controle de congestionamento. Em [92] são utilizadas técnicas de controle robusto para derivar uma condição suficiente para garantir a estabilidade local de uma rede com topologia arbitrária e valores de RTT heterogêneos, enquanto que no trabalho apresentado em [131] é apresentada uma condição suficiente para garantir a estabilidade global do sistema de congestionamento para uma rede nas mesmas condições.

Uma visão geral dos trabalhos desenvolvidos que utilizam técnicas de Teoria de Controle para analisar a estabilidade local e global dos mecanismos de controle de congestionamento é apresentada [50,89]. Além disto, são discutidas várias hipóteses e conceitos que devem ser no modelo utilizado no desenvolvimento de um mecanismo de controle de congestionamento: a presença ou ausência do atraso; se os atrasos existentes são homogêneos ou heterogêneos; se a topologia é restrita ou geral, ou seja, se existem mais de um enlace gargalo; se existem múltiplas fontes ou se a fonte é única.

3.7 ECN - Explicit Congestion Notification

Com o mecanismo de gerenciamento ativo de filas, AQM, a notificação de congestionamento é feita aos nós finais através do descarte de pacotes antes que a fila fique cheia, evitando assim os problemas já descritos com *tail drop* e ainda garantindo um tamanho médio da fila baixo, podendo desta forma absorver tráfegos em rajadas, diminuir o atraso de pacotes e evitar o problema de sincronização global.

Entretanto, o descarte de pacotes não é a única forma de notificação de congestionamento possível de ser utilizada por AQM, ou seja, é possível dissociar a notificação de congestionamento do descarte de pacotes. Ao invés de notificar o emissor da existência de congestionamento através do descarte de pacotes, pode ser utilizado um mecanismo, onde um bit do cabeçalho do pacote é utilizado para sinalizar o congestionamento. Este mecanismo é denominado *Explicit Congestion Notification* (ECN) [132–134]. Quando o roteador verifica que há congestionamento, ele simplesmente sinaliza o congestionamento através da marcação do pacote ao invés de descartá-lo, e caso não exista suporte a ECN, o pacote é simplesmente descartado.

3.7.1 Princípios e Hipóteses

Nesta seção são apresentados os princípios e as hipóteses que guiaram o desenvolvimento de ECN [133, 134]. Foram consideradas as seguintes hipóteses:

1. O congestionamento pode persistir por diferentes escalas de tempo. A escala de tempo considerada é a de eventos de congestionamento que duram mais do que um RTT;

2. A ocorrência de roteamento assimétrico é considerado normal. O caminho seguido por pacotes de dados pode ser diferente do caminho seguido pelos pacotes de reconhecimento na direção oposta;

3. Nem todos os nós irão cooperar com os mecanismos de controle de congestionamento. Entretanto, novos mecanismos de controle de congestionamento não devem tornar mais fácil para as aplicações TCP desabilitar o mecanismo de controle de congestionamento, e ainda devem diminuir ou até mesmo tornar inexistentes os benefícios de se "mentir" a respeito da participação nestes mecanismos.

A seguir são apresentados os princípios que nortearam o desenvolvimento de ECN:

1. O número de pacotes de um fluxo seja TCP ou UDP pode ter uma ampla faixa de variação. Os fluxos considerados são aqueles que causam congestionamento e que

mandam pacotes o suficiente para estarem ativos ainda quando a notificação chega até eles;

2. Novos mecanismos de controle de congestionamento devem coexistir e cooperar com os mecanismos de controle de congestionamento existentes. Em particular, os novos mecanismos devem coexistir com os métodos correntes de controle de congestionamento TCP e com a prática corrente dos roteadores de se descartar pacotes quando o congestionamento ocorrer;

3. ECN deve ser adotado de forma gradual. Desta forma, alguns roteadores irão continuar a descartar pacotes como indicação de congestionamento enquanto outros irão utilizar ECN;

4. Muitos roteadores processam os cabeçalhos IP com suas informações regulares mais eficientemente do que processam as informações contidas nas opções IP. Isto sugere que o bit de notificação de congestionamento seja mantido no cabeçalho IP e não nas opções do cabeçalho.

3.7.2 ECN e IP

A proposta de ECN é que a notificação de congestionamento incipiente seja feita através da marcação de pacotes ao invés de descartá-los. Este mecanismo utiliza os bits 6 e 7 do antigo campo TOS (*Type of Service*) ou atualmente do campo DS (*Differential Service*) do cabeçalho IP. No documento inicial que propôs a inclusão de ECN, [133] foi definido que o primeiro bit denominado ECT - *ECN-Capable Transport*, deve ser selecionado pelo emissor para indicar aos nós finais que possui suporte a ECN. O outro bit, CE - *Congestion Experienced* deve ser selecionado pelo roteador para indicar congestionamento aos nós finais. [133]. No documento que define como será feita esta inclusão, é feita uma pequena alteração no sentido destes bits [134]. Eles são tratados como um único campo com os seguintes significados:

ECT(0), CE(O): estes bits devem ser enviados pelo emissor para indicar aos nós finais que não possui suporte a ECN. Este campo é chamado Not-ECN;

ECT(0), CE(1): estes bits devem ser enviados pelo emissor para indicar aos nós finais que possui suporte a ECN. Este campo com estes bits é denominado ECT(0);

ECT(1), CE(O): estes bits devem ser enviados pelo emissor para indicar aos nós finais que possui suporte a ECN. Este campo com estes bits é denominado ECT(1);

ECT(1), CE(1): estes bits devem ser enviados pelo roteador para indicar a existência de congestionamento aos nós finais. Este campo é chamado CE.

A utilização de dois campos ECT é motivado pelo desejo de se permitir mecanismos para verificar se emissores e receptores estão se comportando de forma correta, ou seja, permite que o emissor possa verificar se os elementos da rede não estão apagando o campo CE, e que os receptores estão notificando corretamente aos emissores o recebimento de pacotes com o campo CE, como exigido pelo protocolo de transporte. Os roteadores tratam ECT(0) e ECT(1) de forma equivalente. As diretrizes para diferenciar ECT(0) e ECT(1) por emissores e receptores devem ser específicas para cada protocolo, e ainda não foram determinadas.

Os algoritmos de controle de congestionamento dos nós finais com suporte a ECN devem ser basicamente os mesmos caso a notificação de congestionamento tenha sido feita pelo descarte de pacotes ou por ECN. A exceção ocorre em relação a algumas implementações TCP. Por exemplo, não recomenda-se seguir o comportamento do TCP Tahoe que ao ter um pacote descartado entra no algoritmo de *Slow Start*, ou o de TCP-Reno que espera aproximadamente metade de um RTT durante *Fast Retransmit*.

A justificativa para este requisito de ter um mesmo tratamento de controle de congestionamento seja para descarte de pacotes ou para ECN é permitir o desenvolvimento incremental de ECN. Desta forma, fluxos sem suporte a ECN terão um tratamento similar, prevenindo que ocorram injustiças em relação a estes fluxos. Um outro requisito é que os nós finais só devem reagir ao congestionamento no máximo uma vez por janela de dados, evitando, assim, que se reaja múltiplas vezes à várias indicações em um mesmo RTT.

Quando o roteador recebe um pacote e a sua fila está cheia, indicando que o congestionamento é intenso, ele deve necessariamente sinalizar o congestionamento através do descarte do pacote e não através do uso de ECN. Espera-se que quando a maioria dos nós tiverem suporte a ECN, o descarte de pacote devido a filas cheias irá ser bastante raro. No ambiente com suporte total a ECN, o descarte só deverá acontecer quando não houver cooperação entre nós.

Uma recomendação para pacotes com suporte a ECN é que estes devem ter o bit DF (*Don't fragment*) marcado, para evitar fragmentação. Caso a fragmentação ocorra, a junção dos pacotes fragmentados deve manter as informações de congestionamento, caso ela existisse no pacote original.

3.7.3 Interação com o Protocolo de Transporte TCP

Além dos bits do cabeçalho IP, ECN também requer suporte do protocolo de transporte, que deve ser responsável pela negociação entre os nós finais durante a fase de inicialização da conexão para determinar se ambos possuem suporte a ECN. Além disto, o protocolo de transporte também deve ser capaz de reagir de forma apropriada quando receber um

pacote com o campo CE marcado.

ECN pode trabalhar com vários protocolos de transporte, mas apenas a integração de ECN com o protocolo TCP foi definida. Para dar suporte a ECN, o TCP deve ter três novas funcionalidades:

- Negociação entre os nós finais durante a fase de inicialização da conexão para determinar se ambos possuem suporte a ECN;
- Um *flag* no cabeçalho TCP, utilizado pelo receptor para informar ao emissor do recebimento de um pacote com o campo CE. Este *flag* é denominado ECN-Echo;
- Um *flag* no cabeçalho TCP, utilizado pelo emissor para informar ao receptor de que a sua janela de congestionamento foi reduzida, devido ao recebimento de um pacote com o campo CE. Este *flag* é denominado CWR - *Congestion Window Reduced*.

Os *flags* CWR e ECN-Echo correspondem, respectivamente, aos dois últimos bits do campo *reserved* do cabeçalho TCP.

A seqüência típica de eventos em uma conexão TCP com suporte a ECN e com indicação de congestionamento é descrita de forma sucinta a seguir:

- O emissor e o receptor durante a fase de inicialização da conexão negociam a utilização de ECN se ambos possuírem suporte. Supondo que ambos concordaram em utilizar ECN a conexão continua;
- Um roteador com suporte a ECN detecta um congestionamento incipiente, e verifica que o emissor do pacote possui suporte a ECN. Ao invés de descartar o pacote ele marca o pacote como CE e encaminha o pacote ao destino;
- O receptor ao receber o pacote com o campo CE, marca o *flag* ECN-Echo do próximo ACK a ser enviado para o emissor. Para evitar que a possibilidade de um ACK perdido deixe de informar ao emissor a existência de congestionamento, o receptor deve enviar continuamente ACK's com o *flag* ECN-Echo marcado, até que receba do emissor um pacote com o *flag* CWR marcado. A exceção é feita apenas para pacotes "ACK puros", ou seja, pacotes que não possuem dados.

  O receptor deve ignorar o campo CE de pacotes que chegam fora de ordem. Esta recomendação é para se evitar o ataque *deny of service*, descrito a seguir;
- O emissor ao receber o ACK com o *flag* ECN-Echo marcado, reage a notificação de congestionamento como se o pacote reconhecido pelo ACK tivesse sido perdido;

- O emissor marca o *flag* CWR do próximo pacote a ser enviado para o receptor, informando-lhe que reagiu ao congestionamento. No caso do protocolo TCP, esta reação é a diminuição da janela de congestionamento.

  É importante salientar que o emissor só deve marcar o *flag* CWR se o pacote a ser transmitido for um novo pacote e não uma retransmissão. Este comportamento é recomendado para evitar o ataque *deny of service*, onde o TCP atacante se passa pelo TCP emissor enviando para o receptor um pacote com o campo CE, para que o receptor informe ao emissor a existência de congestionamento, fazendo com que o emissor diminua sua taxa de transmissão. Em [134] são discutidas várias formas de se evitar este tipo de ataque, e dentre as formas apresentadas é recomendado que o emissor não deve marcar o *flag* CWR quando o pacote transmitido é uma retransmissão.

- O receptor ao receber um pacote com o *flag* CWR marcado, deve parar de enviar ACK's com o *flag* ECN-Echo marcado para o emissor.

A negociação para a utilização de ECN pelos nós finais, o uso dos bits ECT, CE, ECN-Echo e CWR, além do comportamento do emissor, do receptor e dos roteadores no contexto ECN é explicada em detalhes em [133, 134].

3.7.4 Vulnerabilidades de ECN

Nesta seção, são descritas as vulnerabilidades de ECN em vários cenários [134]. Na primeira subseção, é discutido como ECN pode influenciar na detecção e penalização dos fluxos não reagem à indicação de congestionamento, seja por que são fluxos não adaptativos ou mal comportados ou ainda fluxos que possuem suporte a ECN mas não seguem as regras de ECN. Na segunda subseção, serão tratados os problemas que a alteração dos bits ECN pode causar.

Fluxos Mal Comportados

O problema de se detectar e penalizar tráfegos não adaptativos ou mal comportados que não diminuem sua taxa de transmissão diante de congestionamentos é um problema bastante conhecido. Para tratar deste problema, sugere-se que sejam desenvolvidos mecanismos para detectar e eventualmente penalizar estes fluxos mal comportados na presença de congestionamento. Tais mecanismos incluem desde coleta e armazenamento de informações por fluxo, escalonamento por fluxo, isolamento dos fluxos, serviços diferenciados e reserva de recursos. Estes mecanismos usados de forma isolada ou em conjunto podem minimizar ou eventualmente remover os problemas que estes tráfegos podem causar.

A penalidade aplicada aos fluxos mal comportados é, geralmente, o descarte dos pacotes destes fluxos. Assim, pode-se pensar que com ECN estes fluxos possam ser favorecidos ao invés de serem penalizados já que seus pacotes não serão descartados. No entanto, isto não é de fato o que acontece devido a três razões:

- O comportamento dos roteadores em descartar pacotes se mantém quando o congestionamento é intenso. Quando o congestionamento é incipiente, a presença de fluxos mal-comportados não é tão nociva, uma vez que há banda suficiente para todos os fluxos. O problema ocorre quando o congestionamento é intenso, já que estes fluxos irão prejudicar os fluxos adaptativos, conseguindo uma parte da banda bem maior que os demais;

- O descarte de pacotes utilizado como única maneira de penalizar ou controlar fluxos mal comportados não é eficiente;

- Fluxos sensíveis ao atraso e que usam transmissão não confiável, incrementam o uso de FEC -*Forward Error Correction*, em resposta ao aumento da taxa de descarte, aumentando assim a sua taxa de transmissão ao invés de diminuí-la.

Mudanças dos Bits do Cabeçalho IP e TCP

Nas Seções 3.7.2 e 3.7.3, foram apresentados os bits utilizados por ECN tanto no cabeçalho IP como no cabeçalho TCP, como também o seu significado. Através da utilização destes bits é possível indicar se um nó possui suporte a ECN. O roteador pode marcar um pacote de um fluxo indicando a existência do congestionamento; o receptor pode reportar este congestionamento ao emissor e, este, por sua vez pode informar que já reagiu ao congestionamento. Caso alguns destes bits sejam alterados intencionalmente pelos usuários ou por algum problema encontrado no caminho percorrido pelo pacote, ECN pode não funcionar a contento, conseqüentemente não atingindo os seus objetivos.

A seguir são apresentados os problemas causados quando estes bits são alterados.

Falsa Indicação de Suporte a ECN: foi visto na Seção 3.7.2 que um nó faz a indicação de suporte a ECN através do campo ECT(0) ou ECT(1). Suponha que um pacote possui um destes campos, mas o protocolo de transporte utilizado para transmitir este pacote não possui suporte a ECN. Se este pacote encontra no seu caminho congestionamento moderado e passa por um roteador com suporte a ECN, este roteador irá marcar o campo CE deste pacote ao invés de descartá-lo. Como o protocolo de transporte não possui suporte a ECN, irá ignorar a notificação de congestionamento e não irá reduzir a sua taxa de transmissão;

Desabilitando a Indicação de Suporte a ECN: este caso é justamente o oposto do anterior. Um pacote tinha um dos campos ECT(0) ou ECT(1) e foi modificado, ficando com o campo Not-ECN. Caso este pacote encontre algum congestionamento no seu caminho, mesmo que o roteador tenha suporte a ECN, ele será descartado ao invés de ser marcado, pois o roteador não terá como saber que ele também possuía suporte a ECN;

Falsa Indicação de Congestionamento: quando um roteador recebe um pacote que possui suporte a ECN e existe um congestionamento incipiente, o roteador marca este pacote com o campo CE ao invés de descartá-lo para que seja feita a indicação de congestionamento ao emissor, para que este diminua a sua taxa de transmissão. Um usuário mal intencionado pode fazer um ataque *deny of service*, justamente enviando para o receptor em nome do emissor um pacote com o campo CE, ou ainda o próprio receptor também pode fazer este ataque marcando o bit ECN-Echo no seu cabeçalho TCP. Com isto, o emissor irá reduzir sua taxa de transmissão em resposta a uma notificação de congestionamento falsa;

Remoção da Indicação de Congestionamento: caso um pacote estivesse com o campo CE marcado e este campo por algum motivo foi removido, o receptor não poderá informar ao emissor a existência de congestionamento, o que poderá agravar ainda mais o quadro de congestionamento existente. Caso a alteração não tenha sido do campo CE, mas do *flag* ECN-Echo de um ACK, o problema não é tão grave, pois mesmo que um ACK tenha sido alterado, o receptor continua enviando ACK's com o *flag* ECN-Echo marcado até que receba do emissor um pacote com o *flag* CWR marcado.

3.7.5 Avaliação de ECN

ECN diminui sensivelmente as perdas de pacotes quando o congestionamento é incipiente, diminuindo as retransmissões, o que também diminui o tráfego na rede [135]. A diminuição da perda de pacotes também faz com que recursos da rede sejam melhor utilizados, pois quando um pacote é descartado, todos os recursos de rede que ele havia utilizado foram desperdiçados, além do que novos recursos serão utilizados pelo pacote que terá que ser retransmitido.

Como ECN evita perdas desnecessárias de pacotes, aplicações com conexões curtas ou que sejam sensíveis a atraso se beneficiam da utilização de ECN [133,134]. Os mecanismos TCP de recuperação de erro também são favorecidos quando ECN é utilizado, já que ECN evita que TCP entre em *timeouts* [135].

A partir dos experimentos realizados em [135], concluiu-se que TCP com ECN consegue uma melhor vazão quando comparados com outros tráfegos deste mesmo tipo que não usam ECN. Concluiu-se também que tráfegos TCP com ECN são justos em relação a outros tráfegos não ECN e que TCP com ECN é robusto com congestionamento nas duas direções.

Apesar das suas vantagens e de poder ser desenvolvido de forma incremental, ECN ainda está sendo muito pouco utilizado. Isto pode ser justificado pelo fato de ECN exigir que não só os nós finais como os roteadores lhe dêem suporte, como também exige mudanças nos protocolos de transporte.

Capítulo 4

Uma Introdução a Análise de Sistemas Dinâmicos e Controle

Este capítulo visa introduzir os conceitos básicos da área de modelagem matemática e teoria de controle necessários para a compreensão do trabalho desenvolvido nesta tese.

4.1 Conceitos e Definições

Nesta seção, são apresentados alguns dos conceitos e definições necessários para o entendimento dos sistemas de controle. Alguns destes conceitos serão mais detalhados nas seções seguintes. Os conceitos e definições principais são [136, 137]:

Sinal: é uma descrição quantitativa de um determinado fenômeno, associado a um dado meio;

Sistema: é uma combinação de componentes que atuam em conjunto e que se deseja operar com alguma finalidade (objetivo de controle). Um sistema é representado por uma variável de entrada (controle), uma de saída (controlada) e uma relação (função de transferência) entre elas;

Entradas e Saídas de um Sistema: os sinais que relacionam ou comunicam o sistema com o meio são os sinais de entrada e sinais de saída. O meio atua sobre o sistema através dos sinais de entrada e o sistema atua sobre o meio através dos sinais de saída;

Planta ou Sistema a controlar: é qualquer objeto físico a ser controlado;

Processo: é uma operação que evolui progressivamente e se constitui em uma série de ações controladas ou de movimentos sistematicamente dirigidos para alcançar um determinado resultado ou meta;

Variável controlada: a grandeza ou a condição que é medida e controlada;

Variável manipulada: é a grandeza ou condição variada pelo controlador de modo a afetar o valor da variável controlada;

Valor de Referência: valor desejado da variável a ser controlada;

Medidor ou transdutor: responsável pela medição e conversão da variável a ser controlada para fins de comparação e obtenção do erro de saída;

Comparador: dispositivo que constrói o sinal de erro entre o valor desejado e o obtido;

Distúrbio ou perturbação: sinal que tende a afetar de modo adverso o valor da variável de saída de um sistema;

Controlar: significa medir o valor da variável controlada e aplicar o valor conveniente da variável manipulada ao sistema de modo a corrigir ou limitar o desvio entre o valor medido e o valor de referência da variável controlada;

Controlador: manipula o sinal de erro, gerando um sinal de controle que será aplicado no sistema, corrigindo o valor da variável manipulada;

Sistema de Controle: é um sistema que tende a manter uma relação pré-estabelecida entre duas variáveis do sistema através da comparação de funções destas variáveis e utilizando a diferença como meio de controle;

Controle com retroalimentação: refere-se a operação que, em presença de distúrbios, tende a reduzir a diferença entre o sinal de saída e o sinal de referência de um sistema, e que opera com base nesta diferença. Apenas os distúrbios não conhecidos a priori são considerados, uma vez que as perturbações conhecidas podem ser compensadas no sistema.

Pode-se dizer que em geral a análise de um sistema dinâmico e o projeto de um sistema de controle é composto das seguintes fases [138, 139]:

- Estudar o sistema a ser controlado, caracterizando os limites do sistema, o que implica em especificar o escopo no problema a ser tratado e do sistema a ser controlado;
- Estabelecer quais são as entradas para o sistema;

- Formular o modelo do comportamento dinâmico do sistema, incluindo se possível as suas incertezas;
- Simplificar o modelo se necessário para torná-lo tratável;
- Analisar o modelo resultante e determinar suas propriedades e o seu comportamento dinâmico;
- Especificar quais são as variáveis disponíveis para a realimentação do sistema;
- Especificar quais os requisitos desejáveis ou requeridos para o sistema controlado;
- Decidir o tipo do controlador a ser utilizado;
- Projetar o controlador de modo a atender os requisitos de desempenho;
- Simular o sistema controlado para verificação do seu desempenho;
- Repetir os passos anteriores se necessário.

Em muitos dos passos apresentados anteriormente, é crucial a derivação de modelos matemáticos adequados para o sistema, que incluam os sinais de entrada e saída bem como os requisitos de desempenho, visto que o sucesso do projeto de um sistema de controle depende do profundo conhecimento das suas propriedades dinâmicas, bem como dos seus sinais de entrada e saída.

Projetar um sistema de controle é um processo que envolve um número de escolhas e decisões que dependem das propriedades do sistema a ser controlado, bem como dos requisitos a serem satisfeitos pelo sistema controlado. Tais decisões requerem que seja garantido o compromisso entre alguns requisitos conflitantes. O projeto de sistemas de controle visa basicamente aprimorar o desempenho do sistema, de forma a garantir que os seguintes requisitos sejam atendidos:

Estabilidade: um sistema é estável se quando o vetor de estado se afasta ligeiramente do ponto de equilíbrio, ele tende a retornar ao ponto ou, ao menos, não continuar se afastando;

Boa resposta transitória: refere-se a habilidade do controlador em garantir que o sistema apresente uma boa resposta transitória, que se reflete basicamente no curto tempo de resposta que um sistema deve apresentar frente a ruídos ou perturbações para voltar ao ponto de equilíbrio;

Erro nulo ou baixo: refere-se a habilidade do controlador em conseguir para o sistema uma saída muito próxima do valor de desejado. Isto implica em dizer que o sistema deve apresenta erro nulo ou baixo;

Robustez: é a habilidade do sistema em permanecer estável diante de alterações imprevistas.

4.2 Modelagem Matemática

Um dos principais objetivos do uso de matemática em sistemas dinâmicos é o de obter uma representação do sistema. O processo de se obter uma representação é normalmente referenciada como **modelagem**, e o produto final da modelagem é o **modelo** do sistema.

A modelagem de sistemas é essencial para o projeto de sistemas de controle. Os modelos podem ser determinados a partir do conhecimento da estrutura interna do sistema, determinando-se as equações diferenciais que descrevem o sistema.

A escolha do modelo matemático depende das particularidades do sistema que se deseja modelar. Estas particularidades vão determinar se o sistema a representar é linear ou não-linear. Um **sistema** é **linear** se atende ao princípio da superposição, no qual, as respostas produzidas pela aplicação simultânea de duas excitações diferentes é igual à soma das suas respostas individuais a cada uma das excitações. No **sistema não-linear** a resposta a duas entradas não pode ser calculada tratando-se uma entrada de cada vez e adicionando-se os resultados.

Na maioria dos casos, os problemas reais são não-lineares, e a determinação de soluções quando se tem não-linearidade é extremamente complicada. Desta forma, estes sistemas são linearizados de modo a obter um sistema linear aproximado, ao qual se pode aplicar métodos lineares que produzirão informações sobre o comportamento do sistema não-linear.

Em muitas situações não é possível e/ou conveniente representar com precisão todas as dinâmicas de um sistema. Desta forma, o modelo matemático pode ser considerado na prática como uma aproximação da dinâmica do sistema real. Conseqüentemente, o modelo matemático obtido pode apresentar diferentes tipos de incertezas decorrentes das dinâmicas não-modeladas, variações paramétricas, presença de ruídos, ou até mesmo erros decorrentes da etapa de linearização entre outros fatores.

4.2.1 Função de transferência

As relações entre a função resposta e a função de excitação de um sistema representado por equações diferenciais lineares e invariantes no tempo são ditas **função de transferência**. A função de transferência é definida como a relação entre a transformada de Laplace do sinal de saída (função resposta) e a transformada de Laplace do sinal de entrada (função de excitação), supondo-se que as condições iniciais são nulas [136].

Seja o seguinte sistema linear e invariante no tempo definido pela seguinte equação diferencial:

$$\begin{aligned} a_0 \overset{(n)}{y} + a_1 \overset{(n-1)}{y} + \ldots + a_{n-1}\dot{y} + a_n y &= b_0 \overset{(m)}{x} + b_1 \overset{(m-1)}{x} + \ldots + \\ & \quad b_{m-1}\dot{x} + b_m x, \text{ com } (n \geq m) \end{aligned} \tag{4.1}$$

onde y é o sinal de saída e x o sinal de entrada. A função de transferência para o sistema representado pela Equação 4.1 é obtida pelas transformadas de Laplace de ambos os membros, na condição de que $x(0) = 0$, ou seja, que as condições iniciais sejam nulas. Desta forma a função de transferência é representada por:

$$\begin{aligned} G(s) &= \frac{\mathcal{L}(Saida)}{\mathcal{L}(Entrada)}\Big|_{condicoesIniciais=0} \\ &= \frac{Y(s)}{X(s)} \\ &= \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1}s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + a_n} \end{aligned} \tag{4.2}$$

Visto que o conceito de função de transferência é restrito à equações diferenciais lineares e invariantes no tempo, a sua aplicabilidade também se limita na análise e projeto de tais sistemas. Dado que a função de transferência é uma propriedade intrínsica do sistema, uma vez estabelecida, ela fornece uma descrição completa das características dinâmicas do sistema.

Apesar da maioria dos sistemas serem contínuos, para permitir que possam ser implementados, é necessário que sejam escolhida uma freqüência de amostragem f_s, para que uma representação no domínio-z possa ser obtida. Para tanto, é necessário definir uma freqüência e definir qual será o método de discretização utilizado para que se possa obter a função de transferência discreta para o sistema, que uma vez obtida, é convertida numa equação de diferença, podendo então, o algoritmo correspondente ser implementado digitalmente e executado a cada intervalo de amostragem $1/f_s$.

4.2.2 Estabilidade e Linearização

Ao se projetar um controlador para um sistema, deve ser possível prever o comportamento dinâmico deste sistema a partir dos seus componentes. Dentre as características do comportamento do sistema, a mais importante é a **estabilidade**, que se refere ao quão bem comportado é o sistema controlado. Um sistema é dito estável se quando o vetor de

estado se afasta ligeiramente do ponto de equilíbrio, ele tende a retornar ao ponto ou, ao menos, não continuar afastando-se [139].

Um vetor $\bar{x}$ é um **ponto de equilíbrio** de um sistema dinâmico se ele tem a propriedade de que uma vez que o estado do sistema é igual a $\bar{x}$ ele permanece igual a $\bar{x}$ para todo o tempo futuro. [139]. Em particular, se um sistema é descrito por sua série de equações diferenciais na forma:

$$\dot{x}(t) = Ax(t) + b \tag{4.3}$$

O ponto de equilíbrio $\bar{x}$ deve satisfazer a equação:

$$0 = A\bar{x}(t) + b \tag{4.4}$$

Apesar da grande maioria dos sistemas dinâmicos serem não-lineares, os procedimentos utilizados para determinar soluções para problemas envolvendo estes sistemas são, em geral, bastante complexos. Desta forma, faz-se necessário, na maioria dos casos, trabalhar com sistemas lineares equivalentes em substituição aos não-lineares.

Se um sistema não-linear opera em torno de um ponto de equilíbrio, então é possível obter um sistema linear, o qual é dito equivalente ao sistema não-linear considerado. No entanto, os sistemas lineares equivalentes só são válidos dentro de uma limitada faixa de operação [136].

O processo de **linearização** de sistemas não-lineares é extremamente importante, pois através da linearização de equações não-lineares é possível aplicar métodos de análise linear que produzirão informação sobre o comportamento do sistema não-linear considerado. Além disto, Lyapunov provou que se um modelo linear de um sistema é válido para um ponto de equilíbrio e é estável, então existe uma região que contém o ponto de equilíbrio no qual o sistema não-linear equivalente é estável. Desta forma, as propriedades de estabilidade de um sistema não-linear podem ser inferidas a partir do sistema linearizado [139, 140].

Os sistemas lineares obtidos através do processo de linearização são utilizados apenas para o projeto do controlador, cuja função é manter o sistema na região do ponto de equilíbrio. Uma vez que o controlador é sintetizado e é observado que possui as características de desempenho esperadas para o sistema linear, deve-se ser realizada uma simulação cuidadosa e precisa do sistema com todas as suas não-linearidades com o intuito de validar as características de desempenho para o sistema não linear.

Nesta seção, introduz-se, uma técnica de linearização aplicável à maioria dos sistemas não-lineares [139].

Seja um sistema não-linear de primeira-ordem representado pela seguinte equação diferencial:

$$\dot{x}(t) = f(x(t)) \tag{4.5}$$

O processo de linearização do sistema 4.5 é baseado na linearização da função não-linear f, aproximando-a do ponto de equilíbrio $\bar{x}$ por:

$$f(\bar{x} + y) = f(\bar{x}) + \frac{d}{dx} f(\bar{x}) y \tag{4.6}$$

Seja um sistema não-linear de ordem n representado pelas seguintes equações diferenciais:

$$\begin{aligned} \dot{x_1} &= f_1(x_1, x_2, \ldots, x_n) \\ \dot{x_2} &= f_2(x_1, x_2, \ldots, x_n) \\ &\vdots \\ \dot{x_n} &= f_n(x_1, x_2, \ldots, x_n) \end{aligned} \tag{4.7}$$

No caso do sistema descrito por 4.7, o processo de linearização se dá pela aproximação de cada uma das funções pela seguinte relação:

$$\begin{aligned} f_i(\bar{x}_1 + y_1, \bar{x}_2 + y_2, \ldots, \bar{x}_n + y_n) \simeq\ & f_i(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)) + \\ & \frac{\partial}{\partial x_1} f_i(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) y_1 + \\ & \frac{\partial}{\partial x_2} f_i(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) y_2 + \ldots + \\ & \frac{\partial}{\partial x_n} f_i(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) y_n \end{aligned} \tag{4.8}$$

A aproximação linear para o vetor $f(x)$ é então obtida pela aproximação em separado de cada uma das n funções. O resultado é então expresso na notação vetorial por:

$$f(\bar{x} + y) = f(\bar{x}) + \mathbf{F} y \tag{4.9}$$

A matriz $\mathbf{F}$ em 4.9 é uma matriz quadrada de dimensão n, denominada de *matriz de Jacobiano* de f, e dada por:

$$\mathbf{F} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{4.10}$$

Seja $\bar{x}$ o vetor que representa o ponto de equilíbrio para o sistema:

$$\dot{x}(t) = f(x(t)) \tag{4.11}$$

Fazendo-se $x(t) = \bar{x} + y(t)$ e utilizando-se a aproximação obtida em 4.9, obtém-se o seguinte sistema linear equivalente para o sistema 4.11, válido para pequenas variações de $y(t)$:

$$\dot{y} = \mathbf{F}y(t) \tag{4.12}$$

4.2.3 Modelagem no estado de espaço

O **estado** de um sistema dinâmico é o menor conjunto de valores de variáveis, denominadas de variáveis de estado, no qual o conhecimento destes valores no tempo $t = t_0$, junto com o conhecimento dos valores de entrada para o tempo $t \geq t_0$, determina completamente o comportamento do sistema em qualquer instante de tempo $t \geq t_0$.

As **variáveis de estado** de um sistema dinâmico são as grandezas cujo conjunto de valores determina o estado do sistema. As n variáveis, $x_1, x_2, \ldots, x_{n-1}, x_n$, necessárias para descrever completamente o comportamento de um sistema são as componentes do vetor de estado x. Um **vetor de estado** determina univocamente o **estado** $x(t)$ para qualquer instante $t \geq t_0$, uma vez conhecido o estado em $t = t_0$ e a função de entrada $u(t)$ para $t \geq t_0$.

