UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

MARCOS TALAU

EARLY WINDOW TAILORING: UMA NOVA ABORDAGEM PARA MAXIMIZAÇÃO DE FLUXOS TCP CONCORRENTES E MINIMIZAÇÃO DE CONGESTIONAMENTO

TESE

CURITIBA
2020

MARCOS TALAU

EARLY WINDOW TAILORING: UMA NOVA ABORDAGEM PARA MAXIMIZAÇÃO DE FLUXOS TCP CONCORRENTES E MINIMIZAÇÃO DE CONGESTIONAMENTO

EARLY WINDOW TAILORING: A NEW APPROACH TO MAXIMIZING CONCURRENT TCP FLOWS AND MINIMIZING CONGESTION

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, da Universidade Tecnológica Federal do Paraná (UTFPR), como requisito parcial para obtenção do título de “Doutor em Ciências” - Área de Concentração: Telecomunicações e Redes.
Orientador: Prof. Dr. Mauro Sergio Pereira Fonseca
Coorientador: Prof. Dr. Emilio Carlos Gomes Wille

CURITIBA

2020

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

TERMO DE APROVAÇÃO DE TESE

A Tese de Doutorado intitulada **EARLY WINDOW TAILORING: UMA NOVA ABORDAGEM PARA MAXIMIZAÇÃO DE FLUXOS TCP CONCORRENTES E MINIMIZAÇÃO DE CONGESTIONAMENTO**, defendida em sessão pública pelo(a) candidato(a) **Marcos Talau**, no dia 28 de setembro de 2020, foi julgada para a obtenção do Título de Doutor em Ciências, Área de Concentração Telecomunicações e Redes, Linha de Pesquisa Comunicação e Processamento de Dados, e aprovada em sua forma final, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial.

BANCA EXAMINADORA:

Prof. Dr. Mauro Sergio Pereira Fonseca – Presidente - UTFPR
Profa. Dra. Anelise Munaretto - UTFPR
Prof. Dr. Luis Henrique Perreira Fonseca – UFRJ
Profa. Dra. Katia Obraczka - UCSC
Prof. Dr. Igor Moraes - UFF

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 28 de setembro de 2020.

_____

Carimbo e Assinatura do(a) Coordenador(a) do Programa

A minha mãe, tia, e avó (*in memoriam*).

AGRADECIMENTOS

Agradeço primeiramente a minha mãe e minha tia por todo o apoio e amor recebido.

Agradeço aos meus orientadores, prof. Dr. Mauro Sergio Pereira Fonseca, e prof. Dr. Emilio Carlos Gomes Wille, por todo o trabalho e tempo dedicados a todas as fases do doutorado, tudo sempre guiado com base na ciência.

Agradeço a Universidade Tecnológica Federal do Paraná, câmpus Dois Vizinhos, em especial a COENS-DV, pelo afastamento autorizado.

Agradeço ao Núcleo Avançado em Telecomunicações, NATEC, e a todos os colegas, por terem me acolhido no laboratório.

O mundo é a minha representação (Arthur Schopenhauer).

RESUMO

TALAU, Marcos. ***Early Window Tailoring*: Uma nova Abordagem para Maximização de Fluxos TCP Concorrentes e Minimização de Congestionamento**. 2020. 99 f. Tese (Doutorado em Engenharia Elétrica e Informática Industrial) – Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

O TCP (*Transmission Control Protocol*) é um protocolo da camada de transporte que oferece transferência confiável de dados, serviço orientado a conexão, controle de fluxo e controle de congestionamento. O controle de fluxo é utilizado para que a fonte e o receptor informem a quantidade de dados que conseguem receber a cada momento, e o controle de congestionamento é uma medida que tenta prever a quantidade de dados que a rede suportará. O controle de congestionamento funciona através de inferência, segmentos são enviados em volume crescente até a ocorrência de perdas, na ocorrência o volume é diminuído e o processo se repete. Este comportamento leva a problemas que afetam o desempenho do TCP. Neste trabalho é proposto um novo mecanismo, chamado de EWT (*Early Window Tailoring*), buscando melhorar o desempenho do TCP. O EWT provê ao TCP um retorno do estado da memória do roteador para que ele indiretamente o utilize em sua equação de transmissão, produzindo assim um controle no uso da memória para evitar que ela fique cheia e ocorra o descarte de segmentos. O EWT é compatível com qualquer implementação TCP e não impõe alterações no protocolo. Para avaliar o método foram feitas diversas simulações em cenários com redes com fio, ponto de acesso 802.11, e redes sem fio ad hoc. Em geral, resultados obtidos no simulador de rede ns-3, comparando o EWT com as abordagens *drop-tail*, RED, ARED, EWA, e AWM, indicaram que o EWT foi eficaz em reduzir o congestionamento, diminuindo o número de perdas, permitindo a existência de um maior número de fluxos concorrentes, reduzindo o tempo de transmissão fim a fim, melhorando o *goodput* e a justiça.

Palavras-chave: TCP. controle de congestionamento. janela do receptor.

ABSTRACT

TALAU, Marcos. **Early Window Tailoring: A New Approach to Maximizing Concurrent TCP Flows and Minimizing Congestion**. 2020. 99 p. Thesis (PhD in Graduate Program in Electrical and Computer Engineering) – Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

The TCP (Transmission Control Protocol) is a transport layer protocol that offers reliable data transfer, connection-oriented service, flow control and congestion control. Flow control is used so that the source and the receiver inform the amount of data they are able to receive at any given moment, and congestion control is a measure that tries to predict the amount of data that the network will support. The congestion control works through inference, segments are sent in increasing volume until the occurrence of losses, in the event the volume is decreased and the process is repeated. This behavior leads to problems that affect the performance of TCP. In this work, a new mechanism is proposed, called EWT (Early Window Tailoring), seeking to improve the performance of TCP. The EWT provides to TCP a return of the memory state of the router so that it indirectly uses it in its transmission equation, thus producing a control in the use of memory to prevent it from becoming full and discarding segments. EWT is compatible with any TCP implementation and does not impose changes to the protocol. To evaluate the method, several simulations were carried out in scenarios with wired networks, 802.11 access points, and ad hoc wireless networks. In general, results obtained in the ns-3 network simulator, comparing EWT with drop-tail, RED, ARED, EWA, and AWM approaches, indicated that the EWT was effective in reducing congestion, reducing the number of losses, allowing the existence of a greater number of concurrent flows, reducing the end to end transmission time, improving goodput and fairness.

Keywords: TCP. congestion control. receiver's advertised window.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ACW	*Adaptive Congestion Window*
AIMD	*Additive-Increase-Multiplicative-Decrease*
AODV	*On-demand Distance Vector*
AQM	*Active Queue Management*
ARED	*Adaptive RED*
ATM	*Asynchronous Transfer Mode*
AWM	*Active Window Management*
DSR	*Dynamic Source Routing*
ECN	*Explicit Congestion Notification*
EWA	*Explicit Window Adaptation*
EWT	*Early Window Tailoring*
FBCCC	*Fibonacci Based Contention Congestion Control*
FIFO	*First In First Out*
GWA	*Generalized Window Advertising*
MTU	*Maximum Transmission Unit*
NC	*Network Coding*
NGWA	*New Generalized Window Advertising*
PINK	*Proactive INjection into acK*
RED	*Random Early Detection*
RTO	*Retransmission Timeout*
RTT	*Round-Trip Time*
SRA	*Sudden Recovery Algorithm*
TCP	*Transmission Control Protocol*

SUMÁRIO

1 INTRODUÇÃO

O *Transmission Control Protocol*) é um protocolo da camada de transporte que oferece transferência confiável de dados, serviço orientado a conexão, controle de fluxo e controle de congestionamento (KUROSE; ROSS, 2016). O controle de fluxo é utilizado para que o receptor informe a quantidade de dados que consegue receber a cada momento, e o controle de congestionamento é uma medida que tenta prever a quantidade de dados que a rede suportará, previsão esta feita constantemente pela fonte TCP com o uso de diversos algoritmos.

Este trabalho tem enfoque no controle de congestionamento do protocolo TCP, por isto é fundamental revisar o seu funcionamento. Através de inferência o protocolo TCP faz uso de informações da rede no seu controle de congestionamento (BLANTON *et al.*, 2009). Segmentos são enviados com taxa crescente e exponencial com o uso do algoritmo *slow-start*, após um nível, chamado *ssthresh*, o TCP entra no modo *congestion-avoidance* e a quantidade de segmentos (rajada) enviados passa a ser incrementada de forma linear. Este processo é feito até a ocorrência de perdas. Na ocorrência o TCP reduz o tamanho da rajada de dados, passando assim a enviar menos segmentos. Perdas são detectadas com o uso de marcações de tempo (*timeout*) e também podem ser previstas pelo recebimento de um número de ACKs duplicados. Na ocorrência do primeiro caso o tamanho da rajada é reduzido para um segmento. Quando são recebidos uma sequência (normalmente três) de ACKs duplicados e a marcação de tempo do segmento correspondente não foi expirada, o protocolo TCP utiliza o mecanismo *fast-retransmit/fast-recovery* reduzindo assim o tamanho da rajada pela metade. Se chegarem segmentos fora de ordem, o TCP também envia ACKs duplicados, que podem levar a redução da rajada, além disso, o TCP confirma apenas o último segmento recebido. Se ocorrerem perdas de múltiplos segmentos (na rajada) ele entrará na fase *slow-start*, reduzindo assim a sua transmissão. Resumindo, o TCP busca perdas para inferir o gargalo de transmissão entre a fonte e o receptor.

Este comportamento leva a problemas que afetam seu desempenho, principalmente quando múltiplos dispositivos compartilham um mesmo gateway. Os problemas ocorrem pela constante disputa por recursos do gateway, o que leva a degradação do throughput, injustiça e atrasos. Neste contexto, para melhorar o desempenho do TCP foram desenvolvidas diversas propostas que podem ser agrupadas em: gerenciadores de fila (AQMs), como (FLOYD; JACOBSON, 1993) (FLOYD, 1994) (PAN *et al.*, 2013), alterações no TCP (FLOYD *et al.*, 1996) (BOHACEK *et al.*, 2003) (LEE *et al.*, 2017), AQMs + TCP (GERLA *et al.*, 2002) (ALIZADEH

et al., 2010) (TALAU; WILLE, 2013) (SHEWMAKER *et al.*, 2016) e técnicas de retorno da rede (KALAMPOUKAS *et al.*, 2002) (PALAZZI *et al.*, 2006) (BARBERA *et al.*, 2007) (GRAZIA *et al.*, 2015) (CASONI *et al.*, 2017).

O uso de gerenciadores de fila busca evitar a ocorrência de um congestionamento através do descarte/marcação (*Explicit Congestion Notification*)) de segmentos antes do enchimento da fila do gateway. Seu uso pode ser ineficiente se os clientes TCP não utilizarem as marcações de forma adequada e também se eles não forem perfeitamente configurados para o cenário desejado. Já as "alterações no TCP" necessitam a mudança do TCP nos clientes, o que torna inviável a sua utilização. O uso de AQMs + TCP faz o uso de AQMs com alterações do TCP dos clientes, logo passa pelos mesmos problemas. Existem técnicas que atuam no roteador que faz ligação com o gargalo (*bottleneck*) da rede, realizando a alteração do valor da janela do receptor do TCP de segmentos ACKs com base na quantidade de memória disponível no dispositivo — no trabalho de (TALAU *et al.*, 2019) tais técnicas receberam o nome *network-return*, neste o termo foi traduzido para retorno da rede. Com a utilização de técnicas de retorno da rede o TCP ajusta sua taxa de transmissão com base na informação vinda da rede.

Diversos autores (FLOYD; JACOBSON, 1993) (ALLMAN; PAXSON, 1999) (JAIN; DOVROLIS, 2003) (HASEGAWA; MURATA, 2006) (BARBERA *et al.*, 2007) (JIANG *et al.*, 2009) relataram que a forma mais eficaz de se detectar um congestionamento ocorre no equipamento ligado ao gargalo da rede. É neste contexto que as técnicas de retorno da rede atuam. O uso destas técnicas traz as seguintes vantagens: não requer qualquer modificação na implementação TCP nos clientes e para ser aplicada não tem de estar presente em todos os roteadores do caminho fim a fim, sua instalação no gateway de onde se deseje fazer o controle, é o suficiente. O emprego de técnicas de retorno da rede tem se mostrado promissor no controle de congestionamento, conforme verificado em diversos trabalhos (KALAMPOUKAS *et al.*, 2002) (PALAZZI *et al.*, 2006) (BARBERA *et al.*, 2007) (PALAZZI *et al.*, 2009) (GRAZIA *et al.*, 2015) (BUJARI *et al.*, 2016) (CASONI *et al.*, 2017). Porém, grande parte dos estudos publicados se limitou a poucos experimentos. Outro ponto negativo é a complexidade das técnicas, onde muitas delas, para operar necessitam ter a informação do número de fluxos ativos. Visando preencher estas lacunas o presente trabalho propõe e avalia um método para auxílio no controle de congestionamento chamado de *Early Window Tailoring*).

O EWT é uma técnica de retorno da rede que provê ao TCP um dado do estado da memória do gateway para que ele indiretamente o utilize em sua equação de transmissão,

produzindo assim um controle no uso da memória para evitar que ela fique cheia e ocorra o descarte de segmentos. O EWT é compatível com qualquer implementação TCP e não impõe alterações no protocolo.

Para avaliar o desempenho da nova técnica foram utilizados diversos cenários de simulação. Eles podem ser agrupados na seguinte forma: redes com fio, pontos de acesso 802.11, e redes sem fio ad hoc. Diversas técnicas de retorno da rede, como EWA (*Explicit Window Adaptation*), AWM (*Active Window Management*), e PINK (*Proactive INjection into acK*) fizeram uso de redes com fio para validar o seu desempenho. No Capítulo 4 o EWT foi avaliado nestas redes. Para tal foi adotada a topologia *dumbbell*, recomendada em diversos estudos, como em (ANDREW *et al.*, 2008), (HAYES *et al.*, 2014) e (KUHN *et al.*, 2016), sendo seguida a RFC 7928 na definição do número ideal de fluxos para as simulações. No Capítulo 5 o EWT foi utilizado em pontos de acesso 802.11. Este tipo de ambiente foi escolhido por apresentar um RTT significativo até as fontes TCP, além de já ter sido utilizado em outros estudos, como em (HA; CHOI, 2006) (SEYEDZADEGAN *et al.*, 2007) (PALAZZI *et al.*, 2009) (HUANG *et al.*, 2010), e (BUJARI *et al.*, 2015). Por fim, no Capítulo 6 o EWT foi utilizado em redes sem fio ad hoc. Já é conhecido na literatura, por exemplo em (ZHANG *et al.*, 2008) (ZHANG; FENG, 2009) (BHANUMATHI; DHANASEKARAN, 2010), que o TCP não funciona adequadamente onde o meio físico é sujeito a falhas, como em redes sem fio. Nas redes ad hoc com múltiplos nós este problema é ainda maior (CHEN *et al.*, 2003) (FU *et al.*, 2003) (ZHAI *et al.*, 2005). A busca pela melhora de desempenho do TCP em tais redes continua a ser investigado, por isto este tipo de ambiente também foi explorado.

Por mais diferentes que possam ser os cenários estudados nos Capítulos 4, 5, e 6, ambos têm a característica de apresentar um ponto de gargalo, este presente em um gateway, equipamento onde são utilizados AQMs e também as técnicas de retorno da rede. A seguir são descritos e ilustrados alguns problemas do TCP neste tipo de ambiente. Na sequência são apresentados os objetivos da tese e a estrutura do trabalho.

1.1 MOTIVAÇÃO

Desconsiderando o problema da redução enganosa, onde perdas no meio físico fazem o TCP reduzir sua taxa de transmissão indevidamente, existem problemas que ocorrem em um ambiente inter-rede.

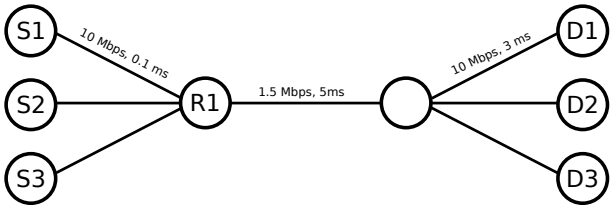
Quando o TCP é utilizado em um ambiente inter-rede é necessário haver memórias

(filas) nos roteadores para possibilitar o encaminhamento de múltiplos segmentos. Tais filas são tradicionalmente gerenciadas por uma técnica chamada de *drop-tail* ou *tail drop*. Para realizar a entrada e saída de segmentos da fila o *drop-tail* segue o modelo *First In First Out*), sendo realizado o descarte de segmentos quando ela atinge seu tamanho máximo (CLARK *et al.*, 1998) (BAKER; FAIRHURST, 2015).

O *drop-tail* apresenta problemas, com destaque para: 1) quando a fila está cheia podem ocorrer múltiplas perdas em um fluxo ou perdas em diferentes fluxos ao mesmo tempo. No primeiro caso o TCP poderá entrar na fase *slow start* e a sua taxa de transmissão será reduzida para um segmento (JACOBSON, 1988). No segundo pode ocorrer uma sincronização de perdas, onde múltiplas conexões reduzem a sua taxa de transmissão; 2) uma ou mais conexões podem monopolizar o espaço da fila (BAKER; FAIRHURST, 2015).

Para ilustrar alguns dos problemas do TCP em um ambiente inter-rede foi utilizado o cenário da Figura 1, onde as fontes S[1-3] transmitiram dados simultaneamente aos receptores D[1-3], de forma a utilizar a memória do roteador R1 até o seu limite (97 kB).

Figura 1 – Topologia utilizada para ilustrar problemas do TCP em um ambiente inter-rede.

