# Eficácia do uso conjunto do mecanismo *Limited Transmit* com diferentes versões do protocolo TCP

**Michele M. de A. E. Lima**[1*], **Nelson L. S. da Fonseca**[2] , **José C. Geromel**[3]

[1]Colegiado de Informática
Universidade Estadual do Oeste do Paraná - UNIOESTE

[2]Instituto de Computação
Universidade Estadual de Campinas - UNICAMP

[3]Faculdade de Engenharia Elétrica e Computação
Universidade Estadual de Campinas - UNICAMP

michele@unioeste.br, nfonseca@ic.unicamp.br, geromel@dsce.fee.unicamp.br

***Abstract.*** *This paper evaluates the use of Limited Transmit in conjunction with TCP Reno, TCP NewReno and TCP SACK under an optimal controller for both FTP and Web traffic. Moreover, a novel optimal controller that includes the timeout mechanism in TCP model used is presented.*

***Resumo.*** *Este artigo avalia o uso em conjunto do mecanismo* Limited Transmit *com as versões do protocolo TCP* Transmission Control Protocol*: TCP Reno, TCP NewReno e TCP SACK, quando é utilizada uma política de AQM baseada em controle ótimo. São utilizados na avaliação tráfego Web e FTP. Além disto, um novo controlador ótimo derivado a partir de equações diferenciais que incluem o mecanismo de temporização é apresentado.*

## 1. Introdução

O protocolo *Transmission Control Protocol* (TCP) ajusta 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. Se três ou mais reconhecimentos forem recebidos para o mesmo segmento, o segmento em questão é considerado perdido e o tamanho da janela de transmissão é reduzido à metade. Entretanto, quando uma perda é detectada pela expiração do intervalo de temporização (*timeout*), a janela de transmissão é reduzida drasticamente para um segmento e o TCP emissor entra na fase de *Slow Start* [1].

Quando o congestionamento é intenso, rajadas de perdas de pacotes podem acontecer, aumentando a probabilidade de ocorrência d*timeouts* e conseqüentemente degradando o desempenho do TCP. O TCP Reno utiliza outros dois algoritmos: *Fast Retransmit* e *Fast Recovery*, para se recuperar de perdas de pacotes. Entretanto, a recuperação não é eficiente quando existem múltiplas perdas numa mesma janela. Medidas obtidas a partir de um servidor Web congestionado indicam que aproximadamente 56% das retransmissões de segmentos ocorrem apenas após um expiração do intervalo de temporização, enquanto que os outros 44% ocorrem depois da ativação do algoritmo de *Fast Retransmit* [2].

Diversas variações do protocolo TCP Reno foram propostos para aprimorar o seu mecanismo de recuperação de perdas. Congestionamento intenso e longos períodos de perda impactam diferentemente o desempenho destas variações. Em especial, o mecanismo *Limited Transmit* foi proposto [3] para transmissões que utilizam um tamanho de janela pequeno. O objetivo deste mecanismo é aumentar

*Este trabalho é suportado em parte pelo CNPQ

o número de reconhecimentos enviados pelo receptor, e conseqüentemente possibilitar a ativação do algoritmo de *Fast Retransmit*.

A notificação de congestionamento é feita no TCP através do descarte de segmentos nos roteadores congestionados. O Gerenciamento Ativo de Filas (*Active Queue Management - AQM*) foi introduzido a fim de se obter o gerenciamento mais eficiente nas filas dos roteadores. A idéia central de AQM é a notificação antecipada da existência de congestionamento incipiente através da marcação/descarte de pacotes, de forma a permitir que os emissores TCP possam reduzir sua taxa de transmissão antes que as filas fiquem cheias, evitando assim, a degradação do desempenho do TCP.

O algoritmo *Random Early Detection, RED* [4], é o algoritmo de AQM recomendado pelo IETF para a Internet. RED estima o tamanho médio da fila, e o compara com dois limiares. O valor da probabilidade de marcação/descarte depende em qual das três regiões, definidas pelos dois limiares, o valor do tamanho estimado da fila encontra-se. Determinar corretamente os parâmetros de RED é um desafio. Apesar de vários estudos baseados em heurísticas e simulações terem sido realizados com o intuito de superar esta dificuldade, tais estudos não garantem que um ponto de equilíbrio seja atingido, nem tampouco garantem a estabilidade do tamanho da fila.

Políticas de AQM baseadas em Teoria do Controle tem sido definidas para superar a incerteza em se determinar corretamente os parâmetros de RED, como também para projetar e desenvolver políticas de AQM que garantam a estabilidade em torno de um ponto de equilíbrio. Nestas políticas, os controladores são responsáveis por determinar qual a probabilidade de descarte/marcação adequada que estabilize o tamanho da fila independentemente das variações das condições da rede.

Em [5] foi introduzido o controlador H2-AQM, um controlador ótimo para o gerenciamento ativo de filas. H2-AQM foi projetado utilizando um modelo simplificado da dinâmica do comportamento do TCP. A síntese do controlador usa uma abordagem não-racional, na qual a estabilidade e os objetivos de desempenho do sistema são completamente expressos e solucionados através de desigualdades matriciais lineares ou LMIs (*Linear Matrix Inequalities*). Apesar de ter sido utilizada uma abordagem não racional, o controlador obtido foi racional. A diferença entre H2-AQM e o H2-TAQM, apresentado neste artigo, é que o projeto do H2-TAQM utiliza um modelo extendido da dinâmica do TCP [6], que inclui o seu mecanismo de temporização.

Em [7], 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. A contribuição principal do presente trabalho é, portanto, avaliar a eficácia do mecanismo *Limited Transmit* quando é utilizada uma política de gerenciamento ativo de filas baseada em controle ótimo.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta um breve resumo das diferentes versões do TCP apresentadas pelo IETF para aprimorar o mecanismo de recuperação de perdas do TCP Reno. A Seção 3 apresenta o modelo do TCP utilizado no projeto do H2-TAQM. A Seção 4 descreve o desenvolvimento do projeto do H2-TAQM. Na seção 5, os resultados numéricos são apresentados. Finalmente, na seção 6, as conclusões são delineadas.