O espaço n-dimensional cujos eixos coordenados consistem nos eixos $x_1, x_2, \ldots, x_{n-1}, x_n$ é chamado **espaço de estado** [136]. Qualquer estado pode, então, ser representado por um ponto no espaço de estado.

A análise no espaço de estado envolve as variáveis de entrada, de saída e as variáveis de estado na modelagem de sistemas dinâmicos. Um sistema pode ter várias representações no espaço de estado, mas em todos o número de variáveis de estado é o mesmo.

Seja o sistema com vetor de entrada u de dimensão r, vetor de saída y m-dimensional e o vetor de variáveis de estado x n-dimensional. Em 4.13 e em 4.14 tem-se, respectivamente, a representação da equação de estado e da equação de saída.

$$\dot{x}(t) = f(x,u,t) \quad (4.13)$$
$$y(t) = g(x,u,t) \quad (4.14)$$

Linearizando-se estas equações em torno do ponto de equilíbrio, obtem-se, respectivamente, as equações lineares para o estado e a saída:

$$\dot{x}(t) = \mathbf{A}(t)x(t) + \mathbf{B}(t)u(t) \quad (4.15)$$
$$y(t) = \mathbf{C}(t)x(t) + \mathbf{D}(t)u(t) \quad (4.16)$$

onde $\mathbf{A}(t)$ é a matriz de estado, $\mathbf{B}(t)$ é a matriz de entrada, $\mathbf{C}(t)$ é a matriz de saída e $\mathbf{D}(t)$ é a matriz de transmissão direta.

4.3 Controle Clássico, Controle Moderno, Controle Ótimo e Controle Robusto

O Controle Clássico consiste de métodos no domínio de freqüência, e baseia-se em métodos gráficos para sintetizar controladores. Tais métodos possuem uma natureza heurística, responsável tanto pelo seu sucesso, bem como por sua limitação.

Para suprir as limitações do controle clássico surge o Controle Moderno que trata de sistemas com entradas e saídas múltiplas, lineares ou não lineares e variantes ou invariantes no tempo. A abordagem adotada tem o seu enfoque no domínio da freqüência e é baseada no conceito de estado, onde, o sistema é representado na forma de espaço de estados [136]. Desta forma, o controlador pode ser formulado como um problema de otimização, possibilitando o uso efetivo de métodos de projeto.

Apesar do formalismo para o desenvolvimento de controladores eficientes, o controle moderno é freqüentemente sensível a erros de modelagem, o que levou ao desenvolvimento do *Controle Pós-moderno* (década de 80 e 90) também chamado de método para síntese de controladores ótimos [141].

Na estratégia utilizada no Controle Ótimo, também denominado de controle H_2, os requisitos de desempenho para o sistema são formulados em termos da minimização de uma função de custo. A utilização de uma função objetivo força o projetista a especificar de forma exata os objetivos de controle para o sistema, que uma vez determinados são atendidos de forma ótima. O conhecimento do estado inicial e a correta modelagem do sistema são fundamentais para a exatidão do controlador. Assim, se o modelo e o estado

inicial estão bem definidos e o desempenho do controlador atende aos objetivos desejados, então, pode-se afirmar que o controlador obtido é ótimo.

Controladores ótimos não são sempre tolerantes a alterações, seja no controle do sistema ou no ambiente. Entretanto, qualquer sistema de controle é vulnerável à incertezas, perturbações externas e a ruídos. Desta forma, incertezas relacionadas com a modelagem do sistema, tais como dinâmicas não modeladas na planta, ou relacionadas a distúrbios ou interferências causados por sinais externos podem degradar o desempenho do sistema.

Em muitas aplicações é importante ter garantias que possíveis perturbações de controle não afetem, de maneira proibitiva, as variáveis de interesse sob controle. Portanto, para sistemas sujeitos a perturbações externas deve-se minimizar de alguma maneira a influência destes sinais nas saídas de interesse do sistema através de uma adequada lei de controle.

A este processo de busca de solução de um problema de controle envolvendo o sistema e uma família de incerteza em torno dele, chama-se de Controle Robusto. O objetivo das técnicas de Controle Robusto, também conhecido como controle H_∞, é manter a estabilidade do sistema, garantir alto desempenho do sistema e uma suave degradação do seu desempenho na presença de alterações no sistema. No projeto de controladores, utilizando Controle Robusto, são consideradas as incertezas causadas por falhas na modelagem do sistema ou por distúrbios externos, de forma a garantir que os controladores produzam resultados que satisfaçam os requisitos desejados para o sistema, identificando-se uma faixa de valores para as incertezas, na qual o sistema permanece estável e com o desempenho requerido.

Como se pode observar, para o projeto do controlador robusto H_∞, é feita uma análise do pior caso, na qual, a adequada operação do controlador é garantida sob circunstâncias diversas, provendo assim, um controle mais rigoroso do que o H_2. Contudo, as técnicas de Controle Robusto devem ser utilizadas com precaução, dado que o projeto para o pior caso pode levar a degradação do desempenho do sistema sob condições normais, principalmente, quando o sistema opera na maior parte do tempo sob estas condições.

4.4 Desigualdades Matriciais Lineares (LMIs)

Desigualdades matriciais lineares (LMI´s) possuem a seguinte forma:

$$F(x) \triangleq F_0 + \sum_{i=1}^{m} x_i F_i > 0; \tag{4.17}$$

onde: $x \in R^m$ é a variável e as matrizes simétricas $F_i = F_i^T \in R^{n \times n}, i = 0, ..., m$ são dadas.

A desigualdade significa que $F(x)$ é definida positiva, o que implica que $u^T F(x)u > 0$ para todo u diferente de zero e $u \in R^n$. O LMI (4.17) representa um problema de restrição convexa sobre x, ou seja, o conjunto $\{x|F(x) > 0\}$ é convexo [142]. Tal representação pode indicar uma grande variedade de restrições convexas sobre x.

Como existem programas computacionais especializados, que convergem em tempo polinomial para a resolução de LMIs, formular um problema em termos de LMI´s equivale basicamente a resolver o problema. Inúmeros problemas em diferentes áreas do conhecimento podem ser reformulados e numericamente resolvidos através de LMIs. Em particular, vários problemas de Teoria do Controle podem ser expressos na forma de LMI´s.

No trabalho apresentado em [143], a síntese de controladores para sistemas lineares com atraso são expressos e solucionados como LMIs, possibilitando, assim, que os parâmetros do controlador possam ser obtidos através da resolução de um simples problema convexo, que pode ser resolvido de forma eficiente.

Capítulo 5

Sistema de Controle de Congestionamento TCP/AQM

Neste capítulo, o sistema de controle de congestionamento é apresentado como um problema de controle. É apresentada, primeiramente, uma simplificação do modelo dinâmico do sistema de congestionamento TCP/AQM, que captura o comportamento da variação da janela do TCP Reno bem como do tamanho da fila. O modelo é linearizado para permitir que se possa fazer o projeto e a análise de políticas de AQM mais eficientes. Em seguida, a dinâmica descrita é analisada em termos dos parâmetros de rede e o problema de controle de congestionamento é descrito como um problema de controle.

5.1 Modelo Dinâmico para o Controle de Congestionamento TCP/AQM

Um modelo dinâmico baseado em análises de fluidos e equações diferenciais estocásticas, que captura o comportamento da variação da janela de TCP em função da variação do tamanho da fila, foi introduzido em [9].

O sistema de equações descreve o comportamento AIMD do TCP Reno na sua fase de estabilização, ou seja, quando o crescimento da sua janela é governado basicamente pelo algoritmo de *Congestion Avoidance*. Desta forma, o modelo é consideravelmente preciso para tráfego de longa duração, onde a fase de *Slow Start* é bastante curta, podendo assim ser desconsiderada. Uma versão simplificada deste modelo, onde se assume que as perdas são detectadas apenas pelo recebimento de ACKs duplicados, o que implica em dizer que a recuperação das perdas se dá através da ativação dos algoritmos de *Fast Retransmit* e *Fast Recovery* e que o mecanismo de *timeout* é ignorado é dada por [10]:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)}{2}\frac{W(t-R(t))}{R(t-R(t))}p(t-R(t)); \quad (5.1)$$

$$\dot{q}(t) = \frac{N(t)}{R(t)}W(t) - C(t) + \omega_q(t); \quad (5.2)$$

$$R(t) = \frac{q(t)}{C(t)} + T_p; \quad (5.3)$$

onde:

$W(t)$: é a média do tamanho da janela TCP em pacotes;
$q(t)$: é o tamanho da fila em pacotes;
$\omega_q(t)$: é o ruído gerado pelo tráfego UDP;
$R(t)$: é o tempo total de viagem (RTT) em segundos;
$C(t)$: é a capacidade do enlace em pacotes/segundo;
T_p: é o tempo de propagação em segundos;
$N(t)$: é o fator de carga em número de conexões TCP;
$p(t)$: é a probabilidade de descarte/marcação de pacotes;

Os valores assumidos por W e q são valores limitados e positivos, ou seja, $[0, \bar{W}]$ e $[0, \bar{q}]$. Além disto, a probabilidade de marcação/descarte, p, assume valores apenas entre $[0, 1]$.

A equação (5.1) descreve a dinâmica da janela TCP, onde o primeiro termo corresponde ao seu crescimento aditivo e o segundo termo modela o seu decréscimo multiplicativo.

A equação (5.2), expressa a variação do tamanho da fila como a diferença entre a taxa de chegada, $NW/R + \omega_q(t)$ e a capacidade do enlace, C.

Os roteadores têm que lidar com fluxos TCP como também com fluxos como o UDP, que são não adaptativos, o que significa que não diminuem sua taxa de transmissão na presença de congestionamento. O modelo original apresentado em [10] considera apenas a existência de fluxos TCP e ignora a existência de outros tipos de fluxos. Desta forma, a equação (5.2) difere da equação apresentada em [10] pela presença do termo $\omega_q(t)$, incluído para que a variação no tamanho da fila também leve em consideração os pacotes referentes aos fluxos UDP.

5.2 Linearização do Sistema

Nesta seção o sistema de equações (5.1-5.3), apresentado para o sistema de congestionamento é linearizado. A razão principal, pela qual se faz a modelagem do sistema de congestionamento, e, posteriormente a linearização do modelo obtido é para permitir que se possa fazer o projeto e a análise de políticas de AQM mais eficientes.

Para linearizar o sistema (5.1-5.3), considera-se que o número de conexões TCP, a capacidade do enlace, bem como o ruído são constantes, ou seja, $N(t) \equiv N_0$, $C(t) \equiv C_0$ e $\omega_q(t) = \omega_{q_0}$,

Seja (W, q) o estado do sistema a ser controlado (5.1-5.3), e p a entrada para o sistema. O ponto de equilíbrio para o sistema de congestionamento pode ser definido como o ponto, no qual, as fontes transmitem a uma taxa máxima, tendo perda mínima. Seja (W_0, q_0, p_0), o ponto de equilíbrio do sistema é obtido através da solução de $\dot{W}(t) = 0$, $\dot{q}(t) = 0$, and $R_0 = \frac{q_0}{C_0} + T_p$. A expressão para o ponto de equilíbrio é dada por:

$$W_0 = \sqrt{\frac{2}{p_0}} = \frac{R_0 C_0}{N_0} = \frac{q_0 + C_0 T_p}{N_0}; \tag{5.4}$$

$$q_0 = N_0\sqrt{\frac{2}{p_0}} - C_0 T_p = C_0 R_0 - C_0 T_p; \tag{5.5}$$

$$p_0 = \frac{2N_0^2}{(R_0 C_0)^2} = \frac{2N_0^2}{(q_0 + C_0 T_p)^2}; \tag{5.6}$$

No processo de linearização do sistema, foram ignorados as dependências em relação ao atraso dos argumentos $(t - R)$, no cálculo da variação do tamanho da fila, assumindo-se que este atraso é fixo e igual a $(t - R_0)$. Desta forma, obtém-se a seguinte dinâmica simplificada:

$$\begin{aligned} \dot{W}(t) &= \frac{1}{\frac{q(t)}{C} + T_p} - \frac{W(t)}{2}\frac{W(t - R_0)}{\frac{q(t-R_0)}{C} + T_p} p(t - R_0); \\ \dot{q}(t) &= \frac{N_0)}{\frac{q(t)}{C_0} + T_p} W(t) - C_0 + \omega_q(t); \end{aligned} \tag{5.7}$$

Para linearizar o sistema (5.7), faz-se necessário definir as funções $f = \dot{W}(t)$ e $g = \dot{q}(t)$ e fazer a avaliação de suas derivadas parciais no ponto de equilíbrio (W_0, q_0, p_0). Seja $W = W(t)$, $W_R \doteq W(t - R_0)$, $q = q(t)$, $q_R \doteq q(t - R_0)$ e $p_R \doteq p(t - R_0)$, as funções f e g são definidas por:

$$f(W, W_R, q, q_r, p_r) \doteq \frac{1}{\left(\frac{q}{C_0} + T_p\right)} - \frac{W.W_R.p_R}{2\left(\frac{q_R}{C_0} + T_p\right)}; \tag{5.8}$$

$$g(W, q) \doteq \frac{N_0 W}{\left(\frac{q}{C_0} + T_p\right)} - C_0 + \omega_{q_0}; \tag{5.9}$$

A seguir é apresentado a avaliação das derivadas parciais de $f(W, W_R, q, q_r, p_r)$ e $g(W, q)$, no ponto de equilíbrio (W_0, q_0, p_0). É importante ressaltar que no ponto de equilíbrio tem-se que $W = W_R = W_0$, $q = q_r = q_0$ e $p = p_r = p_0$. Desta forma, na resolução das derivadas parciais, os valores de W, W_R, q, q_r, p_r são substituídos por W_0, q_0 e p_0. Além disto, são utilizados os valores obtidos em (5.4 - 5.6) para colocar o resultado obtido em função dos parâmetros C_0, N_0 e R_0.

Cálculo das derivadas parciais da função $f(W, W_R, q, q_r, p_r)$:

$$
\begin{aligned}
\frac{\partial f}{\partial W} &= -\frac{W_R p_R}{2\left(\frac{q}{C_0} + T_p\right)}; \\
&= -\frac{W_0 p_0}{2R_0}, \text{ substituindo } p_0 = \frac{2}{W_0^2}, \text{ obtem-se:} \\
&= -\frac{1}{R_0 W_0}, \text{ substituindo } W_0 = \frac{R_0 C_0}{N_0}, \text{ obtem-se:} \\
&= -\frac{N_0}{R_0^2 C_0}; \\
\frac{\partial f}{\partial W_R} &= -\frac{W p_R}{2\left(\frac{q}{C_0} + T_p\right)}; \text{ substituindo } R_0 = \frac{q}{C_0} + T_p : \\
&= -\frac{W_0 p_0}{2R_0}; \\
&= \frac{\partial f}{\partial W}; \\
\\
\frac{\partial f}{\partial q} &= \frac{\partial}{\partial q}\left[\frac{1}{\left(\frac{q}{C_0} + T_p\right)}\right]; \\
&= -\frac{1}{C_0}\frac{1}{\left(\frac{q}{C_0} + T_p\right)^2}; \text{ substituindo } R_0 = \frac{q}{C_0} + T_p : \\
&= -\frac{1}{R_0^2 C_0};
\end{aligned}
$$

$$\begin{aligned}
\frac{\partial f}{\partial q_R} &= \frac{\partial}{\partial q_R}\left[-\frac{W.W_R.p_R}{2\left(\frac{q_R}{C_0}+T_p\right)}\right]; \\
&= \frac{W.W_R.p_R}{2C_0}\frac{1}{\left(\frac{q_R}{C_0}+T_p\right)^2}; \text{ substituindo } R_0 = \frac{q_R}{C_0}+T_p: \\
&= \frac{W_0^2 p_0}{2R_0^2 C_0};, \text{ como } W_0^2 p_0 = 2 \text{ tem-se que:} \\
&= \frac{1}{R_0^2 C_0}; \\
\frac{\partial f}{\partial p_R} &= -\frac{W.W_R}{2\left(\frac{q_R}{C_0}+T_p\right)}; \text{ substituindo } R_0 = \frac{q_R}{C_0}+T_p: \\
&= -\frac{W_0^2}{2R_0}, \text{ substituindo } W_0 = \frac{R_0 C_0}{N_0}, \text{ obtem-se:} \\
&= -\frac{R_0 C_0^2}{2N_0^2};
\end{aligned}$$

Cálculo das derivadas parciais de $g(W, q)$:

$$\begin{aligned}
\frac{\partial g}{\partial W} &= \frac{N_0}{\left(\frac{q}{C_0}+T_p\right)}; \text{ substituindo } R_0 = \frac{q}{C_0}+T_p: \\
&= \frac{N_0}{R_0}; \\
\frac{\partial g}{\partial q} &= \frac{\partial}{\partial q}\left[\frac{N_0 W}{\left(\frac{q}{C_0}+T_p\right)}\right]; \\
&= -\frac{N_0 W}{C_0}\frac{1}{\left(\frac{q}{C_0}+T_p\right)^2}; \text{ substituindo } R_0 = \frac{q}{C_0}+T_p: \\
&= -\frac{N_0 W_0}{C_0 R_0^2}, \text{ substituindo } W_0 = \frac{R_0 C_0}{N_0}, \text{ obtem-se:} \\
&= -\frac{1}{R_0};
\end{aligned}$$

Utilizando-se as derivadas parciais obtidas anteriormente, a linearização do sistema (5.7) em torno do ponto de equilíbrio (W_0, q_0, p_0), resulta em:

$$\begin{aligned}
\dot{x}_1(t) &= -\frac{N_0}{R_0^2 C_0}(x_1(t) + x_1(t - R_0)) - \frac{R_0 C_0^2}{2N_0^2} u(t - R_0) \\
&\quad -\frac{1}{R_0^2 C_0}(x_2(t) - x_2(t - R_0)); \\
\dot{x}_2(t) &= \frac{N_0}{R_0} x_1(t) - \frac{1}{R_0} x_2(t);
\end{aligned} \tag{5.10}$$

onde $x_1(t)$, $x_2(t)$ e $u(t)$ representam a perturbação das variáveis em torno do ponto de equilíbrio e são dadas por:

$$\begin{aligned}
x_1(t) &\doteq W(t) - W_0; \\
x_2(t) &\doteq q(t) - q_0; \\
u(t) &\doteq p(t) - p_0;
\end{aligned}$$

5.3 Controle de Congestionamento como um Problema de Controle

O TCP fonte utiliza os reconhecimentos ou ACKs enviados pelo TCP receptor para estimar a quantidade de banda passante disponível, e assim, adequar sua taxa de transmissão de acordo com o estado de congestionamento da rede. Como apresentado no Capítulo 2, o TCP infere a existência de congestionamento quando tem seus pacotes descartados pelos roteadores, e utiliza a falta de reconhecimento como uma indicação de que perdas aconteceram. As políticas de AQM tem como função principal a monitoração do nível de ocupação das filas, e, assim, fazem a notificação do congestionamento incipiente para os nós emissores através da marcação/descarte de pacotes.

O controle do congestionamento pode ser visto como um sistema de controle por retroalimentação, no qual, a taxa de transmissão dos nós fontes é ajustada de acordo com o estado de congestionamento na rede, inferido pelos roteadores através da monitoração do nível de suas filas e retornado para os nós fontes através da marcação/descarte de pacotes (Figura 5.1).

As políticas de AQM baseadas em Teoria de Controle consideram a natureza intrínseca de retroalimentação do sistema de controle de congestionamento. Busca-se compreender a dinâmica do congestionamento, para poder garantir-se estabilidade e robustez das condições de equilíbrio das métricas de desempenho desejadas.

A taxa de transmissão dos nós fontes deve ser, então, ajustada de acordo com o

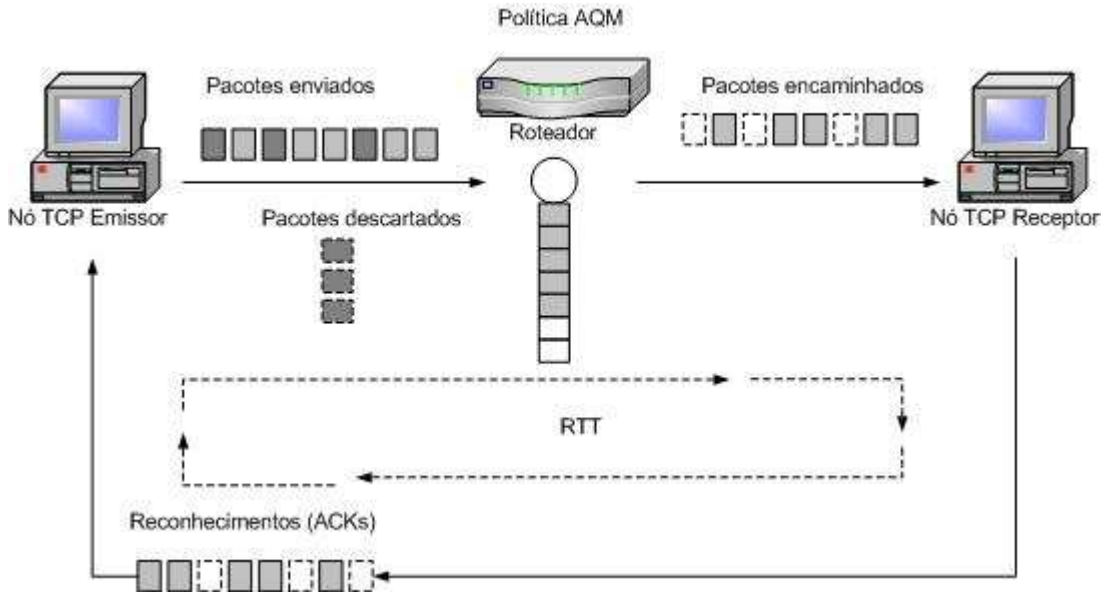

Figura 5.1: Sistema de Controle de Congestionamento

estado de congestionamento, que é determinado pela ocupação da fila. Desta forma, os controladores são responsáveis por determinar qual a probabilidade de descarte/marcação adequada, que maximize a vazão e minimize as perdas, e que ainda garanta a estabilidade do tamanho da fila independentemente das variações das condições da rede.

No caso específico de gerenciamento de filas, a importância da estabilidade está no fato de que sistemas de AQM instáveis produzem grandes variações de retardo e baixa utilização do enlace.

Na Figura 5.2, o sistema de congestionamento é apresentado como um sistema de controle com retroalimentação. A função do controlador AQM, $C(s)$, é marcar os pacotes com probabilidade p, como função do tamanho da fila medido q, como também estabilizar a planta, denotado pela função de transferência $P(s)$, irracional em s, que descreve como a probabilidade de descarte/marcação de pacotes afeta a taxa de transmissão e conseqüentemente o tamanho da fila.

A planta $P(s)$, pode ser determinada através da análise da dinâmica do sistema de controle de congestionamento descrita em (5.10). Tal dinâmica pode ser analisada em termos de parâmetros de rede tais como o número de fluxos TCP, N_0, o tempo total de viagem (RTT), R_0, a capacidade do enlace, C_0, e em termos da natureza de retroalimentação do AQM [10].

Pode-se verificar, na Figura 5.2, que o controlador $C(s)$, estima quão longe o valor do tamanho da fila medido q, está do valor do ponto operacional q_0, e utiliza esta diferença para determinar quanto deve variar a probabilidade de descarte. Tal variação é, então, uti-

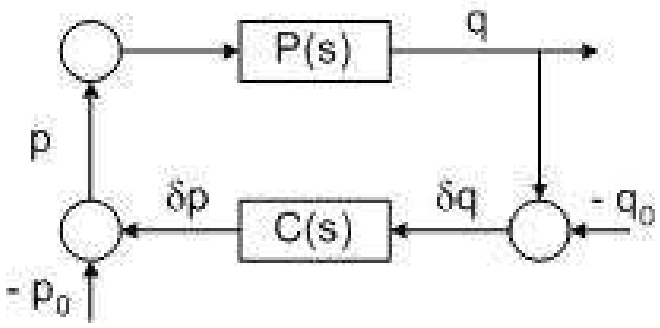

Figura 5.2: Controle de Congestionamento como um Sistema de Controle com Retroalimentação

lizada para obter o valor de p em função do valor da probabilidade no ponto de equilíbrio, p_0. Desta forma, o controlador $C(s)$, através da monitoração do tamanho da fila, determina qual o valor adequado da probabilidade de descarte/marcação, que deve ser utilizada de modo a fazer com que o sistema estabilize em torno do ponto de equilíbrio, independentemente das condições da rede.

Capítulo 6

Projeto de um controlador Ótimo AQM

Neste capítulo, introduz-se o desenvolvimento do projeto do controlador ótimo para AQM, denominado H2-AQM, que utiliza o modelo simplificado da dinâmica do sistema de controle de congestionamento, apresentado no Capítulo 5. Como apresentado na Seção 4.3, o controle moderno é freqüentemente sensível a erros de modelagem, enquanto que na estratégia utilizada no Controle Ótimo, os requisitos de desempenho para o sistema são formulados em termos da minimização de uma função de custo. A utilização de uma função objetivo força o projetista a especificar de forma exata os objetivos de controle para o sistema, que uma vez determinados são atendidos de forma ótima.

A maioria das políticas de AQM baseadas em Teoria de Controle [108, 110] utilizam apenas as informações da dinâmica atual do sistema, e assim, desconsideram os termos dependentes do atraso. É de conhecimento notório que o uso de controle independente do atraso possui limitações de desempenho na presença de grandes atrasos, quando comparado com o controle dependente do atraso [111,112]. No contexto de redes, isto significa que a utilização da abordagem dependente do atraso ou não racional faz com que os recursos de rede possam ser usados eficientemente [110]. Desta forma, neste capítulo, o sistema de controle de congestionamento apresentado em (5.10) é representado na forma de espaço de estado como um sistema linear com atraso e uma abordagem não racional é usada para derivar o controlador ótimo H_2. Utilizando-se uma abordagem não racional, supera-se a principal dificuldade de se projetar controladores racionais para sistemas com atraso, que é incorporar no projeto do problema a matriz de multiplicação utilizada para provar a estabilidade com relação à parte com atraso do sistema [143].

A síntese do controlador é baseada nos resultados apresentados em [143], onde os projetos de controladores para sistemas lineares com atraso são expressos e solucionados como desigualdades matriciais lineares, LMIs (*Linear Matrix Inequalities*), possibilitando

que os parâmetros do controlador possam ser obtidos através da resolução de um simples problema convexo. Neste trabalho, a forma do controlador foi cuidadosamente escolhida de modo a reproduzir a estrutura da planta do sistema linear com atraso contínuo no tempo. A estabilidade do sistema é, então, analisada através da conexão da planta com o controlador. A razão principal para tal escolha é a possibilidade do uso de análise de estabilidade independente de atraso que gera condições de síntese que podem ser expressas e solucionadas como LMIs. Utilizando-se esta forma para o controlador, não apenas a estabilidade, mas também os objetivos de desempenho do sistema de AQM podem ser completamente expressos e solucionados como LMIs [143].

Os objetivos de projeto definem a forma pela qual o controlador faz com que o sistema atinja o seu ponto de equilíbrio. Assim, neste capítulo, são discutidos quais devem ser os objetivos de projeto de um controlador ótimo para AQM, de forma a obter o melhor desempenho. Na obtenção dos controladores foi investigado qual o melhor ponto de equilíbrio para o sistema de controle de congestionamento. Além disto, é feita uma análise da robustez do sistema, utilizando-se técnicas de controle robusto.

Como o trabalho apresentado nesta tese foi baseado no PI-AQM, esta política é apresentada de forma mais detalhada e o controlador é obtido utilizando-se o mesmo ponto de equilíbrio usado para derivar os controladores ótimos.

6.1 Projeto do controlador H2

O sistema linear apresentado em (5.10) pode ser expresso no espaço de estado pelas seguintes equações que descrevem um sistema linear com atraso contínuo no tempo:

$$\begin{aligned} \dot{x}(t) &= A_0 x(t) + A_1 x(t - R_0) + B_w w(t) + B_u u(t); \\ z(t) &= C_{z0} x(t) + C_{z1} x(t - R_0) + D_{zu} u(t); \\ y(t) &= C_y x(t - R_0) + D_{yw} w(t); \end{aligned} \tag{6.1}$$

onde:

$x(t)$ é o vetor de estado;
$u(t)$ é a entrada a ser controlada e representa a probabilidade $p(t)$;
$w(t)$ é o ruído externo produzido pelos fluxos UDP;
$z(t)$ é a saída de referência, ou seja, saída esperada para o sistema;
$y(t)$ é a saída obtida para o sistema.

Devido a questões de retroalimentação, o atraso presente em (5.10) é apresentado em (6.1) na saída $y(t)$. Agora, considere que o sistema descrito em (6.1) esteja conectado

com o controlador:

$$\begin{aligned} \dot{\hat{x}}(t) &= \hat{A}_0\hat{x}(t) + \hat{A}_1\hat{x}(t - R_0) + \hat{B}y(t); \\ u(t) &= \hat{C}_0\hat{x}(t) + \hat{C}_1\hat{x}(t - R_0) + \hat{D}y(t); \end{aligned} \tag{6.2}$$

Este controlador também pode ser expresso no domínio da freqüência pela função de transferência não racional:

$$C_{H_2}(s) = (\hat{C}_0 + \hat{C}_1 e^{-sR_0})(sI - \hat{A}_0 - \hat{A}_1 e^{-sr_0})^{-1}\hat{B} + \hat{D}; \tag{6.3}$$

A forma do controlador $C(s)$, definido por (6.2) ou (6.3), foi cuidadosamente escolhido de forma a reproduzir em detalhes a estrutura da planta do Sistema (6.1) [143]. Desta forma, a estabilização da planta pelo controlador implica na estabilidade do sistema de congestionamento independente das condições de rede.

O objetivo é determinar as matrizes do controlador (6.2) que estabilizem o Sistema (6.1) e minimizem certas medidas na saída de referência $z(t)$ de acordo com a função objetivo. De forma a atingir este objetivo de projeto, é necessário, então, definir quais são os objetivos de desempenho desejados para a saída $z(t)$, bem como, o que deve ser medido na saída, $y(t)$. Isto implica que os objetivos de desempenho para a política AQM projetada devem estar representados no controlador $C_{H_2}(s)$.

As matrizes do controlador (6.2) que estabilizam o sistema (6.1) e minimizem certas medidas na saída de referência $z(t)$ são definidas como:

$$A_0 = \begin{bmatrix} -\frac{N_0}{R_0^2 C_0} & -\frac{1}{R_0^2 C_0} \\ \frac{N_0}{R_0} & -\frac{1}{R_0} \end{bmatrix},$$

$$A_1 = \begin{bmatrix} -\frac{N_0}{R_0^2 C} & \frac{1}{R_0^2 C_0} \\ 0 & 0 \end{bmatrix},$$

$$B_u = \begin{bmatrix} -\frac{R_0 C_0^2}{2N_0^2} \\ 0 \end{bmatrix},$$

$$B_w = \begin{bmatrix} 0 & 0 \\ 0.2C_0 & 0 \end{bmatrix},$$

$$D_{yw} = \begin{bmatrix} 0 & 0.02C_0 \end{bmatrix},$$

$$C_y = \begin{bmatrix} 0 & 1 \end{bmatrix};$$

As matrizes A_0, A_1 e B_u são obtidas diretamente da linearização do sistema.

A_0 representa os termos do sistema sem atraso relacionados a W e q. A primeira linha corresponde a $\frac{\partial f}{\partial W}$ e a $\frac{\partial f}{\partial q}$ e a segunda corresponde a $\frac{\partial g}{\partial W}$ e a $\frac{\partial g}{\partial q}$. A matriz A_1 representa os termos com atraso do sistema. Sua primeira linha corresponde a $\frac{\partial f}{\partial W_R}$ e a $\frac{\partial f}{\partial q_R}$. A segunda linha desta matriz é nula, dado que $\frac{\partial g}{\partial W_R} = 0$ e $\frac{\partial g}{\partial q_R} = 0$. A matriz B_u contém os termos da linearização do sistema referentes a entrada que deve ser controlada, a probabilidade de descarte p. A primeira linha contem $\frac{\partial f}{\partial u}$ e a segunda $\frac{\partial g}{\partial p} = 0$.

O ruído existente no sistema, gerado pelos fluxos UDP, é controlado por B_w. O valor escolhido, $0.2C_0$, permite que os fluxos UDP utilizem até 20% da capacidade do enlace. Este valor é uma margem de tolerância satisfatória, dado que cerca de 83% dos *bytes* transmitidos na Internet são gerados pelo TCP [24].