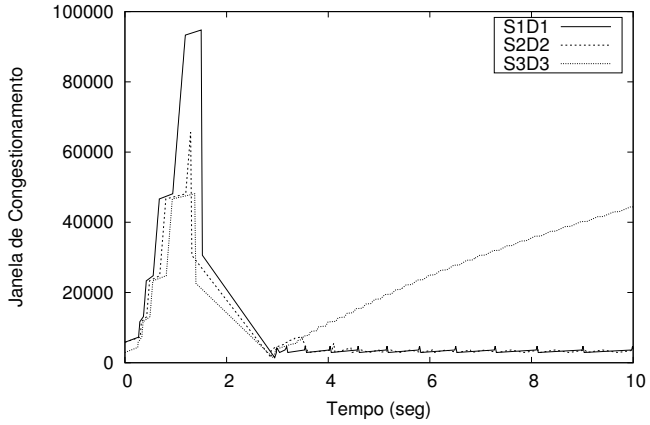


Fonte: Autoria própria.

A Figura 2 mostra as janelas de congestionamento das três conexões. Observa-se que as três apresentam um comportamento inicial semelhante até o momento em que a memória do roteador R1 fica cheia, provocando perdas de segmentos nas três conexões; com uma sequência de perdas as conexões entram na fase *slow start* e ajustam a janela de congestionamento para o tamanho de um segmento. Em seguida observa-se outro comportamento, a conexão S3D3 monopoliza o uso da memória do roteador, pois seu valor cresce continuamente enquanto as outras duas conexões ficam sincronizadas entre o envio de poucos segmentos e ocorrência de perdas.

Na Figura 3 é exibido o aumento do número de sequência das três conexões, este dado, iniciando em zero, indica a quantidade de *bytes* enviados. O comportamento inicial é semelhante, porém a conexão S1D1 tem um maior aumento. O ponto de mudança do gráfico foi exatamente o mesmo onde a janela de congestionamento foi reduzida, e a partir deste ponto a conexão S3D3

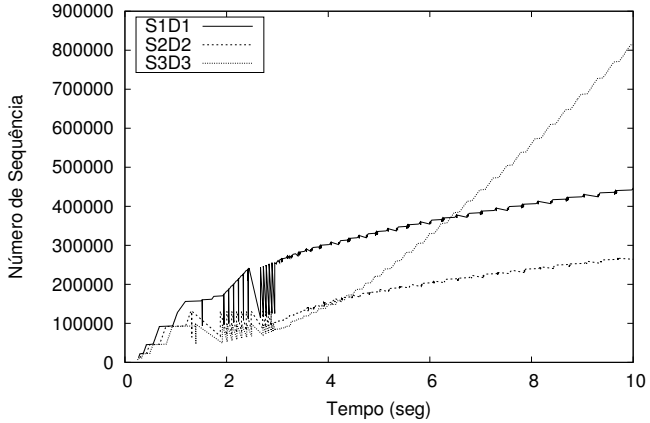
Figura 2 – Janela de congestionamento do TCP obtida em simulação utilizando o cenário da Figura 1.



Fonte: Autoria própria.

passa a ter um crescimento elevado, enquanto as outras duas conexões crescem em igual volume.

Figura 3 – Números de sequência do TCP obtidas em simulação utilizando o cenário da Figura 1.



Fonte: Autoria própria.

1.2 OBJETIVOS DA TESE

A seção anterior evidenciou problemas de injustiça e perdas no protocolo TCP, que ocorrem, principalmente, quando existe a falta de recursos na rede. Levando em conta que uma das causas de um congestionamento (em um tempo t) é a transmissão de dados em quantidade superior às capacidades (no instante t) da rede, este trabalho tem por objetivo desenvolver um método que forneça ao TCP um retorno do estado da memória do roteador para que ele indiretamente o utilize em sua equação de transmissão, produzindo assim um controle no uso da memória para evitar que ela fique cheia e ocorra o descarte de segmentos.

Os objetivos específicos deste trabalho são:

- Propor um esquema para trazer até as fontes TCP um valor referente ao uso da memória de um ponto específico da rede (gargalo), mantendo compatibilidade com as implementações TCP e ao mesmo tempo exigindo o mínimo de alterações nas pontas (cliente/servidor).
- Utilizar um modelo para verificação do método.
- Implementar o novo método em um simulador de rede.
- Realizar a avaliação do método em diferentes tipos de ambientes, como redes com fio, pontos de acesso 802.11, e redes sem fio ad hoc, fazendo uso das métricas de desempenho mais apropriadas (como: perdas de segmentos, atraso, *goodput*, e justiça).
- Com o uso do novo método, buscar:
  - Melhorar a utilização de recursos evitando a perda de pacotes.
  - Reduzir tempo de transmissão fim a fim.
  - Permitir a existência de um maior número de fluxos concorrentes em um ponto de gargalo.
  - Comparar o novo método com propostas similares.
  - Operar adequadamente em redes com alto atraso.
  - Melhorar o *goodput* das transmissões.
  - Manter a justiça entre os fluxos.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte forma: No capítulo a seguir é descrito o funcionamento básico dos controles presentes no protocolo TCP, fornecendo uma descrição dos principais algoritmos de controle de congestionamento, problemas do TCP, gerenciadores de fila e métodos similares ao proposto neste trabalho. O Capítulo 3 detalhada de forma teórica e prática a técnica de retorno da rede desenvolvida neste trabalho. Os Capítulos 4, 5, e 6 tratam da utilização do EWT em diferentes cenários; no 4 ele é explorado em redes com fio; no seguinte o EWT é avaliado em pontos de acesso 802.11 e no Capítulo 6 são utilizadas redes sem fio ad hoc. Por fim o último capítulo apresenta as conclusões.

2 REFERENCIAL TEÓRICO

Neste capítulo é descrito o funcionamento dos controles presentes no TCP, dando-se ênfase ao controle de congestionamento, sendo descritos os controles fundamentais e implementações TCP. Na sequência são relatados alguns problemas do controle de congestionamento do TCP. Em seguida são detalhados gerenciadores de filas relevantes, utilizados neste trabalho. Por fim são analisadas as técnicas de retorno da rede.

2.1 *TRANSMISSION CONTROL PROTOCOL*

O protocolo de controle de transmissão (TCP) é um protocolo da camada de transporte que oferece transferência confiável de dados, serviço orientado a conexão, controle de fluxo e controle de congestionamento (KUROSE; ROSS, 2016). O controle de fluxo é utilizado para que o receptor informe a quantidade de dados que consegue receber a cada momento, e o controle de congestionamento é uma medida que tenta prever a quantidade de dados que a rede suportará, isso normalmente é feito com o uso de diversos algoritmos utilizados pelas fontes TCP.

2.1.1 Controle de Fluxo

O controle de fluxo está presente no protocolo TCP desde a sua primeira especificação. Este controle é utilizado para que uma fonte TCP saiba quantos *bytes* um destino TCP é capaz de receber. Para isto ser possível o cabeçalho dos segmentos TCP contém um campo chamado de janela de recepção [1] (W_r). Sempre antes do envio de um segmento, a implementação TCP da origem/destino copia para o campo janela o valor em *bytes* de sua memória de recepção TCP. Desta forma, na recepção de qualquer segmento, W_r refletirá na capacidade de recepção do emissor do segmento de confirmação (ACK).

O TCP utiliza o valor da janela de recepção em sua equação de transmissão. Para ajudar a definir a equação, considere-se que a cada instante a fonte TCP pode fazer o envio de um número (n) de *bytes*. O número de *bytes* normalmente é maior que o *Maximum Transmission Unit*) da rede, desta forma n é divido em partes (segmentos). Ao atingir n o TCP para de transmitir e aguarda a confirmação dos dados enviados. O valor de n é igual à quantidade de

[1] Neste trabalho os termos “janela“ e “janela de recepção” tem o mesmo significado.

bytes que o receptor consegue receber, ou seja, n é igual a W_r. A Equação 1 define a transmissão do TCP.

$$janela\ de\ transmissão = W_r - u, \tag{1}$$

onde u representa a quantidade de *bytes* não confirmados. A variável u inicia em zero e para cada segmento enviado ela é incrementada pela quantidade de *bytes* enviados, e para cada segmento confirmado ela é decrementada. Com isto dificilmente (dificilmente pois podem haver segmentos considerados perdidos, que foram retransmitidos, mas ainda estavam na rede) haverá mais *bytes* trafegando na rede do que a capacidade do receptor. A forma de transmissão constante na Equação 1 foi inicialmente definida em (CERF *et al.*, 1974), depois especificada em (DARPA, 1981) e evita que o receptor receba mais dados do que ele conseguiria processar.

2.1.2 Controle de Congestionamento

Se no caminho entre os pares de uma conexão TCP existirem equipamentos de rede, como roteadores, eles precisão ter espaço em memória para ser possível o processamento de múltiplos segmentos, e, naturalmente, as memórias podem ficar cheias; neste cenário a equação utilizada no controle de fluxo é inadequada, pois com seu uso uma sequência de *bytes* serão enviados a rede, respeitando apenas o limite do receptor. Um pouco mais de dois anos depois lançamento da especificação do TCP, Nagle observou o primeiro registro de congestionamento envolvendo o protocolo (NAGLE, 1984), e posteriormente Jacobson e Karels (JACOBSON, 1988) modificaram a equação de transmissão do TCP, introduzindo uma nova variável, W_{cwnd}, a janela de congestionamento. A nova variável é atualizada por algoritmos de forma a acompanhar o nível de utilização da rede, e é utilizada para tentar evitar que o TCP envie mais *bytes* que a capacidade da rede. A Equação 2 foi adicionada oficialmente ao TCP em (STEVENS, 1997):

$$bytes\ a\ enviar = min(W_r, W_{cwnd}) - u \tag{2}$$

A variável W_{cwnd} é ajustada pelo transmissor TCP, não sendo compartilhada com o receptor. De forma geral, os algoritmos atualizam a variável com base nas perdas de segmentos; se uma perda ocorreu, a variável será decrementada; sem perdas, a variável vai sendo incrementada. Neste modelo, na ocorrência de perdas a taxa de transmissão é reduzida.

Um congestionamento é associado a ocorrência de perdas de segmentos; é descon-

siderada a existência de perdas por falha na rede, pois ela é menor do que um por cento [2] (JACOBSON, 1988) (STEVENS, 1994). A perda de segmentos pode ser indicada de duas formas (STEVENS, 1994) (GERLA *et al.*, 2001):

- Tempo esgotado (*timeout*): Ocorre quando uma confirmação de recebimento de segmento (ACK) não é recebida em um tempo médio previamente estimado.
- ACK duplicado (*duplicate ACK*): É definido pela recepção de um ACK com um número de sequência que já havia sido recebido anteriormente.

Após a detecção de perdas a variável W_{cwnd} é reduzida. A redução é feita por algoritmos de controle de congestionamento. Nas próximas seções são descritos os algoritmos clássicos de controle de congestionamento e algumas implementações TCP. Finalizando com problemas existentes no controle de congestionamento do TCP e propostas para sua melhoria.

2.1.3 Algoritmos Clássicos

Nesta seção são descritos os algoritmos do modelo aumento-aditivo-redução - multiplicativa (. Os algoritmos do esquema AIMD definem a janela de congestionamento (W_{cwnd}) e como ela se comportará. A utilização dela interferirá diretamente na quantidade de *bytes* a ser transmitida.

2.1.3.1 Slow-start

Este algoritmo tem por objetivo ajustar a janela de congestionamento a um valor adequado a situação de congestionamento da rede. No início da conexão ou após uma perda de segmentos por tempo (*timeout*), a janela de congestionamento será igualada a um segmento. Após cada confirmação de recebimento (ACK) a janela é incrementada em um segmento. É considerado que para cada segmento recebido, um ACK de resposta é enviado, desta forma a janela de congestionamento cresce exponencialmente (JACOBSON, 1988).

[2] O controle de congestionamento do TCP baseia-se no princípio de que uma perda foi causada por um congestionamento. Em redes sem fio este método não deveria ser aplicado, pois neste tipo de rede, erros de transmissão e interferências são frequentes.

2.1.3.2 Congestion-Avoidance

Similar ao *slow-start*, este algoritmo também atua na perda de segmentos por tempo e no recebimento de confirmações. Porém, ao detectar uma perda, a janela de congestionamento é ajustada para a metade do seu valor, e no recebimento de ACKs a janela é incrementada por $1/W_{cwnd}$ (JACOBSON, 1988).

Na prática, os algoritmos *slow-start* e *congestion-avoidance* devem ser usados em conjunto. Para isso ser possível uma nova variável é utilizada. A variável *ssthresh* indica o ponto de troca dos algoritmos. A Figura 4 descreve a operação de alternância dos algoritmos.

Figura 4 – Pseudocódigo de mudança de algoritmo.

```
se W_cwnd < ssthresh
    executar o algoritmo slow-start
senão
    executar o congestion-avoidance
```

Fonte: Autoria própria.

2.1.3.3 Fast-retransmit e Fast-Recovery

Como seu nome sugere, o *fast-retransmit* faz a retransmissão de segmentos considerados perdidos sem aguardar pela expiração de tempo (*timeout*). A sua ativação ocorre por recebimento de um conjunto de ACKs duplicados. Esta quantidade é utilizada, pois, o recebimento de segmentos fora de ordem (que gera ACK duplo) em uma quantidade inferior ao conjunto tende a não indicar uma perda. Mas quando o número de ACKs duplicados for igual ou superior a três, a probabilidade de perda do segmento é alta, e é neste momento que o *fast-retransmit* retransmitirá os segmentos considerados perdidos (STEVENS, 1997).

O *fast-recovery* é definido pela execução do *fast-retransmit* seguida pela execução do algoritmo *congestion-avoidance*. Assim como o *slow-start* e o *congestion-avoidance* são implementados em conjunto, o *fast-retransmit* e o *fast-recovery* também são. A Figura 5 contém o pseudocódigo desta implementação.

2.1.4 Implementações TCP

As implementações de controle de congestionamento do TCP são algoritmos geralmente compatíveis com o TCP padrão (RFC 793). Estes algoritmos podem ser classificados em dois

grupos: baseados em perda (*loss-based*) e baseados em atraso (*delay-based*). No primeiro tipo um congestionamento é detectado quando uma perda de segmentos ocorre, e se ajusta a janela de congestionamento para um nível mais baixo. Os algoritmos baseados em atraso fazem uso de valores RTT (*Round-Trip Time*) para detectar um congestionamento. Os dois métodos também podem ser usados em conjunto (KODAMA *et al.*, 2009) (LA; ANANTHARAM, 2000).

Abaixo são descritas algumas das principais implementações TCP existentes.

2.1.4.1 Tahoe, Reno, New Reno

A primeira implementação TCP criada para o controle de congestionamento foi o TCP Tahoe. Ele basicamente contém os algoritmos *slow-start*, *congestion-avoidance*, e o *fast-retransmit* de Jacobson (JACOBSON, 1988), além de refinamentos no cálculo do *Round-Trip Time*) (FALL; FLOYD, 1996).

A evolução do Tahoe criou o TCP Reno. Além dos três algoritmos clássicos, ele implementa o *fast-recovery* (STEVENS, 1997) (FALL; FLOYD, 1996) (LAI; YAO, 2000). O TCP Reno tem um grande problema: não consegue convergir o tamanho da janela para um valor adequado quando o ambiente de rede é formado por enlaces de natureza diferente (HASEGAWA; MURATA, 2006).

O TCP New Reno traz melhorias no algoritmo *fast-recovery*. Durante o *fast-recovery* ao se receber um ACK (com novo número de sequência) que ainda o mantenha na fase de *fast-*

Figura 5 – Implementação conjunta dos algoritmos *fast-retransmit* e *fast-recovery* (STEVENS, 1997).

```
// Função executada após a recepção do ACK
// do segmento retransmitido
aguardar_chegada_novo_ack()
   // congestion-avoidance
   W_cwnd = ssthresh

// Principal função do código
recepcao_tres_acks_duplicados()
   ssthresh = W_cwnd * 0.5
   se ssthresh < dois_segmentos
      ssthresh = 2
   retransmitir_segmento()
   W_cwnd = ssthresh + 3 * tamanho_segmento
   aguardar_chegada_novo_ack()

// Cada ACK duplicado recebido, executa este código
recebe_dup_ack()
   W_cwnd = W_cwnd + tamanho_segmento
```

Fonte: Autoria própria.

recovery, o segmento será retransmitido imediatamente. O transmissor permanecerá nesta fase até que ocorra um *timeout* ou todos os segmentos perdidos sejam retransmitidos com sucesso.

2.1.4.2 SACK

O reconhecimento seletivo (SACK) não tem perfil de uma implementação TCP, pois ele é uma opção extra que pode ser adicionado ao protocolo. Quando um segmento é perdido um ACK duplicado é enviado, a recepção deste faz com que todos os segmentos pertencentes ao intervalo da janela sejam retransmitidos; o correto seria enviar apenas o segmento perdido, e não todos os segmentos da janela. O SACK foi criado para resolver este problema. Ao receber segmentos, o receptor que implemente o SACK enviará segmentos ao transmissor informando quais segmentos já foram recebidos (FLOYD *et al.*, 1996).

2.1.4.3 Vegas

O TCP Vegas tenta antecipar a percepção de um congestionamento pelo monitoramento da diferença entre a velocidade atual de recebimento de segmentos, e a velocidade esperada por ele. A estratégia do Vegas é ajustar a janela de congestionamento para tentar manter um pequeno número de segmentos armazenados nos roteadores ao longo do caminho (LOW *et al.*, 2001). Em testes realizados, o TCP Vegas obteve de 37 a 71% melhor vazão que o Reno (BRAKMO; PETERSON, 1995).