## 2. Extensões do Protocolo TCP

Esta seção apresenta uma breve descrição das propostas do IETF apresentadas para superar as limitações do TCP Reno em se recuperar de forma efetiva quando ocorrem múltiplas perdas. 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.

### 2.1. TCP Sack

A falta de informações sobre quais pacotes foram perdidos é uma das causas da incapacidade do TCP Reno de se recuperar eficientemente, quando ocorrem múltiplas perdas em uma mesma janela [8]. O mecanismo de reconhecimento seletivo do TCP SACK (*Selective ACKnowledgment*) foi proposto para amenizar tal deficiência [9]. No TCP SACK, o TCP receptor informa ao TCP emissor quais pacotes foram recebidos corretamente, permitindo assim ao TCP emissor implementar um mecanismo de retransmissão seletiva de forma a retransmitir apenas os pacotes extraviados. Uma vez que nem todos os sistemas implementam esta extensão TCP, a sua efetiva utilização deve ser negociada no início do estabelecimento da conexão.

SACK utiliza duas opções TCP. A primeira opção, *SACK-permitted*, é enviada em um pacote SYN, como um indicativo da disponibilidade em se utilizar SACK. A segunda opção, *SACK*, é utilizada pelo TCP receptor para enviar informações extra de reconhecimento para o TCP emissor.

Quando o emissor recebe um reconhecimento que contém a opção *SACK*, ele utiliza as informações nela contidas para ajudar na tomada de decisões futura sobre a retransmissão de um segmento. O TCP emissor deve acrescentar um *flag*, *SACKed*, a cada um dos segmentos armazenados em sua fila de segmentos transmitidos e ainda não reconhecidos. Quando um reconhecimento com a opção *SACK* é recebido, o emissor vai ligar os *flags* de todos os segmentos que foram seletivamente reconhecidos pela opção *SACK*. Os segmentos candidatos a serem retransmitidos são os que estiverem com o *flag* desligado e que sejam menor que o maior número de seqüência do segmento que esteja com o *flag* marcado. Se o intervalo de temporização expirar e ainda restarem segmentos na fila do emissor com os *flags* ligados, o TCP emissor deve então desligá-los, e retransmitir todos os segmentos, a partir do primeiro segmento da borda esquerda da janela de transmissão.

### 2.2. TCP NewReno

O TCP emissor só tem conhecimento sobre o correto recebimento de um pacote quando um reconhecimento é recebido. Se este reconhecimento não confirma o recebimento de todos os pacotes até então enviados, o TCP emissor não tem como inferir quantos segmentos foram perdidos nem tampouco saber quais pacotes devem ser retransmitidos [8].

Após a retransmissão de um pacote, um reconhecimento recebido pode ser um reconhecimento total, ou pode ser apenas um reconhecimento parcial, ou seja, este reconhecimento pode estar confirmando o recebimento de alguns dos pacotes enviados mas não todos. O recebimento de um reconhecimento parcial é uma forte indicação que mais de um pacote foi perdido e que o primeiro pacote não reconhecido deve ser retransmitido como forma de recuperar eficientemente de uma rajada de perdas, evitando, assim, a necessidade de se esperar pela expiração do temporizador, (*timeout*) [8].

No TCP NewReno, quando um pacote é retransmitido, o valor do maior número de seqüência transmitido deve ser armazenado em uma variável denominada $recover$. Quando um reconhecimento é recebido, é verificado se ele é um reconhecimento total, ou seja se ele reconhece todos os pacotes enviados, incluindo o número de segmento armazenado na variável $recover$, caso contrário é um reconhecimento parcial. Caso seja parcial, o pacote com o menor número de seqüência, que ainda não foi reconhecido é retransmitido e a janela de congestionamento deve ser reduzida do número de novos pacotes que foram reconhecidos. O valor da janela de congestionamento deve ser acrescentado de um segmento, e um novo segmento é transmitido caso este novo valor permita.

### 2.3. 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 [3], 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 [3].

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 [10].

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* [3]. 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 deve ser 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.

## 3. Modelo Dinâmico para o Comportamento do TCP

Um modelo dinâmico para o comportamento do TCP baseado em análises de fluidos e equações diferenciais estocásticas que inclui o mecanismo de temporização é apresentado em [6]:

$$\begin{aligned}
\dot{W}(t) &= \frac{1}{R(t)} - (1 - Q(W(t))) \cdot \frac{W(t)}{2} \cdot \frac{W(t - R(t))}{R(t - R(t))} \cdot p(t - R(t)) \\
&\quad + (1 - W(t)) \cdot Q(W(t)) \cdot \frac{W(t - R(t))}{R(t - R(t))} \cdot p(t - R(t)); \qquad (1) \\
\dot{q}(t) &= -C(t) + \frac{N(t)}{R(t)} \cdot W(t) + \omega_q(t); \qquad (2) \\
R(t) &= \frac{q(t)}{C(t)} + T_p; \qquad (3) \\
Q(W(t)) &= min\left(1, \frac{3}{W(t)}\right); \qquad (4)
\end{aligned}$$

onde: $W(t)$ é a média do tamanho da janela TCP em segmentos; $q(t)$ é o tamanho da fila em segmentos; $\omega_q(t)$ 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 segmentos/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 marcação/descarte de segmentos; $Q(W(t))$ é a probabilidade de que uma perda foi detectada pela ocorrência da expiração do intervalo de temporização.

A equação (1) descreve a dinâmica da janela TCP. O primeiro termo modela o crescimento aditivo da janela, enquanto que o segundo termo modela o decréscimo multiplicativo. A equação (2) modela a dinâmica da fila como a diferença entre a taxa de chegada, $NW/R + \omega_q(t)$, e a capacidade do enlace, $C$.