C_y indica que o valor de interesse medido na saída é o tamanho da fila no RTT anterior. D_{yw}, pondera o ruído medido na saída, que é geralmente 10% do valor presente na matriz B_w.

As matrizes C_{z0}, C_{z1} e D_{zu} definem o objetivo de desempenho esperado para a saída de referência do sistema, $z(t)$.

6.2 Objetivos de Projeto

Existem diferentes maneiras pelas quais um controlador faz com que um sistema de AQM saia de um estado qualquer e atinja o ponto de equilíbrio, no qual estabiliza-se. O caminho percorrido para atingir a estabilidade depende dos objetivos utilizados para projetar o controlador.

Em outras palavras, pode-se dizer que os objetivos de projeto definem a forma pela qual o controlador faz com que o sistema atinja o seu ponto de equilíbrio. Objetivos distintos levam a projetos de controladores diferentes, e, conseqüentemente, impactam de forma diferenciada o desempenho obtido para estes controladores.

Na Seção 3.3, foi apresentado que dentre os objetivos de uma política AQM estão: prevenção e controle do congestionamento, eficiente utilização da fila, garantia de baixo retardo e de baixa variação no retardo, escalabilidade e robustez, simplicidade, justiça, prevenção do fenômeno de sincronização global, e prevenção de perdas de pacotes desnecessárias. Apesar de todos os requisitos descritos anteriormente serem importantes, quais deles devem ser utilizados no projeto de uma política AQM é uma questão crucial que precisa ser respondida para que uma política efetiva possa ser definida.

Desta forma, nesta seção, são discutidos quais devem ser os objetivos de projeto de um controlador ótimo para AQM, de forma a obter o melhor desempenho. A discussão leva em consideração a característica de retroalimentação do sistema de congestionamento (6.1) e do seu estado (W, q). Os objetivos de desempenho são expresso através da atribuição de

diferentes valores para as matrizes C_{z0}, C_{z1} e D_{zu}, que são definidas de forma a garantir não apenas a estabilidade do sistema, mas também que, o ponto de equilíbrio seja atingido o mais rápido possível e que a variação em torno deste ponto seja minimizada.

6.2.1 Objetivo 1: Prevenir a Subutilização do Enlace e Minimizar a Ocorrência de *Jitter*

Minimizar a ocorrência de *jitter* garante estabilidade da fila em torno do ponto de equilíbrio, e conseqüentemente, estabilidade para as métricas relacionas com o tamanho da fila, atraso e *jitter*. Para tanto, a primeira linha da matriz C_{z0} está relacionada com o tamanho da fila, e expressa o objetivo de minimizar a distância entre o valor medido para o tamanho da fila q, e o seu valor ideal obtido no ponto de equilíbrio, q_0. A segunda linha está relacionada com a variação do tamanho da fila, e expressa o objetivo de minimizar *jitter*.

Os valores desejados na saída esperada do sistema $z(t)$, são os valores obtidos no intervalo corrente e não no intervalo anterior. Desta forma, a matriz C_{z1} é uma matriz nula.

A matriz D_{zu}, pondera o valor da probabilidade de descarte/marcação na saída. Diferentes valores de pesos foram verificados, variando de 0.3 a 0.9. Os resultados obtidos foram bastante similares, sendo 0.5 o valor adotado.

As matrizes C_{z0} e D_{zu} são:

$$C_{z0} = \begin{bmatrix} 0 & 1 \\ \frac{N_0}{R_0} & -\frac{1}{R_0} \\ 0 & 0 \end{bmatrix},$$

$$D_{zu} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix};$$

6.2.2 Objetivo 2: Atingir a Vazão Ideal mais Rapidamente e Prevenir a Subutilização do Enlace

Para atingir este objetivo, o mais importante é determinar quão rápido o ponto de equilíbrio é atingido independentemente das oscilações que possam vir a ocorrer.

Quando o ponto de equilíbrio é atingido, o TCP emissor obtém a sua vazão ideal, o que significa que obteve a sua porção justa da banda passante. Para atingir a vazão ideal mais rapidamente, deve-se minimizar a distância do valor medido para o tamanho da janela W, e o seu valor ideal obtido no ponto de equilíbrio, W_0, o que é refletido na

primeira linha da matriz C_{z0}. Além disto, como se deseja também prevenir a subutilização do enlace, o tamanho da fila deve ficar em torno do ponto de equilíbrio, ou seja, deve ser minimizada a distância entre o valor medido para o tamanho da fila q, e o seu valor ideal obtido no ponto de equilíbrio, q_0, o que foi refletido na segunda linha da matriz. C_{z0} é então dada por:

$$C_{z0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

As matrizes C_{z1} e D_{zu}, são as mesmas apresentadas para o primeiro objetivo.

6.2.3 Objetivo 3: Garantir Banda Passante e Diminuir a Ocorrência de *Jitter*

As matrizes C_{z0}, C_{z1} e D_{zu} representam o objetivo de minimizar a variação em torno dos valores do ponto de equilíbrio da janela de transmissão e do tamanho da fila, o que garante banda passante mais constante e previne a ocorrência de *jitter*. Este objetivo é especialmente importante para provimento de requisitos de QoS, tais como minimização de *jitter* e garantia mínima de banda passante.

Para se garantir tal objetivo tem-se que as matrizes C_{z0}, C_{z1} e D_{zu} devem ser, respectivamente, iguais as matrizes A_0, A_1 e B_u do Sistema (6.1), que caracterizam justamente como o tamanho da janela e da fila variam.

6.2.4 Objetivo 4: Atingir a Vazão Ideal mais Rapidamente e Diminuir a Ocorrência de *Jitter*

A matriz C_{z0} é determinada de forma a atingir a vazão máxima mais rapidamente e diminuir a ocorrência de *jitter*. Como explicado anteriormente, para atingir a vazão máxima mais rapidamente deve-se minimizar a distância do valor medido para o tamanho da janela W, e o seu valor ideal obtido no ponto de equilíbrio, W_0, o que foi refletido na primeira linha desta matriz. Além disto, para diminuir a ocorrência de *jitter*, deve-se diminuir a variação do tamanho da fila q em torno do ponto de equilíbrio, q_0, o que foi refletido na segunda linha da matriz. Desta forma, a matriz C_{z0} é dada por:

$$C_{z0} = \begin{bmatrix} 1 & 0 \\ \frac{N_0}{R_0} & -\frac{1}{R_0} \\ 0 & 0 \end{bmatrix}$$

As matrizes C_{z1} e D_{zu}, são as mesmas apresentadas para o primeiro objetivo.

6.3 Síntese dos Controladores Ótimos

A síntese do controlador segue a abordagem apresentada em [143], onde os projetos de controladores para sistemas lineares com atraso são expressos e solucionados como desigualdades matriciais lineares, LMIs (*Linear Matrix Inequalities*).

Depois de definidos os objetivos de desempenho, o próximo passo é conectar o Sistema (6.1) com o Controlador (6.2). Seja então, $\bar{x}(t)$ o vetor de estado aumentado que contém o vetor de estado $x(t)$ e o vetor de estado do controlador $\hat{x}(t)$:

$$\bar{x}(t) = \begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix}; \tag{6.4}$$

A conexão do Sistema (6.1) com o Controlador (6.2) resulta no sistema linear com atraso:

$$\begin{aligned} \dot{\bar{x}}(t) &= \mathcal{A}_0\bar{x}(t) + \mathcal{A}_1\bar{x}(t-R_0) + \mathcal{B}w(t); \\ z(t) &= \mathcal{C}_0\bar{x}(t) + \mathcal{C}_1\bar{x}(t-R_0) + \mathcal{D}w(t); \end{aligned} \tag{6.5}$$

onde:

$$\mathcal{A}_0 = \begin{bmatrix} A_0 & B_u\hat{C}_0 \\ 0 & \hat{A}_0 \end{bmatrix},$$

$$\mathcal{A}_1 = \begin{bmatrix} A_1 + B_u\hat{D}C_y & B_u\hat{C}_1 \\ \hat{B}C_y & \hat{A}_1 \end{bmatrix},$$

$$\mathcal{B} = \begin{bmatrix} B_w + B_u\hat{D}D_{yw} \\ \hat{B}D_{yw} \end{bmatrix},$$

$$\mathcal{C}_0 = \begin{bmatrix} C_z & D_{zu}\hat{C}_0 \end{bmatrix},$$

$$\mathcal{C}_1 = \begin{bmatrix} D_{zu}\hat{D}C_y & D_{zu}\hat{C}_1 \end{bmatrix},$$

$$\mathcal{D} = D_{zu}\hat{D}D_{yw};$$

Para garantir a estabilidade do Sistema (6.5), o Teorema 4-b apresentado em [143] é usado. Este teorema especifica que um sistema como (6.5) é assintoticamente estável e $\|H_{wz}(s)\|_2^2 < \gamma$, se $\mathbf{D} = 0$ e se existem matrizes simétricas e definidas positivas W, Y_0

e X_j, e matrizes F, R, L_j e Q_j, com $j = 0, 1$, tais que os seguintes LMIs tenham uma solução factível:

$$\begin{bmatrix} \mathbf{A}_0 + \mathbf{A}_0^T + X_1 & (\bullet)^T & (\bullet)^T \\ \mathbf{A}_1^T & -X_1 & (\bullet)^T \\ \mathbf{C}_0 & \mathbf{C}_1 & -I \end{bmatrix} < 0 \tag{6.6}$$

$$\begin{bmatrix} W & (\bullet)^T \\ \mathbf{B} & \mathbf{P}_0 \end{bmatrix} > 0, \quad trace(W) < \gamma \tag{6.7}$$

onde $\mathbf{A}_0$, $\mathbf{A}_1$, $\mathbf{B}$, $\mathbf{C}_0$, $\mathbf{C}_1$ e $\mathbf{P}_0$ são dadas por:

$$\mathbf{A}_0 = \begin{bmatrix} A_0X_0 + B_uL_0 & A_0 \\ Q_0 & Y_0A_0 \end{bmatrix},$$

$$\mathbf{A}_0 = \begin{bmatrix} A_0X_0 + B_uL_0 & A_0 \\ Q_0 & Y_0A_0 \end{bmatrix},$$

$$\mathbf{A}_1 = \begin{bmatrix} A_1X_0 + B_uL_1 & A_1 \\ Q_1 & Y_0A_1 + FC_y \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} B_w \\ Y_0B_w + FD_{yw} \end{bmatrix},$$

$$\mathbf{P}_0 = \begin{bmatrix} X_0 & I \\ I & Y_0 \end{bmatrix},$$

$$\mathbf{C}_0 = \begin{bmatrix} C_zX_0 + D_{zu}L_0 & C_z \end{bmatrix},$$

$$\mathbf{C}_1 = \begin{bmatrix} D_{zu}L_1 & 0 \end{bmatrix};$$

6.3.1 Determinação do Ponto de Equilíbrio

Para resolver o problema de programação convexa representado pelos LMIS (6.6) e (6.7), faz-se necessário definir qual o ponto de equilíbrio para o sistema, ou seja, devem ser atribuídos valores para os parâmetros de rede R_0, N_0 e C_0 de modo a obter o ponto de equilíbrio (W_0, q_0, p_0).

Uma política AQM tem como função principal gerenciar a fila dos roteadores com

o intuito de fazer a notificação para os nós fontes da ocorrência de congestionamento. Geralmente o congestionamento se inicia nos enlaces gargalo, onde a carga da rede, representada pelo número de conexões ativas, é muito grande.

O TCP Reno não consegue recuperar-se de perdas sem a ocorrência de um *timeout*, quando o tamanho da janela é menor que quatro segmentos. Além disto, a redução da janela à metade quando uma perda é detectada pela recebimento de ACKs duplicados, faz com que o valor da janela deva estar em torno de oito segmentos para evitar que uma futura perda seja detectada apenas pela ocorrência de *timeouts*. Isto pode ser visto como um limite superior para o número de conexões que podem compartilhar um enlace obtendo, uma taxa mínima de transmissão, e prevenindo a ocorrência de *timeouts* nestas conexões e suas conseqüências danosas [86].

O ponto de equilíbrio foi escolhido com o intuito de se obter um ponto de equilíbrio para quando o sistema de congestionamento está mais suscetível, ou seja, quando a carga da rede é alta. O problema de programação convexa foi resolvido numericamente utilizando a rotina LMISol [144]. Os parâmetros de rede utilizados foram escolhidos de forma a obter um valor para a probabilidade p_0 que garanta uma janela mínima de $W_0 = 8$, maximizando assim o número de conexões que podem estar ativas. Os parâmetros escolhidos foram: $R_0 = 0.256$ segundos, $C_0 = 3750$ pacotes/segundos, o que corresponde a um enlace de 15 Mb/s e pacotes com tamanho médio de 500 *bytes* e $N_0 = 120$ sessões TCP.

Uma solução factível foi encontrada, ou seja, foram encontrados valores para as matrizes W, Y_0, X_j, F, R, L_j e Q_j, com $j = 0, 1$. Desta forma, pode-se afirmar que o sistema (6.5) é estável.

6.3.2 Determinação dos Parâmetros dos Controladores

O Teorema 4-b apresentado em [143] diz que se for encontrado uma solução factível para os LMI's (6.6) e (6.7), além de se poder afirmar que o sistema (6.5) é estável, pode-se utilizar as matrizes encontradas W, Y_0, X_0, X_1, F, R, L_0, L_1, Q_0 e Q_1, para determinar os parâmetros do controlador (6.2).

Primeiramente, matrizes arbitrárias U_0 e V_0 devem ser escolhidas tais que $V_0U_0 = I - Y_0X_0$. As matrizes utilizadas foram $U_0 = X_0$ e $V_0 = X_0^{-1} - Y_0$. Assim, os parâmetros do controlador podem ser determinados por [143]:

$$\begin{bmatrix} \hat{A}_0 & \hat{A}_1 & \hat{B} \\ \hat{C}_0 & \hat{C}_1 & \hat{D} \end{bmatrix} = \mathcal{K}.\mathcal{M}.\mathcal{N}. \tag{6.8}$$

onde:

$$\mathcal{K} = \begin{bmatrix} V_0^{-1} & -V_0^{-1}Y_0B_u \\ 0 & I \end{bmatrix};$$

$$\mathcal{M} = \begin{bmatrix} Q_0 - Y_0A_0X_0 & Q_1 - Y_0A_1X_0 & F \\ L_0 & L_1 & 0 \end{bmatrix};$$

$$\mathcal{N} = \begin{bmatrix} U_0^{-1} & 0 & 0 \\ 0 & U_0^{-1} & 0 \\ -C_{y0}X_0U_0^{-1} & -C_{y1}X_0U_0^{-1} & I \end{bmatrix};$$

Nas soluções obtidas, as matrizes $\hat{A}_1$ e $\hat{C}_1$ são aproximadamente zero, sendo portanto ignoradas. Tem-se, conseqüentemente, o cancelamento do termo de atraso do sistema, e os controladores tornam-se racionais. O cancelamento do termo de atraso, quando possível, é a estratégia global ótima para solucionar o problema de minimização da norma H_2 [143]. As funções de transferência no domínio da freqüência para os controladores correspondentes a cada um dos objetivos são, então, dados por:

$$\begin{aligned} C_{H2_1}(s) &= \frac{6.123s + 26}{s^2 + 1.881e^5s + 7.989e^5}; \\ C_{H2_2}(s) &= \frac{0.01341s + 4.817}{s^2 + 431.9s + 1.563e^5}; \\ C_{H2_3}(s) &= \frac{0.02992s + 7.569e^{-5}}{s^2 + 910.3s + 0.0125}; \\ C_{H2_4}(s) &= \frac{6.325s + 0.119}{s^2 + 1.929e^5s + 3623}; \end{aligned} \tag{6.9}$$

onde $C_{H2_i}(s)$ representa o controlador derivado utilizando-se o i_{esimo} objetivo.

6.4 Implementação Digital dos Controladores

O cálculo da probabilidade de descarte em RED é feito sempre quando novos pacotes são recebidos, e é uma função linear do tamanho médio da fila, que varia de forma significativa quando há um grande número de fluxos ativos ou quando a taxa de chegada de pacotes aproxima-se da capacidade do enlace. Para um enlace gargalo congestionado com capacidade de 3750 pacotes/segundo, o valor da probabilidade de descarte é calculado a uma taxa em torno de 3750 Hz, variando a probabilidade de descarte consideravelmente em um curto espaço de tempo, fazendo com que RED falhe em marcar os pacotes de forma aleatória, não obtendo sucesso em prevenir o fenômeno de sincronização global, além de gerar perdas de pacotes desnecessárias.

Para que os controladores obtidos possam ser implementados, é necessário que seja escolhida uma freqüência de amostragem f_s, para que uma representação no domínio-z possa ser obtida. Para tanto, é necessário definir uma freqüência adequada de modo a evitar o descarte desnecessário de pacotes e ainda estabilizando o tamanho da fila. Desta forma, a probabilidade de descarte deve ser calculada a uma freqüência que faça com que o mecanismo de controle de congestionamento possa reagir de forma apropriada a mudanças abruptas, mas ao mesmo tempo não reaja de forma demasiadamente severa, para permitir que mudanças transientes sejam desconsideradas [145].

Para se determinar a freqüência adequada para obter uma representação no domínio-z para os controladores obtidos, foram verificadas valores de freqüências variando de 10% a 50% da capacidade do enlace, C_0. A freqüência escolhida foi de $f_s = 375Hz$, ou 10% do valor da capacidade do enlace, C_0, dado que os controladores obtidos não apresentavam diferenças significativas. Esta também foi a freqüência de amostragem utilizada para derivar o controlador PI-AQM [122].

Os controladores C_{H2_i} no domínio-z, utilizando a freqüência de amostragem de $f_s = 375Hz$, são então dados por:

$$\begin{aligned}
C_{H2_1}(z) &= \frac{a(z+1)}{z+b} = \frac{3.2424^{-5}(z+1)}{z+0.9921};\\
C_{H2_2}(z) &= \frac{az^2+bz-c}{z^2-dz+e} = \frac{1.427^{-5}z^2+9.24e^{-6}z-5.029e^{-6}}{z^2-0.7792z+0.3787};\\
C_{H2_3}(z) &= \frac{a(z+1)}{z+b} = \frac{1.802^{-5}(z+1)}{z+0.09654};\\
C_{H2_4}(z) &= \frac{a(z+1)}{z+b} = \frac{3.2661^{-5}(z+1)}{z+0.9923};
\end{aligned}$$

As funções de transferência entre $\delta p = p - p_0$ e $\delta q = q - q_0$, podem ser convertidas nas seguintes equações de diferenças no tempo discreto kT, onde $T = \frac{1}{f_s}$:

$$\delta p_1(kT) = a[\delta q(kT) + \delta q((k-1)T)] - b\delta p_1((k-1)T); \quad (6.10)$$

$$\begin{aligned}
\delta p_2(kT) &= a\delta q(kT) + b\delta q((k-1)T) - c\delta q((k-2)T) \quad (6.11)\\
&\quad + d\delta p_2((k-1)T) - e\delta p_2((k-2)T);
\end{aligned}$$

$$\delta p_3(kT) = a[\delta q(kT) + \delta q((k-1)T)] - b\delta p_3((k-1)T); \quad (6.12)$$

$$\delta p_4(kT) = a[\delta q(kT) + \delta q((k-1)T)] - b\delta p_4((k-1)T); \quad (6.13)$$

Pelas equações de diferenças apresentadas anteriormente, pode-se verificar que as Equações (6.10), (6.12) e (6.13) tem o mesmo formato. Reescrevendo as equações de

diferenças apresentadas, pode-se ter uma visão intuitiva do comportamento dos controladores. As Equações (6.10), (6.12) e (6.13) foram reescritas de acordo com a Equação (6.14), e a Equação (6.13) de acordo com a Equação (6.16):

$$\delta p_1(kT) = a[2\delta q(kT) - (\delta q(kT) - \delta q((k-1)T))] - b\delta p_1((k-1)T); \quad (6.14)$$

$$\delta p_2(kT) = (a+b)\delta q(kT) - c\delta q((k-1)T) - b(\delta q(kT) - \delta q((k-1)T)) \quad (6.15)$$
$$+c(\delta q((k-1)T) - \delta q((k-2)T)) + d\delta p_2((k-1)T) - e\delta p_2((k-2)T);$$

Analisando a Equação (6.14), pode-se verificar que o sistema converge para o equilíbrio, ou seja, $\delta p(kT) = \delta p((k-1)T) = 0$, quando não apenas o valor da fila converge para o seu valor de referência, $\delta q(kT) = 0$, bem como quando a derivada do tamanho da fila também é nulo, $\delta q(kT) - \delta q((k-1)T) = 0$. O valor da derivada convergindo também para zero implica que a taxa de chegada corresponde a capacidade do enlace, e assim, não há alteração no tamanho da fila. Conseqüentemente, também não há variação da probabilidade de descarte.

A mesma análise pode ser aplicada na Equação (6.14). Esta equação também considera a derivada segunda para calcular a variação de p no intervalo atual. Neste caso, para que o sistema convirja para o equilíbrio, ou seja, $\delta p(kT) = \delta p((k-1)T) = p((k-2)T) = 0$, tem-se que o valor da fila deve convergir para o seu valor de referência, $\delta q(kT) = 0$, e que a derivada do tamanho da fila seja nulo, $\delta q(kT) - \delta q((k-1)T) = 0$, bem como a derivada segunda também seja nula, $\delta q((k-1)T) - \delta q((k-2)T) = 0$.

São necessários apenas dois algoritmos para implementar o cálculo da probabilidade de descarte/marcação correspondente as equações de diferenças para cada um dos controladores obtidos. O Algoritmo 6.1 corresponde a implementação das Equações (6.10), (6.12) e (6.13). O Algoritmo 6.2 corresponde a implementação da Equação (6.12).

Os dois algoritmos são bastante simples, e são executados a cada intervalo de amostragem $1/f_s$. Inicialmente p_0 é computado baseado nos parâmetros de rede dados (N_0, C_0, R_0). Depois, o valor da probabilidade é calculado baseado nas equações de diferenças. Duas variáveis auxiliares q_{old1} e p_{old1}, armazenam os valores de q e p, respectivamente, no intervalo anterior. O Algoritmo 6.2 usa duas outras variáveis: q_{old2} e p_{old2}, que armazenar, respectivamente, os valores de q_{old1} e p_{old1}, no último intervalo.

Algoritmo 6.1 Algoritmo utilizado para o cálculo da probabilidade para os controladores C_{H2_1}, C_{H2_3} e C_{H2_4}

H2-AQM-ProbabilityFunction()
$p_0 \Leftarrow 2N_0^2/(R_0C_0)^2$;
$p \Leftarrow a(q - 2q_0 + q_{old1}) - b.p_{old1} + p_0(1 + b)$;
$p_{old1} \Leftarrow p$;
$q_{old1} \Leftarrow q$;

Algoritmo 6.2 Algoritmo utilizado para o cálculo da probabilidade para o controlador C_{H2_2}

H2-AQM-ProbabilityFunction()
$p_0 \Leftarrow 2N_0^2/(R_0C_0)^2$;
$p \Leftarrow q_0(c - a - b) + a.q + b.q_{old1} - c.q_{old2} + p_0(1 - d + e) + d.p_{old1} - e.p_{old2}$;
$p_{old2} \Leftarrow p_{old1}$;
$q_{old2} \Leftarrow q_{old1}$;
$p_{old1} \Leftarrow p$;
$q_{old1} \Leftarrow q$;

6.5 Considerações em relação a H2-AQM e as demais políticas AQM

Comparando-se H2-AQM com as demais políticas de AQM, pode-se fazer as seguintes considerações:

- H2-AQM é uma política AQM analítica, desenvolvida utilizando-se técnicas de Teoria de Controle, o que garante a estabilidade em torno do ponto de equilíbrio;
- Foram utilizadas técnicas de controle ótimo ao invés de controladores clássicos;
- Apesar do controlador ter sido derivado a partir do modelo linearizado do sistema de congestionamento, a planta utilizada para derivar o controlador representa a dinâmica do sistema de congestionamento em detalhes, e também considera a existência de fluxos não adaptativos. Desta forma, o controlador obtido garante a estabilidade do sistema independente das condições de rede;
- Diferentemente das outras políticas de AQM baseadas em Teoria de Controle, foi investigado o melhor ponto de equilíbrio para o sistema de controle de congestionamento;
- A abordagem utilizada para derivar o controlador foi não-racional, o que faz com que os recursos de rede possam ser usados eficientemente;
- H2-AQM pode ser considerada uma política AQM baseada tanto no tamanho da fila quanto na taxa, dado que a discussão dos objetivos de projeto para o controlador, que obtenha o melhor desempenho para o sistema de congestionamento, leva em consideração a característica de retroalimentação do sistema de congestionamento e do seu estado representado pelo tamanho da janela de transmissão e pelo tamanho da fila (W, q);
- Apesar de ter sido utilizada uma abordagem não-racional, o controlador obtido é racional, dado que foi possível cancelar o termo de atraso.

6.6 Análise da Robustez do Controlador Ótimo

Qualquer sistema de controle é vulnerável a incertezas, perturbações externas e a ruídos. Desta forma, incertezas relacionadas com a modelagem do sistema, tais como dinâmicas não modeladas na planta, ou relacionadas a distúrbios ou interferências causados por sinais externos podem degradar o desempenho do sistema.

O objetivo das técnicas de Controle Robusto, também conhecido como controle H_∞, é manter a estabilidade do sistema, garantir alto desempenho do sistema e uma suave degradação do seu desempenho na presença de alterações no sistema. No projeto de controladores, utilizando Controle Robusto, são consideradas as incertezas causadas por falhas na modelagem do sistema ou por distúrbios externos, de forma a garantir que os controladores produzam resultados que satisfaçam os requisitos desejados para o sistema, identificando-se uma faixa de valores para as incertezas, na qual o sistema permanece estável e com o desempenho requerido.

Desta forma, nesta seção analisa-se o sistema de controle de congestionamento (5.10) através de técnicas de controle robusto, de modo a garantir sua robustez quando ocorrerem variações nos parâmetros da rede.

Da mesma forma que foi feita a análise utilizando-se técnicas de controle ótimo, a análise da robustez do Sistema (5.10) pode ser feita como uma função dos parâmetros de rede, tais como o número de fluxos TCP, N_0, o tempo total de viagem (RTT), R_0, a capacidade do enlace, C_0. Apenas a carga do sistema, representada pelo número de conexões ativas foi considerada na análise, dado que a variação no RTT é diretamente afetada pela variação na carga do sistema.

De forma a garantir a robustez do Sistema (5.10), foi investigada qual é a faixa de variação para a carga do sistema, representada por $|\Delta N = N - N_0|$, para a qual os requisitos ou objetivos do sistema são alcançados, mantendo-se a estabilidade do sistema e garantindo-se uma suave degradação do seu desempenho na presença de alterações no sistema.

Considerando $\Delta N = N - N_0$, as matrizes A_0, A_1 e B_u do Sistema (6.1) podem ser reescritas como:

$$A_0 = \begin{bmatrix} -\frac{N_0}{R_0^2 C_0} & -\frac{1}{R_0^2 C_0} \\ \frac{N_0}{R_0} & -\frac{1}{R_0} \end{bmatrix} + \begin{bmatrix} -\frac{1}{R_0^2 C_0} \\ \frac{1}{R_0} \end{bmatrix} \Delta N \begin{bmatrix} 1 & 0 \end{bmatrix} = A_{00} + B_0 \Delta N D_0,$$

$$A_1 = \begin{bmatrix} -\frac{N_0}{R_0^2 C} & \frac{1}{R_0^2 C_0} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{R_0^2 C} & 0 \\ 0 & 0 \end{bmatrix} \Delta N \begin{bmatrix} 1 & 0 \end{bmatrix} = A_{10} + B_1 \Delta N D_0,$$

$$B_u = \begin{bmatrix} -\frac{R_0 C_0^2}{2N_0^2} \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{R_0 C_0^2}{2N_0^2} \\ 0 \end{bmatrix} \begin{bmatrix} \frac{N_0^2}{(N_0+\Delta N)^2} - 1 \end{bmatrix} = B_{u0} + B_2 E_N$$

Seja $B_\Delta = \begin{bmatrix} B_0 & B_1 & B_2 \end{bmatrix}$, $C_{00} = \begin{bmatrix} D_0 \\ \begin{bmatrix} 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \end{bmatrix} \end{bmatrix}$, $C_{10} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \end{bmatrix} \\ D_0 \\ \begin{bmatrix} 0 & 0 \end{bmatrix} \end{bmatrix}$, $C_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ e

$U = \begin{bmatrix} \Delta N & 0 & 0 \\ 0 & \Delta N & 0 \\ 0 & 0 & E_N \end{bmatrix}$, o Sistema (5.10) pode ser reescrito como:

$$\begin{aligned} \dot{x}(t) &= A_{00}x(t) + A_{10}x(t-R_0) + B_{u0}u(t-R_0) + B_{\Delta}w(t); \\ z(t) &= C_{00}x(t) + C_{10}x(t-R_0) + C_2u(t); \\ w(t) &= Uz(t); \end{aligned} \tag{6.16}$$

onde:
$| \Delta N | < \gamma$;
$| E_N | < \gamma$;
$\|U\| < \gamma^{-1}$;

Para garantir a estabilidade do Sistema (6.5), é usado o Teorema 4-c apresentado em [143] que especifica que um sistema como (6.16) é assintoticamente estável e $\|H_{wz}(s)\|_{\infty}^2 < \gamma$, se $\mathbf{D} = 0$ e se existem matrizes simétricas e definidas positivas W, Y_0 e X_j, e matrizes F, R, L_j e Q_j, com $j = 0, 1$, tais que os seguintes LMIs tenham uma solução factível:

$$\begin{bmatrix} \mathbf{A}_0 + \mathbf{A}_0^T + X_1 & (\bullet)^T & (\bullet)^T & (\bullet)^T \\ \mathbf{A}_1^T & -X_1 & (\bullet)^T & (\bullet)^T \\ \mathbf{C}_0 & \mathbf{C}_1 & -I & (\bullet)^T \\ \mathbf{B}^T & 0 & \mathbf{D}^T & -\gamma I \end{bmatrix} < 0 \tag{6.17}$$

$$\mathbf{P}_0 > 0 \tag{6.18}$$

onde $\mathbf{A}_0$, $\mathbf{A}_1$, $\mathbf{B}$, $\mathbf{C}_0$, $\mathbf{C}_1$ e $\mathbf{P}_0$ são dadas por:

$$\mathbf{A}_0 = \begin{bmatrix} A_{00}X_0 + B_{u0}L_0 & A_{00} \\ Q_0 & Y_0A_{00} \end{bmatrix},$$

$$\mathbf{A}_1 = \begin{bmatrix} A_{10}X_0 + B_{u0}L_1 & A_{10} \\ Q_1 & Y_0A_{10} + FC_y \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} B_{\Delta} \\ Y_0B_{\Delta} + FD_{yw} \end{bmatrix},$$

$$\mathbf{C}_0 = \left[\begin{array}{cc} C_{00}X_0 + C_2L_0 & C_{00} \end{array} \right],$$

$$\mathbf{C}_1 = \left[\begin{array}{cc} C_{10}X_0 + C_2L_1 & C_{10} \end{array} \right],$$

$$\mathbf{P}_0 = \left[\begin{array}{cc} X_0 & I \\ I & Y_0 \end{array} \right];$$

Este problema de programação convexa foi resolvido numericamente utilizando a rotina LMISol [144]. Os parâmetros de rede foram os mesmos usados para a obtenção do controlador ótimo, $R_0 = 0.256$ segundos, $C_0 = 3750$ pacotes/segundos, e $N_0 = 120$ sessões TCP. Apesar de ter sido encontrada uma solução factível para os LMIs (6.17) e (6.18), o valor encontrado para γ foi extremamente alto, e conseqüentemente a faixa de valores para ΔN é muito pequena, não justificando assim, a derivação do controlador H_∞. Desta forma, o Sistema (6.16), apesar de estável, não pode ter sua robustez garantida frente a variações da carga, o que explicita a suscetibilidade do sistema em relação a variação da carga da rede.

Este resultado já era esperado devido principalmente aos problemas do TCP apresentados na Seção 2.3. Primeiramente devido a "Lei do inverso da raíz quadrada" que mostra que para se obter valores altos para a janela de congestionamento, são requeridos valores extremamente baixos para a probabilidade de descarte/marcação p, uma vez que a vazão é inversamente proporcional a raíz quadrada de p. Além disto, para uma pequena variação na carga da rede, tem-se um aumento significativo no valor da probabilidade de descarte, dado que p aumenta com o quadrado do número de conexões ativas. Outro agravante é a instabilidade própria do TCP Reno que faz com que um sistema de congestionamento que o utilize não convirja para um ponto de equilíbrio, e fique oscilando em torno deste ponto. Isto significa que o TCP fonte oscila consideravelmente a sua taxa de transmissão, não estabilizando-a no valor da taxa de transmissão ideal. Desta forma, quando a carga na rede é pequena tem-se períodos em que recursos são subutilizados e quando a carga na rede se intensifica, tem-se a ocorrência do congestionamento e suas conseqüências [34].