O TCP Vegas apresenta alguns problemas. O TCP Reno utiliza um esquema de controle agressivo que expande a taxa de transmissão (banda) até que os segmentos transmitidos sejam perdidos, enquanto que o Vegas é conservador. Quando conexões Reno e Vegas existirem juntas, as conexões TCP Reno ficarão com a banda das conexões TCP Vegas (LAI; YAO, 2000). Outro problema está ligado ao RTT. Como o Vegas utiliza valores de RTT para prever um congestionamento, em redes de alta velocidade, o RTT tende a ficar inexpressivo com o aumento da taxa de transmissão, logo, implementações que são baseadas em RTT não funcionarão adequadamente (HASEGAWA; MURATA, 2006).

2.1.4.4 Westwood

O TCP Westwood (TCPW) faz o controle da janela fim a fim através da estimação contínua da velocidade dos segmentos, pela monitoração da recepção de ACKs. Após uma ocorrência de congestionamento, a estimação obtida é utilizada para computar a janela de congestionamento e a variável *ssthresh* do *slow-start*. Essa característica torna o TCPW mais robusto aos problemas dos enlaces sem fio. O TCPW obteve melhorias significativas de desempenho em comparação ao Reno e o SACK, principalmente em um ambiente misto (com/sem fio) (GERLA *et al.*, 2001).

2.1.4.5 Symbiosis

A maioria das implementações baseia-se na mudança de parâmetros do modelo AIMD para se adequar a determinado tipo de rede. O TCP Symbiosis é bastante diferente dos demais. Ele é baseado no ajuste do tamanho da janela de uma conexão TCP através de informações físicas e de banda disponível no caminho fim a fim da rede. Os algoritmos usados são inspirados na biofísica. Por análises matemáticas demonstrou-se que o TCP Symbiosis apresenta uma boa escalabilidade sobre as implementações Reno, *HighSpeed*, *Scalable TCP* e *FAST TCP* (HASEGAWA; MURATA, 2006).

2.1.4.6 BIC e CUBIC

Novas tecnologias permitiram um aumento na capacidade dos enlaces. Os algoritmos tradicionais de controle de congestionamento do TCP não foram projetados para operar com enlaces de grandes capacidades. A janela de congestionamento destes algoritmos cresce a cada RTT fazendo com que o fluxo TCP demore a atingir a capacidade disponível no enlace. Para atender às capacidades dos enlaces atuais novos algoritmos foram necessários para aumentar, em maior velocidade, a janela de congestionamento (LEVASSEUR *et al.*, 2014). É nesta categoria que se encontram o BIC (XU *et al.*, 2004) e o CUBIC (HA *et al.*, 2008).

O BIC vê o controle de congestionamento como um problema de busca em que o sistema (rede) fornece um retorno. Para realizar a busca são utilizadas duas variáveis, W_{min} para o tamanho mínimo da janela e W_{max} para o tamanho máximo. O BIC repetidamente computa o ponto médio entre W_{min} e W_{max} e ajusta a janela de congestionamento para este valor. Utilizando o critério de perdas de segmentos as variáveis W_{min}/W_{max} são atualizadas.

O CUBIC (HA *et al.*, 2008) é uma evolução do BIC. Nele a alteração da janela de congestionamento é simplificada substituindo o crescimento concavo e convexo do BIC por uma função cubica. A característica principal do CUBIC é que o crescimento da janela depende apenas do tempo real de dois eventos de congestionamento. A sua fase de congestionamento é determinada apenas pela perda de segmentos, assim como a sua taxa de transmissão, e diferentemente do TCP padrão, o RTT não é considerado na definição desta taxa. Desta forma, quando existem muitas perdas, ou o RTT é pequeno, o CUBIC opera no modo TCP padrão. Por fim, como a sua função de crescimento não depende do RTT, a justiça entre fluxos com diferentes RTT é garantida (HA *et al.*, 2008).

2.1.5 Problemas do TCP

Os métodos clássicos de controle de congestionamento da Internet são constituídos de mecanismos fim a fim acompanhados de estratégias de filas nos roteadores. Os procedimentos fim a fim são constituídos de algoritmos que tem por objetivo manter uma alta utilização da rede, tentando evitar que a rede fique ociosa (MORRIS, 1997). O método clássico possuí vários problemas:

Problema 1: Quando existe um número excessivo de fluxos comunicando-se com um dispositivo o controle de congestionamento não funciona adequadamente, pois o recurso do dispositivo diminui e o TCP mantém a forma de como a janela de congestionamento é ajustada (MORRIS, 1997).

Problema 2: Ausência de repasse de informação da rede para os emissores TCP (ALLMAN; PAXSON, 1999) (FLOYD; JACOBSON, 1993) (GERLA *et al.*, 2002) e necessidade de perda de segmentos para reconhecimento de congestionamento (HASEGAWA; MURATA, 2006). A estratégia vê a rede como uma caixa-preta (JACOBSON, 1988) (JACOBSON, 1990), isto é, não retorna informações à origem. O TCP conduzirá a uma perda de segmentos para assim tentar calcular a capacidade de rede (MASCOLO, 1999).

Problema 3: O ajuste não adequado entre a janela do receptor TCP e a banda da rede resultará na acumulação de grandes filas e perdas nos dispositivos de rede (KALAMPOUKAS *et al.*, 2002).

Problema 4: O TCP não consegue distinguir se o segmento foi perdido por falha no enlace ou por descarte de segmentos devido a um congestionamento de rede (GERLA *et al.*, 1999) (JIANG *et al.*, 2009); se a falha foi no enlace não há necessidade de ativar os mecanismos

de controle de congestionamento.

Problema 5: O protocolo TCP não consegue oferecer um compartilhamento por igual de banda quando os fluxos com diferentes RTTs competem em um mesmo ponto (MO; WALRAND, 2000) (GERLA *et al.*, 1999) (HASEGAWA; MURATA, 2006).

Várias pesquisas foram realizadas para tentar resolver os problemas enumerados acima.

Solução 1: Segundo Morris (MORRIS, 1997) a habilidade do TCP em prover um serviço eficiente em um ponto central de tráfego (*bottleneck*) diminui de acordo com o aumento do número de fluxos. Seu trabalho afirma que para resolver tal problema deve-se fazer com que o TCP seja menos agressivo e mais adaptativo quando a janela de congestionamento for pequena.

Solução 2: A adoção de técnicas de gerenciamento ativo de filas visa detectar uma situação de congestionamento antes que a fila do roteador fique cheia, além de prover o repasse da informação da possível ocorrência de congestionamento aos envolvidos na conexão. A notificação explícita de congestionamento (ECN), e a detecção aleatória antecipada (- *Random Early Detection*) foram criadas com o objetivo de serem usadas com os protocolos TCP/IP (FLOYD; JACOBSON, 1993) (RAMAKRISHNAN; FLOYD, 1999) (RAMAKRISHNAN *et al.*, 2001). Floyd descreveu os benefícios e problemas na adoção do ECN no TCP; o uso permite o reconhecimento de uma situação de congestionamento sem chegar ao ponto de ter que descartar segmentos; o ECN apresentou dois problemas: (1) mensagens ECN podem não chegar ao destino, por serem descartadas pela rede, evitando o recebimento da notificação de congestionamento, e (2) problemas de compatibilidade de dispositivos sem suporte a ECN com dispositivos com suporte (FLOYD, 1994). O RED tenta evitar possíveis congestionamentos com base no tamanho médio de fila no roteador. A notificação do congestionamento pode ser feita com a marcação de segmentos ou com o seu descarte. Uma desvantagem do RED é a necessidade do ajuste fino de seus parâmetros (FLOYD; JACOBSON, 1993), que nem sempre é possível.

Solução 3: Quando a conexão TCP é estabelecida por caminhos que apresentem esquemas de velocidade controlada (*rate-controlled*) e sem velocidade controlada (*nonrate-controlled*), caso não ocorra perdas de congestionamento o TCP aumentará sua janela até o tamanho máximo. A diferença entre a janela do TCP e a banda de rede disponível resultará na acumulação de grandes filas e perdas nos dispositivos de borda do esquema com velocidade controlada (KALAMPOUKAS *et al.*, 2002). Kalampoukas e Varna montaram um esquema baseado na modificação do campo janela do TCP receptor. Os resultados demonstraram eficiência no controle de perdas, justiça e vazão (*throughput*) superiores a mecanismos de descarte como o RED

(KALAMPOUKAS *et al.*, 2002).

Solução 4: O TCP não funciona bem em redes que apresentam erros frequentes no meio físico (exemplo, redes sem fio), pois ao detectar uma perda de segmentos, o protocolo acredita que isso foi devido a um congestionamento, e ativa mecanismos para resolver o problema. Com a ativação de mecanismos, como o *slow-start*, haverá uma redução desnecessária de envio de dados (BALAKRISHNAN *et al.*, 1995) (LEUNG; YEUNG, 2004) (ZHANG; FENG, 2009) (VANGALA; LABRADOR, 2003). Uma forma bastante pesquisada na literatura para tentar resolver este problema é a utilização do TCP *Snoop*. O *Snoop* teve sua origem no trabalho de Balakrishnan *et al.* (BALAKRISHNAN *et al.*, 1995) e foi pesquisado e expandido em diversos trabalhos (GARCIA *et al.*, 2002) (ZHANG *et al.*, 2008) (VANGALA; LABRADOR, 2003) (SUN *et al.*, 2004) (ZHANG; FENG, 2009) (ZHANG *et al.*, 2008) (LEUNG; YEUNG, 2004) (MOON; LEE, 2006). Simplificadamente, a ideia consiste na instalação de um serviço no dispositivo que faz a ligação das redes sem/com fio. Este serviço tem por objetivo armazenar temporariamente segmentos TCP, e gerenciar as mensagens TCP do receptor, com isto, atuando no reenvio de segmentos perdidos, e na ocultação de segmentos de resposta desnecessários.

Solução 5: Um fator importante que deve ser mantido em dispositivos que trabalham com diversas conexões TCP é a justiça (*fairness*) [3]. Conexões com RTTs elevados tendem a possuir uma velocidade de transmissão inferior a outras conexões que compartilham uma mesma rota (FLOYD, 1991). Em (MO; WALRAND, 2000) foi criado um protocolo para controlar o tamanho da janela do TCP, baseando-se no tempo total, porém é necessária a intervenção do usuário no seu ajuste. O esquema *bandwidth aware TCP* (BA-TCP) (GERLA *et al.*, 1999) foi criado para ser usado em enlaces de satélites, e possibilita o envio da informação de RTT na camada de rede (campo de extensão do protocolo IPv6). Os resultados demonstraram que a implementação fez uma alocação justa de banda, e a vazão (*throughput*) foi três vezes maior que outras implementações TCP. O TCP Symbiosis (HASEGAWA; MURATA, 2006) faz a alteração de tamanho da janela do TCP de acordo com a banda do caminho de rede, e com isso busca resolver o problema de injustiça causado pela diferença de RTT. Por análises matemáticas o TCP Symbiosis confirmou que tem uma melhor escalabilidade que outras implementações TCP.

[3] A justiça (*fairness*) deve ser interpretada como o uso por igual dos recursos da rede pelos nós.

2.1.6 Problemas em redes ad hoc

Como já citado o controle de congestionamento presente no protocolo de transporte TCP não funciona adequadamente em redes onde o meio físico está sujeito a falhas, como é o caso em redes sem fio (BHANUMATHI; DHANASEKARAN, 2010) (ZHANG *et al.*, 2008) (ZHANG; FENG, 2009). Nestas redes, o principal problema é que ele não consegue diferenciar entre perdas ocorridas por congestionamento daquelas causadas por problemas no enlace, e reage a ambas diminuindo a taxa de transmissão.

Este problema é ainda maior em redes ad hoc com múltiplos nós, como já foi verificado em diversos estudos (CHEN *et al.*, 2003) (FU *et al.*, 2003) (ZHAI *et al.*, 2005). Thangam e Kirubakaran (THANGAM; KIRUBAKARAN, 2009) apresentaram o estado da arte na avaliação de desempenho do TCP em redes ad hoc, identificando três principais problemas: 1) O TCP não é capaz de diferenciar entre perdas por falhas na rota e congestionamento de rede; 2) O TCP passa por frequentes problemas de roteamento; 3) Contenção no canal sem fio.

Existem diversos estudos sobre o desempenho do TCP em redes sem fio ad hoc, onde inúmeras simulações foram feitas para obter um melhor entendimento do seu comportamento e para buscar novas formas para se aumentar o seu desempenho.

Buscando adaptar o TCP para uma rede sem fio ad hoc, diversos trabalhos criaram técnicas objetivando ignorar perdas e mudanças no enlace, não reduzindo desta forma a taxa de transmissão do TCP. Os trabalhos apresentados em (LI *et al.*, 2001) e (CHEN *et al.*, 2003) sugerem que tal procedimento pode não ser eficiente. Este comportamento agressivo pode prejudicar o desempenho do protocolo, induzindo a mais perdas. Um dos principais problemas do TCP sobre redes sem fio é o número excessivo de acessos ao meio pelo TCP, isto é causado não apenas por ACKs que competem com segmentos de dados, mas por retransmissões causadas por perdas (OLIVEIRA; BRAUN, 2007).

Nithya, Mala e Sivasankar criaram um método denominado *Fibonacci Based Contention Congestion Control*) (NITHYA *et al.*, 2014), dividido em duas partes, uma para operar com o problema de contenção da camada de enlace e outra para determinar o tamanho da janela de congestionamento durante a fase *slow start*. Simulações indicaram melhoras em vários pontos, como na taxa de colisão e atrasos. Em (SUNITHA *et al.*, 2014) também foi criado um método *cross-layer*; ele utiliza um mecanismo próprio de controle de congestionamento que opera quando a taxa de ocupação do canal e o nível de sinal recebido é menor que um limite definido.

O mecanismo de controle de congestionamento estima o atraso e a banda disponível. Resultados indicaram melhoras em termos de vazão, taxa de entrega e atraso. Em (GIANG; VI, 2013) os autores também utilizaram informações do canal sem fio, eles criaram um esquema para coletar informações da camada de enlace para estimar a utilização do canal da estação; as informações são utilizadas para determinar valor de janela de contenção, este valor é repassado para a camada de transporte, que ajusta a taxa de transmissão de acordo. O método melhorou a justiça e sensivelmente a vazão. Wang e Perkins criaram um algoritmo *cross-layer*, objetivando aumentar a vazão de dados em redes sem fio *multihop*. O método utiliza dados da camada de enlace para calcular o nível de contenção aplicado a um nó, além de obter o uso da fila do nó adjacente. Com estas duas informações cada nó realiza o ajuste da transmissão de dados. Os resultados apresentados indicaram um aumento da vazão em relação ao TCP Reno (WANG; PERKINS, 2008). Em (ZHAI *et al.*, 2007) os autores afirmam que a contenção de mídia é um dos principais problemas do TCP em redes ad hoc, e criaram um protocolo chamado de WCCP *(Wireless Congestion Control Protocol)*, que é composto por dois componentes: o primeiro situado na camada de transporte, que substitui o algoritmo de ajuste de janela do TCP por um algoritmo de controle de taxa fim a fim, e o segundo entre as camadas de rede e enlace, realizando uma alocação justa de recursos entre os nós de um caminho. Os resultados indicaram que o esquema superou o TCP em termos de utilização do canal, atraso fim a fim e justiça no compartilhamento de recursos.

Quando a codificação de rede (*Network Coding*) (AHLSWEDE *et al.*, 2006) é utilizada em enlaces sem fio, os segmentos de diferentes fluxos são combinados para se tentar reduzir o número de transmissões (GOMEZ *et al.*, 2014). Vu, Boukhatem e Nguyen reformularam essa codificação de rede para que o TCP destino informe quantos segmentos perdidos existem na combinação realizada. Os resultados indicaram melhoras na vazão e no tempo de entrega (VU *et al.*, 2014). Em (GOMEZ *et al.*, 2014) foram exploradas as possibilidades da codificação de rede para encapsular segmentos TCP ACK junto a combinação padrão. Resultados obtidos mostraram que houve melhora de 15% de desempenho em relação a codificação de rede padrão.

Em (HOLLAND; VAIDYA, 2002) foi proposto o método chamado ELFN *(Explicit Link Failure Notification)* para prevenir a perda de desempenho do TCP por causa de alterações na rota. Com o uso desta técnica quando ocorre uma falha no enlace é enviada uma mensagem para o nó de origem para que ele pare de transmitir dados, e quando uma nova rota é descoberta, a origem é novamente notificada. Utilizando o método ELFN, o trabalho em (YU, 2004) apresentou dois

outros mecanismos: o EPLN *(Early Packet Loss Notification)* que busca informar as fontes TCP sobre os segmentos perdidos, e o BEAD *(Best-Effort ACK Delivery)* que busca retransmitir os ACKs perdidos pelos nós intermediários ou pelo TCP emissor. Os resultados indicaram melhoras sobre o ELFN, e o autor conclui que uma possível solução para o baixo desempenho do TCP em redes sem fio é a combinação de dados de diferentes camadas da rede.