O ponto de equilíbrio ($W_0$, $q_0$, $p_0$) do sistema é obtido através da solução de $\dot{W}(t) = 0$ e $\dot{q}(t) = 0$, resultando em:

$$\begin{aligned}
W_0 &= \frac{-3 + \sqrt{\frac{8+33p_0}{p_0}}}{2} = \frac{R_0 C_0}{N_0}; \qquad (5) \\
q_0 &= N_0 W_0 - C_0 T_p = C_0 (R_0 - T_p); \qquad (6)
\end{aligned}$$

$$p_0 = \frac{2N_0^2}{(R_0C_0)^2 + 3N_0R_0C_0 - 6N_0^2}; \quad (7)$$

onde: $N(t) \equiv N_0$, $C(t) \equiv C_0$ e $R_0 = \frac{q_0}{C_0} + T_p$.

As equações (1) e (2) são linearizadas em torno do ponto de equilíbrio ($W_0$, $q_0$, $p_0$) resultando em:

$$\begin{aligned} \dot{x}_1(t) &= -\frac{-N_0R_0^2C^2 - 6N_0^3}{R_0^4C^3 + 3N_0R_0^3C^2 - 6N_0^2R_0^2C_0}x_1(t) - \frac{N_0}{R_0^2C}x_1(t-R_0) \\ &\quad -\frac{1}{R_0^2C_0}(x_2(t) - x_2(t-R_0)) + \frac{6N_0^2 - R_0^2C^2 - 3N_0R_0C}{2N_0^2R_0}u(t-R_0); \\ \dot{x}_2(t) &= \frac{N_0}{R_0}x_1(t) - \frac{1}{R_0}x_2(t) + \omega_q(t); \end{aligned} \quad (8)$$

onde: $x_1(t) \doteq W(t) - W_0$; $x_2(t) \doteq q(t) - q_0$; $u(t) \doteq p(t) - p_0$;

## 4. Projeto do Controlador Ótimo para o Gerenciamento Ativo de Filas

Nesta seção, uma abordagem não racional é usada para derivar o controlador ótimo $H_2$ para o sistema (8). A síntese do controlador é baseada nos resultados apresentados em [11], onde os projetos de controladores para sistemas lineares com atraso são expressos e solucionados como desigualdades matriciais lineares, LMIs (*Linear Matrix Inequalities*).

O sistema linear apresentado em (8) pode ser expresso no estado de espaço pelas seguintes equações que descrevem um sistema linear com atraso contínuo no tempo:

$$\begin{aligned} \dot{x}(t) &= A_0x(t) + A_1x(t-R_0) + B_ww(t) + B_uu(t); \\ z(t) &= C_zx(t) + D_{zu}u(t); \\ y(t) &= C_yx(t-R_0) + D_{yw}w(t); \end{aligned} \quad (9)$$

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. O atraso presente em $u(t - R_0)$ em (8) é apresentado em (9) na saída $y(t)$. Agora, considere que o sistema descrito em (9) esteja conectado com o seguinte controlador:

$$C(s) = (\hat{C}_0 + \hat{C}_1e^{-sR_0})(sI - \hat{A}_0 - \hat{A}_1e^{-sR_0})^{-1}\hat{B} + \hat{D}; \quad (10)$$

O controlador (10) foi escolhido de forma a reproduzir a estrutura da planta do sistema (9). O objetivo é determinar as matrizes do controlador (10) que estabilizam (9) e minimizam certas medidas na saída de referência $z(t)$. Para atingir estes objetivos de projeto é necessário, então, definir quais são os objetivos de desempenho desejados para a saída $z(t)$ e o que deve ser medido na saída obtida do sistema, $y(t)$. Isto implica que os objetivos de desempenho para a política de AQM projetada devem estar representados no controlador $C_{TH2}(s)$. O valor ideal da probabilidade de descarte/marcação deve tentar ao mesmo tempo maximizar as taxas de transmissão e minimizar o tamanho da fila sujeito às condições da rede, de forma a evitar perdas de segmentos desnecessárias. Para atingir tais objetivos, as matrizes do sistema (9), são definidas como:

$$A_0 = \begin{bmatrix} -\frac{N_0}{R_0^2 C_0} & -\frac{1}{R_0^2 C_0} \\ \frac{N_0}{R_0} & -\frac{1}{R_0} \end{bmatrix}, \quad A_1 = \begin{bmatrix} -\frac{N_0}{R_0^2 C} & \frac{1}{R_0^2 C_0} \\ 0 & 0 \end{bmatrix}, \quad B_w = \begin{bmatrix} 0 & 0 \\ 0.2C_0 & 0 \end{bmatrix},$$

$$B_u = \begin{bmatrix} -\frac{R_0 C_0^2}{2N_0^2} \\ 0 \end{bmatrix}, \quad C_z = \begin{bmatrix} 0 & 1 \\ \frac{N_0}{R_0} & -\frac{1}{R_0} \\ 0 & 0 \end{bmatrix}, \quad D_{zu} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix},$$

$$D_{yw} = \begin{bmatrix} 0 & 0.02C_0 \end{bmatrix}, \quad C_y = \begin{bmatrix} 0 & 1 \end{bmatrix};$$

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, enquanto $A_1$ representa os termos com atraso. A primeira coluna destas matrizes corresponde a $\frac{\partial f_1}{\partial x_i}$, e a segunda corresponde $\frac{\partial f_2}{\partial x_i}$, com $i = 1, 2$. $B_u$ contém $\frac{\partial f_1}{\partial u}$ na primeira coluna, e $\frac{\partial f_2}{\partial u}$ na segunda coluna, onde $f_1$ é o lado direito da Equação (1) e $f_2$ o lado direito da Equação(2), $x_1 = W$, $x_2 = q$ e $u = p$.

A quantidade de ruído existente no sistema, gerado pelos fluxos UDP, é controlada pela matriz $B_w$. O valor considerado permite que os fluxos UDP utilizem até 20% da capacidade do enlace. Este valor é uma margem de tolerância satisfatória, dado que 95% dos bytes transmitidos na Internet são gerados pelo TCP [12].