Este é um resultado importante e indica que o aprimoramento do sistema de congestionamento depende de se garantir a estabilidade do tamanho da fila governado pela política AQM, bem como depende também que seja garantida a estabilidade da janela de transmissão governada pelo TCP.

6.7 PI-AQM - Proportional Integrator AQM

A política PI-AQM, (*Proportional Integrator* AQM) [122], usa um controlador do tipo PI, e foi desenvolvida utilizando o modelo do sistema de congestionamento (5.10).

Para simplificar a dinâmica do sistema (5.10), os projetistas de PI focalizaram no comportamento nominal (baixa-freqüência) da dinâmica da janela de forma a determinar o resíduo de alta-freqüência $\Delta_{PI}(s)$:

$$\Delta_{PI}(s) \doteq = \frac{2N_0^2 s}{R_0^2 C^3}(1 - e^{-sR_0}); \tag{6.19}$$

A função do controlador AQM, $C_{PI}(s)$, é marcar os pacotes com probabilidade $p(t)$, como função do tamanho da fila medido $q(t)$ e estabilizar a planta $P_{PI}(s)$ e a dinâmica de alta freqüência da janela $\Delta_{PI}(s)$. A planta $P_{PI}(s)$ é simplificada de forma a isolar as contribuições devido ao atraso no resíduo $\Delta_{PI}(s)$, que é tratado como uma dinâmica não modelada do sistema. Nesta abordagem, para que o sistema seja estável o controlador $C_{PI}(s)$ também tem que estabilizar o resíduo $\Delta_{PI}(s)$ e estabelecer um limite superior para $\Delta_{PI}(s)V_{PI}(s)$, onde $V(s)_{PI}$ é a função de sensibilidade do sistema. Isto significa que devem ser identificados valores de N_0, R_0 e C_0, para os quais o controlador consegue estabilizar o sistema. De modo contrário, a planta utilizada no desenvolvimento dos controladores H2-AQM obtidos reproduz em detalhes a estrutura do Sistema (6.1), assim, a estabilização da planta pelo controlador, implica na estabilidade do sistema de congestionamento independente das condições de rede. A planta simplificada para o sistema, $P_{PI}(s)$ é, então, dada por:

$$P_{PI}(s) = \frac{\frac{C^2}{2N_0}}{\left(s + \frac{2N_0}{R_0^2 C_0}\right)\left(s + \frac{1}{R_0}\right)}; \tag{6.20}$$

A função de transferência, $C_{PI}(s)$, do controlador PI-AQM é dada por:

$$\begin{aligned} C_{PI}(s) &= K_{PI}\frac{\frac{s}{z}+1}{s}; \\ K_{PI} &= \omega_g \frac{2N_0}{R_0^2 C_0}\left|\frac{\left(j\omega_g + \frac{1}{R_0}\right)}{\frac{C_0^2}{2N}}\right|; \\ \omega_g &= \frac{2N_0}{R_0^2 C_0}; \end{aligned} \tag{6.21}$$

Usando-se o mesmo ponto de equilíbrio empregado para derivar o controlador H2-AQM, ($R_0 = 0.256$, $C_0 = 3750$ e $N_0 = 120$), o controlador PI-AQM tem sua função de transferência no domínio da freqüência $C_{PI}(s)$ para este ponto dada por:

$$C_{PI}(s) = \frac{8.138^{-5}s + 7.947^{-5}}{s}; \tag{6.22}$$

Algoritmo 6.3 Algoritmo utilizado para o cálculo da probabilidade para o controlador $PI - AQM$

PI-AQM-ProbabilityFunction()
$p \Leftarrow a(q - q_0) - b(q_{old} - q_0) + p_{old}$;
$p_{old} \Leftarrow p$;
$q_{old} \Leftarrow q$;

Utilizando-se a freqüência de amostragem $f_s = 375Hz$, o controlador PI no domínio-z, é dado por:

$$C_{PI}(z) = \frac{8.149^{-5}z + 8.127^{-5}}{z - 1}; \tag{6.23}$$

Para a implementação do controlador PI, os autores em [122] consideraram que $p_0 = 0$. Desta forma, a função de transferência entre $\delta p = p - p_0 = p$ e $\delta q = q - q_0$ pode ser convertida na seguinte equação de diferença no tempo discreto kT, onde $T = \frac{1}{f_s}$:

$$p(kT) = a\delta q(kT) - b\delta q((k-1)T) + p((k-1)T); \tag{6.24}$$

Os algoritmo Algoritmo 6.3, que implementa o controlador PI é bastante simples [122]. O valor da probabilidade é calculado baseado na Equação 6.24 e duas variáveis auxiliares q_{old} e p_{old}, são utilizadas para armazenar os valores de q e p, respectivamente, no intervalo anterior.

Capítulo 7

Efetividade do Controlador H2-AQM

Uma vez que os controladores correspondentes aos quatro objetivos foram sintetizados e foi observado através de Simulações no Simulink que possuem as características de desempenho esperadas para o sistema linear, devem ser realizadas simulações cuidadosas e precisas do sistema com todas as suas não-linearidades com o intuito de validar as características de desempenho para o sistema não linear. Desta forma, os Algoritmos 6.1 e 6.2 foram implementados no NS [65], de forma a permitir a validação das características de desempenho do sistema de em um ambiente dinâmico de rede.

Neste capítulo, os algoritmos correspondentes a implementação dos controladores obtidos na Seção 6.4 são comparados entre si em um ambiente dinâmico de rede com o intuito de verificar qual deles apresenta o melhor desempenho. O desempenho do algoritmo do controlador correspondente ao objetivo que apresentou o melhor resultado, é, então, comparado ao desempenho das políticas RED e PI-AQM.

7.1 Cenário de Simulação

A topologia, a capacidade dos enlaces, como também o tempo de propagação para cada enlace são apresentados na Figura 7.1. O enlace entre os nós R_1 e R_2 é o enlace gargalo. O tamanho do *buffer* utilizado por estes dois nós é de 800 segmentos.

Como foi apresentado no Capítulo 2, o TCP varia sua taxa de transmissão de acordo com o nível de congestionamento. Desta forma, com o intuito de se verificar a robustez das políticas de AQM sob diferentes condições de rede, a carga da rede foi variada, e foi empregado um gerador de tráfego, denominado TrafficGen [146], para gerar cargas específicas para tráfegos distintos. Apesar do modelo utilizado para derivar os controladores ótimos considerar basicamente tráfego de longa-duração (FTP), também foi gerado tráfego de curta duração (Web), de modo a verificar a eficácia dos controladores sob este tipo de tráfego. Além disto, como os roteadores têm que lidar tanto com fluxos TCP quanto com

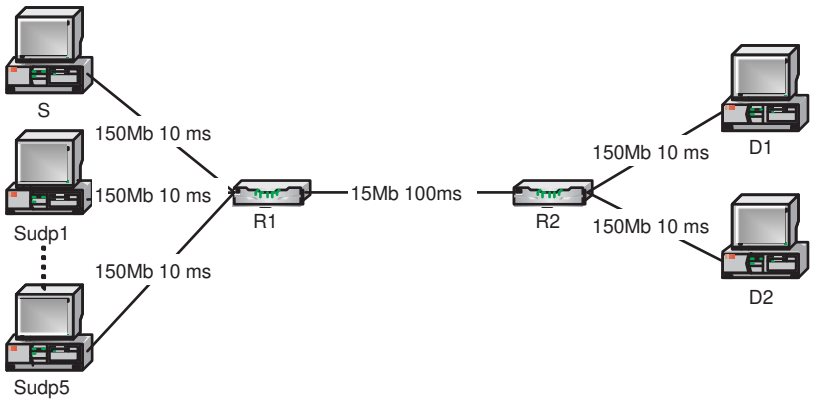

Figura 7.1: Topologia utilizada nos experimentos

fluxos não-adaptativos, foi inserido também tráfego de ruído.

As simulações foram geradas tendo um tráfego TCP de longa ou de curta duração como tráfego principal. Foram incluídos fluxos não-adaptativos do tipo CBR/UDP que podem representar até 20% da capacidade do enlace, de forma a gerar o tráfego de ruído. Estes fluxos são gerados e finalizados em diferentes intervalos a partir dos nós S_{udp_i} com destino ao nó D_2.

O tráfego principal TCP foi gerado do nó S_1 para o nó D_1. A carga foi variada de 0.4 à 0.9 de modo a obter cenários com congestionamento incipiente ou intenso. Uma distribuição híbrida Lognormal/Pareto foi utilizada para gerar o tráfego WEB. O corpo da distribuição, correspondente a uma área de 0.88, é modelado por uma distribuição Lognormal com média de 7247 *bytes*, enquanto que a cauda é modelada por uma distribuição Pareto com média de 10558 *bytes* [147]. O tráfego FTP é gerado, utilizando-se uma distribuição exponencial com média de 512 *KBytes*.

A variação TCP empregada foi o TCP Reno. O tamanho dos segmentos gerados foi de 500 *bytes*, devido ao fato de que muitas implementações TCP que não implementam o algoritmo *Path MTU Discovery* utilizam 512 ou 536 *bytes* como valores *default* para o seu tamanho máximo de segmento (MSS), para transmissões com destino IP fora da rede local. Além disto, diferentes artigos contendo levantamentos estatísticos com amostras da ordem de milhões de pacotes apontam que este é o valor médio do tamanho dos pacotes trafegando na Internet. O site [148] contem este artigos.

O valor da janela de recepção do TCP receptor utilizado foi bastante alto (1000), de forma a fazer com que o crescimento da janela de transmissão do TCP emissor fosse governado apenas pela rede e não pelo TCP receptor, ou seja, o controle de fluxo foi desativado.

Controlador	a	b	c	d	e
$C_{H2_1}(z)$	3.2424^{-5}	0.9921			
$C_{H2_2}(z)$	1.427^{-5}	$9.24e^{-6}$	$-5.029e^{-6}$	-0.7792	0.3787
$C_{H2_3}(z)$	1.802^{-5}	0.09654			
$C_{H2_4}(z)$	3.2661^{-5}	0.9923			

Tabela 7.1: Parâmetros utilizados na implementação dos controladores ótimos

7.2 Comparação entre os Controladores Ótimos Obtidos

Com o intuito de verificar o desempenho e a eficácia dos controladores obtidos na Seção 6.4, os algoritmos 6.1 e 6.2 foram implementados no simulador de redes NS [65]. O ponto de equilíbrio introduzido na Seção 6.3.1 foi utilizado. A freqüência de amostragem utilizada para derivar os controladores digitais foi de 375 Hz, gerando respectivamente os valores para os parâmetros de cada um dos quatro controladores apresentados na Tabela 7.1. O cenário utilizado para a simulação foi o apresentado na Seção 7.1.

Todos os controladores obtidos no Capítulo 6 atingem o ponto de equilíbrio diferindo apenas pelo caminho percorrido por cada um deles para atingir o ponto de equilíbrio. Desta forma, todos os controladores derivados apresentam um desempenho equivalente para todas as métricas quando a média é utilizada para compará-los. Os caminhos distintos para atingir o ponto de equilíbrio, derivados dos diferentes objetivos de projeto influenciam na variação do desempenho em torno do seu valor médio. Desta forma, uma medida estatística adequada para comparar o desempenho dos diferentes controladores é o coeficiente de variação, que é a razão entre o desvio padrão e a média ($CV = s/\bar{X}$).

Para cada tipo de tráfego FTP e WEB, os valores de CV obtidos para as métricas relacionadas com o ponto de equilíbrio, tais como o tamanho da janela por conexão ($Cwnd$), a probabilidade de descarte/marcação e o tamanho da fila em função da carga da rede são apresentados. O controlador com o melhor resultado é o que apresenta os valores mais baixos de CV para estas métricas, o que implica em uma pequena variação em torno do ponto de equilíbrio e por conseguinte, garante maior estabilidade para o sistema.

7.2.1 Experimento com Tráfego de Longa Duração - FTP

Nesta seção, são apresentados os resultados dos valores de coeficiente de variação, CV, para o tamanho da janela por conexão ($Cwnd$), a probabilidade de descarte/marcação e o tamanho da fila em função da carga, para a simulação realizada com o tráfego de longa-duração (FTP).

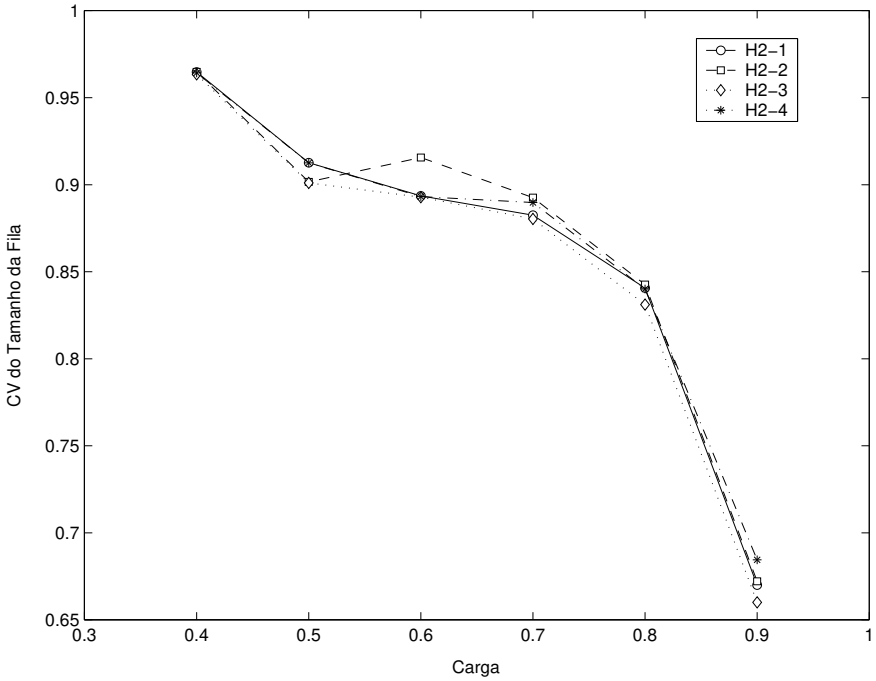

Figura 7.2: Tráfego FTP: coeficiente de variação do tamanho da fila

Na Figura 7.2 podem ser vistos os valores de CV para o tamanho da fila para os diferentes controladores. Pode-se verificar que os valores de CV diminuem a medida que a carga aumenta, ou seja, quanto maior a carga, mais os controladores vão conseguindo manter o valor do tamanho da fila mais próximo do ponto de equilíbrio. O resultado obtido para os controladores diferem muito pouco, dado que com exceção apenas do controlador C_{H2_2}, todos os demais controladores têm como objetivo a minimização do *jitter*. Para cargas acima de 0.7, todos os controladores apresentam resultados equivalentes. Como o ponto de equilíbrio foi escolhido para cargas grandes, todos os controladores a partir desta carga, 0.7, apresentam desempenho similar. O controlador que apresentou o melhor resultado, ou seja, menores valores de CV para o tamanho da fila foi o C_{H2_3}.

A Figura 7.3 apresenta os valores de CV para a probabilidade de descarte/marcação produzida por cada controlador em função da carga. Pode-se verificar basicamente o mesmo comportamento para o gráfico da variação do tamanho da fila, entretanto, a diferença entre os valores de CV obtidos por cada controlador é quase inexistente. Mais uma vez, o controlador C_{H2_3} apresentou o melhor resultado, ou seja, menores valores de CV para a probabilidade de descarte/marcação. Valores baixos de CV implica em uma pequena oscilação para a componente de probabilidade de descarte/marcação para o ponto de equilíbrio definido, e conseqüentemente em uma menor variação da taxa instantânea de perda.

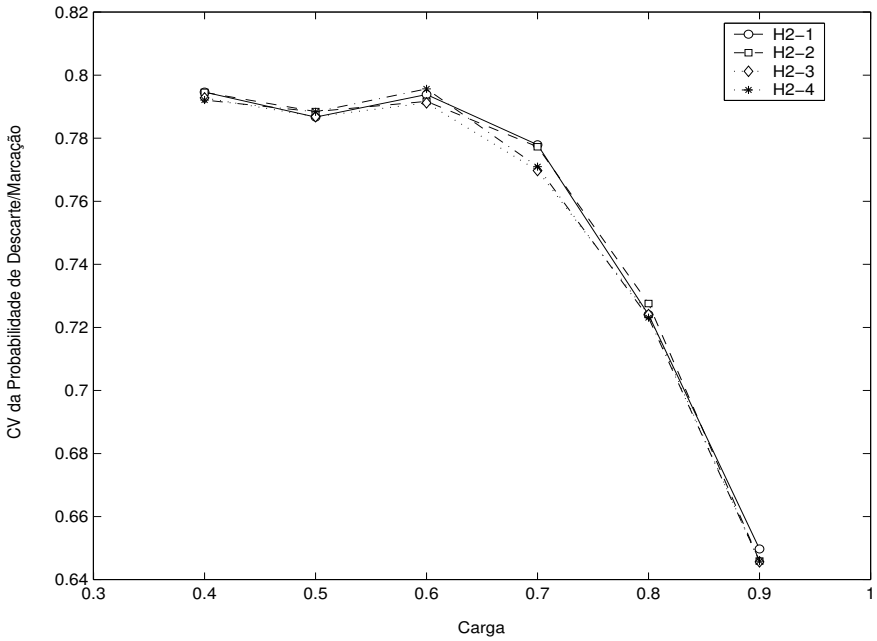

Figura 7.3: Tráfego FTP: coeficiente de variação da probabilidade de descarte/marcação

A Figura 7.4 apresenta os valores de CV para a janela de congestionamento ($Cwnd$) por conexão ativa. Pode-se verificar que da mesma forma como aconteceu para os valores de CV para as outras componentes do ponto de equilíbrio, os valores de CV para a janela de congestionamento dos controladores diferiu muito pouco. Outra observação que pode ser feita é que os valores de CV aumentam a medida que se aumenta a carga na rede, ou seja, a medida que aumenta o número de conexões ativas. Como foi apresentado na Seção 2.3.4, a vazão TCP, definida em função da sua janela de transmissão, é inversamente proporcional a raíz quadrada da probabilidade de descarte/marcação p, que por sua vez aumenta com o quadrado do número de conexões ativas. Desta forma, um pequeno aumento no número de conexões ativas faz com que a janela tenha uma grande variação. Quando o número de conexões aumenta, conseqüentemente, também aumenta a variação a janela. O controlador C_{H2_3} apresenta a menor variação para a janela de congestionamento. Este resultado já era esperado dado que um dos objetivos deste controlador é garantir uma taxa de transmissão estável, através da minimização da variação em torno do valor do ponto de equilíbrio da janela de transmissão.

De forma geral, o controlador que produz os melhores resultados para tráfego FTP é C_{H2_3}. Este controlador minimiza a variação do tamanho da janela e garante uma taxa de transmissão mais constante. Além disto, produz o menor valor de CV para a probabilidade de descarte/marcação, o que significa uma menor variação na taxa instantânea de perda.

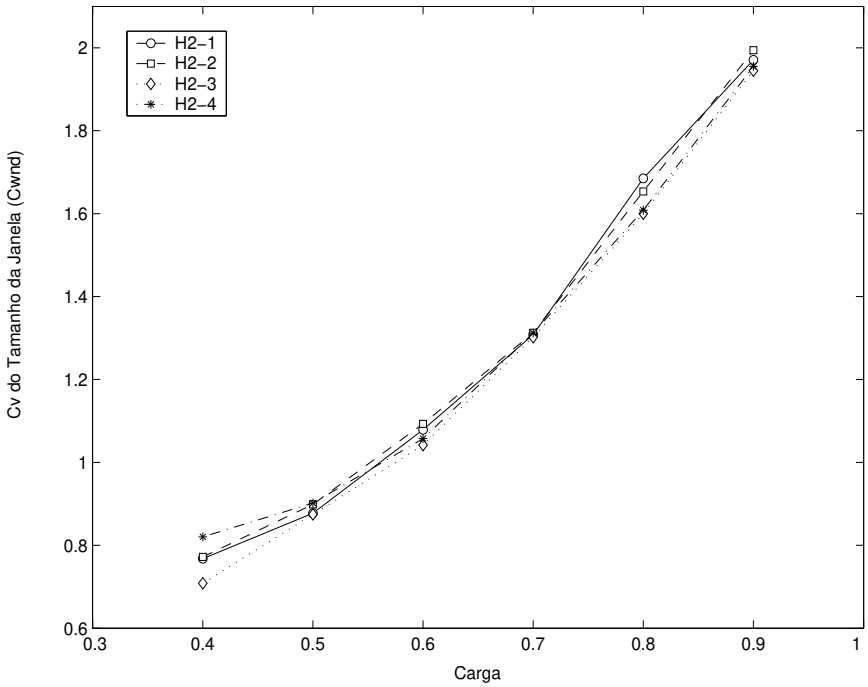

Figura 7.4: Tráfego FTP: coeficiente de variação da janela de congestionamento (cwnd) por conexão ativa

Como o tráfego FTP é de longa duração, ele é favorecido pela taxa mais estável de transmissão e pela pequena variação na taxa de perda. O controlador C_{H2_3} também apresenta o melhor valor de CV para o tamanho da fila, e conseqüentemente apresenta os menores valores de *jitter*. Outrossim, a minimização de *jitter* implica na estabilidade do tamanho da fila.

7.2.2 Experimento com Tráfego de Curta Duração - Web

Nesta seção, são apresentados os resultados dos valores de coeficiente de variação, para o tamanho da janela por conexão ($Cwnd$), a probabilidade de descarte/marcação e o tamanho da fila em função da carga, para a simulação realizada com o tráfego principal sendo de curta-duração (Web).

O sistema de equações do modelo utilizado para derivar os controladores descreve o comportamento AIMD do TCP Reno na sua fase de estabilização, ou seja, quando o crescimento da sua janela é governado basicamente pelo algoritmo de *Congestion Avoidance*. Desta forma, o modelo é consideravelmente preciso para tráfego de longa duração, onde a fase de *Slow Start* é bastante curta, podendo assim ser desconsiderada. No entanto, o modelo não é tão preciso para tráfego de curta duração, visto que neste tipo de tráfego

a fase de *Slow Start* tem um papel determinante no comportamento da janela do TCP. Desta forma, os controladores desenvolvidos a partir deste modelo podem não ser tão eficazes na presença de tráfego de curta-duração. Apesar disto, também foi investigado a eficácia dos controladores sob tráfego Web.

A Figura 7.5 mostra os valores de CV para o tamanho da fila. O comportamento dos valores de CV para o tamanho da fila sob Tráfego Web é o mesmo que para tráfego FTP. No entanto, os valores de CV produzidos pelos quatro controladores sob tráfego Web apresentam basicamente dois comportamentos distintos: C_{H2_1} apresenta um comportamento similar ao do C_{H2_2} e, o C_{H2_3} do C_{H2_4}. Os controladores que apresentaram o melhor resultado, ou seja, menores valores de CV para o tamanho da fila foram o C_{H2_3} e o C_{H2_4}. Os controladores C_{H2_1} e C_{H2_2} apresentaram maiores valores de CV, devido a um de seus objetivos que é evitar a subutilização do enlace, o que faz com que o tamanho da fila seja empurrado para o valor do ponto de equilíbrio o mais rápido possível, podendo levar a grandes variações quando o tamanho da fila se afasta do ponto de equilíbrio. Para o tráfego FTP, este objetivo não gerou grandes diferenças para os valores de CV obtidos para cada controlador. Entretanto, para o tráfego Web, tal objetivo fez com que os controladores tivessem um desempenho diferenciado em relação aos valores de CV obtidos. A justificativa para este fato é que neste tipo de tráfego, a fase de *Slow Start* não é desprezível em relação ao tempo de vida da conexão, fazendo com que a taxa de transmissão seja dobrada a cada RTT, e assim, a taxa de chegada de pacotes oscila de forma considerável, e por conseguinte, o tamanho da fila. Ao empurrar rapidamente o tamanho da fila de volta para o valor do ponto de equilíbrio, os controladores C_{H2_1} e C_{H2_2} irão gerar uma maior oscilação no tamanho da fila que os controladores C_{H2_3} e C_{H2_4}.

A Figura 7.6 apresenta o valor de CV da probabilidade de descarte/marcação para os diferentes controladores sob tráfego Web. Pode-se verificar também para tráfego Web, que o gráfico do CV da probabilidade de descarte/marcação apresenta o mesmo comportamento para o gráfico do CV do tamanho da fila. Os controladores C_{H2_3} e o C_{H2_4}, mais uma vez, apresentam o melhor resultado e quase se equivalem. Pode-se verificar também uma diferença significativa entre os valores apresentados pelos controladores C_{H2_1} e C_{H2_2} em relação aos valores de CV apresentados pelos controladores C_{H2_3} e o C_{H2_4}.

Pode-se verificar, na Figura 7.7, que o comportamento dos valores de CV para a janela de congestionamento ($Cwnd$) por conexão ativa sob tráfego Web é diferente do apresentado para tráfego FTP. O pequeno intervalo que contém os valores de CV, de 1.82 para 1.88, mostra que sob tráfego Web basicamente não há diferenças entre os valores de CV para os distintos controladores. Apesar disto, fica claro na figura que os controladores C_{H2_2} e C_{H2_4}, que possuem como um de seus objetivos atingir a vazão ideal mais rapidamente produzem os melhores resultados, sendo o controlador C_{H2_4} ainda mais eficiente.

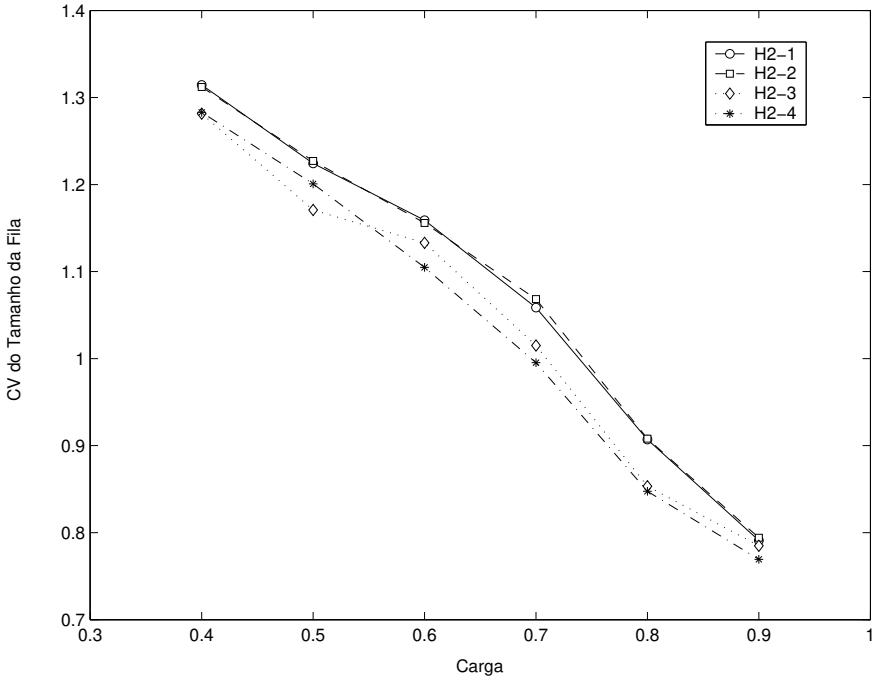

Figura 7.5: Tráfego Web: coeficiente de variação do tamanho da fila

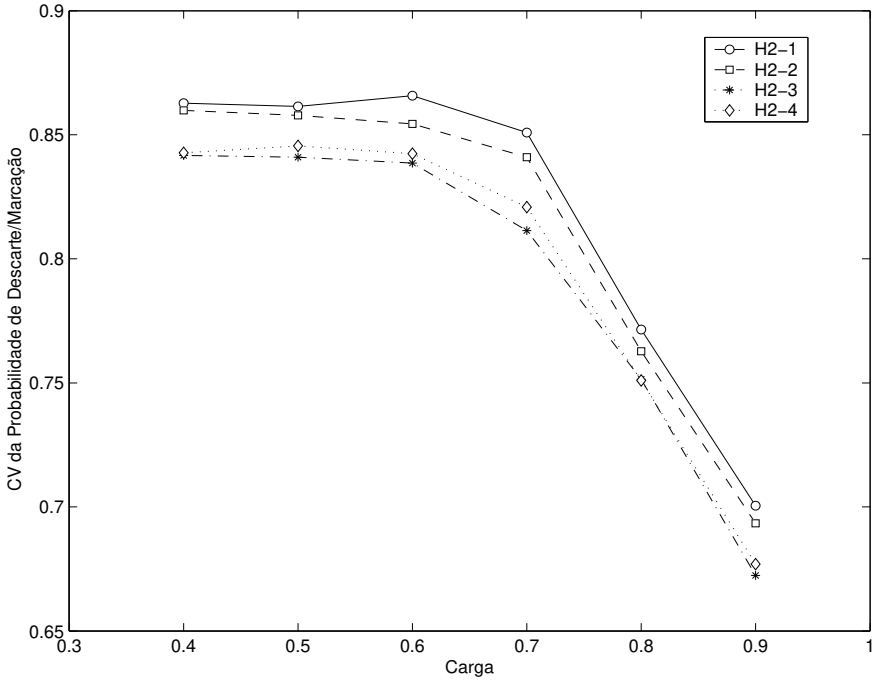

Figura 7.6: Tráfego Web: coeficiente de variação da probabilidade de descarte/marcação

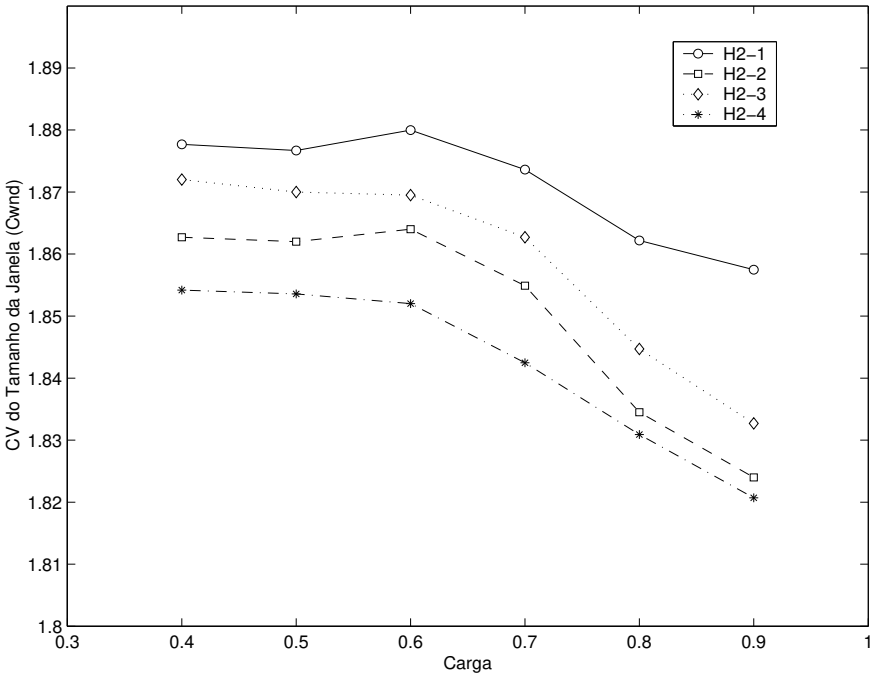

Figura 7.7: Tráfego Web: coeficiente de variação da janela de congestionamento (cwnd) por conexão ativa

De uma forma geral, os controladores C_{H2_3} e C_{H2_4} foram os controladores que apresentaram o melhor desempenho para o tráfego Web. O C_{H2_4} supera o controlador C_{H2_3} apenas em minimizar a variação da janela. Entretanto, como para esta componente do ponto de equilíbrio, a diferença foi pequena para os valores de CV obtidos por todos os controladores, pode-se considerar que o controlador C_{H2_3} obteve um desempenho equivalente ao controlador C_{H2_4}.

Considerado o desempenho de uma forma geral dos controladores tanto sob tráfego FTP como Web, o controlador C_{H2_3} é o controlador escolhido como o que apresentou o melhor desempenho. Este controlador minimiza a variação do tamanho da janela e garante uma taxa de transmissão mais constante. Além disto, produz o menor valor de CV para a probabilidade de descarte/marcação, o que significa uma menor variação na taxa instantânea de perda. O controlador C_{H2_3} também apresenta o melhor valor de CV para o tamanho da fila, e conseqüentemente apresenta os menores valores de *jitter*. Este resultado é coerente com os objetivos deste controlador que é garantir banda passante mais constante e minimizar a ocorrência de *jitter*.