Em (OLIVEIRA; BRAUN, 2007), os autores apresentaram uma proposta denominada TCP-DAA *(TCP-Dynamic Adaptive Acknowledgment)*, o método foca no lado TCP receptor, embora utilize técnicas no lado transmissor para minimizar retransmissões desnecessárias por *timeout*. No lado transmissor são feitas duas mudanças: 1) O número de ACKs duplos que ativam o algoritmo *fast-retransmit* passa de três para dois; 2) O aumento do *Retransmission Timeout*) em cinco vezes. No lado TCP receptor sempre é realizado o agrupamento de quatro ACKs, ou seja, para cada quatro segmentos de dados recebidos o receptor envia somente um ACK. Os resultados indicaram que o método melhorou a vazão e também o consumo de energia quando comparado com o TCP. Em (PRASANTHI *et al.*, 2013) foi proposto o *Sudden Recovery Algorithm*), que é aplicado na implementação TCP do transmissor. O algoritmo atua na fase de *fast-retransmit*, quando ocorrem múltiplas perdas de segmentos recentemente enviados. Resultados indicaram melhoras em termos de vazão e taxa de perdas. E em (PATIL; PATIL, 2013) foi criado o *Adaptive Congestion Window*), que estima o estado de contenção da rede e dependendo dele a janela de congestionamento do TCP é ajustada, o trabalho também apresenta uma forma de estimar o RTT utilizando informações de roteamento. O método aumentou a vazão da rede. Em (SUGANO; MURATA, 2003) foram utilizadas as técnicas de atraso de ACKs e envio de ACKs de forma preferencial. O atraso de ACKs permite que ACKs sejam coletados e depois seja somente enviado um único ACK (STEVENS, 1994). Quando ocorrem colisões de segmentos, pode-se dar preferência ao envio de ACKs a certos segmentos de dados, isto é feito pela técnica de envio de ACKs de forma preferencial. A combinação destes dois métodos aumentou a vazão do TCP em 20% como foi demonstrado em (SUGANO; MURATA, 2003). Em (CHEN *et al.*, 2003) o limite da janela de congestionamento foi dinamicamente ajustado de acordo com o tamanho do caminho dos fluxos TCP. Em (FU *et al.*, 2002) foi proposto o esquema ADTCP (Ad Hoc TCP), que se baseia em um mecanismo fim a fim para detectar congestionamento, desconexões, mudanças na rota e erros no canal. Para ele ser implementado é necessário algumas mudanças no TCP emissor e receptor. Gajjar e Gupta (GAJJAR; GUPTA, 2008) utilizaram o ADTCP para criar o (I-ADTCP *(Improved-ADTCP)*. Esse novo método considera que haverá banda disponível

no caminho fim a fim. Os resultados indicaram que o I-ADTCP obteve um desempenho superior, com vazão de 10% a 30% superior quando comparado com o método anterior.

Buscando adaptar o TCP para uma rede sem fio ad hoc, diversos trabalhos criaram técnicas objetivando ignorar perdas e mudanças no enlace, não reduzindo desta forma a taxa de transmissão do TCP. Os trabalhos apresentados em (LI *et al.*, 2001), (LI *et al.*, 2001) e (CHEN *et al.*, 2003) sugerem que tal procedimento pode não ser eficiente. Este comportamento agressivo pode prejudicar o desempenho do protocolo induzindo a mais perdas. Um dos principais problemas do TCP sobre redes sem fio é o número excessivo de acessos ao meio pelo TCP, isto é causado não apenas por ACKs que competem com segmentos de dados, mas por retransmissões causadas por perdas (OLIVEIRA; BRAUN, 2007).

Outros trabalhos exploraram a utilização de *Active Queue Management*). Em (XU *et al.*, 2003) foi proposto um esquema similar ao RED (FLOYD; JACOBSON, 1993), conhecido como NRED *(Neighborhood RED)*, que busca aumentar a justiça do TCP observando informações do canal para realizar o ajuste das probabilidades de descarte. Assim como o RED, este método requer o ajuste fino dos parâmetros. Para aliviar este problema, os autores em (MEDINA *et al.*, 2007) propõem o NDEM *(Neighborhood Diffusion Early Marking)*, que faz uso de propriedades estatísticas de redes ad hoc.

Resumindo, diversas pesquisas foram realizadas buscando melhorar o desempenho do protocolo TCP em redes sem fio ad hoc. Vários destes estudos propuseram alterações no TCP, que levam a alteração nos pontos finais, o que pode ser difícil de implementar. Outras propostas buscaram ocultar do TCP problemas no enlace, que já se mostraram não serem eficazes. Uma alternativa promissora para tratar do problema é o uso de AQMs, que por atuarem na rede, podem ser facilmente aplicados. O EWT é um AQM que provê ao TCP um retorno do estado da memória nos roteadores para que ele indiretamente o utilize em sua equação de transmissão.

2.2 GERENCIAMENTO ATIVO DE FILAS

A memória dos roteadores é utilizada na forma de fila. A fila é gerenciada por uma política definida chamada de AQM (*Active Queue Management*). A seguir são descritos os AQMs considerados neste trabalho.

2.2.1 Random Early Detection (RED)

O RED pode ser utilizado em uma rede onde o protocolo de transporte responde a indícios de congestionamento na rede. Ele busca simplificar o trabalho do controle de congestionamento da camada de transporte. Embora não se aplique exclusivamente as redes TCP/IP, algumas características são exclusivas; a marcação ou o descarte de um segmento deve ser suficiente para indicar a existência de um congestionamento.

O mecanismo de controle de congestionamento presente no RED monitora o tamanho médio da fila realizando um descarte probabilístico. Além do descarte, o algoritmo pode marcar um segmento para descarte com o uso do ECN; quando a fila excede um nível máximo, os segmentos não são marcados, mas descartados diretamente (FLOYD; JACOBSON, 1993).

2.2.1.1 Formulação

O algoritmo RED mantém uma média móvel do nível de utilização da fila, cujo valor é dado por avg. A cada novo segmento, esta média é recalculada utilizando a equação:

$$avg = (1 - w)\ avg + w \cdot U, \qquad U \geq 0, \tag{3}$$

onde w é um fator de peso e U corresponde ao tamanho corrente da fila. Caso a fila esteja vazia, a média é reduzida proporcionalmente ao tempo (t) em que ela esteve vazia, conforme a Equação 4:

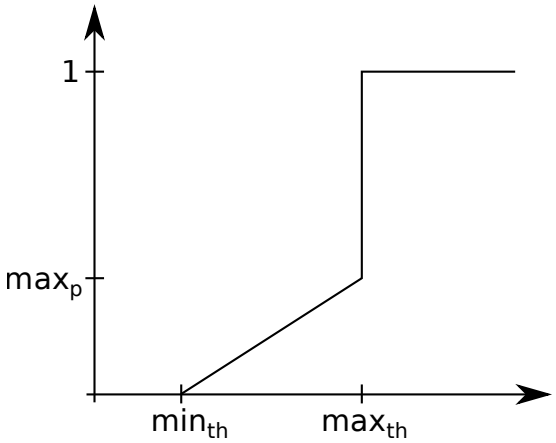
$$avg = (1 - w)^t\ avg, \tag{4}$$

A variável avg é utilizada para decidir se um segmento vai ser descartado. Os segmentos são descartados probabilisticamente usando a função de probabilidade definida como:

$$f = \begin{cases} 0, & avg \leq min_{th} \\ \frac{avg - min_{th}}{max_{th} - min_{th}}\ max_p, & min_{th} < avg < max_{th} \\ 1, & avg \geq max_{th} \end{cases} \tag{5}$$

onde min_{th} e max_{th} são parâmetros para definição do tamanho da fila, e max_p corresponde a probabilidade máxima de descarte de segmentos (FLOYD; JACOBSON, 1993). A Figura 6 ilustra a função de probabilidade do RED.

Figura 6 – Gráfico da função de probabilidade do RED.



Fonte: Autoria própria.

2.2.2 Adaptive RED (ARED)

A ideia do ARED (*Adaptive RED*) é verificar o tamanho médio da fila e tornar o RED mais ou menos agressivo. Se o tamanho médio da fila oscilar entre o valor de min_{th}, então o mecanismo do RED está muito agressivo, mas se ele oscilar em max_{th}, então o mecanismo é muito conservador. Tendo como base a variação do tamanho médio da fila o ARED ajusta o valor de max_p com base em duas variáveis, α e β (FENG *et al.*, 1999). A Figura 7 exibe o algoritmo do ARED.

Figura 7 – Algoritmo do ARED.

```
1  if (min_th < avg < max_th)
2      status = entre
3  if (avg < min_th && status != abaixo)
4      status = abaixo
5      max_p = max_p / α
6  if (avg > max_th && status != acima)
7      status = acima
8      max_p = max_p * β
```

Fonte: Autoria própria.

2.3 TÉCNICAS DE RETORNO DA REDE

Técnicas de retorno da rede são caracterizadas por inserirem ou alterarem valores no cabeçalho dos segmentos IP/TCP. Normalmente tais informações estão associadas ao estado da memória do roteador. O *Generalized Window Advertising*) (GERLA *et al.*, 2002), além da capacidade do receptor (W_r), fornece ao TCP a capacidade do meio. Para isto o TCP do receptor deve ser alterado e o método deve ser instalado nos roteadores da rede. Quando um segmento

passa pelo roteador o GWA insere em um campo (*options*) do cabeçalho IP a quantidade de memória disponível (B_n), se um valor já existir neste campo, este é extraído e é inserido o mínimo do valor atual e B_n. No momento de chegada do segmento ao receptor ele extrai o valor inserido pelo GWA e o utiliza no preenchimento do campo janela do ACK, sendo inserido o mínimo da memória do receptor e o valor do GWA. Resultados com o uso do GWA indicaram que ele deixou a rede mais estável, sem perdas e com um alto grau de justiça. Em (TALAU; WILLE, 2013) foi desenvolvido o *New Generalized Window Advertising*), um esquema similar ao GWA. Na inserção da quantidade de memória disponível o método tem dois modos: (1) utilizar a quantidade total de memória disponível ou (2) dividir a memória disponível pelo número de fluxos ativos. Outra diferença do NGWA para o GWA, é que o receptor do NGWA faz uso de uma equação de suavização no valor extraído do cabeçalho IP. Em experimentos realizados com o NGWA observou-se uma melhora na justiça e redução de segmentos perdidos. Um grande problema do GWA/NGWA é a necessidade de alteração da implementação TCP dos receptores e a instalação do método em roteadores — por necessitarem a alteração do TCP eles não são considerados técnicas de retorno da rede puras. Existem métodos que atuam apenas em roteadores, não necessitando a alteração do TCP nos pontos finais. O *Explicit Window Adaptation*) (KALAMPOUKAS *et al.*, 2002) foi desenvolvido para atuar em redes *Asynchronous Transfer Mode*). O método atua em roteadores monitorando o uso de sua memória. Na presença de segmentos na memória o EWA reduz o valor contido no campo janela do receptor (W_r) dos segmentos TCP ACK. A alteração do valor de W_r é feita com base em uma função logarítmica do espaço disponível na memória ponderada por uma variável constantemente ajustada. Resultados indicaram que o EWA foi eficaz em controlar o uso da memória do roteador obtendo uma melhora na redução das perdas, justiça e *throughput*. Para utilizar o EWA o usuário necessita configurar uma série de parâmetros. Outro método é o *Active Window Management*) (BARBERA *et al.*, 2007), que também atua em roteadores e altera o valor de W_r do cabeçalho TCP de ACKs. A alteração é feita somente quando o valor da janela do receptor for maior que $swnd$ (janela sugerida), neste caso W_r receberá o valor constante em $swnd$. Resultados com o uso do AWM indicaram um constante nível de utilização de memória, com poucas variações, e uma redução no número de perdas. Porém, um problema na sua utilização é a necessidade de saber, previamente, o número de fluxos ativos. Por fim, o *Proactive INjection into acK*) (GRAZIA *et al.*, 2015) é outro método que atua em roteadores alterando o valor do campo janela de segmentos ACKs. A alteração é feita para o valor resultante da divisão da banda pelo número de fluxos. Para operar,

o método necessita do número de fluxos ativos e do RTT de cada um (o trabalho não fornece detalhes de como o método funciona). Resultados indicaram que o método obteve uma maior justiça entre os fluxos e um menor tempo na transferência de arquivos.

A seguir são fornecidos mais detalhes das técnicas EWA, e AWM, pois tais foram utilizadas em diversos experimentos realizados neste trabalho.

2.3.1 Explicit Window Adaptation (EWA)

O EWA (KALAMPOUKAS *et al.*, 2002) foi desenvolvido para atuar em redes ATM. O método atua em roteadores monitorando o uso de sua memória. Na presença de segmentos na memória, o EWA reduz o valor contido no campo janela do receptor (W_r) do cabeçalho dos segmentos TCP ACK. A alteração do valor de W_r é feita com o uso da seguinte equação:

$$\begin{aligned} &B_a(t) = B - U(t), \\ &W'_r(t) = min(W_r(t), max(f(B_a(t)), MSS)), \end{aligned} \tag{6}$$

onde B é a quantidade de memória total, $U(t)$ é a ocupação da memória no tempo t e $B_a(t)$ indica a quantidade de memória disponível no tempo t, todas expressas em *bytes*. MSS é o tamanho máximo de um segmento e $f(B_a(t))$ é uma função definida por:

$$f(B_a(t)) = \alpha log_2\ B_a(t) \tag{7}$$

O parâmetro α é atualizado dinamicamente com base no tamanho médio da memória ($\overline{U}_t$)

$$\overline{U}_t = (1-g)\overline{U}(t-1) + gU(t), \tag{8}$$

onde g é ajustado para $\frac{1}{128}$. Dois limiares (*threshold*) são definidos em torno de $\overline{U}(t)$. Se o tamanho médio da memória for inferior a marca inferior (T_l), então α é incrementada por uma quantidade (W_{up}) a cada T segundos; se a ocupação média for maior que a marcação superior (T_h), então α é reduzida por W_{down}.

$$\alpha = \begin{cases} \alpha + W_{up}, & se\ \overline{U}(t) < T_l \\ \alpha - W_{down}, & se\ \overline{U}(t) > T_h \end{cases} \tag{9}$$

onde W_{up}, W_{down} e T são parâmetros ajustados pelo usuário.

2.3.2 Active Window Management (AWM)

O AWM (BARBERA *et al.*, 2007) também atua em roteadores e altera o valor da janela do receptor dos ACKs. A alteração do valor da janela é feita somente se ela for maior que $swnd$ (janela sugerida), quando for recebe o valor constante em $swnd$.

A variável $swnd$ é atualiza quando um segmento entra/sai da memória do roteador. A atualização é realizada com o uso da equação:

$$swnd_t = max(swnd_{t-1} + DQ_t + DT_t, MTU), \tag{10}$$

onde MTU é a unidade máxima de transmissão. O termo DQ_t é definido por:

$$DQ_t = \frac{1}{N}(U_{t-1} - U_t), \tag{11}$$

em que N é o número estimado de fluxos e U é a quantidade de memória em uso. Por fim, DT_t é igual a:

$$DT_t = \alpha(target - U_t), \tag{12}$$

onde α e $target$ são parâmetros definidos pelo usuário. DT_t gera valores positivos quando o tamanho da fila é menor que o $target$, e negativos no caso contrário.

3 EARLY WINDOW TAILORING

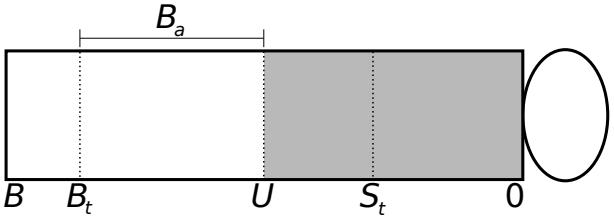
Neste capítulo é detalhado de forma teórica e prática a técnica de retorno da rede desenvolvida neste trabalho, o EWT (*Early Window Tailoring*). Inicialmente é descrito a forma de como ele opera, na sequência é fornecido um exemplo de operação, pseudocódigo, modelagem, e ao final sua implementação em simulador de rede.

3.1 O MÉTODO

O EWT traz para o TCP um controle adicional da sua transmissão pelo repasse de informações da rede que serão utilizadas em sua equação de transmissão. Essa informação é indiretamente utilizada pelo TCP, pois para o seu uso não é necessário modificar o protocolo, a instalação do método no ponto mais crítico da rede é o suficiente para prover o controle adicional. Este controle é feito pela atualização do valor contido no campo janela do receptor (W_r) de segmentos TCP ACK com base no número de bytes disponíveis de memória.

O valor da janela do receptor será atualizado pelo EWT se o seu novo valor for maior que o MSS, do contrário, o valor será ajustado para um MSS. Durante o processamento dos ACKs, não é considerado a qual fluxo eles pertencem, desta forma todos os fluxos são tratados da mesma forma e recebem a mesma atualização (proporcional ao valor de sua janela). Por somente alterar o valor da janela do receptor, o EWT é compatível com o TCP padrão e não requer mudanças no protocolo, pois a janela do receptor é naturalmente utilizada pelo transmissor para limitar a sua transmissão. A Figura 8 ilustra as principais variáveis utilizadas pelo EWT.

Figura 8 – Principais variáveis do EWT em uma memória com tamanho B.



Fonte: Autoria própria.

O retorno da memória fornecido pelo EWT é feito com base na quantidade de *bytes* disponíveis (B_a) na memória do roteador. Se os fluxos respeitarem este limite dificilmente ocorrerão perdas, porém, caso a memória esteja com um nível alto de utilização e surja uma nova rajada de tráfego, a memória ficará cheia e segmentos serão descartados. Para acomodar

rajadas de tráfego o EWT pode não utilizar toda a memória no cálculo de B_a, ficando:

$$B_a = B_t - U, \tag{13}$$

onde B_t é a quantidade máxima de memória a ser utilizada e U indica a quantidade de memória em uso. B_t é sempre menor que a quantidade de memória total (B) e $B - B_t$ é a quantidade de memória que ficará ociosa, servindo para acomodar rajadas de tráfego.

Novos fluxos tendem a ter sua janela de congestionamento baixa, por isto o retorno provido pelo EWT não teria um efeito imediato, para mitigar isto e poupar recursos quando a memória está baixa, o parâmetro S_t é utilizado para definir o ponto de início para a operação do método. Com isto, o EWT passará a entrar em funcionamento apenas quando o uso de memória ultrapasse a marca S_t. Quando isto ocorre o EWT altera o valor da janela dos segmentos proporcionalmente a B_a. Considerando W_{ewt} como o valor de retorno do EWT, pode-se escrever:

$$\begin{aligned} W_{ewt} &= \frac{B_a}{B} \cdot W_r, \\ W'_r &= max(W_{ewt}, MSS) \end{aligned} \tag{14}$$

onde W'_r é o novo valor para W_r calculado pelo EWT.