$C_z$ representa o objetivo do projeto, que é tentar obter o equilíbrio entre maximizar as taxas de transmissão e ao mesmo tempo minimizar o tamanho da fila, como também minimizar a sua variação. A primeira linha está relacionada com o tamanho da fila e a segunda com a sua variação. $D_{zu}$, pondera o valor da probabilidade de descarte/marcação na saída. Diferentes valores foram verificados para esta ponderação, variando de $0,3$ a $0,9$. Os resultados obtidos foram bastante similares, sendo $0,5$ o valor adotado.

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

Depois de definidos os objetivos de desempenho, o próximo passo é conectar o sistema (9) com o controlador (10). Seja $\bar{x}(t)$ o vetor de estado aumentado que contém o vetor de estado $x(t)$ e o vetor de estado do controlador $\hat{x}(t)$:

$$\bar{x}(t) = \begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix}; \tag{11}$$

A conexão do sistema (9) com o controlador(10) resulta no sistema linear com atraso:

$$\begin{aligned} \dot{\bar{x}}(t) &= \mathcal{A}_0 \bar{x}(t) + \mathcal{A}_1 \bar{x}(t - R_0) + \mathcal{B} w(t); \\ z(t) &= \mathcal{C}_0 \bar{x}(t) + \mathcal{C}_1 \bar{x}(t - R_0) + \mathcal{D} w(t); \end{aligned} \tag{12}$$

onde:

$$\mathcal{A}_0 = \begin{bmatrix} A_0 & B_u \hat{C}_0 \\ 0 & \hat{A}_0 \end{bmatrix}, \quad \mathcal{A}_1 = \begin{bmatrix} A_1 + B_u \hat{D} C_y & B_u \hat{C}_1 \\ \hat{B} C_y & \hat{A}_1 \end{bmatrix},$$

$$\mathcal{B} = \begin{bmatrix} B_w + B_u \hat{D} D_{yw} \\ \hat{B} D_{yw} \end{bmatrix}, \quad \mathcal{C}_0 = \begin{bmatrix} C_z & D_{zu} \hat{C}_0 \end{bmatrix},$$

$$\mathcal{C}_1 \;=\; \left[\; D_{zu}\hat{D}C_y \quad D_{zu}\hat{C}_1 \;\right], \quad \mathcal{D} \;=\; D_{zu}\hat{D}D_{yw};$$

Para garantir a estabilidade do sistema (12), o Teorema 4-b apresentado em [11] é usado. Este teorema especifica que um sistema como (12) é assintoticamente estável e $\|H_{wz}(s)\|_2^2 < \gamma$, se existem matrizes simétricas e definidas positivas $W$, $Y_0$ e $X_j$, e matrizes $F$, $R$, $L_j$ e $Q_j$, com $j = 0, 1$,

$$\left[\begin{array}{lll} \mathbf{A}_0 + \mathbf{A}_0^T + X_1 & (\bullet)^T & (\bullet)^T \\ \mathbf{A}_1^T & -X_1 & (\bullet)^T \\ \mathbf{C}_0 & \mathbf{C}_1 & -I \end{array}\right] < 0 \tag{13}$$

$$\left[\begin{array}{ll} W & (\bullet)^T \\ \mathbf{B} & \mathbf{P}_0 \end{array}\right] > 0, \quad trace(W) < \gamma \tag{14}$$

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

$$\mathbf{A}_0 \;=\; \left[\begin{array}{ll} A_0X_0 + B_uL_0 & A_0 \\ Q_0 & Y_0A_0 \end{array}\right], \quad \mathbf{A}_1 \;=\; \left[\begin{array}{ll} A_1X_0 + B_uL_1 & A_1 \\ Q_1 & Y_0A_1 + FC_y \end{array}\right],$$

$$\mathbf{B} \;=\; \left[\begin{array}{c} B_w \\ Y_0B_w + FD_{yw} \end{array}\right], \qquad \mathbf{P}_0 \;=\; \left[\begin{array}{ll} X_0 & I \\ I & Y_0 \end{array}\right],$$

$$\mathbf{C}_0 \;=\; \left[\; C_zX_0 + D_{zu}L_0 \quad C_z \;\right], \qquad \mathbf{C}_1 \;=\; \left[\; D_{zu}L_1 \quad 0 \;\right];$$

Este problema de programação convexa foi resolvido numericamente utilizando a rotina LMISol [13]. Os parâmetros de rede utilizados foram $N_0 = R_0C_0$ fluxos TCP, $R_0 = 0,246$ segundos e $C = 3750$ segmentos/segundos, o que corresponde a um enlace de 15 Mb/s com segmentos com tamanho médio de 500 *bytes*. Os valores para estes parâmetros foram escolhidos de forma a obter um ponto de equilíbrio com um valor extremamente reduzido para a probabilidade de perda $p_0$ que permita tamanhos de janelas grandes. Como foi obtida uma solução factível para os LMIs ((12) e (13), pode-se afirmar que o sistema (11) é estável.

Obtida uma solução factível, o próximo passo é determinar os parâmetros do controlador (9). Primeiramente, matrizes arbitrárias $U_0$ e $V_0$ devem ser escolhidas de forma a satisfazer $V_0U_0 = I - Y_0X_0$. 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:

$$\left[\begin{array}{lll} \hat{A}_0 & \hat{A}_1 & \hat{B} \\ \hat{C}_0 & \hat{C}_1 & \hat{D} \end{array}\right] = \mathcal{K}.\mathcal{M}.\mathcal{N}. \tag{15}$$

onde:

$$\mathcal{K} \;=\; \left[\begin{array}{ll} V_0^{-1} & V_0^{-1}Y_0B_u \\ 0 & I \end{array}\right]; \quad \mathcal{M} \;=\; \left[\begin{array}{lll} Q_0 - Y_0A_0X_0 & Q_1 - Y_0A_1X_0 & F \\ L_0 & L_1 & 0 \end{array}\right];$$

$$\mathcal{N} \;=\; \left[\begin{array}{lll} U_0^{-1} & 0 & 0 \\ 0 & U_0^{-1} & 0 \\ -C_{y0}X_0U_0^{-1} & -C_{y1}X_0U_0^{-1} & I \end{array}\right];$$

Na solução obtida, as matrizes $\hat{A}_1$ e $\hat{C}_1$ são aproximadamente zero, sendo portanto ignoradas. Como resultado tem-se o cancelamento do termo de atraso do sistema, e o controlador torna-se racional.