O controlador C_{H2_3}, referenciando daqui em diante simplesmente como H2-AQM, tem sua eficácia avaliada na próxima seção através de uma comparação com as políticas RED e PI-AQM.

7.3 Avaliação da Eficácia do Controlador H2-AQM

Nesta subseção, a eficácia da política H2-AQM é avaliada através de simulações no NS, comparando-se o seu desempenho com a políticas de AQM padrão RED e com a política PI-AQM, que serviu como base para o desenvolvimento deste trabalho.

De modo a obter uma comparação justa, o controlador PI utilizado nas comparações foi derivado usando-se o mesmo ponto de equilíbrio e a mesma freqüência de amostragem utilizados para derivar o controlador H2-AQM. Os valores obtidos para os coeficientes de PI foram, respectivamente, $8.149e^{-5}$ e $8.127e^{-}5$. RED foi simulado usando-se os valores *default* do NS, que são: $min_{th} = 5$, $max_{th} = 15$ e $max_p = 0.1$.

Os experimentos foram conduzidos tanto para tráfego de longa-duração, FTP, como para tráfego de curta-duração, Web. O cenário utilizado foi o mesmo apresentado na Seção 7.1. Para cada tipo de tráfego várias métricas foram investigadas e seus valores médios obtidos em função da carga na rede são apresentados na forma de gráficos, onde também são visíveis os intervalos de confiança com 95% de nível de confiança, que foram gerados pelo método de replicação independente.

Para cada um dos experimentos foram coletadas e analisadas métricas relacionadas ao enlace e métricas relacionadas com as conexões ativas. São elas:

- **Métricas Relacionadas ao enlace**: as duas métricas apresentadas a seguir, foram coletadas com o intuito de verificar como a política de AQM afeta o desempenho da rede analisando-se o tráfego agregado que chega ao enlace gargalo. Para tanto, são apresentadas informações relativas ao tamanho médio da fila e da taxa de perda média:
  - **Tamanho médio da fila**: o tamanho da fila é uma das métricas que dá indicação de como a política de AQM atua de forma a prevenir e controlar o congestionamento. Além disto, esta métrica é importante, dado que o RTT e o *jitter* dependem dela. O valor do tamanho da fila é dado em pacotes, cujo tamanho é de 500 *bytes*;
  - **Taxa de perda média**: esta métrica é a razão entre a quantidades de *bytes* descartados e a quantidade total de *bytes* enviados. O valor da taxa de perda é um valor no intervalo $[0, 1]$;
- **Métricas Relacionadas às conexões ativas**: estas métricas são apresentadas com o propósito de verificar como as políticas de AQM afetam o desempenho das conexões TCP:
  - **Vazão média obtida**: a vazão obtida é dada pela razão entre a quantidade

total de *bytes* enviados e a capacidade do enlace, e é apresentada em porcentagem;

- **Tamanho médio da janela** ($Cwnd$): durante o período da simulação, quando uma conexão é finalizada, o valor do tamanho da sua janela é armazenado e no final é calculada a média destes valores. A vazão TCP se dá basicamente em função do tamanho da sua janela. Desta forma, quanto maior for o tamanho da fila, maior é a vazão obtida pela conexão. O valor do tamanho da janela é dado em pacotes, cujo tamanho é de 500 *bytes*;
- **Goodput médio**: a utilização apenas da métrica de vazão para indicar quão eficiente é o uso da banda passante pela conexão não é a mais adequada. O ideal é que o uso da banda passante pela conexão não leve a desperdícios, ou seja, que os dados enviados não sejam descartados, de forma a prevenir retransmissões. Desta forma, para cada uma das conexões ativas foi calculado o valor de *goodput* ou vazão útil, que é dado por: $(nbytes - nretbytes)/nbytes$, onde *nbytes* indica o numero total de *bytes* enviados, e *nretbytes* é a quantidade de *bytes* que foram retransmitidos. O valor de *goodput* é um valor no intervalo $[0, 1]$;
- **RTT médio**: esta métrica verifica o atraso sofrido pelas conexões em decorrência da variação no tamanho da fila gerado pelo uso de uma determinada política de AQM. Como o atraso de propagação é constante e que o cenário de simulação considera um único enlace gargalo, tem-se que a variação no tamanho da fila ocorre apenas em função da variação do tamanho da fila;
- **RTO médio**: a perda de múltiplos segmentos pode ter um efeito extremamente danoso na vazão TCP e na latência na transferência dos arquivos. O TCP Reno só consegue recuperar-se eficientemente da primeira perda, enquanto que as demais, só serão tratadas depois da ocorrência de um *timeout* como apresentado na Seção 2.3.1. Quando o congestionamento é intenso, rajadas de perdas de pacotes podem acontecer, aumentando a probabilidade de ocorrência de *timeouts* e, conseqüentemente, degradando o desempenho do TCP. As políticas de AQM são diretamente responsáveis pelo descarte de pacotes. Desta forma, a política de AQM deve apresentar uma taxa de descarte de pacotes proporcional, evitando, assim, a continuidade de perdas de pacotes bem como reduzindo a ocorrência de RTO´s;
- **Tempo médio de transferência**: como apresentado anteriormente, as políticas de AQM são diretamente responsáveis pelo descarte de pacotes. Por ser um protocolo adaptativo, o TCP reduz sua taxa de transmissão na presença de congestionamento, indicado pelo descarte de pacotes. Desta forma, as perdas

sofridas por uma conexão TCP irão impactar a sua vazão, e conseqüentemente a latência na transferência do arquivo;

– **Número médio de conexões ativas**: avalia a eficácia da política de AQM em fazer o melhor uso dos recursos da rede de forma a permitir um maior número de conexões faça uso dos recursos para um mesmo período de simulação.

7.3.1 Experimentos com Tráfego de Longa Duração - FTP

Esta seção apresenta os resultados obtidos nos experimentos realizados com tráfego FTP. Primeiramente, são apresentados e analisados os resultados referentes às métricas relativas ao enlace, e posteriormente, as métricas obtidas por conexão.

As Figuras 7.8 e 7.9, apresentam, respectivamente, o tamanho médio da fila e a taxa de perda em função da carga.

Pode-se verificar na Figura 7.8 que o tamanho médio da fila resultante do uso das políticas H2-AQM e PI-AQM aumenta a medida que cresce a carga na rede, ou seja, o congestionamento se intensifica. De modo oposto, RED apresenta uma pequena variação na média do tamanho da fila mesmo quando a carga aumenta, mantendo basicamente o tamanho médio da fila constante. Como conseqüência, tem-se que RED subutiliza os recursos da rede (*buffer* e banda passante). A manutenção de um tamanho médio de fila constante é ortogonal com os objetivos de H2-AQM e PI-AQM, que é maximizar a taxa de utilização, utilizar os recursos eficientemente e minimizar o descarte desnecessário de pacotes. H2-AQM é a política que apresenta os maiores valores para o tamanho médio da fila, chegando a cerca de 350. Entretanto, este valor ainda é baixo, considerando que o tamanho máximo da fila é de 800 pacotes.

Considerando-se a métrica de taxa de perda, pode-se verificar que para todas as políticas, ela cresce a medida que a carga na rede aumenta (Figura 7.9). H2-AQM e PI-AQM produzem taxas de perdas bem menores que as produzidas por RED. Para congestionamento incipiente, cargas inferiores a 0.7, H2-AQM supera PI-AQM, evitando o descarte desnecessário de pacotes, e as suas conseqüências, e, assim, utilizando mais eficientemente os recursos da rede. Quando o congestionamento intensifica, os valores apresentados por H2-AQM e PI-AQM são basicamente equivalentes. Comparando-se com RED e PI-AQM, a taxa de perda média obtida por H2-AQM chega a ser até 91% e 76% menor para carga de 0.4.

As Figuras 7.10 à 7.16 apresentam os resultados das métricas obtidas por conexões ativas. Um dos objetivos da política H2-AQM é garantir uma taxa máxima de transmissão. Para tanto, deve-se maximizar o tamanho da janela. A Figura 7.10 apresenta a porcentagem média de banda passante obtida, ou a vazão obtida, no enlace gargalo por conexão ativa. Pode ser verificado que H2-AQM obtém vazão por conexão consideravel-

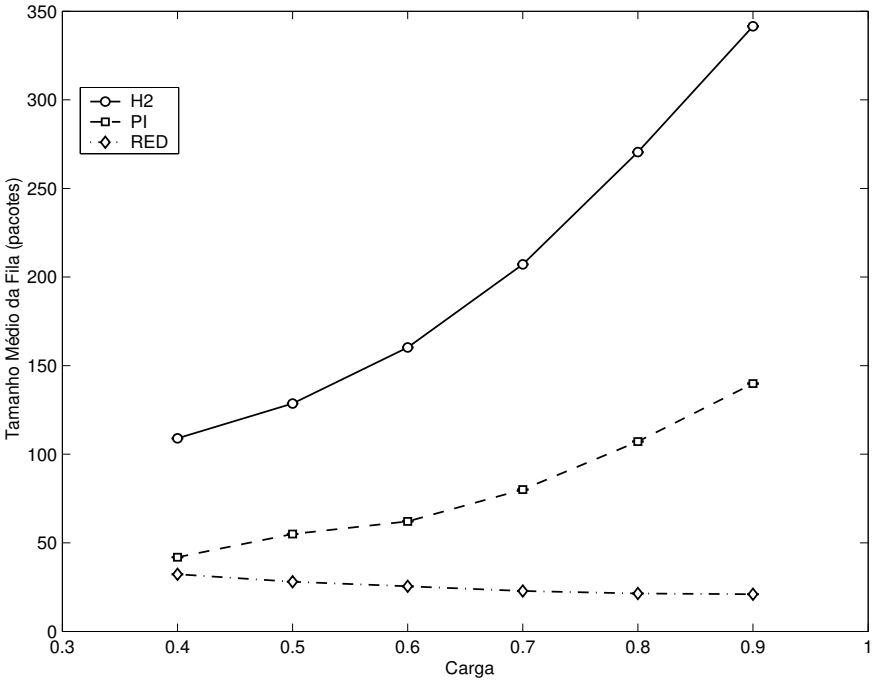

Figura 7.8: Tráfego FTP: valor médio para o tamanho da fila

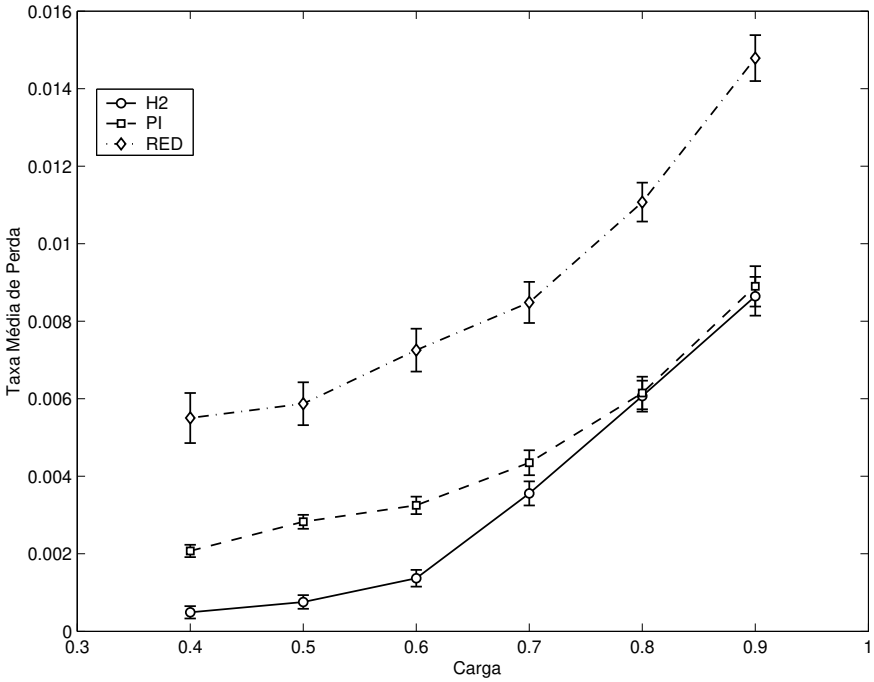

Figura 7.9: Tráfego FTP: valor médio para a taxa de perda

mente maior as outras políticas de AQM são utilizadas. A diferença se acentua quando o congestionamento é incipiente. Este resultado é uma conseqüência direta dos altos valores obtidos para o tamanho da janela, como pode ser observado na Figura 7.11.

Depois da ocorrência de um *timeout* a janela de transmissão é reduzida drasticamente para um segmento e o TCP emissor entra na fase de *Slow Start*. O retorno para o tamanho original só irá ocorrer após um considerável intervalo de tempo. Quando o congestionamento é intenso, rajadas de perdas de pacotes podem acontecer, aumentando a ocorrência de *timeouts* e conseqüentemente degradando o desempenho do TCP, já que a vazão é reduzida drasticamente, além de se aumentar a latência na transferência dos arquivos. Quando a política H2-AQM é utilizada, tem-se as menores taxas de perda. Além disto, como o descarte de pacotes realizado é proporcional, minimizando-se, assim, a perda de pacotes em rajadas, e conseqüentemente, reduz-se a ocorrência de RTO´s. Comparando H2-AQM com RED e PI-AQM, a redução na ocorrência de RTO´s é de no mínimo 33% e 12% para cargas de 0.8 e de no máximo 73% e 59% para cargas de 0.4 (Figura 7.13).

Uma conseqüência direta da redução na ocorrência de RTO´s é o aumento da vazão e a redução no tempo de vida da conexão, ou seja, a latência na transferência dos arquivos é reduzida. Como pode ser observado na Figura 7.14, H2-AQM conseguiu reduzir de forma significativa o tempo de vida das conexões. Comparando-se com RED e PI a redução foi de no mínimo 39% e 15% para cargas de 0.9 e de no máximo 72% e 51% para cargas de 0.4.

A Figura 7.12 apresenta o valor médio de *goodput* obtido por conexão ativa. Esta métrica fornece uma indicação mais precisa da eficácia da utilização do enlace por uma conexão. Se a conexão possui um tamanho de janela grande, mas experimenta uma alta taxa de perda, apenas uma fração da banda obtida é utilizada para enviar novos dados, dado que o restante será utilizado para retransmitir os dados que foram descartados. H2-AQM faz com que se obtém os maiores valores de goodput. Pode-se verificar na figura que a diferença entre os valores de *goodput* obtidos quando H2-AQM é utilizadas são consideravelmente maiores. Isto é uma conseqüência dos maiores valores obtidos para o tamanho médio da janela, *Cwnd*, bem como das menores taxas de perda e menores números de ocorrências de RTO´s. Uma outra observação é que a medida que o congestionamento intensifica, os valores obtidos de *goodput* diminuem. Apesar disto, H2-AQM ainda produz os maiores valores de *goodput*

Por obter maiores valores de *goodput*, diminuir a ocorrência de RTO´s e conseqüentemente reduzir o tempo de vida das conexões, H2-AQM desperdiça menos recursos, otimizando, assim, a utilização dos recursos. Desta forma, o número de conexões ativas que pode utilizar os mesmos recursos durante um certo período de tempo é maior quando H2-AQM é utilizada como política de AQM do que quando as demais políticas são uti-

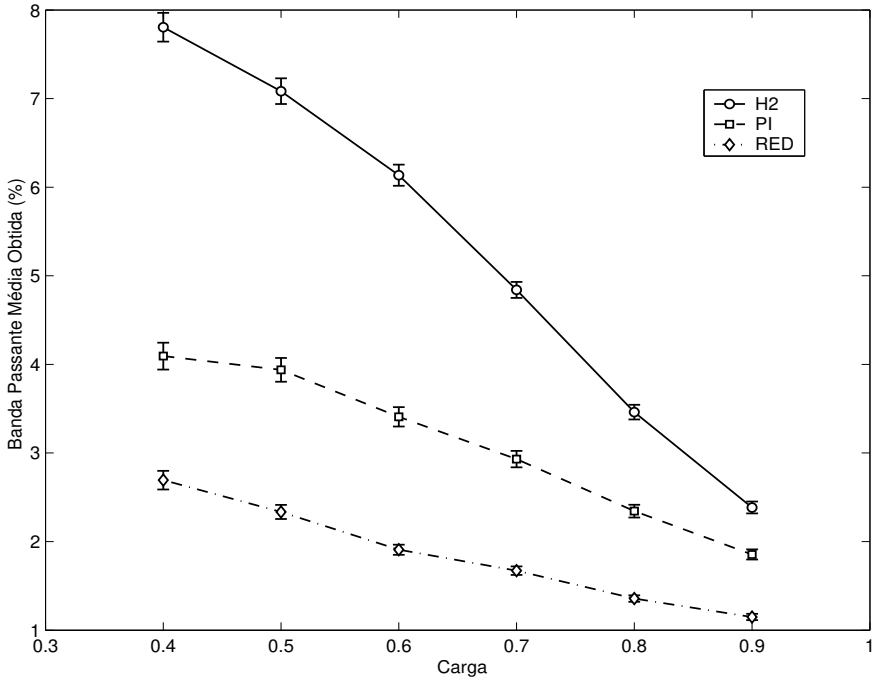

Figura 7.10: Tráfego FTP: vazão média por conexão ativa como função da carga

lizadas (Figura 7.15). Este aumento quando se compara com RED e PI-AQM, é de no mínimo 6% e 3% para cargas de 0.9 e de no máximo 13% e 5% para cargas de 0.7.

Para todas as métricas investigadas, com exceção apenas do RTT, H2-AQM supera PI-AQM e RED. Apesar do tamanho médio da fila produzido por H2-AQM ser consideravelmente maior que os produzidos por RED e PI-AQM, os valores médios para o RTT obtidos por H2-AQM são de no máximo 28% e 16% maiores para cargas de 0.9, do que os produzidos por RED e PI-AQM. Como no cenário da simulação, tem-se apenas um único enlace gargalo, o RTT será dado basicamente só em função do tempo de propagação. A porção referente ao tempo de propagação é igual a $0,22$ segundos. Mesmo que o valor máximo do tamanho da fila seja atingido ($q = 800$ pacotes), o valor do RTT no máximo dobraria. Como o tamanho médio máximo da fila apresentado pela política H2-AQM foi em torno de 350 pacotes, tem-se que a componente do RTT devido ao atraso de fila seria de $350/3750 = 0,09$ segundos, o que dá um RTT em torno de $0,31$ segundos. RED mantém sua fila em torno de 50 pacotes, o que dá um componente de atraso para RED de $0,013$ segundos, dando um RTT em torno de $0,233$ segundos.

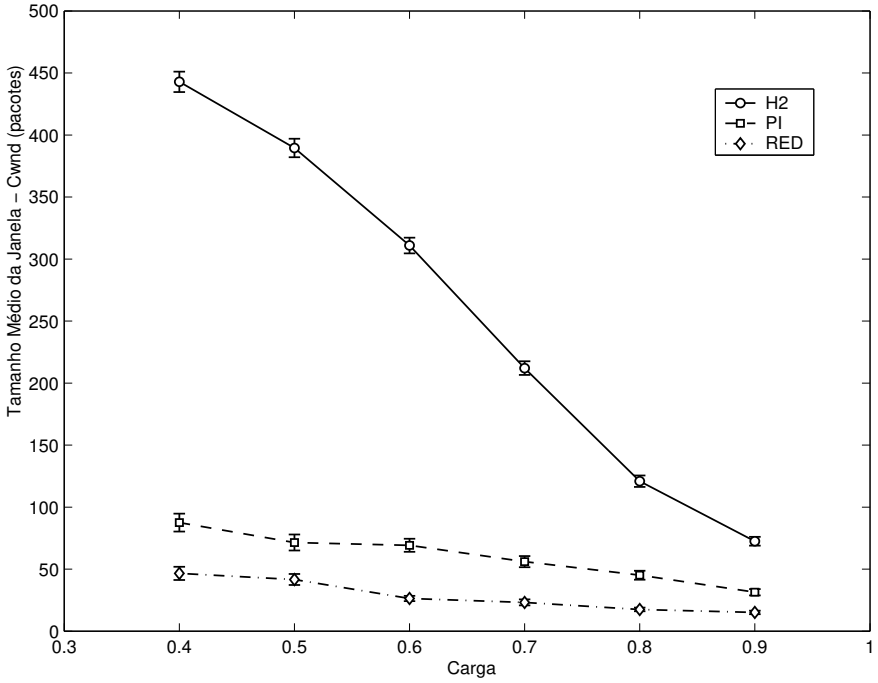

Figura 7.11: Tráfego FTP: valor médio da janela de congestionamento (Cwnd) obtido por conexão ativa como função da carga

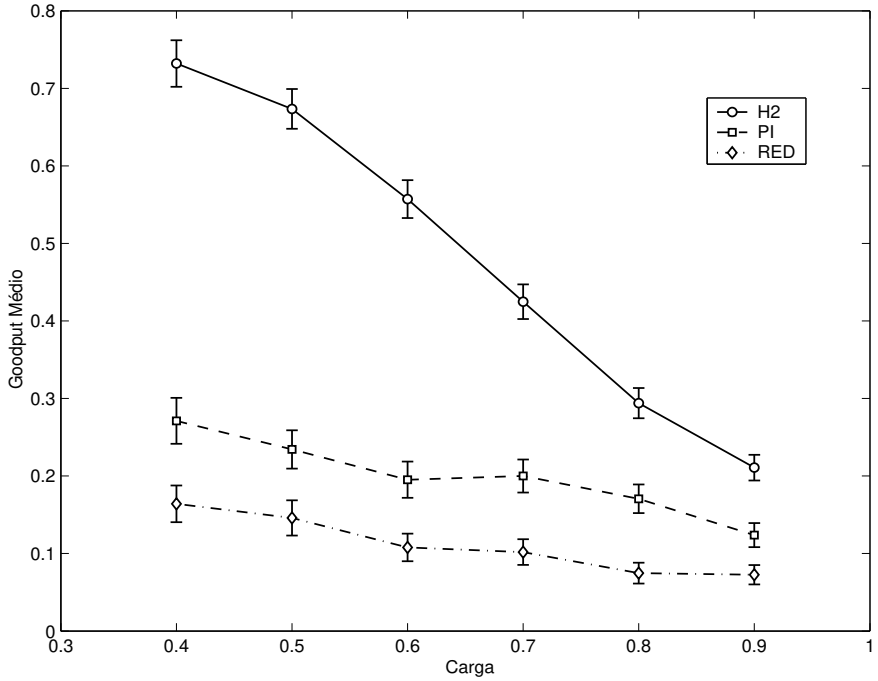

Figura 7.12: Tráfego FTP: valor médio de goodput por conexão ativa como função da carga

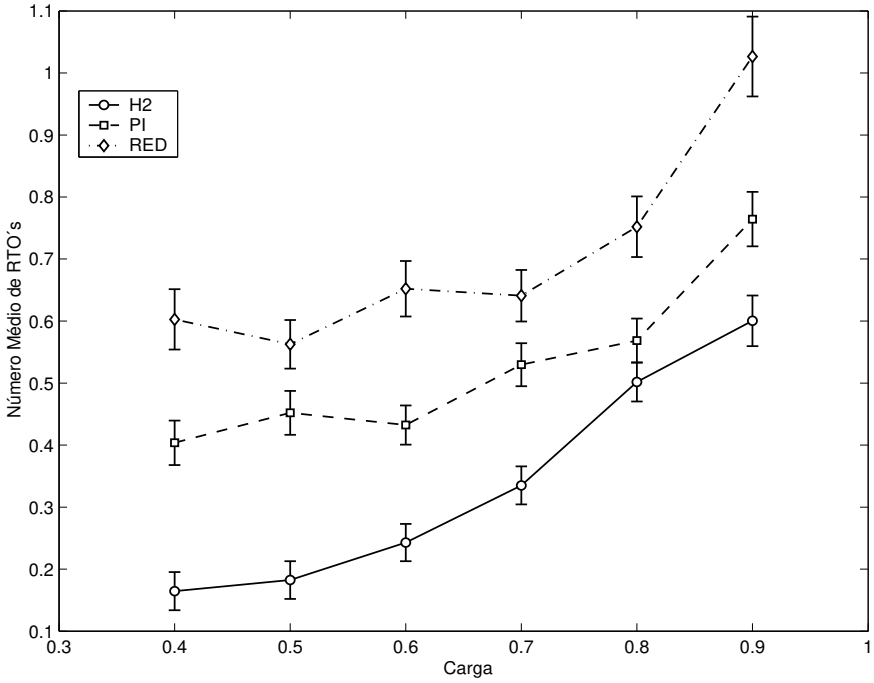

Figura 7.13: Tráfego FTP: número médio de RTO por conexão ativa como função da carga

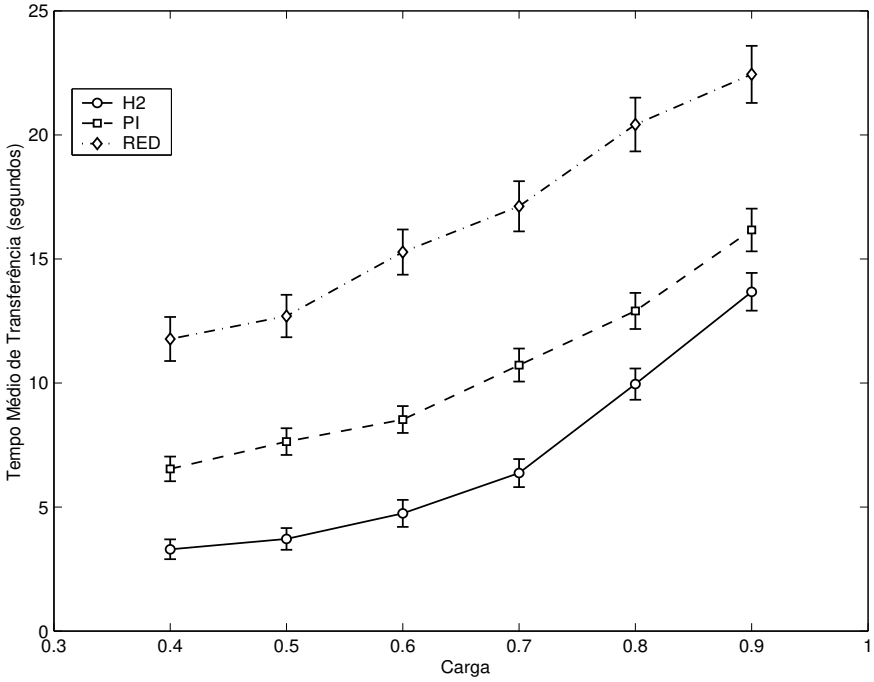

Figura 7.14: Tráfego FTP: tempo médio de transferência por conexão ativa como função carga

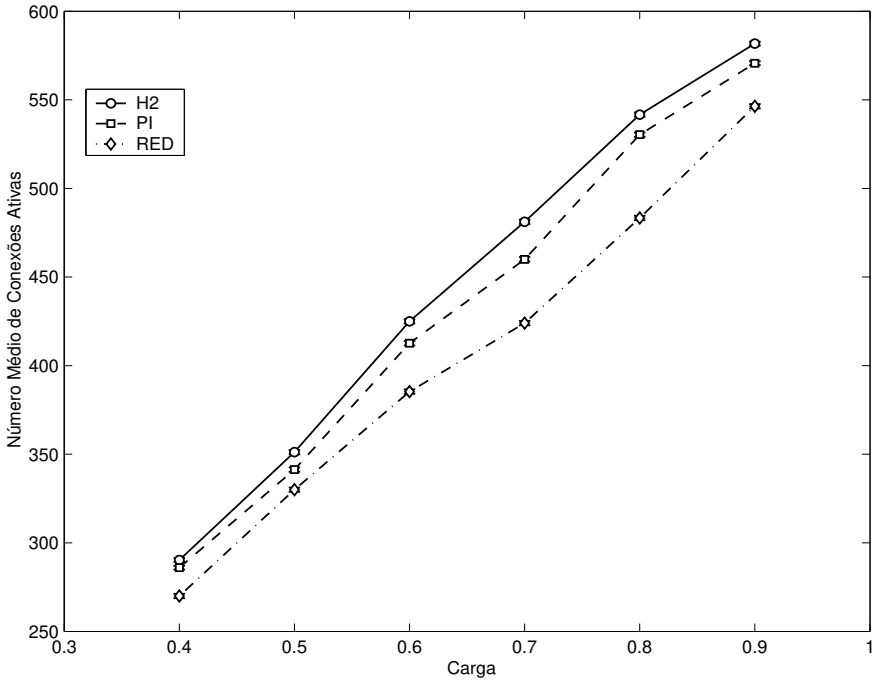

Figura 7.15: Tráfego FTP: número médio de conexões ativas como função da carga

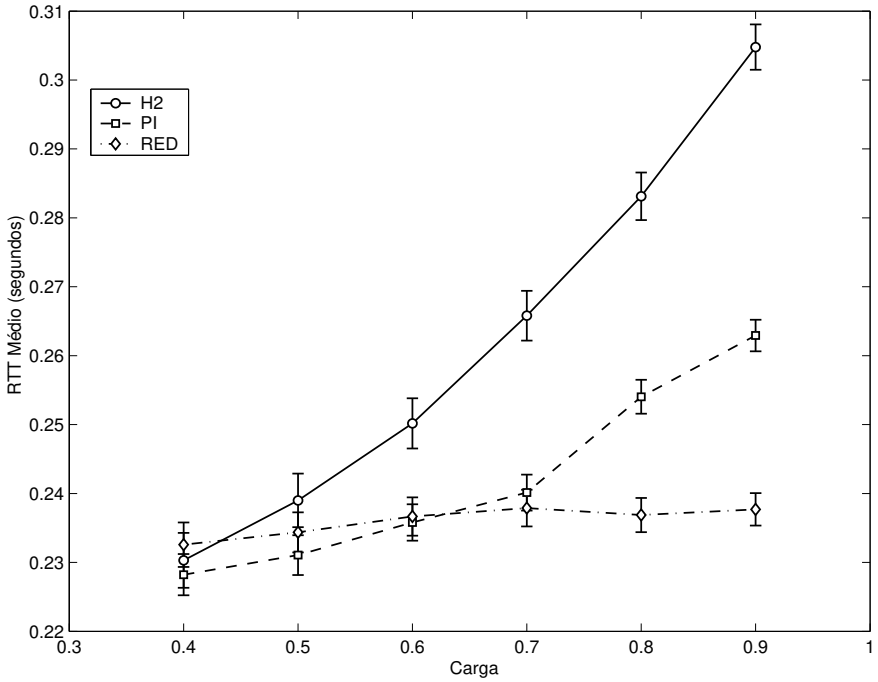

Figura 7.16: Tráfego FTP: valor médio de RTT por conexão ativa como função da carga

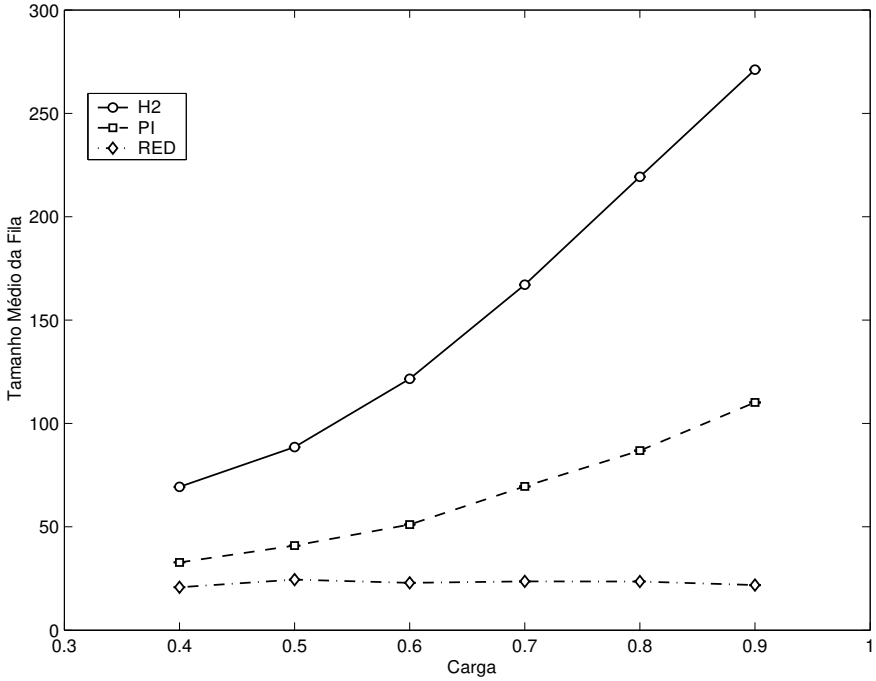

Figura 7.17: Tráfego Web: valor médio para o tamanho da fila

7.3.2 Experimento com Tráfego de Curta Duração - Web

As Figuras 7.17 e 7.18, apresentam, respectivamente, o valor médio para o tamanho da fila e o valor médio para a taxa de perda para as políticas de AQM avaliadas em função da carga, sob tráfego WEB.