Utilizando estas equações o EWT reduzirá o valor de W_r proporcionalmente ao espaço disponível na memória, isto é, os valores das janelas serão reduzidos conforme o uso de memória aumenta. O EWT foi projetado desta forma para prover um controle da taxa de transmissão de todos os fluxos TCP com base no uso de memória, com isto minimizam-se perdas por memória cheia enquanto se mantém uma justiça entre os fluxos.

O local mais apropriado para se utilizar o EWT é no gateway (mas ele também pode ser utilizado em roteadores), onde a disputa por recursos é maior. Para utilizar o controle adicional fornecido pelo EWT não são necessárias mudanças no protocolo TCP, logo ele pode ser utilizado com qualquer algoritmo de controle de congestionamento do TCP. No TCP, a equação $min(W_c, W_r)$ é respeitada, desta forma o EWT torna-se efetivo quando a janela de congestionamento (W_c) é maior que W_r. Relacionado a isto, três casos principais podem ocorrer: (1) *sem congestionamento*: O valor de W_c é alto e a ocupação de memória atinge um certo nível. O EWT reduzirá W_r, e com isto, o fluxo reduzirá a sua taxa de transmissão, isto porque W_c provavelmente será mais alto que W_r; (2) *congestionamento*: O TCP transmite dados utilizando W_c, mas se a janela de congestionamento é alta, o W_r calculado pelo EWT pode ser menor que W_c e o volume dos será reduzido; (3) *recuperação de congestionamento*: O W_c é utilizado na

transmissão, mas W_r também pode ser utilizado se a memória tiver um nível considerável de ocupação. Com o uso do EWT procura-se evitar a ocorrência de congestionamento, por esta razão, os casos (2) e (3) tendem a ocorrer apenas por rajadas inesperadas de tráfego ou por perdas na camada física.

3.1.1 EWT Versão Cliente

O EWT traz a fonte TCP um retorno do estado da memória do ponto em que ele é instalado, este retorno é utilizado pela fonte de modo a evitar congestionamento de rede onde o EWT se encontra. Ou seja, as fontes TCP ajustam a sua taxa de transmissão utilizando a informação fornecida pelo EWT. Quanto maior o tempo que a informação leva para chegar até as fontes, maiores serão as chances de a informação não refletir no estado de memória atual. Por isto o EWT deve ser utilizado próximo às fontes TCP, onde o RTT até as fontes é baixo.

Mas podem existir casos onde se deseje instalar o EWT no lado do cliente, como, por exemplo, em um AP WiFi, que normalmente apresenta um maior RTT até as fontes TCP. Para lidar com casos como este foi desenvolvida a versão EWT cliente. Nesta versão considera-se uma média móvel para a quantidade de *bytes* disponíveis (B_a) na memória do roteador, sendo definida por:

$$\overline{B_a} = \begin{cases} (1-g)\cdot\overline{B_a} + g\cdot B_a, & se\ B_a \geq \overline{B_a} \\ B_a, & caso\ contrário \end{cases} \tag{15}$$

onde g é uma constante pertencente ao intervalo [0, 1]. Nesta versão é utilizado $\overline{B_a}$ ao invés de B_a, buscando assim fornecer as fontes um retorno mais conservador da quantidade de bytes disponíveis. Por fim, para evitar ao máximo perdas, quando $\overline{B_a}$ for maior que B_a o seu valor é reduzido para B_a.

3.2 EXEMPLO DE OPERAÇÃO

Para um melhor entendimento da operação do EWT são apresentados alguns exemplos numéricos. Foram considerados os valores: $B = 100$ kB, $S_t = 30$ kB e $W_r = 60$ kB. A Tabela 1 mostra alguns valores para W_{ewt} em função do nível de ocupação U.

A primeira linha apresenta o caso onde 20% da fila está sendo utilizada, neste caso, $U < S_t$, por isto as janelas não serão atualizadas. Pode-se considerar que, neste caso, $W_{ewt} =$

Tabela 1 – W_{ewt} em função do nível de ocupação U.

U	B_a	W_{ewt}
20 kB	80 kB	60 kB
50 kB	50 kB	30 kB
80 kB	20 kB	12 kB

Fonte: Autoria própria.

$W_r = 60$ kB.

A segunda linha considera um uso médio da fila. Como $U > S_t$, o EWT irá operar, produzindo $W_{ewt} = 30$ kB. O último caso corresponde a uma fila com alto uso, com isto o EWT força uma maior redução nas janelas, pois $W_{ewt} = 12$ kB.

3.3 PSEUDOCÓDIGO

Os gateways fazem uso de métodos AQM para o gerenciamento de memória. Estes são modulares, por exemplo, um gateway pode suportar vários métodos, mas o administrador deste escolhe qual será utilizado. Considerando isto o EWT foi implementado na forma de um AQM.

A implementação do EWT como um AQM tem por base a operação do *drop-tail*, inserindo e removendo segmentos de uma fila utilizando a política FIFO. A Figura 9 exibe a implementação do EWT como um AQM.

Figura 9 – Implementação do EWT como um AQM.

```
função dequeue()
    seg ← fila.obter_primeiro()
    ewt(seg)
    return seg
fim

função ewt(seg)
    se tipo_segmento != TCP_ACK então
        return
    fim se
    U ← fila.uso
    se U < S_t então
        return
    fim se
    W_r ← seg.janela
    B ← fila.tamanho
    B_a ← B_t − U
    W_ewt ← (B_a / B) · W_r
    seg.janela ← max (W_ewt, MSS)
fim
```

Fonte: Autoria própria.

O EWT entra em operação depois que o segmento sai da fila. Na linha dois é obtido

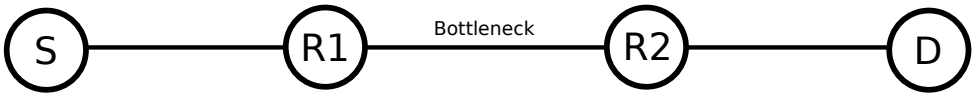
o segmento que se encontra na primeira posição da fila, depois disto ele é passado como um parâmetro para a função *ewt*. Na função é checado se o segmento é do tipo TCP ACK, caso não seja a função é finalizada e nada é feito. Na sequência é checado se o uso de memória ultrapassa o ponto de início do EWT, se o nível S_t não foi atingido, a função termina. Depois disto é obtido o valor da janela do receptor (W_r) e o tamanho da memória do gateway. Continuando, é calculada a quantidade de memória disponível (B_a) com base em B e é calculada a janela do EWT (W_{ewt}). Por fim, W_r do segmento é alterada para o maior valor entre W_{ewt} e o MSS.

3.4 MODELAGEM

Para realizar a modelagem do EWT foi utilizado como base o modelo desenvolvido no trabalho de Zhang (ZHANG *et al.*, 2006). O modelo faz uso do controle de congestionamento do TCP em conjunto com um AQM, sendo o controle de congestionamento realizado com o uso dos algoritmos *slow-start* e o *congestion-avoidance*.

O modelo considera a rede apresentada na Figura 10. Ela é composta por uma fonte (S), dois roteadores, R1 e R2, e um receptor (D). A ligação entre S-D é feita pelos roteadores R1 e R2. É assumido que os enlaces S-R1 e R2-D tem capacidade suficiente, já o enlace R1-R2 é o de menor capacidade, sendo o único *bottleneck* da rede, por isto o AQM é utilizado em R1. Segmentos de dados são transmitidos de S para D, enquanto os ACKs fazem o caminho inverso. Considera-se um único fluxo TCP persistente. O RTT entre S e D e o tamanho dos segmentos são constantes. Segmentos ACKs nunca são perdidos. Perdas somente ocorrem quando a fila de R1 atinge seu limite.

Figura 10 – Rede utilizada no modelo de (ZHANG *et al.*, 2006).



Fonte: Autoria própria.

No modelo é utilizado somente uma variável para se referir a janela de congestionamento e a janela de transmissão TCP. A esta variável é atribuído o nome W_k e nesta seção ela é chamada de janela. O modelo considera instantes discretos de tempo (amostras), sendo k utilizado como seu índice. O crescimento da janela é governado pela seguinte equação:

$$W_{k+1} = \begin{cases} min(2W_k,\ ssthresh,\ W_r), & se\ W_k < ssthresh \\ min(W_k + 1,\ W_r), & se\ W_k \geq ssthresh \end{cases} \tag{16}$$

onde $ssthresh$ é o valor do limite para a execução do algoritmo *slow-start*, e W_r é o valor da janela do receptor TCP. Quando a janela for menor que $ssthresh$ é executado o algoritmo *slow-start*, onde a janela é aumentada de forma exponencial, do contrário é executado o *congestion-avoidance*, gerando um aumento linear. Em ambos os casos é respeitado o valor de W_r como limite de tamanho. A redução do valor da janela ocorre somente quando existem perdas, onde seu valor é reduzido pela metade.

Outro ponto que o modelo trata é o da utilização da fila no instante k. Seu cálculo é dado por:

$$\begin{aligned} U_{k+1} &= U_k + W_{k+1} - \frac{C \cdot RTT_{k+1}}{M} \\ U_{k+1} &= U_k + W_{k+1} - \frac{C}{M}(d + \frac{U_k \cdot M}{C}) \\ U_{k+1} &= W_{k+1} - \frac{C \cdot d}{M} \end{aligned} \tag{17}$$

onde U_k é o valor do uso da fila, W_k é a janela, C é a capacidade do enlace, M é o tamanho do segmento e d é o retardo de propagação do meio físico. U_k e W_k são expressos em segmentos, C e M em *bytes*, e RTT e d em segundos.

O pseudocódigo descrevendo o modelo completo é exibido na Figura 11. Executando o algoritmo por K rodadas ao final pode-se verificar ao longo do tempo o comportamento da janela W_k e da utilização da fila (U). As perdas ocorrem quando W_k, aplicada na Equação 17, resultar em um uso maior ou igual à quantidade total de memória (B). A utilização deste mecanismo de detecção de perdas representa o comportamento do *drop-tail*. Para fazer uso do EWT o algoritmo foi expandido. Antes de o modelo fazer uso da janela do receptor (W_r) é realizada uma chamada a *função* EWT, esta retorna o valor de W_r processado pelo EWT. Após este processo o valor de W_r já foi atualizado pelo EWT e o modelo não necessita de alterações, pois W_r já é utilizado nos cálculos que definem a janela W_k. O modelo contendo o EWT é mostrado na Figura 12.

Para demonstrar o comportamento do EWT foi utilizado o modelo EWT (Figura 12) com 100 amostras sendo coletados os valores de U_k e W_k. O mesmo processo também foi executado para o modelo *drop-tail* (Figura 11). Os parâmetros utilizados estão na Tabela 2.

Na Figura 13 pode-se observar que a janela W_k teve um crescimento exponencial até o ponto $ssthresh$, depois houve um crescimento linear. Com o uso do EWT a janela do receptor foi reduzida conforme o crescimento da fila, como pode ser observado na Figura 14. A redução de W_r passou a segurar o crescimento da janela W_k quando o valor de W_r foi menor que W_k, neste

Figura 11 – Algoritmo adaptado do modelo de (ZHANG *et al.*, 2006).

$W_0 \leftarrow 1$
$U_0 \leftarrow 0$
Para cada rodada k:
  se $W_k - \frac{C \cdot d}{M} < B$ **então**
    se $W_k < ssthresh$ **então**
      $W_{k+1} \leftarrow min(2W_k,\ ssthresh,\ W_r)$
    senão
      $W_{k+1} \leftarrow min(W_k + 1,\ W_r)$
    fim se
  senão
    $W_{k+1} \leftarrow \frac{1}{2} W_k$
  fim se
  $U_{k+1} \leftarrow W_{k+1} - \frac{C \cdot d}{M}$

Fonte: Autoria própria.

Figura 12 – Algoritmo adaptado do modelo de (ZHANG *et al.*, 2006) fazendo uso do EWT.

função $EWT(W_r, W_k)$
  $U \leftarrow W_k - \frac{C \cdot d}{M}$
  $B_a \leftarrow B - U$
  se $U < S_t$ **então**
    return W_r
  fim se
  $W_{ewt} \leftarrow \frac{B_a}{B} \cdot W_r$
  return $max(W_{ewt}, 1)$
fim

$W_0 \leftarrow 1$
$U_0 \leftarrow 0$
Para cada rodada k:
  se $W_k - \frac{C \cdot d}{M} < B$ **então**
    $W_r \leftarrow EWT(W_r, W_k)$
    se $W_k < ssthresh$ **então**
      $W_{k+1} \leftarrow min(2W_k,\ ssthresh,\ W_r)$
    senão
      $W_{k+1} \leftarrow min(W_k + 1,\ W_r)$
    fim se
  senão
    $W_{k+1} \leftarrow \frac{1}{2} W_k$
  fim se
  $U_{k+1} \leftarrow W_{k+1} - \frac{C \cdot d}{M}$

Fonte: Autoria própria.

ponto o sistema entrou em equilíbrio, a janela W_k passou a utilizar W_r e não houve ocorrência de perdas na fila (Figura 15).

3.5 IMPLEMENTAÇÃO NO SIMULADOR DE REDES NS-3

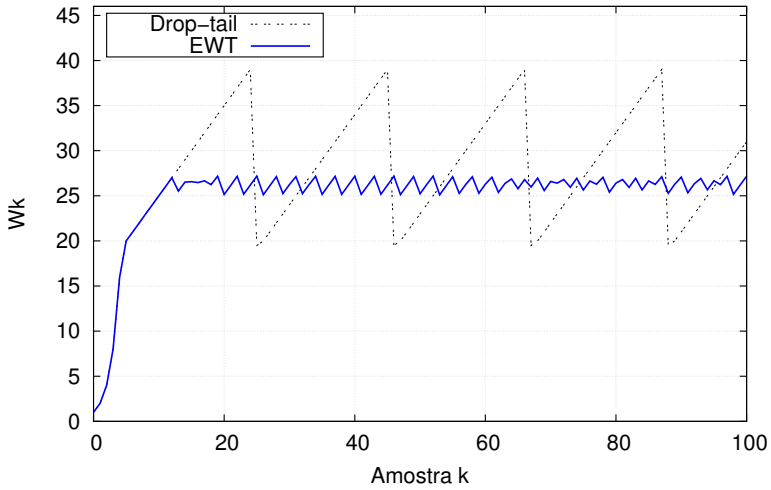
Para implementar e avaliar a proposta deste trabalho foi utilizado o simulador ns-3 (NS-3(a), 2019). O ns-3 é um simulador de rede baseado em eventos discretos projetado com foco para o uso em pesquisas e ensino. Seu desenvolvimento teve início em 2006, sendo desde a

Tabela 2 – Parâmetros utilizados nos modelos.

Parâmetro	Valor
M	500 bytes
C	192500 bytes
d	0.0228 seg.
B	30 segmentos
W_r	65 segmentos
$ssthresh$	20 segmentos
S_t	3 segmentos

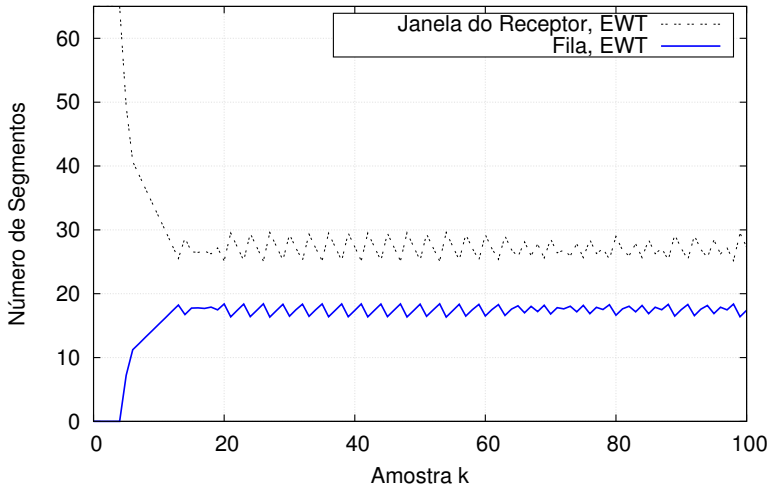
Fonte: Autoria própria.

Figura 13 – Registro da janela W_k para k amostras dos modelos *drop-tail* e EWT.



Fonte: Autoria própria.

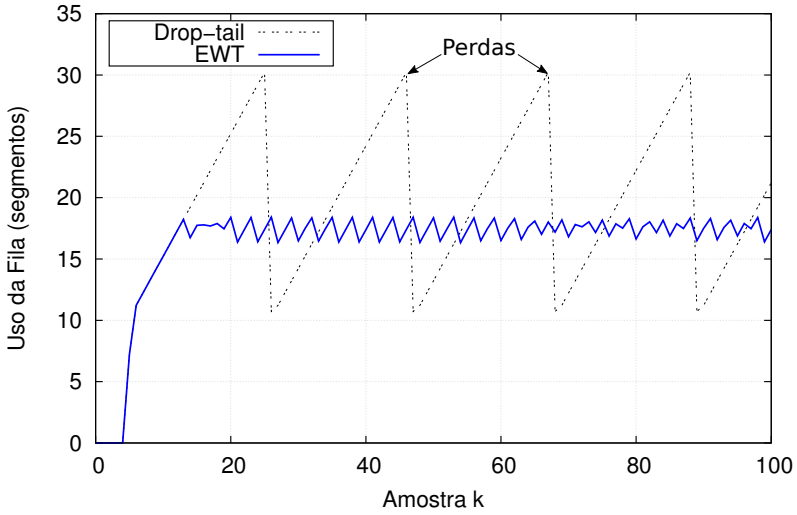
Figura 14 – Registro conjunto da janela do receptor e do uso da fila no modelo EWT.



Fonte: Autoria própria.

primeira versão um software livre (GPLv2) com código fonte orientado a objetos em linguagem C++. No geral o ns-3 oferece modelos de como uma rede orientada a segmentos opera, provendo

Figura 15 – Registro do uso da fila para k amostras dos modelos *drop-tail* e EWT.