H2-TAQM-ProbabilityFunction()

$p_0 \Leftarrow \frac{2N_0^2}{(R_0C_0)^2+3N_0R_0C_0-6N_0^2}$;

$p \Leftarrow a*(q-2*q_0+q_{old})-b*p_{old}+p_0*(1+b)$;

$p_{old} \Leftarrow p$;

$q_{old} \Leftarrow q$;

**Figure 1: Algoritmo para o cálculo da probabilidade de descarte/marcação em H2-TAQM**

O cancelamento do termo de atraso, quando possível, é a estratégia global ótima para solucionar o problema de minimização da norma $H_2$ [11]. O controlador $C_{TH2}(s)$ representado por sua função de transferências no domínio da freqüência é dado por:

$$C_{TH2}(s) = \frac{3.62e^{-6}s + 1.582e^{-5}}{s^2 + 1601s + 7022}; \quad (16)$$

Para que $C_{TH2}$ possa ser implementado, é necessário que seja escolhida uma freqüência de amostragem $f_s$, para que uma representação no domínio-$z$ possa ser obtida. A freqüência escolhida foi $f_s = 375Hz$, o que representa 10% do valor da capacidade do enlace $C$. O controlador $C_{TH2}$ no domínio-$z$ é dado por:

$$C_{TH2}(z) = \frac{a(z+1)}{z+b} = \frac{1.5424^{-9}(z+1)}{z+0.3608}; \quad (17)$$

Observe que $C_{H2}(s)$, a função de transferência apresentada em (15) é de primeira ordem e não de segunda ordem como esperado, dado que com a fatoração da função de transferência, o termo $(z-0.9885)$ presente tanto no numerador quanto no denominador, foi cancelado.

A função de transferência entre $\delta p = p - p_0$ e $\delta q = q - q_0$, apresentada em (17), pode ser convertida em uma equação de diferenças no tempo discreto $kT$, onde $T = \frac{1}{f_s}$:

$$\delta p(kT) = a[\delta q(kT) + \delta q((k-1)T)] + b\delta p((k-1)T); \quad (18)$$

O algoritmo para o cálculo da probabilidade de descarte/marcação em H2-TAQM é bastante simples, e é executado a cada intervalo de amostragem $1/f_s$ (Figura 1). Inicialmente, o valor de $p_0$ é calculado baseado nos parâmetros $N_0$, $C_0$ e $R_0$. Depois, o valor da probabilidade é calculado baseado em (17). O algoritmo necessita de duas variáveis auxiliares: $q_{old}$ e $p_{old}$, que armazenam os valores de $q$ e $p$ respectivamente no intervalo anterior.

## 5. Resultados Numéricos

Para avaliar o impacto da probabilidade de marcação/descarte no desempenho do mecanismo *Limited Transmit* utilizado em conjunto com diferentes versões TCP, o algoritmo apresentado na Figura 1 foi implementado no simulador de redes NS na versão 2.26 [14]. A freqüência de amostragem utilizada para derivar o controlador digital foi de 375 Hz, gerando os valores de $1.5424^{-9}$ e $0.3608$ 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 $max_p = 0.1$.

A topologia, a capacidade dos enlaces, como também o tempo de propagação para cada enlace são apresentados na Figura 2. 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.

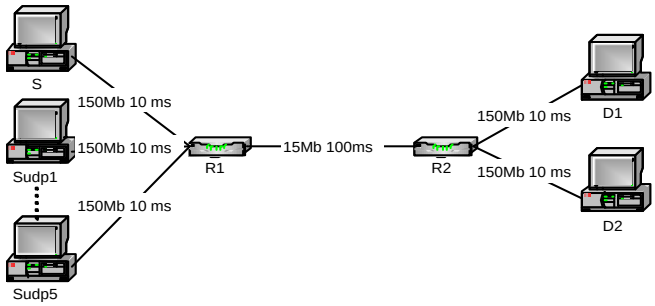



**Figure 2: Topologia utilizada nas simulações**

Um gerador de tráfego, denominado, TrafficGen [15], foi empregado para gerar cargas específicas para diferentes tipos de tráfego. A carga varia de $0.4$ à $0.9$ de forma a verificar a robustez do controlador sob diferentes cargas. Uma distribuição híbrida Lognormal/Pareto foi utilizada para gerar o tráfego WEB. O corpo da distribuição, correspondente a uma área de $0.88$, é modelada por uma distribuição Lognormal com média de $7247$ bytes, enquanto que a cauda é modelada por uma distribuição Pareto com média de $10558$ bytes [16]. O tráfego FTP é gerado utilizando-se uma distribuição exponencial com média de $512$ KBytes. Tanto o tráfego WEB como o tráfego FTP são gerados do nó $S_1$ para o nó $D_1$.

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 valor é o valor médio do tamanho dos pacotes trafegando na Internet. O site [17] 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.

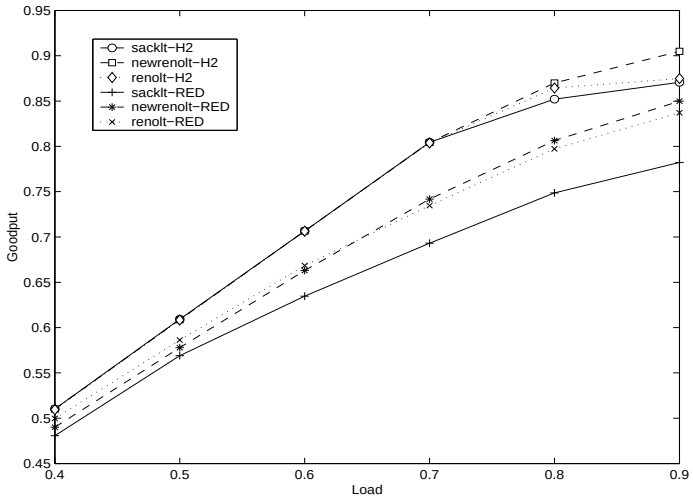
Fluxos não adaptativos ou que não respondem ao congestionamento, tais como fluxos do tipo CBR/UDP, foram incluídos para gerar tráfego de interferência. Tais fluxos podem representar até 20% da capacidade do enlace em um dado momento da simulação. Estes fluxos foram gerados e finalizados em diferentes intervalos de tempo dos nós $S_{udp_i}$ para o nó $D_2$.