Da mesma forma que no experimento com tráfego FTP, no experimento com tráfego Web, o tamanho da fila produzido por H2-AQM e PI-AQM aumentam em função da carga, enquanto que RED mantém os valores basicamente constantes (Figura 7.17). H2-AQM, mais uma vez é a política que produz os maiores valores para o tamanho médio da fila, que apresenta os maiores valores para o tamanho médio da fila, chegando a cerca de 275 pacotes. Entretanto, este ainda é um valor baixo, considerando que o tamanho máximo da fila é de 800 pacotes. Pode-se verificar que sob tráfego Web (Figura 7.17), os valores obtidos para o tamanho médio da fila são menores que os obtidos para tráfego FTP (Figura 7.8).

A Figura 7.18 apresenta a taxa de perda do enlace gargalo. Resultados semelhantes aos obtidos para tráfego FTP podem ser observados também para tráfego Web. H2-AQM produz os menores valores para a taxa de perda para cargas até 0.8. Além disso, as taxas de perda obtidas por H2-AQM e PI-AQM são similares, mas ainda consideravelmente menores que as obtidas por RED. A taxa de perda produzida por H2-AQM é no mínimo 51% e 1% menor que as taxas produzidas por RED e PI-AQM para cargas de 0.9 e no

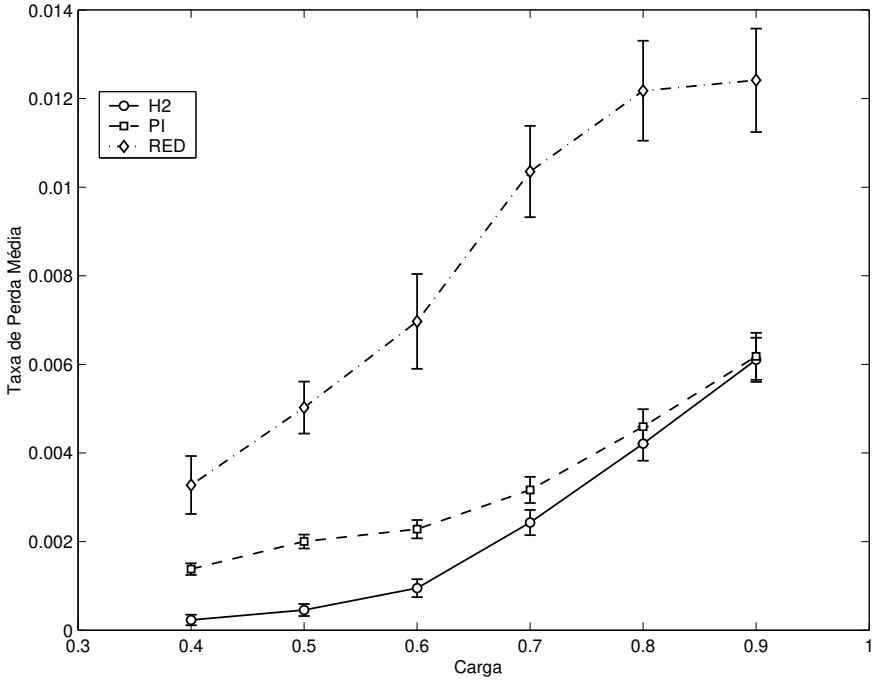

Figura 7.18: Tráfego Web: valor médio para a taxa de perda

máximo 93% e 83% menores para cargas de 0.4. Além disto, a taxa de perda produzida por H2-AQM e PI-AQM dependem basicamente da carga da rede e independem do tipo de tráfego.

As Figuras 7.19 à 7.25 apresentam os resultados das métricas obtidas por conexões ativas, sob tráfego Web. A Figura 7.20 mostra a porcentagem média de banda passante obtida por conexão ativa. Pode ser observado que H2-AQM obtém uma maior quantidade de banda passante do que RED e PI-AQM. Mais uma vez, este resultado é conseqüência dos maiores valores de janela, *Cwnd*, obtidos quando H2-AQM é utilizada (Figura 7.19).

Para congestionamento incipiente, ou seja, para cargas baixas à médias, os valores de banda passante obtidos quando H2-AQM é utilizada são maiores do que os obtidos pelas demais políticas, ou seja, a medida que o congestionamento intensifica a diferença entre os valores obtidos quando H2-AQM é utilizada e quando as demais políticas são utilizadas vai diminuindo, mas ainda continua obtendo um melhor resultado. Comparando-se com os resultados obtidos com tráfego FTP (Figura 7.10), pode-se observar, que sob tráfego Web, os valores de banda passante obtidos são consideravelmente menores (Figura 7.20). Este resultado já era esperado dado que por ser um tráfego de curta duração e por ter uma quantidade pequena de dados a transferir, os valores para o tamanho da janela, *Cwnd*, são bem menores.

O grau de tolerância à perda de segmentos de uma conexão é dependente do tamanho

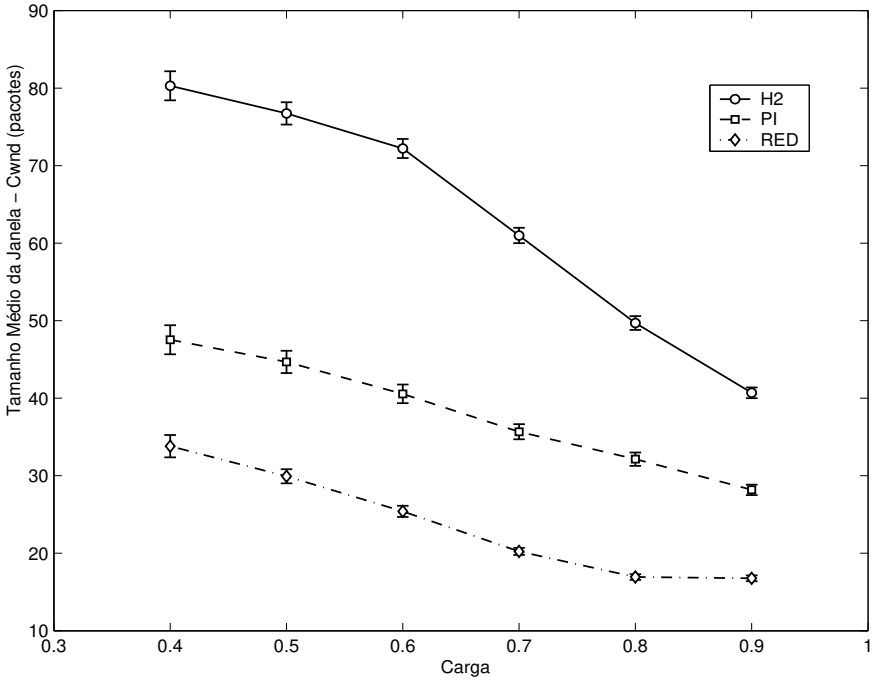

Figura 7.19: Tráfego Web: valor médio da janela de congestionamento (Cwnd) obtido por conexão ativa como função da carga

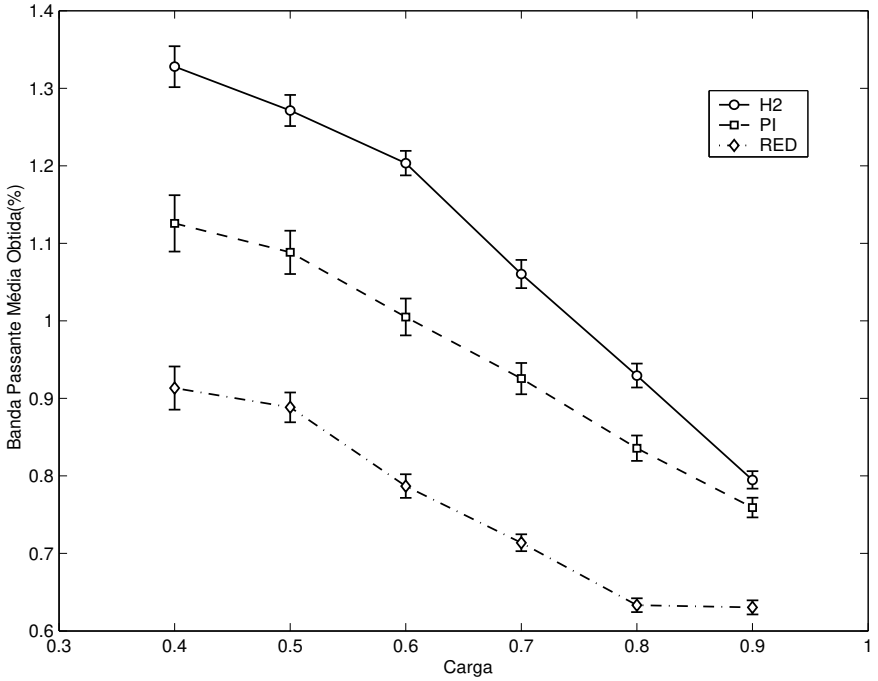

Figura 7.20: Tráfego Web: vazão média por conexão ativa como função da carga

de sua janela de transmissão. Conexões com janelas grandes, podem vir a se recuperar de múltiplas perdas em um único RTT ativando seu mecanismo de recuperação de perdas, enquanto que conexões com janelas pequenas têm que esperar por um intervalo considerável para se recuperar das perdas, dado que terão que esperar pela ocorrência da expiração do intervalo de temporização para detectar a ocorrência de perdas. Desta forma, tráfegos TCP de curta duração, ou que tenham poucos dados a transmitir são mais sensíveis a perda de segmentos e a ocorrência de expiração do intervalo de temporização.

Os baixos valores para a taxa de perda e o descarte de pacotes proporcional fazem com que H2-AQM minimize a perda de pacotes em rajadas, e conseqüentemente, reduza a ocorrência de RTO´s por conexão, como pode ser verificado na Figura 7.21. Comparando-se com RED e PI-AQM, H2-AQM reduz a ocorrência de RTO´s em no mínimo 68% e 29% para cargas de 0.9 e no máximo em 86% e 70% para cargas de 0.4. Pode-se observar que quando RED é utilizada o número de ocorrência de RTO´s é consideravelmente maior que quando as demais políticas são utilizadas, acentuando-se mais a diferença a medida que a carga na rede aumenta, quando então o número de RTO´s aumenta abruptamente. H2-AQM é a política que produz os menores valores para o número de RTO´s. Outra observação que pode ser feita é que sob tráfego FTP, o número de ocorrência de RTO´s é bem maior do que os valores obtidos para tráfego WEB, dado que por ser um tráfego de curta-duração, tem-se uma probabilidade menor de ocorrência de RTO´s.

Por reduzir a ocorrência de RTO´s e por apresentar as mais baixas taxas de perda (Figura 7.18), H2-AQM é mais robusto que RED e PI-AQM na presença de tráfego Web.

Como conseqüência direta da redução do RTO´s e das baixas taxas de perda, tem-se que também para tráfego WEB, H2-AQM reduz o tempo de vida das conexões, como pode ser observado na Figure 7.22. Comparando-se com RED e PI-AQM, H2-AQM reduz, respectivamente, o tempo médio de vida das conexões em no mínimo 28% para cargas de 0.4 e 14% para cargas de 0.9 e no máximo 40% e 17% para cargas de 0.7.

Pode-se observar, na Figura 7.23, que sob tráfego Web, os valores de *goodput* obtidos pelas conexões são maiores dos que os obtidos sob tráfego FTP, como conseqüência da curta duração do tráfego Web, como também devido a pequena quantidade de dados a transmitir. H2-AQM produz os maiores valores de *goodput* sob tráfego Web (Figura 7.23), assim como sob o tráfego FTP (Figura 7.12). Entretanto, a diferença dos valores obtidos por H2-AQM em relação às demais políticas sob tráfego Web não é tão acentuada, mas ainda é significante. A diferença é de no mínimo 24% e 12% para cargas de 0.4 e de no máximo 53% e 13% para cargas de 0.7, quando comparados respectivamente com RED e PI-AQM. Os maiores valores de *goodput* são resultado dos maiores valores de janela, *Cwnd*, das baixas taxas de perda, bem como dos menores números de RTO´s produzidos quando H2-AQM é utilizada.

Sob tráfego Web, H2-AQM também otimizou a utilização dos recursos, permitindo que

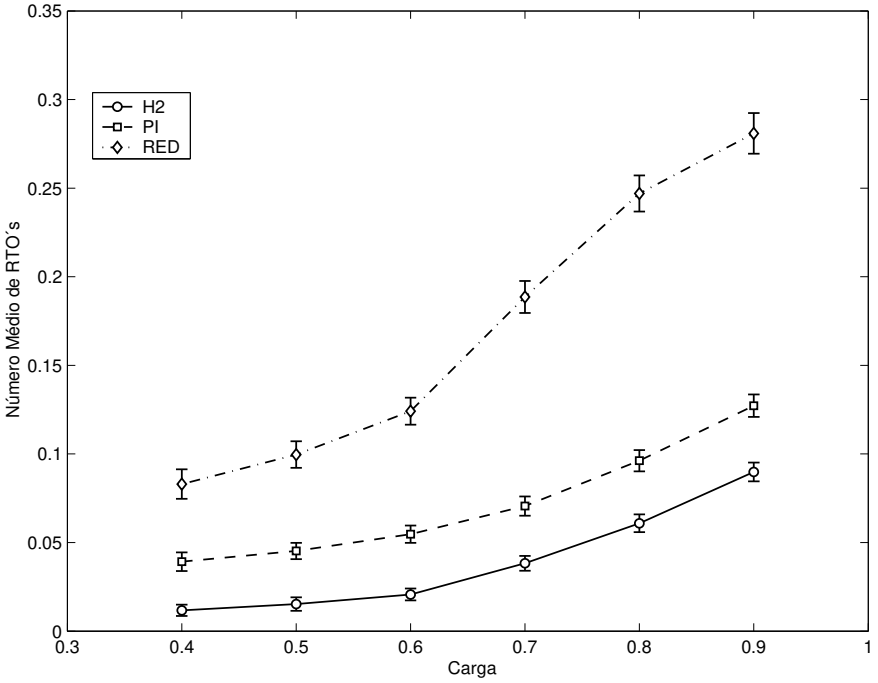

Figura 7.21: Tráfego Web: número médio de RTO por conexão ativa como função da carga

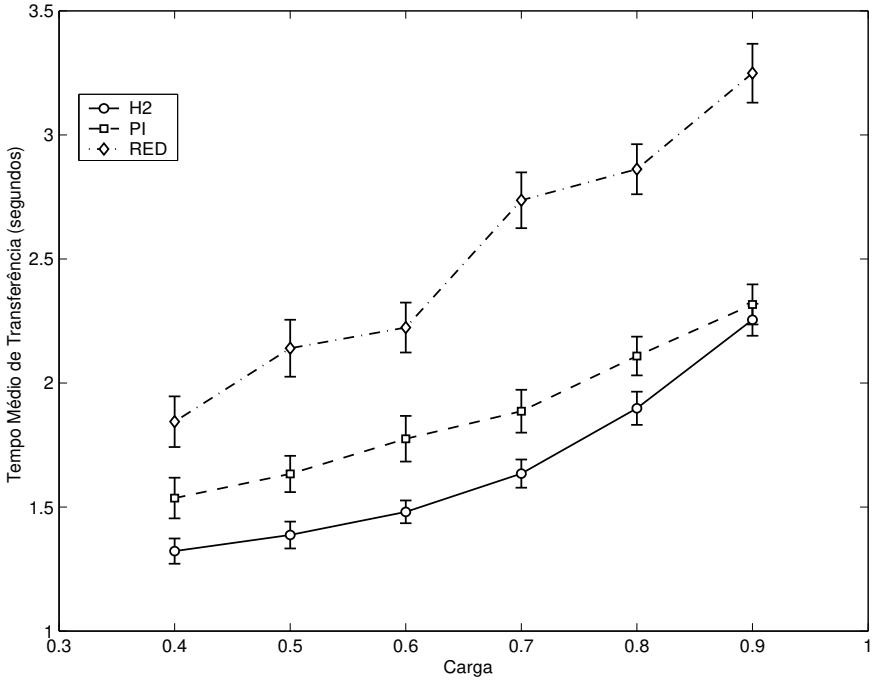

Figura 7.22: Tráfego Web: tempo médio de transferência por conexão ativa como função carga

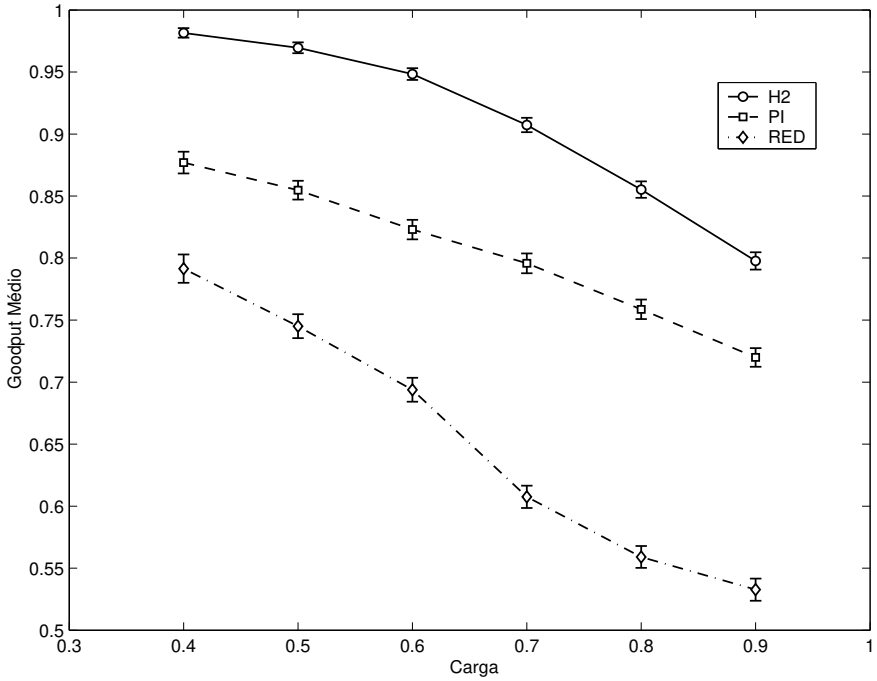

Figura 7.23: Tráfego Web: valor médio de goodput por conexão ativa como função da carga

um maior número de conexões utilizasse os recursos em um mesmo intervalo, assim como ocorreu sob tráfego FTP. (Figura 7.15). Comparando-se com RED e PI-AQM, o aumento conseguido por H2-AQM foi de no mínimo 5% e 3% para cargas de 0.8 e no máximo 26% e 14% para cargas de 0.4. A diferença é mais acentuada para cargas menores, o que é uma conseqüência das menores taxas de perda apresentadas para estas cargas.

Para todas as métricas investigas sob tráfego Web, H2-AQM supera RED e PI-AQM, com exceção apenas do RTT. H2-AQM produz maiores tamanhos de fila que as demais políticas, e, conseqüentemente, maiores valores de RTT. Entretanto, os valores médios de RTT produzidos por H2-AQM são no máximo 28% e 16% maiores para cargas de 0.9 do que os valores produzidos por RED e PI-AQM respectivamente (Figura 7.25).

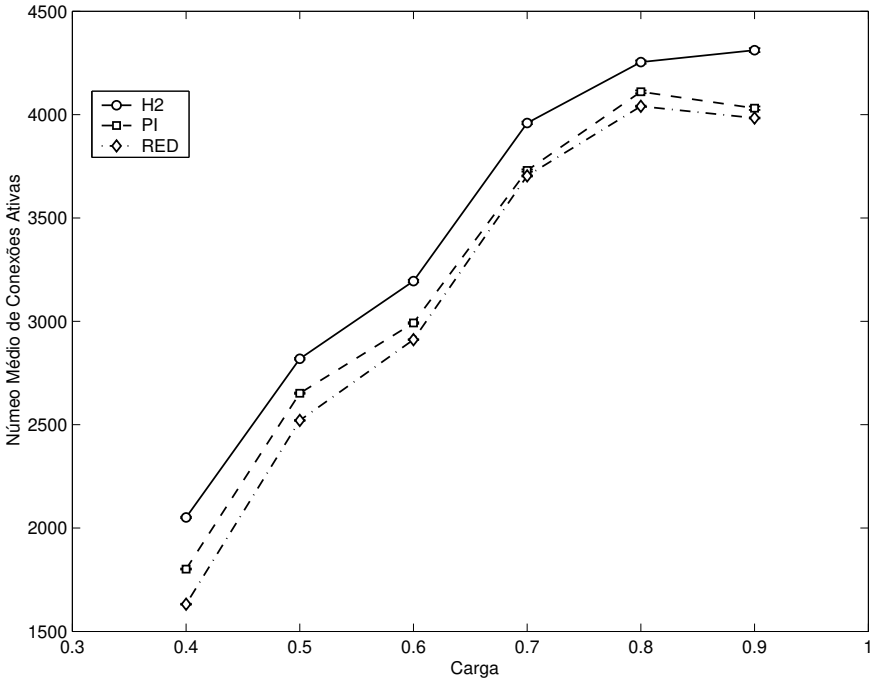

Figura 7.24: Tráfego Web: número médio de conexões ativas como função da carga

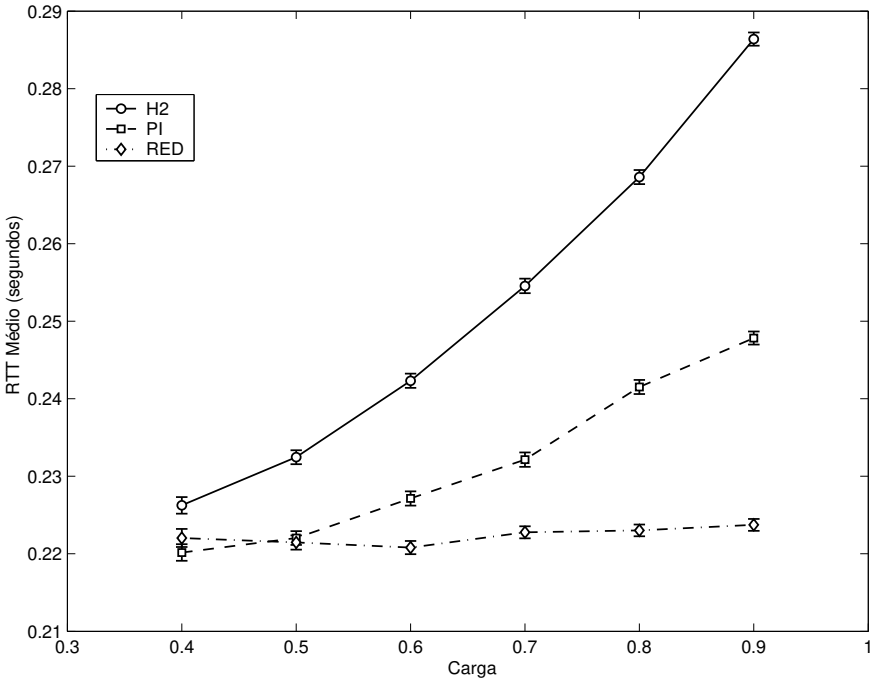

Figura 7.25: Tráfego Web: valor médio de RTT por conexão ativa como função da carga

Capítulo 8

H2-TAQM - Controlador TCP/AQM com Mecanismo de Temporização

O controlador H2-AQM, apresentado no Capítulo 6, foi projetado utilizando um modelo simplificado da dinâmica do comportamento do TCP. Neste capítulo, será apresentado o H2-TAQM, uma variação do controlador H2-AQM que utiliza um modelo estendido da dinâmica do TCP [9], que inclui o mecanismo de temporização.

Em [11], foi apresentada uma comparação entre diferentes versões do TCP utilizando RED como a política de AQM. Foi verificado que o uso em conjunto do algoritmo *Limited Transmit* com TCP Reno, TCP NewReno e TCP Sack pode melhorar o desempenho do TCP na recuperação de perdas. Entretanto, a extensão destes benefícios quando políticas de AQM mais eficientes são utilizadas ainda não foi verificado. Desta forma, neste capítulo avalia-se a eficácia do uso do mecanismo *Limited Transmit* em conjunto com TCP Reno, TCP NewReno e TCP Sack quando submetidos a uma política de gerenciamento ativo de filas baseada em controle ótimo, desenvolvida utilizando-se um modelo que inclui o mecanismo de temporização.

8.1 Modelo do Sistema TCP/AQM que inclui o Mecanismo de Temporização

O modelo dinâmico para o sistema de controle de congestionamento TCP/AQM apresentado na seção 5.1 é uma versão simplificada do modelo apresentado em [9] que ignora o mecanismo de *timeout*. O modelo estendido da dinâmica do sistema de congestionamento apresentado em [9], que inclui o seu mecanismo de temporização é dado por:

$$\dot{W}(t) = \frac{1}{R(t)} - (1 - Q(W(t))) \cdot \frac{W(t)}{2} \cdot \frac{W(t - R(t))}{R(t - R(t))} \cdot p(t - R(t)) \quad (8.1)$$
$$+(1 - W(t)) \cdot Q(W(t)) \cdot \frac{W(t - R(t))}{R(t - R(t))} \cdot p(t - R(t));$$

$$\dot{q}(t) = -C(t) + \frac{N(t)}{R(t)} \cdot W(t) + \omega_q(t); \quad (8.2)$$

$$R(t) = \frac{q(t)}{C(t)} + T_p; \quad (8.3)$$

$$Q(W(t)) = min\left(1, \frac{3}{W(t)}\right); \quad (8.4)$$

onde:

$W(t)$:	é a média do tamanho da janela TCP em pacotes;
$q(t)$:	é o tamanho da fila em pacotes;
$\omega_q(t)$:	é o ruído gerado pelo tráfego UDP;
$R(t)$:	é o tempo total de viagem (RTT) em segundos;
$C(t)$:	é a capacidade do enlace em pacotes/segundo;
T_p:	é o tempo de propagação em segundos;
$N(t)$:	é o fator de carga em número de conexões TCP;
$p(t)$:	é a probabilidade de descarte/marcação de pacotes;
$Q(W(t))$:	é a probabilidade de que uma perda foi detectada pela ocorrência de um *timeout*.

A Equação (8.1) descreve a dinâmica da janela TCP enquanto que a Equação (8.2) modela a dinâmica da fila.

O ponto de equilíbrio para este sistema, obtido através da solução de $\dot{W}(t) = 0$, $\dot{q}(t) = 0$, and $R_0 = \frac{q_0}{C_0} + T_p$, é dada por:

$$W_0 = \frac{-3 + \sqrt{\frac{8+33p_0}{p_0}}}{2} = \frac{R_0 C_0}{N_0}; \quad (8.5)$$

$$q_0 = N_0 W_0 - C_0 T_p = C_0(R_0 - T_p); \quad (8.6)$$

$$p_0 = \frac{2N_0^2}{(R_0 C_0)^2 + 3N_0 R_0 C_0 - 6N_0^2}; \quad (8.7)$$

Para linearizar o Sistema (8.1-8.4), foram feitas as mesmas considerações utilizadas na linearização do Sistema (5.1-5.3). Desta forma, as Equações (8.1) e (8.2) são linearizadas em torno do ponto de equilíbrio (W_0, q_0, p_0) resultando em:

$$\begin{aligned}
\dot{x}_1(t) &= -\frac{-N_0R_0^2C_0^2 - 6N_0^3}{R_0^4C_0^3 + 3N_0R_0^3C_0^2 - 6N_0^2R_0^2C_0}x_1(t) \\
&\quad -\frac{N_0}{R_0^2C_0}x_1(t-R_0) - \frac{1}{R_0^2C_0}(x_2(t) - x_2(t-R_0)) \\
&\quad +\frac{6N_0^2 - R_0^2C_0^2 - 3N_0R_0C_0}{2N_0^2R_0}u(t-R_0); \qquad (8.8)\\
\dot{x}_2(t) &= \frac{N_0}{R_0}x_1(t) - \frac{1}{R_0}x_2(t) + \omega_q(t); \qquad (8.9)
\end{aligned}$$

onde:

$x_1(t) \doteq W(t) - W_0$;

$x_2(t) \doteq q(t) - q_0$;

$u(t) \doteq p(t) - p_0$;

8.2 Projeto do Controlador Ótimo

Nesta seção, a abordagem não racional utilizada para derivar os controladores ótimos, apresentada no Capítulo 6, é utilizada para derivar um controlador ótimo para o sistema (8.8), referenciado como $C_{H2_T}(s)$.

O sistema linear apresentado em (8.8) pode ser expresso no espaço de estado de acordo com (6.1). As matrizes do sistema são definidas como:

$$A_0 = \begin{bmatrix} \frac{-N_0R_0^2C_0^2-6N_0^3}{R_0^4C_0^3+3N_0R_0^3C_0^2-6N_0^2R_0^2C_0} & -\frac{1}{R_0^2C_0} \\ \frac{N_0}{R_0} & -\frac{1}{R_0} \end{bmatrix}, \quad A_1 = \begin{bmatrix} -\frac{N_0}{R_0^2C} & \frac{1}{R_0^2C_0} \\ 0 & 0 \end{bmatrix},$$

$$B_u = \begin{bmatrix} \frac{6N_0^2-R_0^2C_0^2-3N_0R_0C_0}{2N_0^2R_0} \\ 0 \end{bmatrix}, \qquad B_w = \begin{bmatrix} 0 & 0 \\ 0.2C_0 & 0 \end{bmatrix},$$

$$D_{yw} = \begin{bmatrix} 0 & 0.02C_0 \end{bmatrix}, \qquad C_y = \begin{bmatrix} 0 & 1 \end{bmatrix};$$

$$C_{z0} = A_0 \qquad C_{z1} = A_1$$

$$D_{zu} = B_u$$

As matrizes A_0, A_1 e B_u foram obtidas diretamente da linearização do sistema.

As matrizes B_w, D_{yw} e C_y foram mantidas idênticas às utilizadas para obter os controladores apresentados no Capítulo 6.

As matrizes C_{z0}, C_{z1} e D_{zu} representam o objetivo de minimizar a variação em torno dos valores do ponto de equilíbrio da janela de transmissão e do tamanho da fila, o que garante banda passante mais constante e previne a ocorrência de *jitter*. Para se garantir tal objetivo, tem-se que as matrizes C_{z0}, C_{z1} e D_{zu} devem ser, respectivamente, iguais as matrizes A_0, A_1 e B_u. A escolha deste objetivo deve-se ao fato de que o controlador que apresentou o melhor desempenho, dentre os controladores desenvolvidos no Capítulo 6, foi o $C_{H2_3}(s)$.

A síntese do controlador segue a abordagem apresentada no Capítulo 6. Depois de definidos os objetivos de desempenho, o Sistema (8.8), apresentado na forma de estado de espaço de acordo com (6.1), é conectado a um controlador da forma (6.2) resultando em um sistema linear com atraso, como o Sistema (6.5).

Para garantir a estabilidade do Sistema foi utilizado o Teorema 4-b apresentado em [143], que especifica que um sistema como o (6.5) é assintoticamente estável e $\|H_{wz}(s)\|_2^2 < \gamma$, se $\mathbf{D} = 0$ e se existem matrizes simétricas e definidas positivas W, Y_0 e X_j, e matrizes F, R, L_j e Q_j, com $j = 0, 1$, tais que os (6.6) e (6.7) tenham uma solução factível.

Este problema de programação convexa foi resolvido numericamente utilizando a rotina LMISol [144]. Os parâmetros de rede utilizados foram os mesmos utilizados para obter o controlador H2-AQM, $R_0 = 0.256$ segundos, $C_0 = 3750$ pacotes/segundos e $N_0 = 120$ sessões TCP.

Como foi obtida uma solução factível, pode-se afirmar que o sistema é estável, bem como determinar os parâmetros do controlador. Os parâmetros do controlador foram determinados utilizando-se (6.8). Na solução obtida, as matrizes $\hat{A}_1$ e $\hat{C}_1$ são aproximadamente zero, sendo portanto ignoradas. O controlador $C_{TH2}(s)$, representado por sua função de transferência no domínio da freqüência, é dado por:

$$C_{TH2}(s) = \frac{0.02992s + 7.569^{-5}}{s^2 + 910.3s + 0.0125}; \tag{8.10}$$

A freqüência utilizada para obter o controlador C_{TH2} no domínio-z foi a mesma utilizada para derivar os controladores H2-AQM, $f_s = 375Hz$. O controlador C_{TH2} no domínio-z é, então, dado por:

$$C_{TH2}(z) = \frac{a(z+1)}{z+b} = \frac{1.802^{-5}(z+1)}{z + 0.09654}; \tag{8.11}$$

A função de transferência entre $\delta p = p - p_0$ e $\delta q = q - q_0$, apresentada em (8.11), pode ser convertida em uma equação de diferenças no tempo discreto kT, onde $T = \frac{1}{f_s}$:

$$\delta p(kT) = a[\delta q(kT) + \delta q((k-1)T)] - b\delta p((k-1)T); \tag{8.12}$$

O Algoritmo 8.1, correspondente a implementação da Equação (8.12), difere apenas

Algoritmo 8.1 Algoritmo para o cálculo da probabilidade de descarte/marcação em H2-TAQM

H2-TAQM-ProbabilityFunction()

$p_0 \Leftarrow \frac{2N_0^2}{(R_0C_0)^2+3N_0R_0C_0-6N_0^2}$;
$p \Leftarrow a * (q - 2 * q_0 + q_{old}) - b * p_{old} + p_0 * (1 + b)$;
$p_{old} \Leftarrow p$;
$q_{old} \Leftarrow q$;

do apresentado no Algoritmo 6.1, pelo cálculo do valor de p_0.