Fonte: Autoria própria.

um ambiente para os usuários executarem as suas simulações, que podem ser escritas em linguagem C++ ou em Python (NS-3(b), 2017). Em (WEINGARTNER *et al.*, 2009) foi feita uma comparação entre os simuladores ns-2, ns-3, OMNet++, SimPy e o JiST/SWANS, avaliando desempenhos quanto ao tempo de simulação e ao uso de memória. No geral foi verificado que o ns-3 obteve melhores resultados, demandando poucos recursos computacionais e consumindo pouca memória.

Na implementação do método uma escolha natural seria a utilização do ns-2, porém ele não oferece uma implementação TCP que suporte o uso de janelas (NS-2-LIMITATIONS, 2011). A seguir é descrita de forma resumida a implementação do EWT em redes sem fio e com fio.

Redes sem fio: A implementação foi feita junto ao módulo *wifi*. Neste, o AQM padrão fica na classe *WifiMacQueue*. Utilizou-se herança para criar uma classe. Na nova classe foi reescrito o método de *Dequeue*, que é responsável pela retirada de segmentos da fila, realizando ao seu término o retorno do respectivo segmento. Os códigos do EWT foram inseridos antes deste retorno ser realizado, sendo retornado um segmento possivelmente atualizado pelo método; possivelmente pois o EWT age apenas quando o nível S_t é atingido.

Redes com fio: A implementação do EWT para operar em redes com fio foi feita junto ao módulo *Traffic-control* (IMPUTATO; AVALLONE, 2016). Neste foi criada uma nova classe com o uso de herança da classe *QueueDisc*. Na nova classe seguiu-se a estrutura do *drop-tail*, sendo que na saída do segmento da fila ele é repassado para o EWT.

Em testes preliminares no simulador de redes foi verificado que o custo de operação do EWT foi satisfatório, não gerando atrasos significativos a ponto de inviabilizar o uso do método.

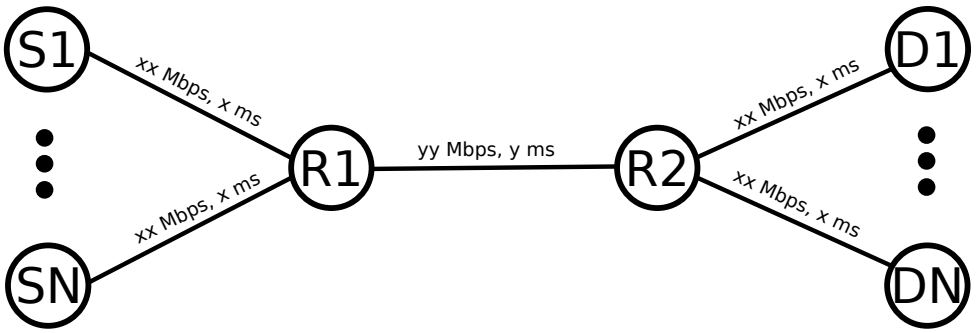
Em (CASONI *et al.*, 2017) foi discutido o custo de implementação de uma técnica de retorno da rede em um cenário real. Os autores afirmaram que é possível a realização das operações necessárias para o funcionamento das técnica de retorno da rede de forma paralela em hardware. Já em redes com velocidades maiores que um Gigabit os autores afirmam que para evitar a degradação da performance é necessário que as operações sejam executadas em um harware dedicado.

4 ANÁLISE DO EWT EM REDES COM FIO

Diversas técnicas de retorno da rede, como EWA, e AWM, fizeram uso de redes com fio para validar o seu desempenho. Neste capítulo o EWT também foi avaliado nestas redes. Foram utilizados três cenários que seguem a metodologia descrita a seguir.

Para testar o EWT em redes com fio foi adotada uma topologia bastante utilizada na literatura, a *dumbbell*. Esta topologia já foi recomendada em (ANDREW *et al.*, 2008), (HAYES *et al.*, 2014) e (KUHN *et al.*, 2016). Ela é constituída por $n \times n$ nós e dois roteadores ($R1$ e $R2$). Neste caso, n nós são diretamente conectados ao roteador $R1$ e os outros n nós são conectados ao roteador $R2$. Por fim $R1$ é conectado a $R2$ formando uma *bottleneck*. A Figura 16 ilustra esta topologia. As características dos enlaces dependem dos cenários de simulação.

Figura 16 – Topologia *dumbbell*.



Fonte: Autoria própria.

Nas simulações executadas os nós da esquerda de $R1$ foram configurados para serem fontes TCP, tendo como receptores os nós a direita de $R2$. Ao início das simulações, cada fonte, em tempo aleatório (de um a oito segundos), estabelece uma conexão com o respectivo receptor e realiza a transferência de um arquivo de 5 MB. As simulações terminam quando todas as fontes completam a transmissão de seus arquivos.

Durante as simulações o número de fontes/receptores é igual ao número de fluxos, ou seja, cada fonte/receptor cria um único fluxo. Para definir o número ideal de fluxos a ser utilizado nos experimentos foi seguida a recomendação da RFC 7928 (KUHN *et al.*, 2016), que define três níveis de congestionamento: baixo, médio e alto. O nível baixo é caracterizado por apresentar uma perda (de segmentos no AQM) aproximada de 0,1%, o médio de 0,5% e o alto de 1%. Para encontrar estes níveis foram feitas simulações preliminares para cada cenário utilizado. Nestas foi utilizado o método *drop-tail* com i (≥ 1) fluxos, sendo verificado ao final da simulação se

a porcentagem de perdas se encaixava em alguma das três categorias. Até não encontrar um número i, para cada categoria, que correspondesse de forma aproximada a porcentagem de perda da respectiva categoria, i foi sendo incrementado em uma unidade e o processo foi repetido. Ao término deste processo obteve-se três valores para o número de fluxos, um para cada nível de congestionamento. Neste capítulo foram estudados três cenários, o número de fluxos utilizado em cada um dos cenários é apresentado na Tabela 3.

Tabela 3 – Número de fluxos utilizados nas simulações com fio.

	Baixo	*Médio*	*Alto*
Cenário 1	2	4	6
Cenário 2	8	17	24
Cenário 3	13	24	31

Fonte: Autoria própria.

Na avaliação do desempenho do EWT foram realizadas simulações e os resultados foram comparados àqueles das abordagens *drop-tail*, RED, adaptive RED (ARED), EWA e AWM. Para cada um dos métodos foram executadas 30 rodadas com diferentes sementes de números aleatórios. Considerou-se um intervalo de confiança de 95%. Os parâmetros do RED foram ajustados conforme o padrão encontrado no *kernel* do Linux, e os do EWA e do AWM de acordo com a recomendação dos autores em (KALAMPOUKAS *et al.*, 2002) (BARBERA *et al.*, 2007). As demais configurações de simulação consideradas importantes são apresentadas na Tabela 4.

Tabela 4 – Parâmetros de simulação redes com fio.

Parâmetro	**Valor**
Tamanho da memória	97 kB
min_{th} (RED)	tamanho_memória / 12
max_{th} (RED)	tamanho_memória / 4
Q_w (RED)	0.002
max_p (RED)	0.02
Modo EWT	Padrão
S_t (EWT)	29 kB [1]
Tamanho segmento TCP	1458 bytes
TCP	CUBIC

Fonte: Autoria própria.

Os resultados apresentados fazem uso das configurações aqui descritas, variando o atraso e a capacidade dos enlaces da topologia *dumbbell* (Figura 16).

[1] Foram realizados testes com diversos valores de S_t, sendo que os resultados foram semelhentes para os valores considerados.

4.1 MÉTRICAS

Para avaliação de desempenho foram calculadas as métricas a seguir, algumas indicadas pela RFC 7928.

- *Eficiência do TCP*: Esta métrica foi extraída de (SCHRAGE *et al.*, 2011) e representa a porcentagem de dados (em *bytes*) que não foram retransmitidos. Seu cálculo é definido por:

$$\frac{Bytes\ transmitidos - Bytes\ retransmitidos}{Bytes\ transmitidos} \times 100 \tag{18}$$

  Na apresentação dos resultados é exibido o valor médio da métrica calculada a partir da eficiência registrada por fluxo.

- *Tempo de transferência de arquivo*: O tempo de transferência de arquivo indica quantos segundos são gastos para que um arquivo seja transmitido com sucesso. Nos resultados é apresentada a média deste tempo calculada a partir dos tempos registrados por fluxo.

- *Goodput*: A quantidade de dados recebidos pela aplicação, em um período de tempo, define o *goodput*. Nos resultados é apresentado o valor médio do *goodput* (por segundo) dos fluxos em Mbits/seg. Esta métrica não contabiliza o recebimento de segmentos já recebidos (duplicados), diferentemente do *throughput*.

- *Justiça do Goodput*: Foi aplicado o índice de Jain (JAIN *et al.*, 1984) no *goodput* (gp) obtido pelos n fluxos da simulação, para assim verificar a justiça entre os fluxos. A equação utilizada para tal foi:

$$\frac{(\sum_{n=1}^{n} gp)^2}{n \cdot \sum_{n=1}^{n} gp^2} \tag{19}$$

- *Porcentagem média de perda*: Este valor representa a porcentagem média de perda de segmentos durante a simulação. As perdas consideradas são as registradas pelas camadas de rede e transporte. O calculo desta métrica é feito com a equação:

$$\frac{Total\ segmentos\ perdidos}{Total\ segmentos\ recebidos + Total\ segmentos\ perdidos} \times 100 \tag{20}$$

Para se obter as métricas foi utilizado o método *FlowMonitor* (CARNEIRO *et al.*, 2009), presente no simulador ns-3.

4.2 CENÁRIO 1

Este cenário foi baseado no usado para a avaliação do RED nos simuladores ns-2/3 (FLOYD; FALL, 1997) (TALAU, 2011). As configurações utilizadas para a topologia *dumbbell* estão na Tabela 5.

Tabela 5 – Parâmetros do Cenário 1

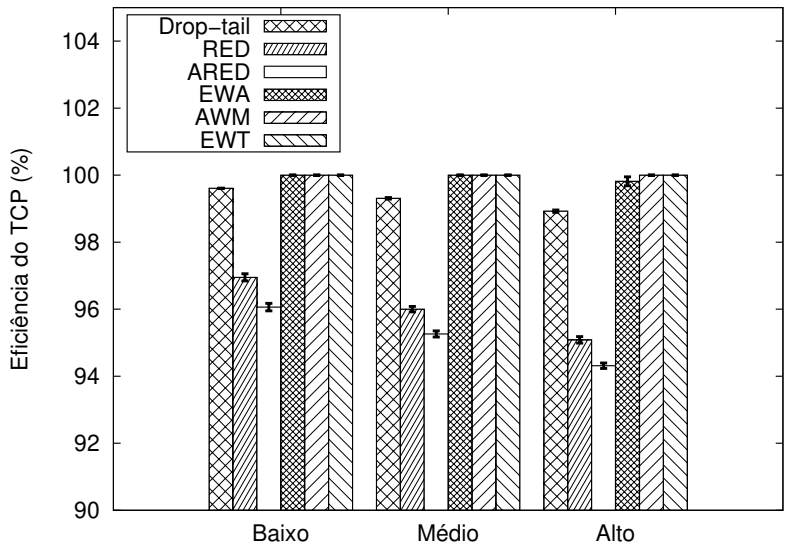
Parâmetro	Valor
Capacidade enlace (Fontes)	10 Mbps
Atraso de propagação (Fontes)	2 ms
Capacidade enlace (R1-R2)	1,5 Mbps
Atraso de propagação (R1-R2)	20 ms
Capacidade enlace (Receptores)	10 Mbps
Atraso de propagação (Receptores)	2 ms

Fonte: Autoria própria.

4.2.1 Resultados

Eficiência do TCP: Os resultados são apresentados na Figura 17. Os métodos *drop-tail*/RED/ARED tiveram uma perda na eficiência com o aumento do congestionamento, o ARED foi o menos eficiente, seguido pelo RED e pelo *drop-tail*. As técnicas de retorno da rede (AWM/EWA/EWT) mantiveram o TCP eficiente, porém com um alto nível de congestionamento o EWA perdeu eficiência.

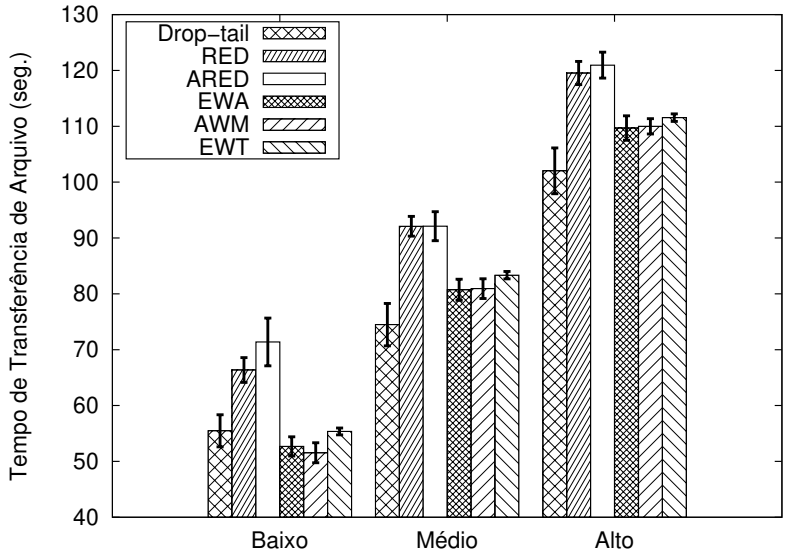
Figura 17 – Eficiência do TCP – Cenário 1.



Fonte: Autoria própria.

Tempo de transferência de arquivo: A Figura 18 exibe os resultados. Nas simulações os métodos RED/ARED apresentaram um tempo médio de transferência semelhante, assim como as técnicas de retorno da rede. Por fim, o *drop-tail* registrou o menor tempo.

Figura 18 – Tempo de transferência de arquivo – Cenário 1.



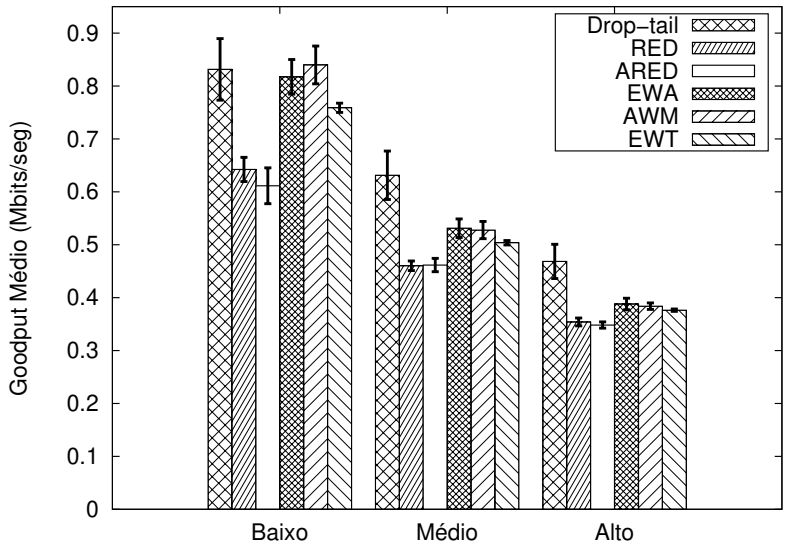
Fonte: Autoria própria.

Goodput: Os resultados são apresentados na Figura 19. O *drop-tail* apresentou o maior *goodput* médio, os métodos RED/ARED, considerando o intervalo de confiança, registraram um mesmo *goodput*. No nível baixo houve uma maior variação, onde o *drop-tail* obteve o maior *goodput*, seguido do AWM e do EWA. Quando o congestionamento foi médio/alto, o *drop-tail* manteve o maior *goodput*, já as técnicas de retorno da rede tiveram resultados semelhantes.

Justiça do Goodput: A Figura 20 exibe os resultados. Esta métrica indica a variação do *goodput* entre os fluxos, e pode ser utilizada como uma medida para verificar a justiça. Com congestionamento baixo, o *drop-tail*, que tinha registrado o maior *goodput* (Figura 19), foi o método mais injusto, seguido pelo EWA e o AWM. O EWT manteve a justiça entre os fluxos em todos os níveis de tráfego.

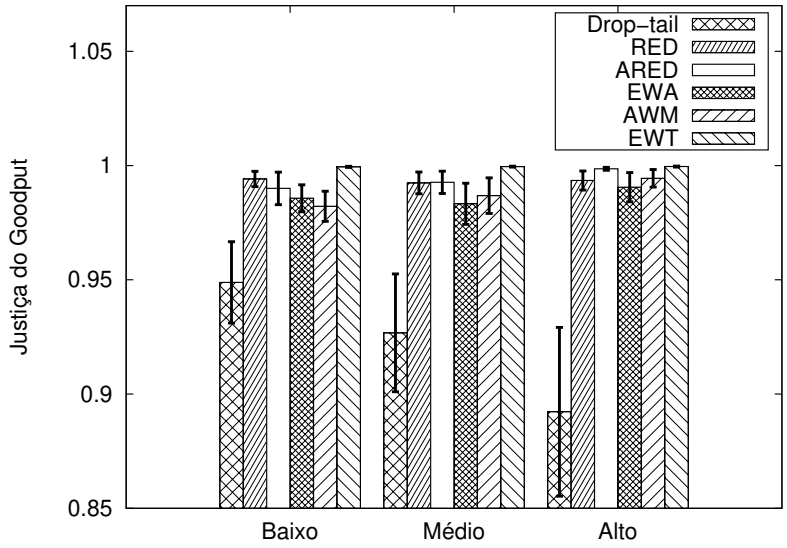
Porcentagem média de perda: Os resultados são apresentados na Figura 21. Eles indicam a porcentagem média de perdas registradas durante as simulações. Para os métodos RED/ARED o aumento foi proporcional ao nível de congestionamento, tendo o ARED registrado uma maior média de perdas. O *drop-tail*, que foi previamente utilizado para definir os níveis de congestionamento, teria de registrar uma média em torno de 0,1% para um congestionamento baixo, 0,5% para um médio e 1% para o alto, como foi o caso. O método EWA apresentou perdas com o tráfego alto. Com a utilização do AWM/EWT a porcentagem média de perdas foi zero.