Para cada política de AQM, o TCP Reno, TCP NewReno, e TCP SACK foram comparados sob diferentes condições de carga. O uso em conjunto destes protocolos com o *Limited Transmit*, denotados por *renolt*, *newrenolt* e *sacklt*, foram também simulados para o mesmo cenário. Em [7] foi mostrado que o uso em conjunto destas versões TCP com *Limited Transmit* melhora o desempenho do TCP. Desta forma, apenas os resultados obtidos pelos uso conjunto destes protocolos serão apresentados.

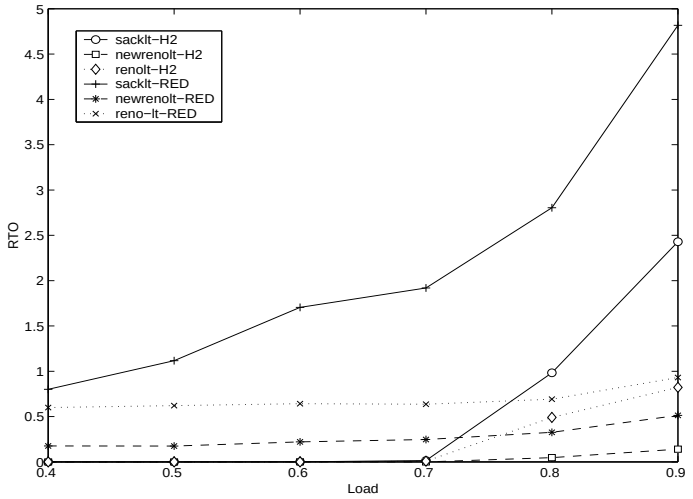
O *goodput*, a taxa de perda e o número médio de ocorrências de retransmissões devido a expiração do intervalo de temporização (RTO´s) por conexão como função da carga da rede são apresentados nas Figuras 3 e 4.

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, dado que são visualmente desprezíveis devido a pequena largura do intervalo.

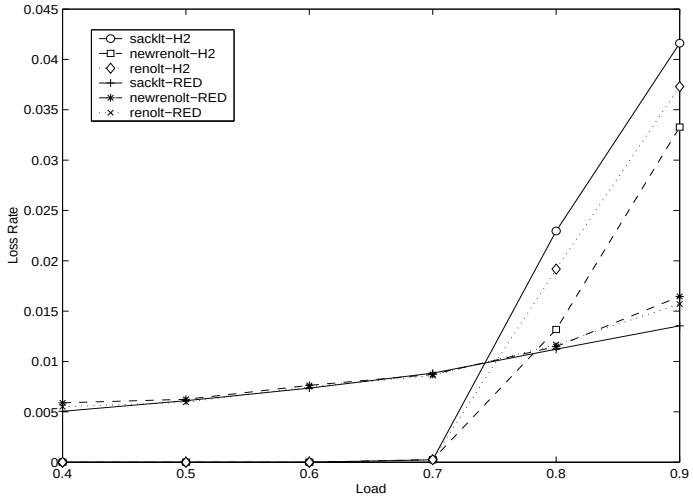
Pode ser observado na Figura 3 que para cargas menores que 0.8, não existe uma diferença significativa para o *goodput* obtido para as diferentes versões TCP, utilizando H2-TAQM. Para cargas iguais a 0.9, *newrenolt* supera as demais versões TCP em relação ao *goodput* obtido em cerca de 5%. O *goodput* obtido para as diferentes versões TCP utilizando-se RED como política de AQM é menor do que o obtido utilizando-se H2-TAQM. A diferença é de no mínimo 6%, 4% e 2% e de no máximo 16%,