8.3 Resultados Numéricos

Para avaliar o impacto da probabilidade de descarte/marcação no desempenho do mecanismo *Limited Transmit* utilizado em conjunto com diferentes versões TCP, o Algoritmo 8.1 foi implementado no simulador de redes NS, versão 2.26 [65]. A freqüência de amostragem utilizada para derivar o controlador digital foi de 375 Hz, gerando os valores de 1.802^{-5} e 0.09654, respectivamente, para os coeficientes a e b do H2-TAQM. RED foi simulado utilizando os valores *default* do NS, que são: $min_{th} = 5$, $max_{th} = 15$ e $max_p = 0.1$.

Os experimentos foram conduzidos tanto para tráfego de longa-duração, FTP, como para tráfego de curta-duração, Web. O cenário utilizado foi o mesmo apresentado na Seção 7.1. Para cada tipo de tráfego e para cada política de AQM, o uso em conjunto do TCP Reno, do TCP NewReno, e do TCP SACK com o *Limited Transmit*, foram comparados sob diferentes condições de carga.

Várias métricas relacionadas com as conexões ativas foram investigadas e seus valores médios obtidos em função da carga na rede são apresentados na forma de gráficos. Intervalos de confiança com 95% de nível de confiança foram gerados pelo método de replicação independente e não são apresentados em favor da interpretação visual dos gráficos. As métricas analisadas são: o tamanho médio da janela, Cwnd; o *goodput* médio; o número médio de ocorrência de RTO´s e o tempo de médio de transferência.

8.3.1 Experimento com Tráfego de Longa Duração - FTP

As Figuras 8.2 à 8.3 apresentam os resultados dos experimentos realizados com tráfego FTP.

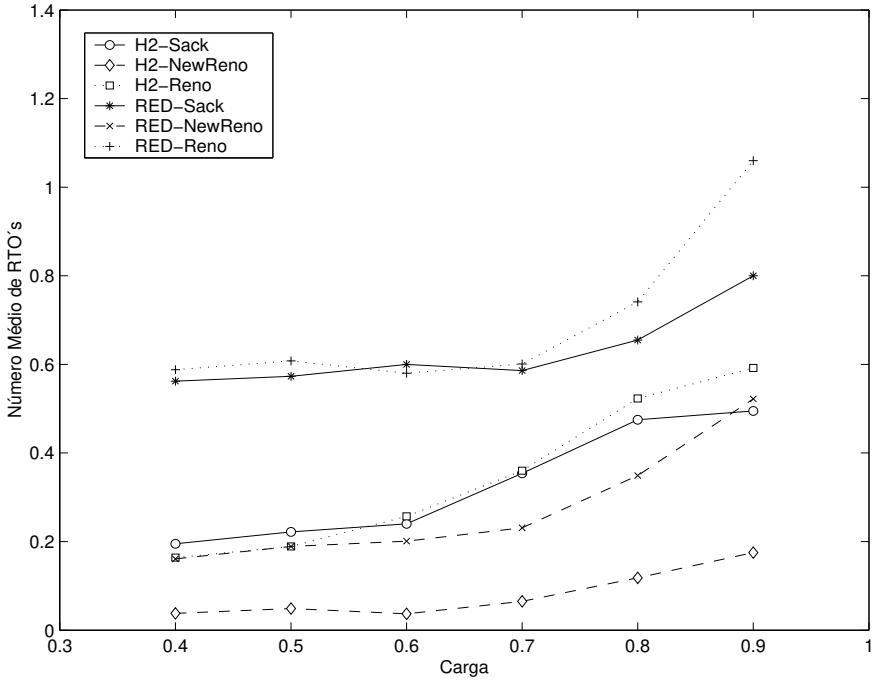

Figura 8.1: Tráfego FTP: número médio de RTO´s por conexão ativa como função da carga

A ocorrência de *timeouts* é danoso para uma conexão TCP dado que reduz drasticamente a vazão, além de aumentar a latência na transferência dos arquivos. Na Figura 8.1, é apresentado o número médio de ocorrências de RTO´s para cada uma das versões TCP quando RED e H2-AQM são utilizadas como políticas AQM. Pode-se verificar que todas as versões TCP obtêm melhor resultado quando são utilizadas em conjunto com H2-AQM, diminuindo consideravelmente a ocorrência de RTO´s. Isto deve-se ao fato de que H2-AQM realiza o descarte de pacotes de forma proporcional, minimizando, assim, a perda de pacotes em rajadas, e conseqüentemente, reduzindo a ocorrência de RTO´s. Entretanto, o TCP NewReno utilizado em conjunto com RED supera as outras duas versões TCPs utilizadas em conjunto com H2-AQM, e quando o NewReno é utilizado com H2-AQM, a redução no número de RTO´s é ainda maior, o que convalida o resultado apresentado em [11], que o TCP NewReno mostra um resultado superior em relação as outras duas versões TCP em minimizar a ocorrência de RTO´s. Outro observação que pode ser feita é que independente da política de AQM utilizada, o TCP NewReno só apresenta um resultado superior ao TCP Reno, considerando-se uma mesma política AQM, quando a carga na rede é intensa.

Para se garantir uma taxa máxima de transmissão, deve-se maximizar o tamanho da janela. Com a redução na ocorrência de RTO´s, tem-se um aumento no tamanho

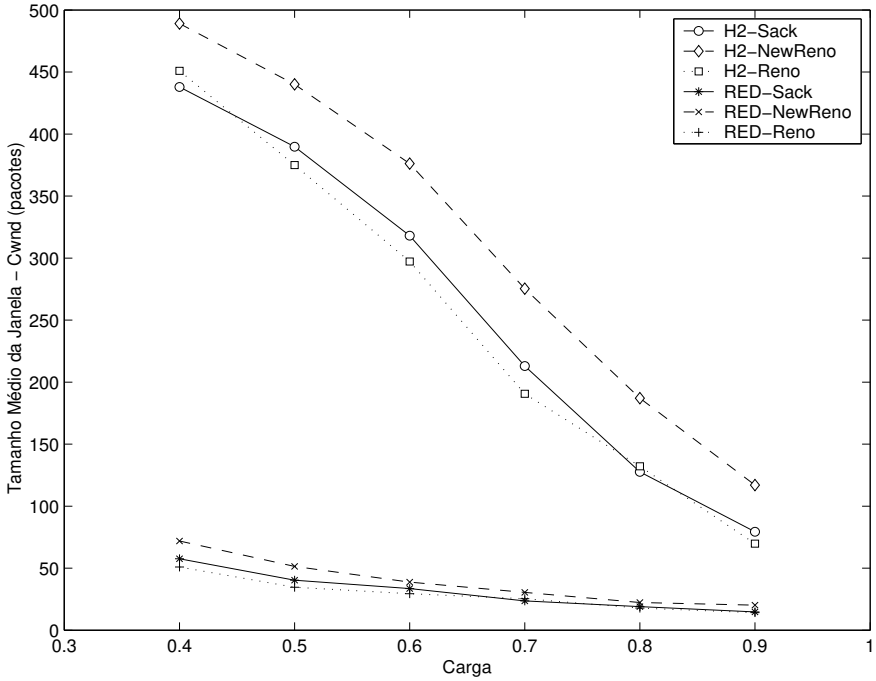

Figura 8.2: Tráfego FTP: valor médio da janela de congestionamento (Cwnd) obtido por conexão ativa como função da carga

médio da janela. A Figura 8.2 apresenta o tamanho médio da janela para cada uma das versões TCP quando RED e H2-AQM são utilizadas como políticas AQM. Quando RED é utilizada como política de AQM, apesar da diferença entre os valores obtidos ser muito pequena, o TCP NewReno apresenta o melhor resultado. Quando a política de AQM utilizada é H2-AQM, pode-se ver claramente que o TCP NewReno obtém um resultado superior ao TCP Sack, que por sua vez supera o TCP Reno para quase todas as cargas. Outra observação que pode ser feita é que quando a política de AQM utilizada é H2-AQM, os valores de vazão obtidos para todas as diferentes versões TCP são consideravelmente maiores que os obtidos para as mesmas versões quando RED é utilizada. A diferença é mais acentuada para cargas menores, e vai diminuindo a medida que a carga aumenta.

Uma conseqüência direta da redução na ocorrência de RTO´s e de um maior tamanho de janela é a redução no tempo de vida da conexão. Como pode ser observado na Figura 8.3, a versão TCP que conseguiu o menor tempo médio de transferência, o TCP NewReno, seguido por Sack e Reno, foi justamente a que conseguiu maiores valores para o tamanho da janela e menores valores de ocorrência de RTO´s. Pode-se verificar que quando H2-AQM foi utilizada como política de AQM, o resultado de todas as versões TCP foi bem superior ao resultado apresentado para as mesmas versões quando RED é utilizada.

A Figura 8.4 apresenta o valor médio de *goodput* obtido por conexão ativa, para cada

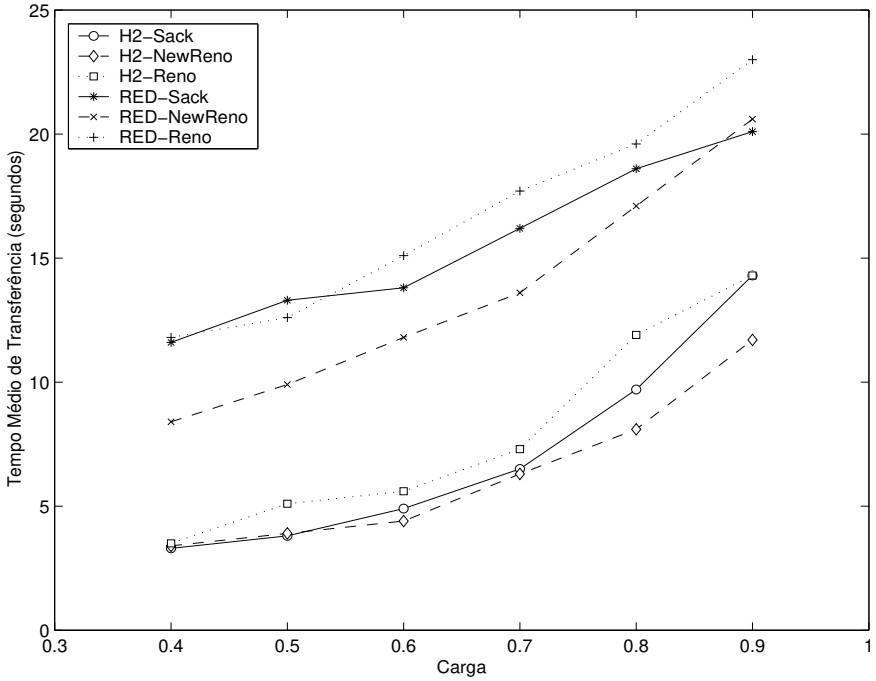

Figura 8.3: Tráfego FTP: tempo médio de transferência por conexão ativa como função carga

uma das versões TCP utilizadas em conjunto com H2-AQM e RED. Esta métrica fornece uma indicação mais precisa da eficácia da utilização do enlace por uma conexão. Quando RED é utilizada como política de AQM, os valores de *goodput* mantêm-se basicamente constantes independentemente da carga, enquanto que quando H2-AQM os valores de *goodput* obtidos são maiores para cargas menores, mostrando que H2-AQM faz uma utilização mais eficiente dos recursos que RED, independentemente da versão TCP utilizada. Mesmo para cargas altas, o uso em conjunto de todas as versões TCP com H2-AQM apresentam resultado bem melhor do que quando RED é utilizada. Uma observação que pode ser feita é que apesar do TCP NewReno ter apresentado um resultado superior às demais versões TCP em relação ao tamanho da janela e em relação ao número de ocorrência de RTO´s, tem-se que os valores obtidos de *goodput* não diferem muito entre as versões TCP, considerando-se a mesma política de AQM. A justificativa para isto é que o cálculo de *goodput* é dado por: $(nbytes - nretbytes)/nbytes$, onde *nbytes* indica o numero total de *bytes* enviados, e *nretbytes* é a quantidade de *bytes* que foram retransmitidos. Desta forma, considerando-se a mesma política de AQM, tem-se basicamente a mesma taxa de perda para todas as versões, o que implica em dizer que todas terão que fazer retransmissões.

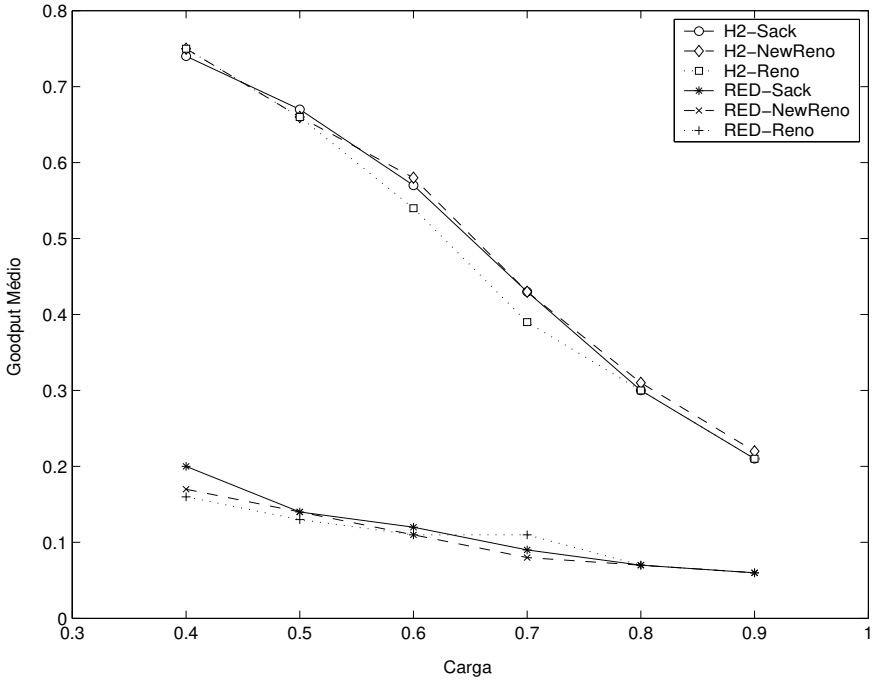

Figura 8.4: Tráfego FTP: valor médio de goodput por conexão ativa como função da carga

8.3.2 Experimento com Tráfego de Curta Duração - Web

As Figuras 8.6 à 8.7 apresentam os resultados dos experimentos realizados com tráfego Web.

Como já apresentado anteriormente, o grau de tolerância à perda de segmentos de uma conexão é dependente do tamanho de sua janela de transmissão. Desta forma, tráfegos TCP de curta duração ou que tenham poucos dados a transmitir são mais sensíveis a perda de segmentos e a ocorrência RTO´s. Na Figura 8.5, é apresentado o número médio de ocorrências de RTO´s para cada uma das versões TCP quando RED e H2-AQM são utilizadas como políticas AQM. Quando H2-AQM é utilizada, todas as diferentes versões TCP reduzem a ocorrência RTO´s para zero para cargas até 0.7. Mesmo para cargas maiores, o número de retransmissões devido à expiração do intervalo de temporização gerados, utilizando-se H2-TAQM, é menor que para as mesmas versões TCP que utilizam RED. Diferentemente do que aconteceu sob tráfego FTP, considerando-se apenas uma mesma política de AQM, não foi apresentada uma diferença significativa entre as distintas versões TCP. Entretanto, o TCP NewReno apresenta melhor resultado para cargas acima de 0.8, quando RED e H2-AQM são, respectivamente, utilizadas. Esta relativa equivalência entre as três versões TCP deve-se ao fato de que apenas 10% das conexões onde ocorrem RTO´s tem janelas de congestionamento maiores que 10 segmentos [33]. Desta forma, como o TCP *Limited Transmit* foi desenvolvido para aprimorar a eficiência

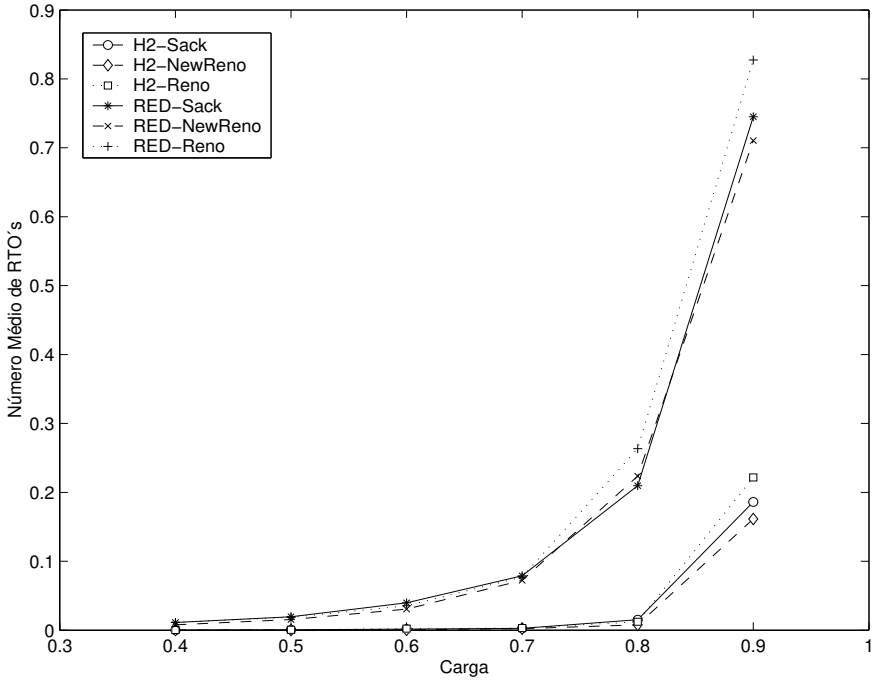

Figura 8.5: Tráfego Web: número médio de RTO´s por conexão ativa como função da carga

do TCP na presença de janelas de congestionamento pequenas, reduzindo a ocorrência de *timeouts*, tem-se que, o uso em conjunto com o *Limited Transmit*, faz com que as três versões TCP avaliadas tenham um resultado equivalente sob tráfego Web, que é caracterizado justamente por ter tamanho de janelas pequenos.

A Figura 8.6 apresenta o tamanho médio da janela para cada uma das versões TCP quando RED e H2-AQM são utilizadas como políticas AQM. Quando RED é utilizada como política de AQM, tem-se que todas as diferentes versões TCP apresentam resultado equivalente. Além disto, os valores obtidos decrescem a medida que a carga aumenta. Quando H2-AQM é utilizada, tem-se que o TCP Reno e o TCP Sack apresentam resultado similar, e o TCP NewReno apresenta um resultado ligeiramente melhor. Diferentemente do que aconteceu com RED, os valores obtidos para o tamanho médio da janela mantém-se basicamente constantes para cargas baixas, sendo reduzido seu valor apenas para cargas acima de 0.7.

Na Figura 8.7, são apresentados os valores para o tempo médio de transferência. Pode-se verificar que independentemente da política AQM, não existem diferenças significativas entre os valores obtidos pelo TCP NewReno e o TCP Sack. Diferentemente do que aconteceu com tráfego FTP, quando H2-AQM é utilizada como política de AQM, o tempo de transmissão foi mantido basicamente constante para todas as versões TCP investigadas

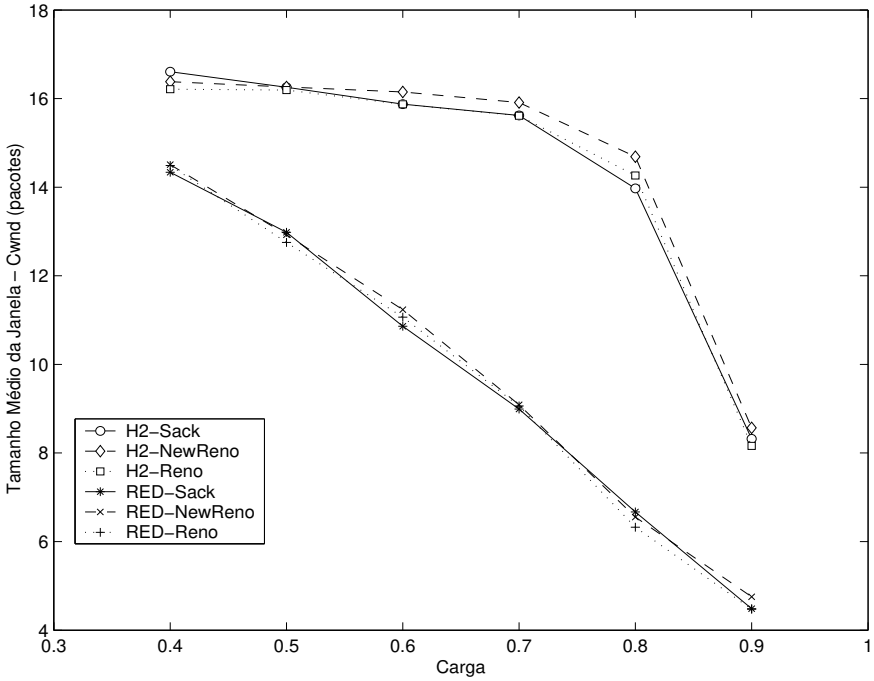

Figura 8.6: Tráfego Web: valor médio da janela de congestionamento (Cwnd) obtido por conexão ativa como função da carga

para cargas até 0.7. Quando RED é utilizada, tem-se que o tempo de transmissão aumenta a medida que a carga na rede intensifica.

Na Figura 8.8, pode-se verificar os valores médio de *goodput* obtidos para cada uma das versões TCP utilizadas em conjunto com H2-AQM e RED. Comparando-se com os resultados obtidos para tráfego FTP, tem-se para tráfego Web, maiores valores de *goodput* por conexão, independente das versões TCP e das políticas AQM utilizadas. Isto se deve ao fato das conexões TCP para tráfego Web serem de curta duração, e, assim, apresentarem em média menos perdas e retransmissões por conexão. Quando H2-AQM é utilizada como política de AQM, tem-se valores máximos de *goodput* para cargas até 0.6, e apenas para cargas de 0.9, tem-se valores de *goodput* próximos a 0.8. Quando RED é utilizada os valores de *goodput*, decrescem com o aumento da carga, chegando no máximo a 0.55 para cargas de 0.9. Da mesma forma, que para as demais métricas avaliadas para tráfego WEB, tem-se que para uma mesma política AQM, os resultados para as diversas versões TCP é basicamente equivalente.

De uma forma geral, pode-se afirmar que o uso em conjunto do TCP *Limited Transmit*, do TCP NewReno e da política H2-AQM é a combinação que apresenta o resultado mais eficiente entre as versões TCP e as duas políticas de AQM investigadas, sob tráfego FTP e WEB. Apesar de para tráfego Web, o uso do TCP *Limited Transmit* fazer com que

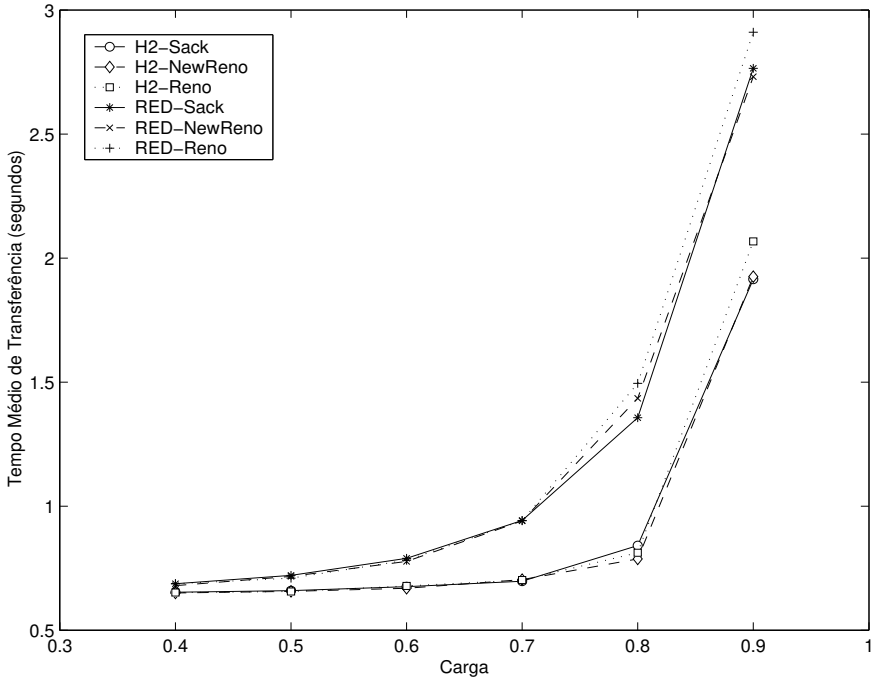

Figura 8.7: Tráfego Web: tempo médio de transferência por conexão ativa como função carga

as diferentes versões TCP apresentem resultados similares, o TCP NewReno mostra-se mais eficiente do que as demais versões TCP para cargas acima de 0.8, quando o congestionamento é mais intenso e a taxa de perda é maior.

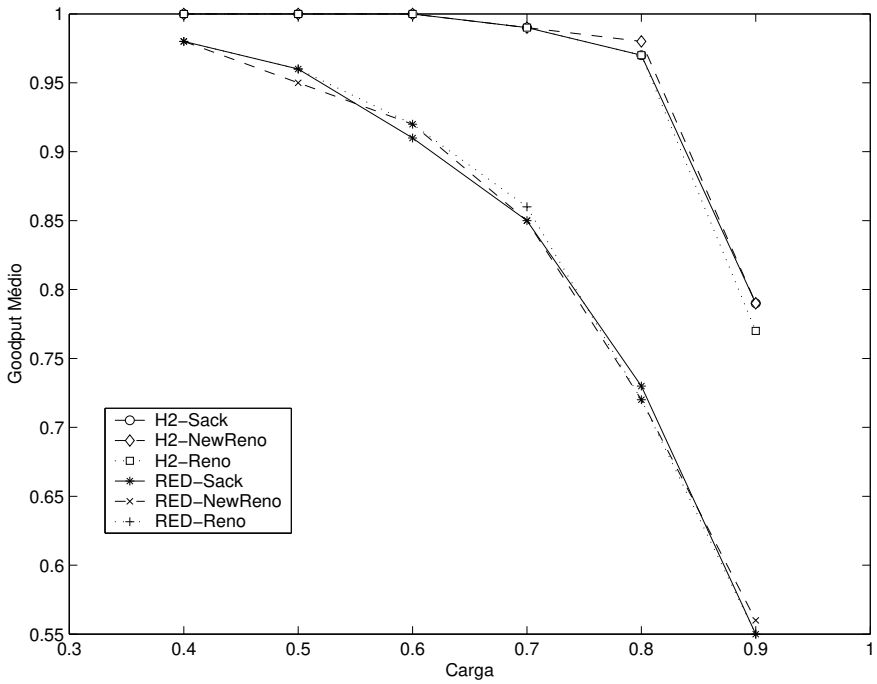

Figura 8.8: Tráfego Web: valor médio de goodput por conexão ativa como função da carga

Capítulo 9

Conclusões e Trabalhos Futuros

Neste capítulo, conclui-se a tese através de um resumo das contribuições apresentadas ao longo do trabalho, bem como através da discussão de questões que não foram tratadas no desenvolvimento deste trabalho, mas que são relevantes e que podem trazer resultados importantes. São apresentados também alguns direcionamentos para a realização de trabalhos futuros.

9.1 Resumo das Contribuições

Esta tese utilizou técnicas da Teoria de Controle Ótimo para desenvolver um controlador ótimo para AQM. O sistema de controle de congestionamento foi tratado como um sistema de controle de retroalimentação, onde a taxa de transmissão dos nós fontes deve ser ajustada de acordo com a ocupação da fila, de forma a evitar o congestionamento. Um modelo simplificado da dinâmica do sistema de controle de congestionamento TCP/AQM foi utilizado para derivar os controladores, que são, então, responsáveis por determinar qual a probabilidade de descarte/marcação adequada, que maximiza a vazão, minimiza as perdas, e ainda garante a estabilidade do tamanho da fila independentemente das variações das condições da rede. Nesta seção é apresentado o resumo das contribuições deste trabalho.

No Capítulo 2, foram apresentados os conceitos, mecanismos e procedimentos utilizados pelo TCP para fornecer o serviço de transferência confiável de dados. Foram introduzidos os algoritmos que compõem o mecanismo de controle de congestionamento da implementação padrão do TCP, o TCP Reno. A contribuição principal deste capítulo é a discussão sobre os problemas relacionados a esta versão do TCP: quando múltiplas perdas ocorrem em uma mesma janela, o TCP Reno não se recupera de forma eficiente; a estratégia utilizada para governar o comportamento da sua janela de transmissão é inerentemente oscilatória, o que faz com que o uso da banda passante disponível também

oscile de forma significante; a taxa de transmissão obtida pelo TCP Reno é inversamente proporcional a raiz quadrada da probabilidade de perda, o que implica em dizer que para se obter altas taxas de transmissão, a taxa de perda tem que ser extremamente baixa. Este problema e o problema anterior fazem com que o TCP Reno tenha uma baixo desempenho em redes com elevado produto banda-atraso; fluxos com grande RTT são prejudicados em detrimento dos que possuem RTT pequeno, não conseguindo a sua porção justa da banda passante; o TCP Reno tem que criar perdas para detectar o estado de congestionamento da rede, e conseqüentemente interpreta de forma errada a ocorrência de uma perda devido a erros de transmissão, reduzindo sua taxa de transmissão.

No Capítulo 3, foi apresentado o mecanismo de controle de congestionamento presente nos roteadores, AQM. A política de AQM recomendada pelo IETF para a Internet, RED, foi apresentada, e foram discutidas suas vantagens bem como suas deficiências. A principal contribuição deste capítulo é a extensa revisão bibliográfica sobre as políticas analíticas propostas para minimizar e solucionar os problemas de RED. Foram apresentados os desenvolvimentos mais recentes e significativos de projeto de mecanismos AQM estáveis, focando nos trabalhos que utilizam ferramentas de Otimização e Teoria de Controle.

No Capítulo 4, foram apresentados os conceitos e fundamentos básicos da área de modelagem matemática e teoria de controle necessários para a compreensão do trabalho desenvolvido. No Capítulo 5, o sistema de controle de congestionamento foi apresentado como um problema de controle e o modelo simplificado da dinâmica deste sistema foi introduzido.