Figura 19 – *Goodput* – Cenário 1.



Fonte: Autoria própria.

Figura 20 – Índice de justiça do *goodput* – Cenário 1.

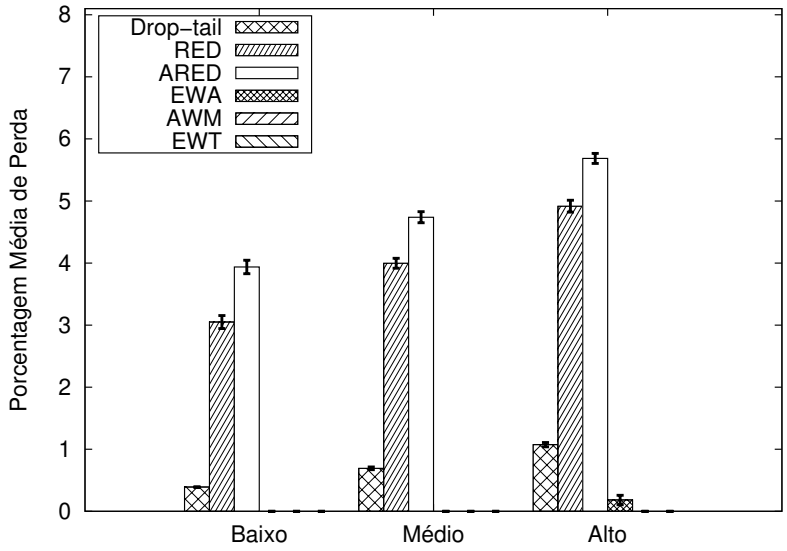


Fonte: Autoria própria.

4.2.2 Análise dos Resultados

O método EWT foi eficiente no controle de congestionamento. Sua aplicação impediu que a memória do roteador ficasse cheia, assim perdas foram evitadas. A não ocorrência de perdas se deu pelo correto ajuste de W_r dos segmentos TCP ACK, sendo feita a sua redução proporcionalmente ao nível de utilização da memória do roteador. Com isto foram enviados menos segmentos por instante de tempo e a fila não ficou cheia, evitando assim retransmissões e a redução da janela de congestionamento (W_c). A redução de W_c poderia impactar na redução

Figura 21 – Porcentagem média de perda – Cenário 1.



Fonte: Autoria própria.

da taxa de transmissão de segmentos.

O método *drop-tail* obteve uma diminuição no tempo médio de transferência de arquivos e um maior *goodput*, que pode ser explicado pela injustiça entre os fluxos. Esta injustiça é caracterizada pela sincronização das perdas no roteador da *bottleneck*, isto faz com que constantemente segmentos dos mesmos fluxos sejam perdidos, o que privilegia os fluxos que sobram, que com um menor número de perdas reduzem menos a W_c, assim transmitindo mais segmentos que os demais fluxos. A sincronização também leva a uma redução no tempo de transferência médio, pois ocorrendo menos disputa por recursos haverá um menor número de perdas e com isto será reduzido o tempo de transferência. O EWT não apresentou perdas e mesmo assim teve um tempo médio de transferência um pouco maior que o *drop-tail*, isto se deu pelo custo de manter a justiça entre os fluxos, pois transferindo dados de forma justa gera um aumento na disputa por recursos da rede.

4.3 CENÁRIO 2

Neste caso foi utilizado o cenário utilizado no trabalho presente em (GRAZIA *et al.*, 2015) (CASONI *et al.*, 2017). Este possui um valor de atraso na *bottleneck* característico de redes distantes ou comunicações via satélite. As configurações utilizadas na topologia estão na Tabela 6.

Tabela 6 – Parâmetros do Cenário 2

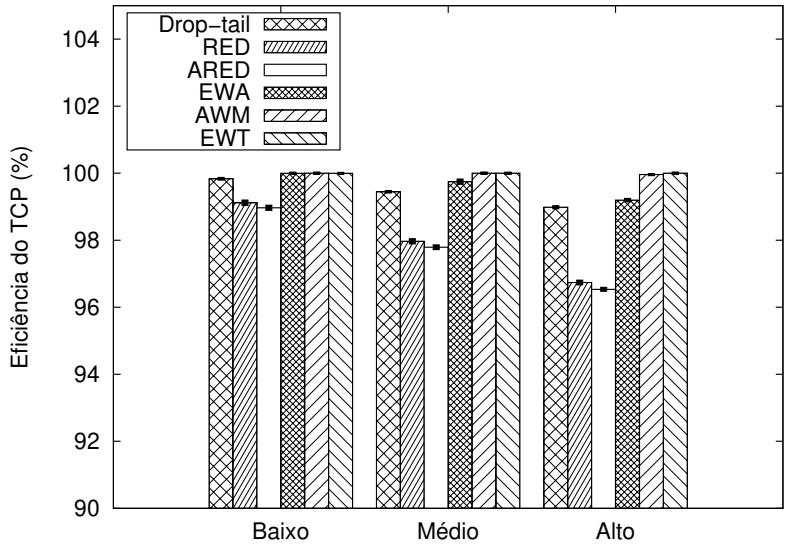
Parâmetro	Valor
Capacidade enlace (Fontes)	4 Mbps
Atraso de propagação (Fontes)	5 ms
Capacidade enlace (R1-R2)	4 Mbps
Atraso de propagação (R1-R2)	360 ms
Capacidade enlace (Receptores)	4 Mbps
Atraso de propagação (Receptores)	5 ms

Fonte: Autoria própria.

4.3.1 Resultados

Eficiência do TCP: Os resultados são apresentados na Figura 22. Os métodos *drop-tail*/RED/ARED tiveram uma queda gradual na eficiência. O método EWA apresentou uma queda na eficiência a partir do nível baixo. O EWT manteve-se eficiente independentemente do nível de congestionamento.

Figura 22 – Eficiência do TCP – Cenário 2.

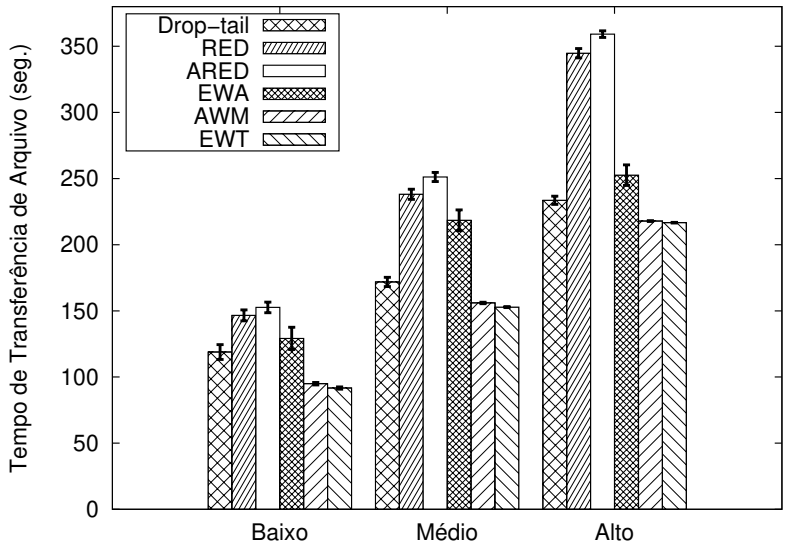


Fonte: Autoria própria.

Tempo de transferência de arquivo: A Figura 23 exibe os resultados. No nível de congestionamento baixo, o EWT apresentou um tempo de transferência menor que *drop-tail*/RED/ARED/EWA. Com um congestionamento médio, os métodos tiveram um aumento no tempo de transferência, tendo o EWT registrado o menor tempo. Com tráfego alto, o EWT também apresentou o menor tempo. E, no geral, o EWT e AWM apresentaram resultados semelhantes.

Goodput: Os resultados são mostrados na Figura 24. Na presença de tráfego médio o

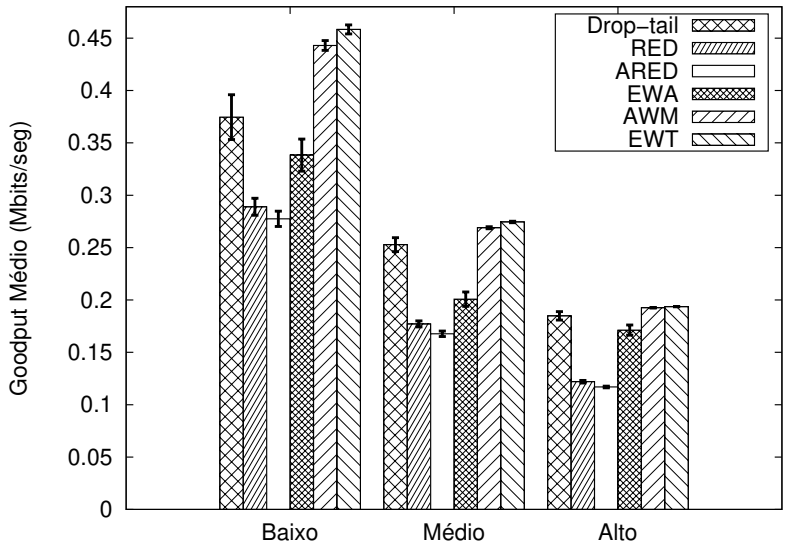
Figura 23 – Tempo de transferência de arquivo – Cenário 2.



Fonte: Autoria própria.

EWT teve um *goodput* superior ao *drop-tail*/RED/ARED/EWA. Com tráfego médio, tendo mais fluxos, houve uma queda no *goodput* médio, mas o EWT manteve seu desempenho em relação a outros métodos. Finalmente, com tráfego alto, o EWT continuou registrando a maior taxa de *goodput*. O AWM seguiu o EWT nos resultados.

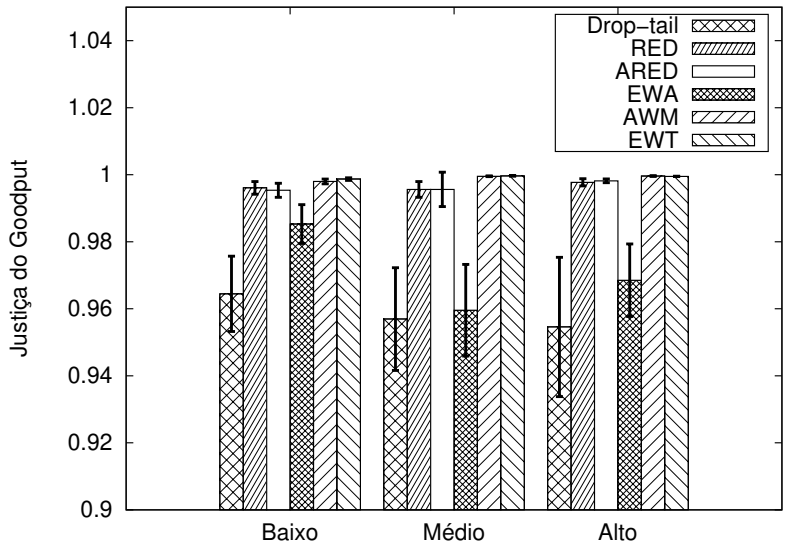
Figura 24 – *Goodput* – Cenário 2.



Fonte: Autoria própria.

Justiça do Goodput: A Figura 25 exibe os resultados. O *drop-tail* foi o método com maior injustiça no *goodput*. Considerando o intervalo de confiança os métodos RED/ARED/AWM/EWT tiveram uma justiça semelhante.

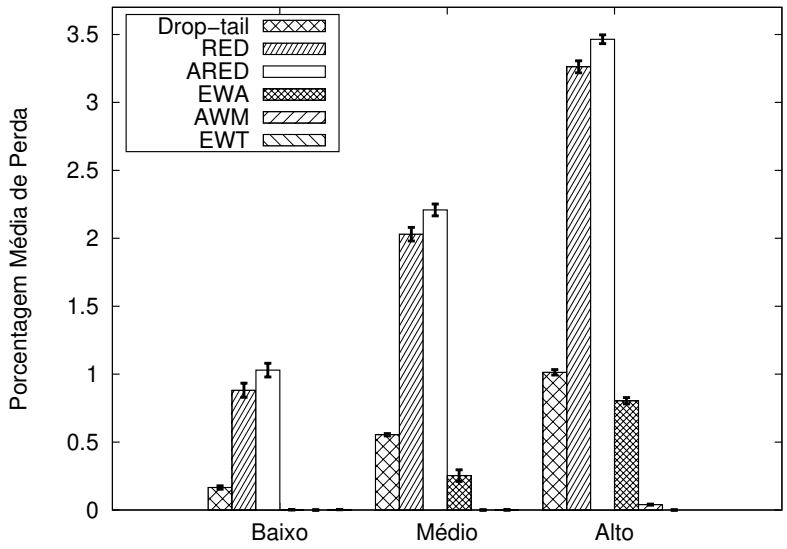
Figura 25 – Índice de justiça do *goodput* – Cenário 2.



Fonte: Autoria própria.

Porcentagem média de perda: Os resultados são apresentados na Figura 26. A maior porcentagem de perda ocorreu com os métodos RED/ARED, seguido pelo método *drop-tail*. O EWA apresentou perdas no nível médio/alto, já o AWM apenas no nível alto. No EWT a porcentagem foi zero, independentemente do nível de congestionamento.

Figura 26 – Porcentagem média de perda – Cenário 2.



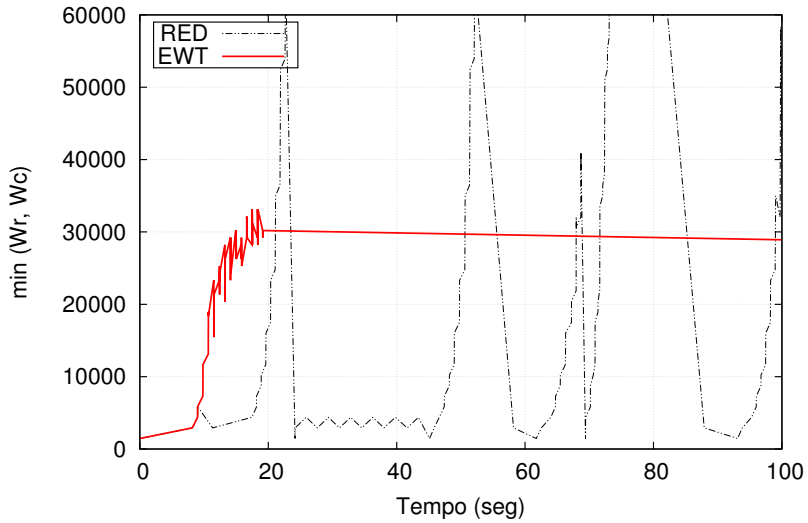
Fonte: Autoria própria.

4.3.2 Análise dos Resultados

Diferentemente do cenário anterior, este apresenta um grande atraso no *bottleneck*, que afetou o desempenho do TCP com os métodos tradicionais. O EWT demonstrou ser eficiente neste cenário, reduzindo significativamente o congestionamento, trazendo ganhos expressivos no tempo de transferência e no *goodput*, mantendo a justiça entre os fluxos. Não sendo registradas perdas, a eficiência do TCP foi mantida. O AWM foi o método que mais se aproximou dos resultados obtidos pelo EWT, porém o método registrou perdas.

Com base nesta simulação foi verificado o bom comportamento do EWT em redes com alto atraso. O seu mecanismo de ajuste das janelas trouxe ganhos ao TCP. Neste cenário os métodos tradicionais, como o RED, apresentaram um baixo desempenho se comparado ao EWT. O principal motivo disto é o incorreto ajuste da janela de congestionamento, que leva aos fluxos enviarem poucos segmentos. Isto pode ser observado na Figura 27. Para produzir o gráfico foi coletada a janela efetiva de transmissão ($min(W_r, W_{cwnd})$) de um [2] fluxo. Analisando a figura pode se observar que o RED levou o TCP a apresentar uma grande oscilação na janela de transmissão, enquanto o EWT manteve a janela equilibrada sem atingir valores baixos.

Figura 27 – Janela utilizada na transmissão de segmentos para um fluxo do Cenário 2.



Fonte: Autoria própria.

[2] Um fluxo foi o suficiente para exibir o comportamento, pois o fluxo registrado foi dos métodos RED e EWT, e ambos apresentaram uma justiça equivalente entre os fluxos.

4.4 CENÁRIO 3

Este cenário foi elaborado para verificar a eficiência do EWT na presença de tráfego UDP. Foi utilizada a topologia *dumbbell* (Figura 16) com as configurações da Tabela 7.

Tabela 7 – Parâmetros do Cenário 3

Parâmetro	Valor
Capacidade enlace (Fontes)	100 Mbps
Atraso de propagação (Fontes)	2 ms
Capacidade enlace (R1-R2)	45 Mbps
Atraso de propagação (R1-R2)	80 ms
Capacidade enlace (Receptores)	100 Mbps
Atraso de propagação (Receptores)	2 ms

Fonte: Autoria própria.

Foram feitas simulações variando-se o número de fluxos TCP na presença de tráfego UDP concorrente. O tráfego UDP foi gerado de forma constante com velocidade de 13,35 Mbps (30% da capacidade do enlace) sendo transmitidos segmentos de 1400 *bytes*. Para isto foi instalado um cliente UDP no primeiro nó a esquerda de $R1$ que transmite dados a um receptor UDP localizado no primeiro nó ao lado direito de $R2$.