Goodput x Carga

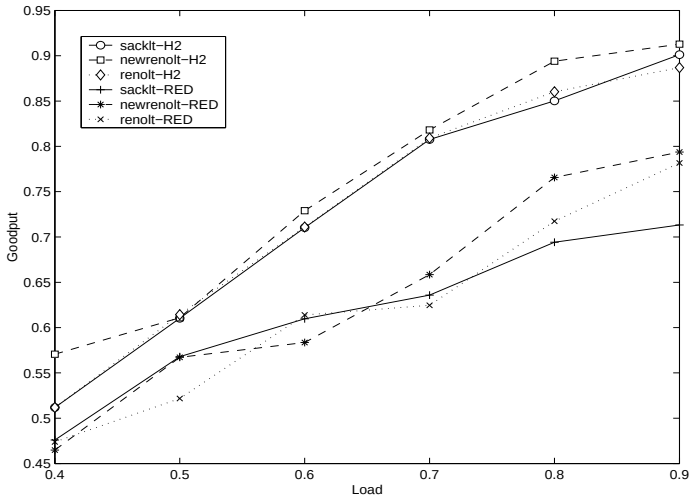



Número médio de RTO´s por conexão x Carga

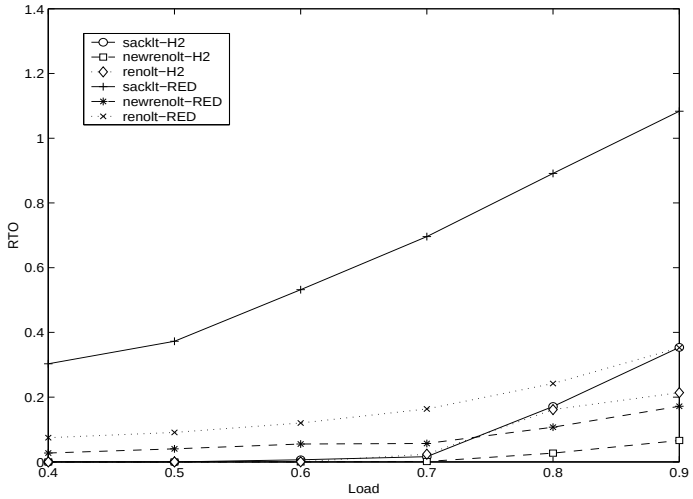



Taxa de perda x Carga

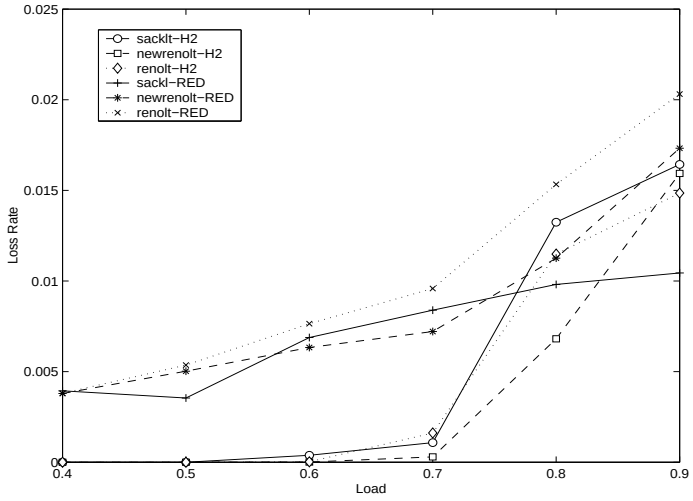
**Figure 3: Comparação entre H2-TAQM e RED para Tráfego FTP**




Goodput x Carga




Número médio de timeout por conexão x Carga




Taxa de perda x Carga

**Figure 4: Comparação entre H2-TAQM e RED para Tráfego WEB**

8% e 9% para *sacklt*, *newrenolt* e *renolt*, respectivamente. Quando RED é utilizado, o desempenho difere entre as diferentes versões TCP. Para cargas de 0.9, o *goodput* obtido por *newrenolt* é 9% maior que o obtido por *sacklt*.

Pode-se verificar que quando H2-TAQM é utilizado, todas as versões TCP reduzem a ocorrência de retransmissões devido à expiração do intervalo de temporização à zero para cargas entre 0.4 e 0.7. Mesmo para cargas maiores, o número de ocorrência de retransmissões devido à expiração do intervalo de temporização é reduzido quando são comparadas aos valores obtidos para as mesmas versões TCP que utilizam RED.

Apesar do fato de que para cargas maiores que 0.7, a taxa de perda apresentada pelas versões TCP que utilizam H2-TAQM serem maiores que para as mesmas versões que utilizam RED, os valores de *goodput* obtido quando H2-TAQM é utilizada são maiores. Isto pode ser explicado pelo fato de que H2-TAQM reduz a ocorrência de retransmissões devido à expiração do intervalo de temporização, evitando, desta forma, a degradação da eficiência do TCP depois da ocorrência deste evento. A redução da ocorrência de retransmissões devido à expiração do intervalo de temporização para as diferentes versões TCP utilizando-se H2-TAQM quando comparada com estas mesmas versões utilizando-se RED é de até 65%, 86% e 29% para *sacklt*, *newrenolt* e *renolt* respectivamente.

Sacklt é a versão TCP que apresenta o resultado menos satisfatório. Uma possível explicação é o fato de que múltiplas perdas contribuem apenas para cerca de 10% das retransmissões devido à expiração do intervalo de temporização, enquanto que a não ativação do algoritmo de *Fast Retransmit* é responsável por cerca de 85% destas retransmissões [10]. Dado que o TCP Sack foi desenvolvido para aprimorar o TCP na ocorrência de múltiplas perdas em uma mesma janela, ele não é eficiente em evitar retransmissões devido à expiração do intervalo de temporização. O TCP Sack é mais eficiente para conexões TCP com janelas muito grandes, típicas de redes de satélite. *Newrenolt* é a versão TCP que apresenta o melhor desempenho, mesmo quando RED é utilizada. Uma razão para isto é a sua habilidade de inferir que mais de uma perda ocorreu, recuperando-se rapidamente quando perdas ocorrem evitando assim a ocorrência de RTO´s, apresentando os maiores valores de *goodput* quando o congestionamento aumenta.

O grau de tolerância à perda de segmentos de uma conexão é dependente do tamanho 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. As políticas de AQM são diretamente responsáveis pelo descarte de pacotes. Desta forma, para não prejudicar conexões curtas, 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.

Apesar do modelo TCP utilizado para derivar o H2-TAQM considerar apenas tráfegos TCP de longa duração, foi também investigado o impacto da probabilidade de descarte/marcação de H2-TAQM e de RED em diferentes versões TCP, quando o tráfego é de curta duração, como é a característica do tráfego WEB. 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.

Pode ser verificado na Figura 4 que não existe uma diferença significativa entre o *goodput* obtido para diferentes conexões TCP que utilizam H2-TAQM. Entretanto, utilizando-se RED, a diferença entre o *goodput* obtido pode ser de até 10% entre o *newrenolt* e *sacklt* para cargas de 0.9. Comparando-se as mesmas versões TCP utilizando-se H2-TAQM e RED tem-se uma diferença que pode ser de no mínimo 8%, 17% e 8% para cargas de 0.5 e de no máximo 27%, 31% e 23% para cargas de 0.7 para *sacklt*, *newrenolt* e *renolt* respectivamente.

Para o tráfego WEB, todas as diferentes versões TCP utilizando H2-TAQM, reduzem a ocorrência de retransmissões devido à expiração do intervalo de temporização para zero para cargas de 0.4 à 0.6. 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. A redução nos valores obtidos para as diferentes versões TCP quando se utiliza H2-TAQM ao se comparar com as mesmas versões utilizando-se RED pode ser de até 81%, 75% e 33% para *sacklt*, *newrenolt* e *renolt*, respectivamente.