Nos Capítulos 6 e 7, foram apresentados os principais resultados desta tese. No Capítulo 6 o desenvolvimento do projeto do controlador ótimo para AQM, denominado H2-AQM, foi introduzido e no Capítulo 7 são apresentados e discutidos os resultados dos experimentos realizados para investigar a eficácia do controlador H2-AQM. As principais conclusões destes capítulo são:

- H2-AQM foi desenvolvida utilizando-se técnicas de Teoria de Controle Ótimo ao invés de técnicas de controle clássicos. A abordagem utilizada para derivar o controlador foi não-racional, o que faz com que os recursos de rede possam ser usados eficientemente;

- Apesar do controlador ter sido derivado a partir do modelo linearizado do sistema de congestionamento, a planta utilizada para derivar o controlador representa a dinâmica do sistema de congestionamento em detalhes, e também considera a existência de fluxos não adaptativos. Desta forma, o controlador obtido garante a estabilidade do sistema independente das condições de rede;

- Foi investigado o melhor ponto de equilíbrio para o sistema de controle de congestionamento. A escolha do ponto de equilíbrio levou em consideração o ponto em que

o sistema de congestionamento está mais suscetível, ou seja, quando a carga da rede é grande. Com a utilização deste ponto de equilíbrio, foi demonstrado através de simulações que o H2-AQM supera PI-AQM e RED em atingir os objetivos esperados para uma política de AQM. O controlador PI-AQM derivado com este ponto de equilíbrio apresenta um desempenho superior ao PI-AQM original;

- H2-AQM pode ser considerada uma política AQM baseada tanto no tamanho da fila quanto na taxa, dado que a discussão dos objetivos de projeto para o controlador, que obtenha o melhor desempenho para o sistema de congestionamento, leva em consideração a característica de retroalimentação do sistema de congestionamento e do seu estado representado pelo tamanho da janela de transmissão e pelo tamanho da fila (W, q);

- Todos os controladores, derivados a partir dos quatro objetivos definidos no Capítulo 6, apresentam um desempenho equivalente para todas as métricas quando a média é utilizada para compará-los. Desta forma, foi utilizado o coeficiente de variação (CV) como medida estatística adequada para compará-los. O controlador que obteve o melhor resultado nos experimentos realizados para tráfego FTP como WEB, apresentando os menores valores de CV para o tamanho da fila, do tamanho da janela e da probabilidade de descarte/marcação foi o correspondente ao objetivo de garantir banda passante e diminuir a ocorrência de *jitter*;

- Apesar de ter sido utilizada uma abordagem não-racional, o controlador obtido é racional, dado que foi possível cancelar o termo de atraso, garantindo a estratégia global ótima para solucionar o problema de minimização da norma H_2;

- A freqüência de amostragem utilizada na implementação digital do controlador foi escolhida de forma a garantir que o controlador possa reagir de forma apropriada a mudanças abruptas, e também não reagir de forma demasiadamente severa, para permitir que mudanças transientes sejam desconsideradas. O Algoritmo 6.1 e o Algoritmo 6.2, obtidos a partir das equações de diferenças correspondentes a representação no domínio-z destes controladores, são bastante simples, e utilizam, respectivamente, apenas duas e quatro variáveis auxiliares;

- Na Seção 6.6, é feita uma análise da robustez do sistema, utilizando-se técnicas de controle robusto. Desta análise, obteve-se um importante resultado: o aprimoramento do sistema de congestionamento depende de se garantir a estabilidade do tamanho da fila e também da janela do TCP;

- Não foi objetivo do projeto dos controladores H2-AQM, tratar questões relacionadas a justiça. Entretanto, foi verificado via simulação que H2-AQM, por garantir um

descarte proporcional, evita a continuidade de perdas de pacotes bem como reduz a ocorrência de RTO´s. Conseqüentemente, tem-se que conexões curtas não são prejudicadas. Além disto, a planta utilizada considera a existência de fluxos não adaptativos. Desta forma, a presença destes fluxos não impacta negativamente o desempenho dos fluxos TCP, como foi verificado nos experimentos realizados;

- O modelo utilizado para derivar os controladores H2-AQM considera que o tráfego principal é de longa duração. Desta forma, os controladores desenvolvidos a partir deste modelo podem não ser tão eficazes na presença de tráfego de curta-duração, como o são na presença de tráfego de longa-duração. No Capítulo 7, foi verificada a eficácia dos controladores sob a presença de tráfego de longa-duração bem como de curta-duração. Os resultados obtidos mostram que H2-AQM supera RED e PI-AQM em ambos os tipos de tráfego.

No Capítulo 8, é apresentada uma extensão do controlador H2-AQM, denominada H2-TAQM, derivado utilizando um modelo estendido da dinâmica do TCP que inclui o seu mecanismo de temporização. A política de AQM obtida foi utilizada para avaliar a eficácia do mecanismo *Limited Transmit* em conjunto com TCP Reno, TCP NewReno e TCP Sack quando é utilizada uma política de gerenciamento ativo de filas baseada em controle ótimo. Foram feitas simulações em um ambiente dinâmico para cada uma das versões TCP, utilizando-se RED e H2-TAQM como políticas de AQM. Da análise dos resultados obtidos, pode-se afirmar que o uso conjunto do TCP *Limited Transmit* com TCP NewReno é a combinação que apresenta o resultado mais eficiente entre as versões TCP. O TCP NewReno mostra-se mais eficiente do que as demais versões TCP para cargas acima de 0.8, quando o congestionamento é mais intenso e a taxa de perda é maior.

9.2 Trabalhos Futuros

Nesta seção, são apresentados algumas questões que não foram tratadas no desenvolvimento deste trabalho e que são relevantes e podem trazer resultados importantes, como também são delineados alguns direcionamentos para a realização de trabalhos futuros.

9.2.1 Cenários de Simulação

No cenário utilizado nos experimentos realizados com o intuito de verificar a efetividade do controlador H2-AQM e H2-TAQM, a topologia utilizada considera a existência de apenas um único enlace gargalo e o tempo de propagação nos enlaces é o mesmo, o que leva a valores de RTT homogêneos. Além disto, não foram feitas variações no tamanho do *buffer* dos roteadores e o tamanho do pacote utilizado foi o mesmo para todas as simulações.

Apesar de ser um pouco restrito, este cenário foi utilizado dado que o modelo utilizado para a derivação dos controladores considera a existência apenas de tráfego de longa duração, de um único enlace gargalo e que os atrasos são homogêneos. Para tornar o cenário mais realista, foram feitas simulações com um gerador de tráfego, para gerar tanto tráfego de longa bem como de curta duração. Além disto, foi inserido tráfego de ruído e a carga da rede foi variada.

No entanto, é importante verificar a eficácia dos controladores obtidos definindo diferentes cenários de rede, nos quais, podem ser feitas variações de topologia, o que possibilita obter RTTs heterogêneos, múltiplos enlaces gargalo e a inserção de tráfego cruzado. Além disto, deve ser verificado como o tamanho do *buffer*, e o tamanho do pacote também influenciam a eficácia dos controladores.

9.2.2 Comparação entre as diversas Políticas Analíticas

No Capítulo 7, a política H2-AQM foi comparada com RED, a política padrão de AQM e com PI-AQM, por ser o trabalho utilizado como base para o desenvolvimento de H2-AQM e por ambas utilizaram o mesmo modelo.

Um estudo importante é a comparação entre as diversas políticas analíticas de acordo com os seguintes critérios: baseadas em teoria de otimização (solução primal, dual ou primal-dual) ou baseadas em teoria de controle (controladores clássicos, controladores ótimos e robustos); baseadas em critérios virtuais ou reais; baseadas em taxa ou no tamanho da fila; dependentes ou não do atraso. Tal estudo pode trazer resultados importantes sobre a efetividade destas políticas e de como suas vantagens e desvantagens podem ser utilizadas no aprimoramento do sistema de congestionamento.

9.2.3 Extensão do H2-AQM para o modelo de QoS

Evitar a ocorrência de congestionamento e controlá-lo são de capital importância para a operação da rede. Em redes que oferecem o serviço tipo melhor-esforço, o congestionamento pode degradar ainda mais os baixos níveis de qualidade de serviço. Em redes com suporte a QoS, o congestionamento pode inviabilizar o compromisso de atendimento dos requisitos de qualidade de serviços das aplicações. Assim sendo, a prevenção e o controle de congestionamento, o que permite uma degradação suave do desempenho das redes, são fatores essenciais para garantir a otimização de recursos da rede, bem como prover desempenho previsível ou garantido às aplicações.

A política H2-AQM foi desenvolvida para o modelo de serviço tradicional da Internet, ou seja, para o modelo melhor-esforço, onde nenhuma garantia pode ser dada às aplicações.

A política padrão de AQM utilizada no modelo QoS, denominada RIO ou RED with IN/OUT [149], é uma variação de RED, que foi desenvolvido para diferenciar e penalizar

fluxos que estão fora das especificações de contrato no modelo QoS. Para isto, RIO utiliza dois algoritmos RED, um para os pacotes que estão em conformidade com as especificações de contrato, denominados pacotes In e outro para os pacotes que não estão em conformidade, denominados pacotes Out. Por utilizar dois algoritmos RED, pode-se dizer que RIO herda suas vantagens bem como suas deficiências. Desta forma, devem ser investigados o sistema de controle de congestionamento nas redes TCP/IP com suporte a QoS, de forma que um modelo que caracterize este sistema possa ser obtido, e, assim, derivar um controlador ótimo para AQM que garanta a estabilidade e robustez das métricas de desempenho desejadas para um sistema de controle de congestionamento no modelo de QoS. É importante salientar que apesar da metodologia utilizada na obtenção do controlador ótimo no modelo melhor esforço poder ser utilizada para a obtenção do controlador para o modelo QoS, os objetivos de projeto, bem como a escolha do ponto de equilíbrio adequados diferem e são dependentes do modelo.

9.2.4 Modelo mais Completo para o Sistema de Controle de Congestionamento

O modelo dinâmico utilizado para derivar o controlador H2-AQM, é um modelo simplificado que descreve o comportamento AIMD do TCP Reno na sua fase de estabilização. Desta forma, o modelo é consideravelmente preciso para tráfego de longa duração, onde a fase de *Slow Start* é curta, podendo assim ser desconsiderada. No entanto, o modelo não é tão preciso para tráfego de curta duração, visto que neste tipo de tráfego a fase de *Slow Start* tem um papel determinante no comportamento da janela do TCP. Além disto, o modelo considera a existência de apenas um único enlace gargalo e que os atrasos são homogêneos. O modelo considera também que a notificação de congestionamento se dá apenas através do descarte de pacotes, ou seja, não é considerado o suporte a ECN. Finalmente, o tráfego não adaptativo, UDP, foi incluído no modelo como um ruído de entrada e não como um tráfego que tenha um comportamento próprio.

O conhecimento do estado inicial e a correta modelagem do sistema são fundamentais para a exatidão do controlador. Desta forma, um trabalho importante a ser realizado é a utilização de modelos mais acurados e completos do sistema de congestionamento, para que controladores mais eficientes possam ser obtidos.

9.2.5 Investigação de versões TCP estáveis

Da análise da robustez do sistema de congestionamento TCP Reno/AQM realizada na Seção 6.6, obteve-se um importante resultado: o aprimoramento do sistema de congestionamento depende de se garantir a estabilidade do tamanho da fila bem como da janela do

TCP. Este resultado deve-se aos problemas do TCP apresentados na Seção 2.3. Primeiramente devido a "Lei do inverso da raíz quadrada" que mostra que para se obter valores altos para a janela de congestionamento são requeridos valores extremamente baixos para a probabilidade de descarte/marcação p. Outro agravante é a instabilidade própria do TCP Reno que faz com que um sistema de congestionamento que o utilize não convirja para um ponto de equilíbrio, e fique oscilando em torno deste ponto. Isto significa que o TCP fonte oscila consideravelmente a sua taxa de transmissão, não estabilizando-a no valor da taxa de transmissão ideal. Desta forma, quando a carga na rede é pequena tem-se períodos em que recursos são subutilizados e quando a carga na rede intensifica, tem-se a ocorrência do congestionamento e suas conseqüências.

Várias propostas de protocolos TCP estáveis foram apresentadas. Assim, deve-se analisar a eficácia e a estabilidade destas propostas, visando verificar quais delas é mais eficaz em prevenir o congestionamento, ou ainda fazer o controle eficiente quando ele ocorre, minimizando assim, os seus danos. Posteriormente, o modelo que caracteriza esta versão TCP deve ser linearizado, e as técnicas de controle ótimo utilizadas para o desenvolvimento do H2-AQM devem ser usadas para derivar um controlador ótimo que garanta a estabilidade não apenas no tamanho da fila, mas também no tamanho da janela, e, assim, o sistema de controle de congestionamento como um todo pode ser aprimorado.

Bibliografia

[1] X. Xiao, "Providing Quality of Service in The Internet," Ph.D. dissertation, Michigan State Universty, USA, 2000.

[2] X. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Network*, vol. 13, no. 2, pp. 8–18, Mar 1999.

[3] J. Nagle, "Congestion control in IP/TCP internetworks," RFC 896, Jan 1984.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," RFC 2702, Sep 1999.

[5] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of Internet Traffic Engineering," RFC 3272, May 2002.

[6] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM '88*, Stanford, CA, aug 1988, pp. 314–329.

[7] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, Apr 1999.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[9] V. Misra, W.-B. Gong, and D. F. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, September 2000, pp. 151–160.

[10] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Trans. Automat. Contr.*, vol. 47, no. 6, pp. 945 –959, Jun 2002.

[11] M. M. A. E. Lima, N. L. S. Fonseca, and J. F. Rezende, "On the performance of TCP loss recovery mechanisms," in *Proceedings of IEEE International Conference on Communications*, Anchorage, Alaska, May 2003, pp. 1812–1816.

[12] M. M. A. E. Lima, J. C. Geromel, and N. L. S. Fonseca, "H2-AQM - an optimal AQM controller," State University of Campinas, Institute of Computing, Campinas, Brazil, Tech. Rep. IC-03-09, April 2003.

[13] M. M. A. E. Lima, N. L. S. Fonseca, and J. C. Geromel, "Eficácia do uso conjunto do mecanismo Limited Transmit com diferentes versões do protocolo TCP," in *Anais do XXII Simpósio Brasileiro de Redes de Computadores*, Gramado-RS, Maio 2004, pp. 395–408.

[14] ——, "Uma avaliação da eficácia do controlador Ótimo H2-AQM para o gerenciamento ativo de filas," in *Anais do XXII Simpósio Brasileiro de Redes de Computadores*, Gramado-RS, Maio 2004, pp. 423–436.

[15] ——, "Um controlador Ótimo para o gerenciamento ativo de filas," in *Anais do 20º Simpósio Brasileiro de Telecomunicações*, Rio de Janeiro-RJ, Out 2003, pp. 471–477.

[16] ——, "Design objetives of optimal active queue management controllers," in *Proceedings of IEEE Global Telecommunications Conference*, Dallas, EUA, Dec 2004.

[17] ——, "On the efficacy of TCP Limited Transmit under active queue management," in *Proceedings of IEEE ICC2004*, Paris, France, Jun 2004, pp. 1038–1043.

[18] ——, "An optimal active queue management controller," in *Proceedings of IEEE ICC2004*, Paris, France, Jun 2004, pp. 2261–2266.

[19] M. M. de A. Lima, N. L. S. da Fonseca, and J. C. Geromel, "Design of optimal active queue management controllers," 2005, submited to The Computer Networks Journal.

[20] J. F. Rezende, M. M. de A. Lima, and N. L. S. da Fonseca, "Mobility over Transport Control Protocol/Internet Protocol (TCP/IP)," in *The Handbook of Ad hoc Wireless Networks*, M. Ilyas, Ed. CRC Press, 2002, pp. 19.1–19.15.

[21] M. M. de A. Lima and N. L. S. da Fonseca, "Controle de tráfego Internet," in *Minicursos SBRC2002*, 2002, pp. 187–249.

[22] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an Internet backbone," in *Proceedings of INET*, Geneva, Switzerland, July 1998.

[23] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns - a view from ames Internet exchange," in *Proceedings of 13th ITC Specialist Seminar on Internet Traffic Measurement and Modelling*, Sep 2000. [Online]. Available: http://www.caida.org/outreach/papers/2000/AIX0005/

[24] M. Fomenkov, K. Keys, D. Moore, , and K. Claffy, "Longitudinal study of Internet traffic in 1998-2003," in *Proceedings of Winter International Symposium on Information and Communication Technologies (WISICT)*, Cancun, Mexico, Jan 2004.

[25] J. Postel, "Transmission control protocol," RFC 793, Sep 1981.

[26] V. Paxson and M. Allman, "Computing TCP's retransmission timer," RFC 2988, November 2000.

[27] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 26,4. ACM Press, Aug 1996.

[28] V. Jacobson. (2000) Modified TCP congestion avoidance algorithm. End2end-interest mailing list. [Online]. Available: ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt

[29] W. Feng, "Improving Internet congestion control and queue management algorithms," Ph.D. dissertation, University of Michigan, USA, 1999.

[30] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 3, pp. 5–21, 1996.

[31] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's Fast Recovery Algorithm," RFC 3782, April 2004.

[32] K. Balakrishnan, V. Padmanabhan, S. Seshan, S. Stemm, and R. Katz, "TCP behavior of a busy Internet server: Analysis and improvements," in *Proceedings of IEEE Infocom*, San Francisco, CA, USA, March 1998, pp. 252–262.

[33] D. Lin and H. T. Kung, "TCP fast recovery strategies: Analysis and improvements," in *Proceedings of IEEE Infocom*, San Francisco, CA, USA, March 1998.

[34] F. Gunnarsson, "Problems and improvements of Internet traffic congestion control," Master's thesis, Linköpings Universitet, Oct. 2000.

[35] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *SIGCOMM*, 1994, pp. 24–35. [Online]. Available: citeseer.nj.nec.com/brakmo94tcp.html

[36] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: Emulation and experiment," in *SIGCOMM*, 1995, pp. 185–205.

[37] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Netw.*, vol. 5, no. 3, pp. 336–350, 1997.

[38] C. Barakat, E. Altman, and W. Dabbous, "On TCP performance in a heterogeneous network: A survey," *IEEE Communication Magazine*, vol. 38, no. 1, pp. 40–46, January 2000.

[39] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proceedings of IEEE Infocom*, Hong Kong, March 2004.

[40] M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, 1997.

[41] T. J. Ott, J. H. B. Kemperman, and M. Mathis, "The stationary behavior of ideal TCP congestion avoidance," in *Proceedings of IEEE INFOCOM*, New York, March 1999.

[42] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 458–472, 1999.

[43] S. Floyd, S. Ratnasamy, and S. Shenker, "Modifying TCP's congestion control for high speeds," May 2002, rough draft. [Online]. Available: http://www.icir.org/floyd/papers/hstcp.pdf

[44] S. Floyd, "Highspeed TCP for large congestion windows," RFC 3649, December 2003.

[45] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, 2003.

[46] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *In Proceedings of IEEE Infocom 2004*, Hong Kong, 03 2004.

[47] J. Chen, F. Paganini, R. Wang, M. Y. Sanadidi, and M. Gerla, "Fluid-flow Analysis of TCP Westwood with RED," in *Proceedings of Globecom*, San Francisco, Ca, November 2003.

[48] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, 1989.

[49] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle, "Linear stability of TCP/RED and a scalable control," *Comput. Networks*, vol. 43, no. 5, pp. 633–647, 2003.

[50] S. H. Low and R. Srikant, "A mathematical framework for designing a low-loss, low-delay Internet," *Journal of Networks and Spacial Economics, special issue on 'Crossovers between transportation planning and telecommunications'*, vol. 4, pp. 75–101, Mar 2004.

[51] C. Jin, D. X. Wei, and S. H. Low, "The case for delay-based congestion control," in *Proc. of IEEE Computer Communication Workshop (CCW)*, Laguna Beach, CA, 10 2003.

[52] H. Choe and S. Low, "Stabilized Vegas," in *Proceedings of IEEE INFOCOM*, vol. 3, San Francisco, CA, April 2003, pp. 2290–2300.

[53] L. A. Grieco and S. Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 25–38, 2004.

[54] P. Dykstra, "Issues impacting gigabit networks: Why don't most users experience high data rates?" August 2000. [Online]. Available: http://sd.wareonearth.com/~phil/issues.html

[55] C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi, and R. Wang, "TCP Westwood: End-to-end congestion control for wired/wireless networks," *Wireless Networks Journal*, vol. 8, no. 5, pp. 467–479, 2002.

[56] T. Hatano, M. Fukuhara, H. Shigeno, and K. Okada, "TCP-friendly SQRT TCP for High Speed Networks," in *Proceedings of APSITT*, November 2003, pp. 455–460.

[57] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM Sigcomm*, Pittsburgh, PA, August 2002, pp. 89–102.

[58] R. N. Shorten and D. J. Leith, "H-TCP: TCP for high-speed and long-distance networks," in *Proc. PFLDnet*, Argonne, 2004.

[59] S. F. M. Mathis, J. Mahdavi and A. Romanow, "TCP selective acknowledgment options," RFC 2018, October 1996.

[60] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An extension to the selective acknowledgment (SACK) Option for TCP," RFC 2883, July 2000.

[61] J. C. Hoe, "Start-up dynamics of TCP's congestion control and avoidance schemes," Master's thesis, Massachusetts Institute of Technology, May 1995.

[62] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, April 1999.

[63] H. B. M. Allman and S. Floyd, "Enhancing TCP's loss recovery using Limited Transmit," RFC 3042, January 2001.

[64] U. Ayesta and K. Avrachenkov, "The effect of the initial window size and Limited Transmit algorithm on the transient behaviour of TCP transfers," in *Proceedings of The 15th ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, Würzburg, Germany, July 2002.

[65] V. project. NS: Network Simulator. [Online]. Available: http://www.isi.edu/nsnam/ns

[66] J. Pahdye and S. Floyd, "On inferring TCP behavior," in *SIGCOMM '01.* New York, NY, USA: ACM Press, 2001, pp. 287–298.

[67] M. Mathis and J. Mahdavi, "Forward acknowledgement: refining TCP congestion control," in *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM Press, 1996, pp. 281–291.

[68] E. Blanton, M. Allman, K. Fall, and L. Wang, "A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP," RFC 3517, April 2003.

[69] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the Internet," RFC 2309, Apr 1998.

[70] M. A. Labrador and S. Banerjee, "Packet dropping policies for ATM and IP networks." *IEEE Communications Surveys and Tutorials*, vol. 2, no. 3, 1999.

[71] S. Floyd. (1997) RED: Discussions of setting parameters. [Online]. Available: http://www.aciri.org/floyd/REDparameters.txt

[72] ——. (1997) RED: Optimum functions for computing the drop probability. [Online]. Available: http://www.aciri.org/REDfunc.txt

[73] ——. (2000) Recommendation on using the gentle variant of RED. [Online]. Available: http://www.aciri.org/floyd/red/gentle.html

[74] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED gateway," in *Proc. IEEE INFOCOM'99*, vol. 3, New York, NY, Mar 1999, pp. 1320–1328.

[75] D. Lin and R. Morris, "Dynamics of random early detection," in *SIGCOMM '97*, Cannes, France, september 1997, pp. 127–137. [Online]. Available: citeseer.nj.nec.com/lin97dynamics.html

[76] S. Floyd, "Congestion control principles," RFC 2914, September 2000.

[77] T. Bonald, M. May, and J.-C. Bolot, "Analytic evaluation of RED performance," in *Proc. IEEE INFOCOM'00*, Tel Aviv, Israel, March 2000, pp. 1415–1424.

[78] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 249–264, 2001.

[79] S. Floyd, "A report on some recent developments in TCP congestion control," *IEEE Communication Magazine*, vol. 39, no. 4, pp. 84–90, 2001.

[80] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. IEEE INFOCOM'00*, Tel Aviv, Israel, March 2000, pp. 1435–1444.

[81] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *IEEE/IFIF-(IWQoS'99)*, London, June 1999, pp. 260–262.

[82] Y. Zhang and L. Qiu, "Understanding the end-to-end performance impact of RED in a heterogeneous environment," Cornell University, Ithaca, NY, USA, Tech. Rep., 2000.

[83] V. Rosolen, O. Bonaventure, and G. Leduc, "A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 3, pp. 23–43, 1999.

[84] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED," 2001, unpublished Technical Report. [Online]. Available: http://www.icir.org/floyd/papers/adaptiveRed.pdf

[85] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "BLUE: A new class of active queue management algorithms," University of Michigan, Tech. Rep. CSE-TR-387-99, April 1999. [Online]. Available: citeseer.nj.nec.com/feng99blue.html

[86] R. Morris, "Scalable TCP congestion control," in *Proceedings of INFOCOM 2000*, Tel-Aviv, Israel, 2000, pp. 1176–1183. [Online]. Available: citeseer.nj.nec.com/article/morris99scalable.html

[87] P. Kuusela, P. Lassila, and J. Virtamo, "Stability of TCP-RED congestion control," in *Proc. ITC-17*, Salvador da Bahia, Brasil, dec 2001, pp. 655–666.

[88] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[89] R. Srikant, *The Mathematics of Internet Congestion Control*, ser. Systems & Control: Foundations & Applications. Birkhauser, 2004.

[90] S. L. Basar and T. Srikant, "Controlling the Internet: A survey and some new results," in *Proceedings of 42nd IEEE Conference on Decision and Control*, vol. 3, 12 2003, pp. 3048– 3057.

[91] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: utility functions, random losses and ECN marks," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 689–702, 2003.

[92] G. Vinnicombe, "On the stability of networks operating TCP-like congestion control," in *Proceedings of the 15th IFAC World Congress on Automatic Control*, 07 2002.

[93] R. Srikant, "Models and methods for analyzing Internet congestion control algorithms," in *In Advances in Communication Control Networks in the series Lecture Notes in Control and Information Sciences (LCNCIS)*, C. Abdallah, J. Chiasson, and S. Tarbouriech, Eds. Springer-Verlag, 2004.

[94] S. H. Low and D. E. Lapsley, "Optimization flow control — I: basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[95] R. R. M. H. Yaiche and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing of elastic connections in broadband networks: Theory and algorithms," *IEEE/ACM Trans. Networking*, pp. 667–678, October 2000.

[96] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking*, vol. 11, no. 4, pp. 525–536, 2003.

[97] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," in *In Proceedings of Conference on Decision and Control*, Orlando, FL, USA, 12 2001, pp. 185–190. [Online]. Available: citeseer.ist.psu.edu/paganini01scalable.html

[98] F. Paganini, Z. Wang, J. C. Doyle, and S. H. Low, "Congestion control for high performance, stability, and fairness in general networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 43–56, 2005.

[99] J. Wen and M. Arcak, "A unifying passivity framework for network flow control," in *Proceedings of IEEE Infocom*, April 2003.

[100] K. B. Kim, A. Tang, and S. H.Low, "A stabilizing AQM based on virtual queue dynamics in supporting TCP with arbitrary delays," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, 2003.

[101] J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, pp. 1969–1985, 1999.

[102] S. Kunniyur and R. Srikant, "A time-scale decomposition approach to adaptive ECN marking," *Special Issue on System and Control Methods in Communication Networks for the IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 882–894, 2002.

[103] S. S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue algorithm for active queue management," *IEEE/ACM Trans. Networking*, no. 4, pp. 286–299, 2004.

[104] A. Lakshmikantha, C. L. Beck, and R. Srikant, "Robustness of real and virtual queue-based active queue management schemes," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 81–93, 2005.

[105] S. Athuraliya and S. Low, "Optimization flow control – II: Implementation," May 2000. [Online]. Available: http://netlab.caltech.edu

[106] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, 2001.

[107] F. Paganini, J. C. Doyle, and S. H. Low, "A control theoretical look at Internet congestion control," in *Lecture Notes in Control and Information Sciences*, L. Giarre' and B. Bamieh, Eds., Springer-Verlag, Berlin, 2003, vol. 289.

[108] K. B. Kim and S. H. Low, "Analysis and design of AQM for stabilizing TCP," California Institute of Technology, Tech. Rep. CSTR:2002.009, 03 2002. [Online]. Available: https://caltechcstr.library.caltech.edu

[109] D. Agrawal and F. Granelli, "Redesigning an active queue management system," in *Proceedings of IEEE Global Telecommunications Conference*, Dallas, EUA, Dec 2004.

[110] K. B. Kim, A. Tang, and S. H.Low, "Design of AQM in supporting TCP based on the well-known AIMD model," in *Proceedings of IEEE Globecom 2003*, San Francisco, California, USA, 12 2003.

[111] J. H. Lee, S. W. Kim, and W. H. Kwon, "Memoryless H_∞ controllers for state delayed systems," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 159 –162, 2002.

[112] P. Park, "A delay-dependent stability criterion for systems with uncertain time-invariant delays," *IEEE Trans. Automat. Contr.*, vol. 44, no. 4, pp. 876–877, 1999.

[113] G. Vinnicombe, "Robust congestion control for the Internet," 2002, unpublished Technical Report, University of Cambridge.

[114] R. Fengyuan, L. Chuang, Y. Xunhe, and X. S. W. Fubao, "A robust active queue management algorithm based on sliding mode variable structure control," in *Proceedings of IEEE INFOCOM 2002*, New York, USA, 06 2002.

[115] Z. Heying, L. Baohong, and D. Wenhua, "Design of a robust active queue management algorithm based on feedback compensation," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2003, pp. 277–285.

[116] P. Yan, Y. Gao, and H. Ozbay, "A variable structure control approach to active queue management for TCP with ECN," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 2, pp. 203–215, 2005.

[117] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM'01*, Anchorage, Alaska, April 2001, pp. 1510–1519.

[118] C. Wang, B. Li, T. Hou, K. Sohraby, and Y. Lin, "LRED: A robust active queue management scheme based on packet loss ratio," in *Proceedings of IEEE Infocom 2004*, 03 2004.

[119] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proc. IEEE INFOCOM'99*, vol. 3, New York, NY, Mar. 1999, pp. 1346–1355.

[120] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management," *Computer Networks*, vol. 36, no. 2-3, pp. 203–235, 2001.

[121] E.-C. Park, H. Lim, K.-J. Park, and C.-H. Choi, "Analysis and design of the virtual rate control algorithm for stabilizing queues in TCP networks," *Comput. Networks*, vol. 44, no. 1, pp. 17–41, 2004.

[122] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. IEEE INFOCOM'01*, Anchorage, Alaska, April 2001, pp. 1726–1734.

[123] H. Lim, K.-J. Park, E.-C. Park, and C.-H. Choi, "Active queue management algorithm with a rate regulator," in *Proceedings of the 15th IFAC World Congress*, Jul 2002, pp. 21–26.

[124] J. Sun, G. Chen, K.-T. Ko, S. Chan, and M. Zukerman, "PD-RED: to improve the performance of RED," *IEEE Communications Letters*, vol. 7, no. 8, pp. 406– 408, Aug 2003.

[125] J. Sun, G. Chen, K.-T. Ko, and M. Z. Sammy Chan, "PD-controller: A new active queue management scheme," in *Proceedings of Globecom 2003*, 12 2003.

[126] Y. Li, K.-T. Ko, G. Chenand, and J. Sun, "Designing a stable and effective PD-control AQM," in *Proceedings of ICARCV 2004*, 12 2004.

[127] X. Deng, S.Yi, G. Kesidis, and C. R. Das, "A control theoretic approach for designing adaptive active queue management schemes," in *Proceedings of IEEE GlOBECOM'03*, San Francisco, CA, dec 2003.

[128] S. Deb and R. Srikant, "Rate-based versus queue-based models of congestion control," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 246–257, 2004.

[129] C. Long, B. Zhao, X. Guan, and J. Yang, "The Yellow active queue management algorithm," *Comput. Netw. ISDN Syst.*, vol. 47, no. 4, pp. 525–550, 2005.

[130] K. B. Kim and S. H. Low, "Cost of AQM in stabilizing TCP," California Institute of Technology, Tech. Rep. CSTR:2002.008, 07 2002.

[131] L. Ying, G. Dullerud, and R. Srikant, "Global stability of Internet congestion controllers with heterogeneous delays," in *Proceedings of American Control Conference ACC2004*, 06 2004.

[132] S. Floyd, "TCP and Explicit Congestion Notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, pp. 8–23, 1994.

[133] K. K. Ramakrishanan and S. Floyd, "A proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481, January 1999.

[134] K. K. Ramakrishanan, S. Floyd, and D. Black, "The addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, September 2001.

[135] J. H. Salim and U. Ahmed, "Performance evaluation of Explicit Congestion Notification (ECN) in IP Networks," RFC 2884, July 2000.

[136] K. Ogata, *Engenharia de Controle Moderno*, 3rd ed. LTC - Livros Técnicos e Ciêntíficos S.A., 2000.

[137] A. S. e Silva, "Modelagem, análise e síntese de sistemas de controle," 2004, notas de Aula da disciplina de Sistemas de Controle. http://www.labspot.ufsc.br/ aguinald/ensino/eel7063/controle.pdf. [Online]. Available: http://www.labspot.ufsc.br/˜aguinald/ensino/eel7063/controle.pdf

[138] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*. Macmillan Coll Div, 1992.

[139] D. G. Luenberger, *Introduction to Dynamic Systems: Theory, Models and Applications*. Stanford University: John Wiley & Sons, 1979.

[140] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Addison-Wesley Publishing Company, 1994.

[141] M. D. Lemmon, "Lectures in robust optimal control," 2004.

[142] S. Boyd, L. Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, ser. SIAM studies in applied mathematics. Philadelphia: SIAM, 1994, vol. 15.

[143] M. C. Oliveira and J. C. Geromel, "Synthesis of Non-Rational controllers for linear delay systems," *Automatica*, vol. 40, no. 2, pp. 171–188, Feb 2004.

[144] D. P. F. M. C. Oliveira and J. C. Geromel, *LMI Solver User's Guide*. [Online]. Available: http://www.dt.fee.unicamp.br/~mauricio/lmisol/userguide.ps.gz

[145] S. Floyd, "Metrics for the evaluation of congestion control mechanisms," IETF Draft draft-floyd-transport-metrics-00.txt, May 2005.

[146] K. V. Cardoso and J. F. de Rezende, "HTTP traffic modeling: Development and application," in *Proceedings of ITS2002*, Natal Brazil, Sep 2002.

[147] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web client access patterns: Characteristics and caching implications," Boston University - Computer Science, Boston, MA, Tech. Rep. BUCS-TR-1998-023, 4 1998.

[148] Papers by CAIDA. [Online]. Available: http://www.caida.org/outreach/papers/

[149] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 362–373, 1998.