A taxa de perda obtida para as diferentes versões TCP utilizando-se H2-TAQM são menores que as obtidas utilizando-se RED quando o tráfego é WEB. Apenas para cargas de 0.9, a taxa de perda obtida por *sacklt* ao se utilizar RED é menor que as demais. Mesmo para esta carga, a taxa de perda apresentada para *renolt* e *newrenolt* utilizando-se H2-TAQM é menor que a apresentada por estas versões utilizando-se RED. Para o tráfego WEB *newrenolt* é também a versão TCP que apresenta o melhor resultado. Mesmo quando RED é utilizada como política de AQM, *newrenolt* apresenta o menor número de ocorrência de retransmissões devido à expiração do devido à expiração do intervalo de temporização, bem como o maior valor obtido para o *goodput*.

Para todas as versões TCP o desempenho obtido quando H2-TAQM é utilizada é superior ao desempenho das mesmas versões quando se utiliza RED como política de AQM. O número médio de retransmissões devido à expiração do intervalo de temporização produzidos com tráfego FTP é maior que o número produzido com tráfego WEB. Sob H2-TAQM, o número de retransmissões devido à expiração do intervalo de temporização gerados com tráfego FTP para *sacklt*, *newrenolt* e *renolt* pode ser até seis, duas e três vezes maior do que o número gerado para estas mesmas versões TCP com tráfego WEB. Utilizando-se RED, o número de retransmissões devido à expiração do intervalo de temporização gerados com tráfego FTP para *sacklt*, *newrenolt* e *renolt* pode ser até quatro, seis e oito vezes maior do que o número gerados para estas mesmas versões TCP com tráfego WEB. A taxa de perda apresentada para tráfego FTP quando H2-TAQM é utilizada pode ser até o dobro da apresentada para tráfego WEB considerando as mesmas versões TCP. Entretanto para RED, a taxa de perda é basicamente a mesma para ambos os tráfegos.

Quando se compara o *goodput* obtido considerando os dois tipos de tráfego, pode ser observado que quando H2-TAQM é utilizada como política de AQM, o *goodput* obtido é quase o mesmo para ambos os tráfegos. Quando a política utilizada é RED, o *goodput* obtido para tráfego FTP é maior que o obtido para o tráfego WEB. A diferença pode ser de até 10% para *sacklt*, 19% para *newrenolt* e 15% para *renolt*.

Quando RED é utilizada, existe uma diferença significativa no *goodput* obtido para ambos os tráfegos. A versão TCP que apresenta o maior *goodput* é o *newrenolt*, o que confirma os resultados apresentados em [7].

## 6. Conclusões

Este artigo investigou a eficácia do mecanismo *Limited Transmit* quando utilizado em conjunto com o TCP SACK, TCP Reno e o TCP NewReno, sob as políticas de gerenciamento ativo de filas RED e H2-TAQM. O estudo considerou tráfego WEB e tráfego FTP. Os resultados obtidos indicam que todas as versões TCP apresentam valores similares de *goodput* quando H2-TAQM é utilizado como a política de AQM, dado que H2-TAQM produz uma baixa taxa de perda quando o congestionamento é incipiente e também reduz a ocorrência de retransmissões devido à expiração do intervalo de temporização (RTO´s). Mesmo quando H2-TAQM produz uma taxa de perda maior que a apresentada por RED, H2-TAQM apresenta uma taxa de descarte de pacotes mais proporcional, evitando, desta forma a continuidade de perdas de pacotes bem como reduzindo a ocorrência de RTO´s. Além disto, mostrou-se que o uso em conjunto do TCP NewReno com o *Limited Transmit* e H2-TAQM apresenta o melhor desempenho, não apenas por que maiores valores de *goodput* são obtidos, mas também devido a redução na ocorrência de RTO´s.

Como trabalhos futuros sugere-se a utilização de cenários diferentes, com uma topologia que contenha mais de um enlace gargalo, que diferentes padrões de tráfego possam ser utilizados, tenha-se tráfego cruzado, além de se ter diferentes tamanhos de pacotes. Além disto, sugere-se a extensão deste trabalho, onde seja avaliado o impacto da utilização em conjunto de ECN, do mecanismo *Limited Transmit* com as demais versões TCP e as políticas de AQM.

## References


[1] V. P. M. Allman and W. Stevens, "TCP congestion control," RFC 2581, April 1999.

[2] 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.

[3] H. B. M. Allman and S. Floyd, "Enhancing TCP's loss recovery using limited transmit," RFC 3042, January 2001.

[4] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[5] J. C. G. M. M A. E. Lima 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.

[6] 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'00*, Stockholm, Sweden, September 2000, pp. 151–160.

[7] M. M. de A. E. Lima, N. L. S. da Fonseca, and J. F. de Rezende, "On the performance of TCP loss recovery mechanisms," in *Proceedings of the IEEE ICC*, Anchorage, Alaska, May 2003.

[8] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, April 1999.

[9] S. F. M. Mathis, J. Mahdavi and A. Romanow, "TCP selective acknowledgment options," RFC 2018, October 1996.

[10] D. Lin and H. T. Kung, "TCP fast recovery strategies: Analysis and improvements," in *Proceedings of IEEE Infocom*, San Francisco, CA, USA, March 1998.

[11] 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.

[12] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an Internet backbone," in *Proceedings of INET'98*, Geneva, Switzerland, July 1998.

[13] 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

[14] S. Floyd. NS: Network simulator. [Online]. Available: http://www.isi.edu/nsnam/ns

[15] K. V. Cardoso and J. F. de Rezende, "HTTP traffic modeling: Development and application," in *Proceedings of ITS2002*, Natal Brazil, September 2002.

[16] 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.

[17] Papers by CAIDA. [Online]. Available: http://www.caida.org/outreach/papers/