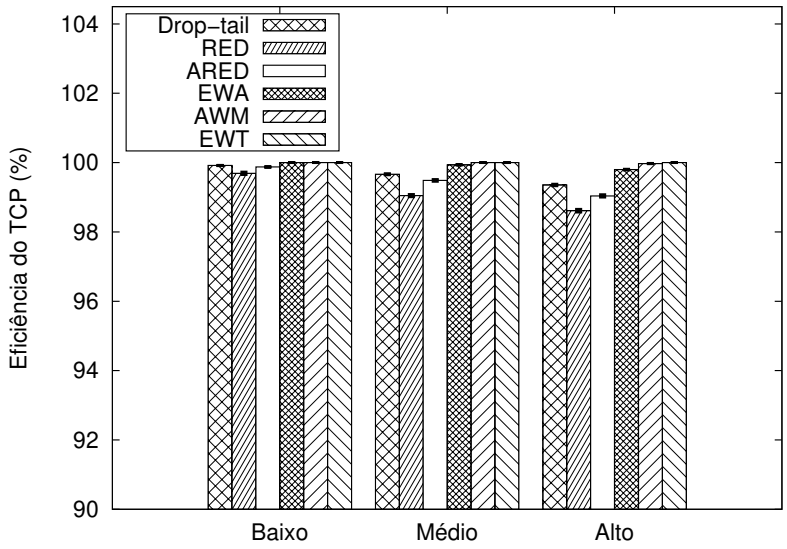
4.4.1 Resultados

Eficiência do TCP: Os resultados são apresentados na Figura 28. A eficiência dos métodos *drop-tail*/RED/ARED foi semelhante, com uma pequena diferença para congestionamento moderado e alto. O EWA/AWM tiveram uma pequena perda de eficiência com um congestionamento alto. O EWT manteve o TCP eficiente nos três níveis de congestionamento.

Eficiência do UDP: Como este cenário apresentou tráfego UDP, também foi calculado a eficiência do UDP (calculada utilizando a mesma equação da eficiência do TCP). Na Figura 29 os resultados. Com tráfego baixo a eficiência dos métodos foi semelhante. No nível médio as abordagens *drop-tail*/RED/ARED tiveram perda de eficiência e com tráfego alto houve uma queda ainda maior. No geral as técnicas de retorno da rede mantiveram o UDP eficiente.

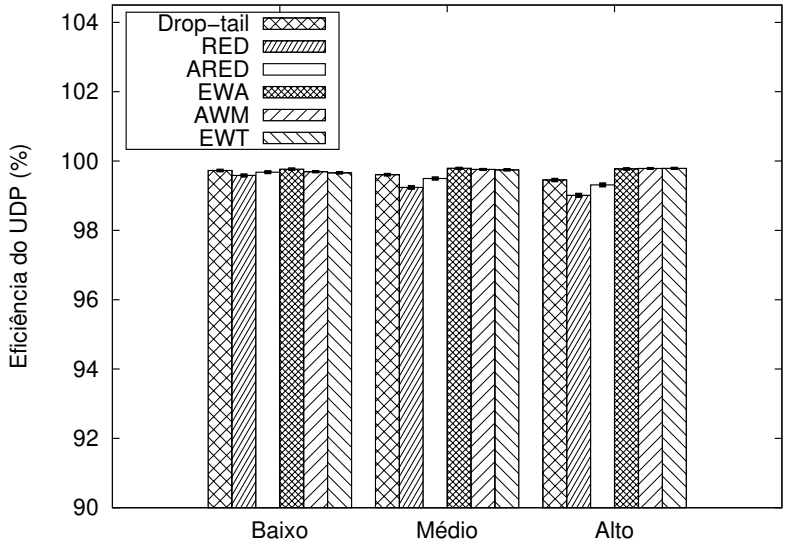
Tempo de transferência de arquivo: A Figura 30 exibe os resultados. O EWT obteve um menor tempo de transferência para os três níveis de congestionamento. Com tráfego médio e alto, considerando o intervalo de confiança, os métodos *drop-tail*/RED/ARED apresentaram um tempo de transferência similar. O método EWA apresentou o pior tempo de transferência. O AWM apresentou resultados semelhantes ao EWT.

Figura 28 – Eficiência do TCP – Cenário 3.



Fonte: Autoria própria.

Figura 29 – Eficiência do UDP – Cenário 3.

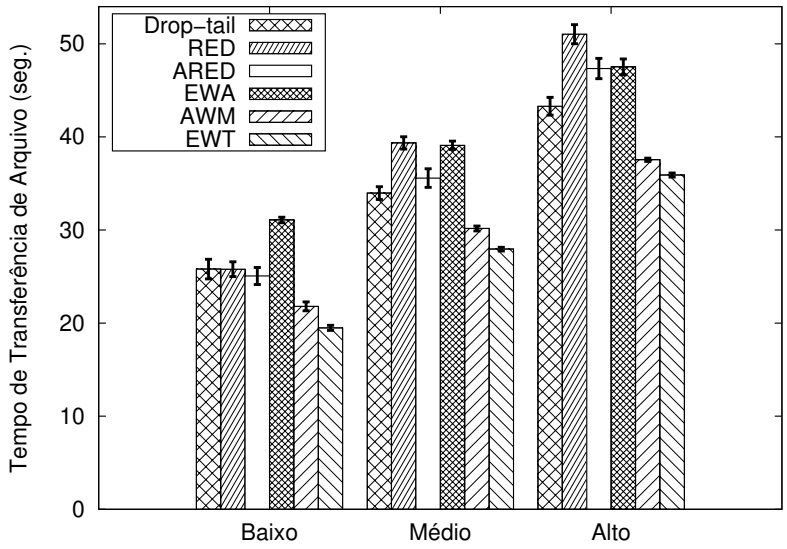


Fonte: Autoria própria.

Goodput: Os resultados são mostrados na Figura 31. Com o nível moderado de congestionamento, o EWT registrou o maior *goodput*. Também no nível médio, considerando o intervalo de confiança, os métodos *drop-tail*/RED/ARED tiveram um goodput semelhante. Com tráfego alto, o mesmo ocorreu. O EWA apresentou o menor valor de *goodput*. O AWM apresentou resultados semelhantes aos do EWT nos níveis de tráfego médio/alto.

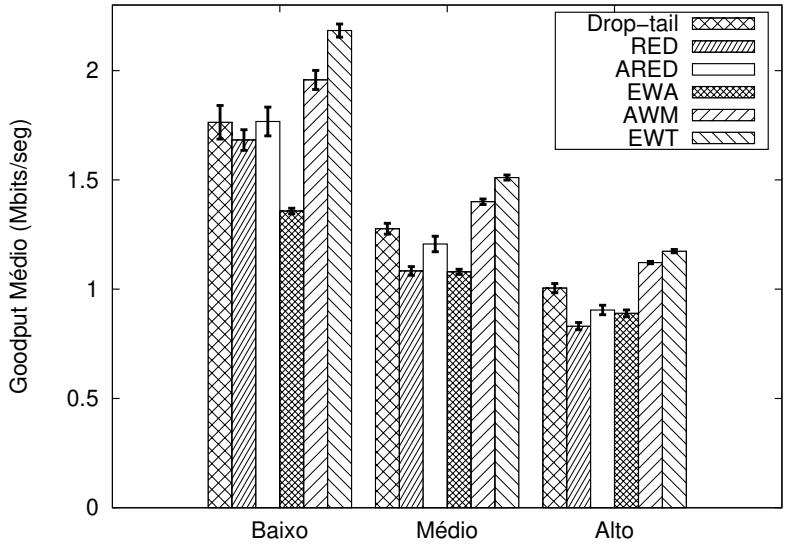
Justiça do Goodput: A Figura 32 exibe os resultados. Nos diferentes níveis de congestionamento o método *drop-tail* foi o método mais injusto. O EWA manteve-se justo até o nível

Figura 30 – Tempo de transferência de arquivo – Cenário 3.



Fonte: Autoria própria.

Figura 31 – *Goodput* – Cenário 3.

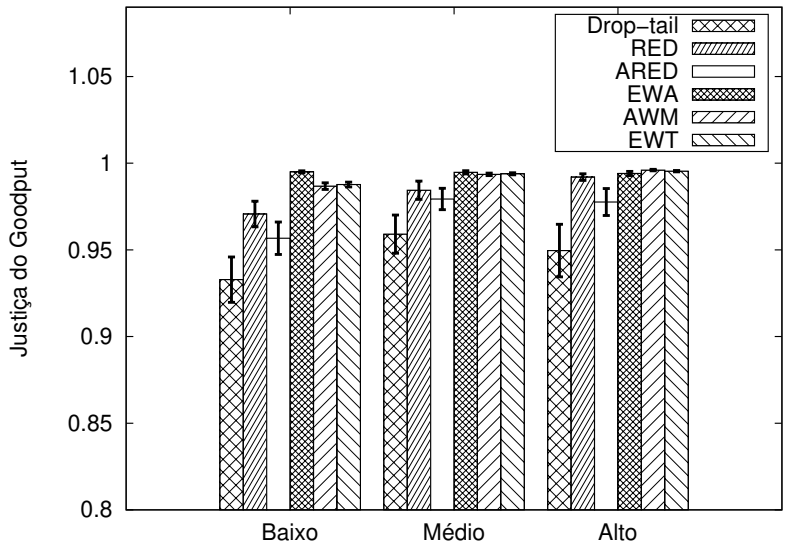


Fonte: Autoria própria.

médio, após este ele teve uma perda de justiça. Nos níveis médio/alto, considerando o intervalo de confiança, o RED/ARED/AWM/EWT tiveram uma mesma justiça.

Porcentagem média de perda: Como neste cenário houve a presença de tráfego TCP e UDP, os resultados são apresentados na Tabela 8. A porcentagem média de perda cresceu conforme o aumento do congestionamento, tendo registrado a maior porcentagem o método RED. Os métodos EWA/AWM apresentaram perdas nos níveis médio/alto. O EWT não registrou perdas em nenhum dos níveis.

Figura 32 – Índice de justiça do *goodput* – Cenário 3.



Fonte: Autoria própria.

Tabela 8 – Porcentagem média de perda do TCP e UDP – Cenários 3.

	Baixo		**Médio**		**Alto**	
	TCP	UDP	TCP	UDP	TCP	UDP
drop-tail	0.07	0.029	0.31	0.31	0.52	0.43
RED	0.11	0.07	0.54	0.37	0.83	0.62
ARED	0.09	0.05	0.41	0.29	0.62	0.50
EWA	0	0	0.07	0.02	0.19	0.05
AWM	0	0	0	0	0.04	0.02
EWT	0	0	0	0	0	0

Fonte: Autoria própria.

4.4.2 Análise dos Resultados

A presença do tráfego UDP não teve influência negativa no desempenho do EWT. Em grande parte das métricas o EWT obteve melhores resultados que os demais métodos, com destaque para o aumento no *goodput*, diminuição do tempo médio de transferência de arquivos até 35% e a ausência de perdas. A justiça do EWT com tráfego médio e alto foi preservada, mas com tráfego baixo ela foi um pouco inferior (1,40%) que a do AWM.

Mesmo o EWT sendo projetado para atuar no controle de congestionamento do TCP, ele operou de forma satisfatória na presença de tráfego UDP. O EWT manteve a ausência de perdas, isto se deu pelo seu mecanismo de ajuste de janelas, que realiza o cálculo com base na quantidade de bytes disponíveis na memória do roteador, não considerando apenas o tráfego TCP. Naturalmente, se houver tráfego UDP excessivo, perdas ocorrerão. Mas a atuação do EWT em conjunto com o controle de congestionamento presente no TCP rapidamente fará com que

os fluxos ajustem o tamanho da rajada para mitigar o congestionamento, isto pois o retorno do EWT é menor que um RTT, porque ele atua diretamente nos segmentos TCP ACK que passam pelo roteador. Outro ponto interessante da aplicação do EWT é que o UDP acaba se tornando prioritário, pois o TCP é controlado e o UDP mantém sua taxa. Como normalmente os fluxos UDP transmitem dados em tempo real, este comportamento pode ser desejado.

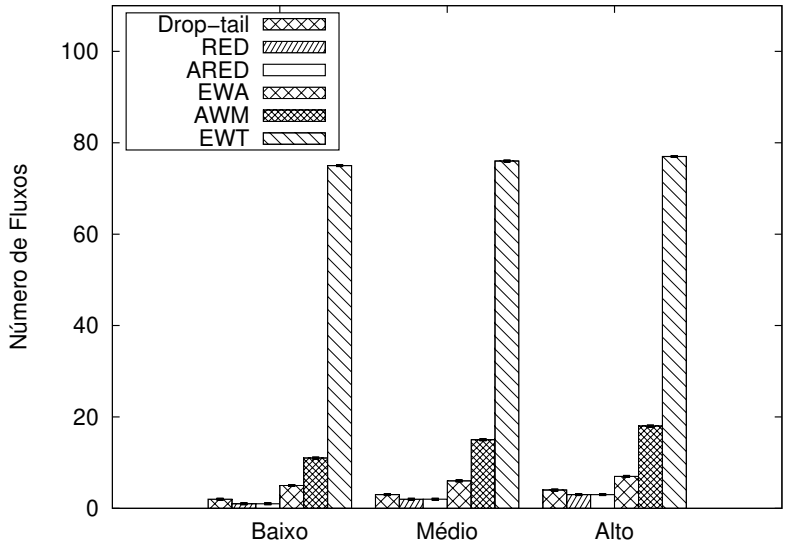
4.5 NÚMERO DE FLUXOS TCP PARA ATENDER A RFC 7928

A RFC 7928 define três níveis de congestionamento com base na taxa de perda de segmentos. Nas simulações, previamente apresentadas neste capítulo, os níveis foram encontrados pelo aumento do número de fluxos TCP até que a taxa de perda de cada nível fosse alcançada. Como a RFC indica, nenhum esquema AQM foi usado nesta etapa.

Nesta seção, usamos o RED, ARED, EWA, AWM e EWT para encontrar os três níveis de congestionamento sugeridos pela RFC. Isso foi feito com o objetivo de encontrar o número de fluxos que cada método pode satisfazer em cada nível de congestionamento.

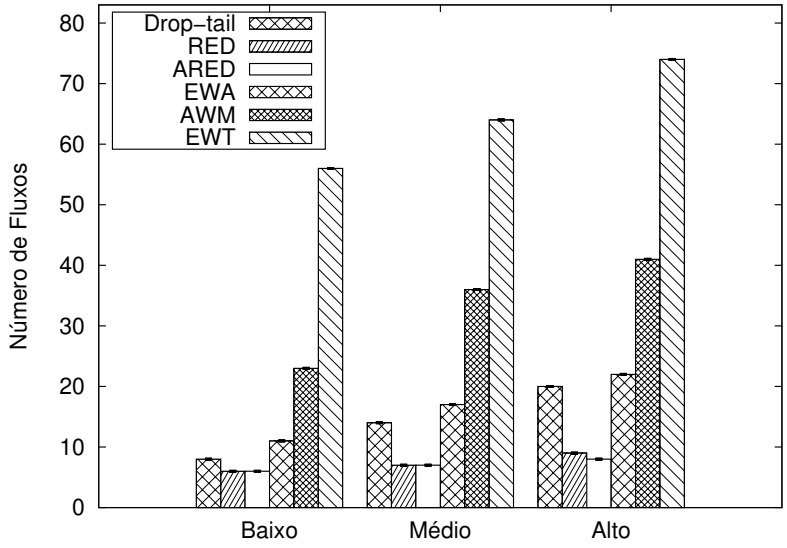
Os resultados são apresentados nas Figuras 33, 34 e 35. O método que atingiu o maior número de fluxos para cada nível de congestionamento foi o EWT. O método permitiu a existência de um número considerável de fluxos, sendo, em média, 59,83% melhor que seu melhor concorrente e 82,59% melhor quando nenhum esquema de AQM foi usado. O algoritmo do EWT foi responsável por esse bom resultado. Com a redução de W_r de segmentos ACK, menos segmentos foram enviados quando o número de fluxos aumenta, devido a isso, a memória só ficará cheia e as perdas ocorrerão principalmente quando $\sum_{f=0}^{nf} f_i \times MSS > queueSize$ (em que nf é o número total de fluxos) ou quando um tráfego que não é TCP passar pelo gateway.

Figura 33 – Número de fluxos necessários para atender a RFC 7928 no Cenário 1.



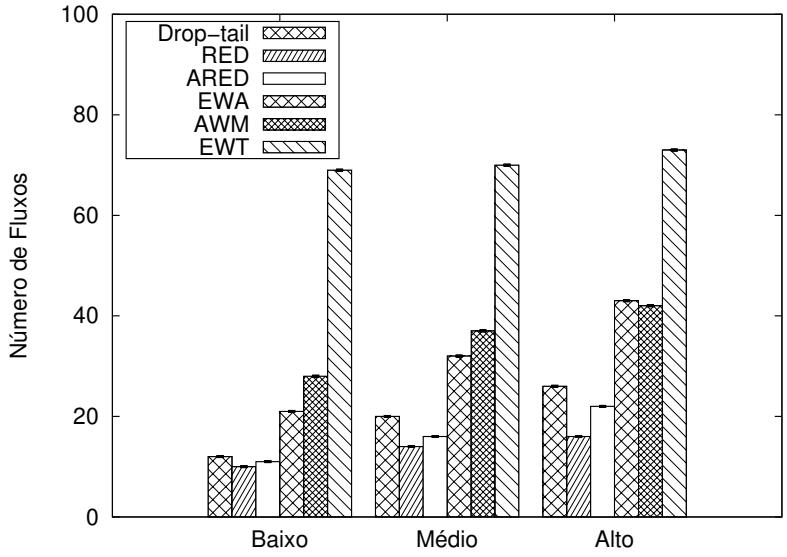
Fonte: Autoria própria.

Figura 34 – Número de fluxos necessários para atender a RFC 7928 no Cenário 2.



Fonte: Autoria própria.

Figura 35 – Número de fluxos necessários para atender a RFC 7928 no Cenário 3.



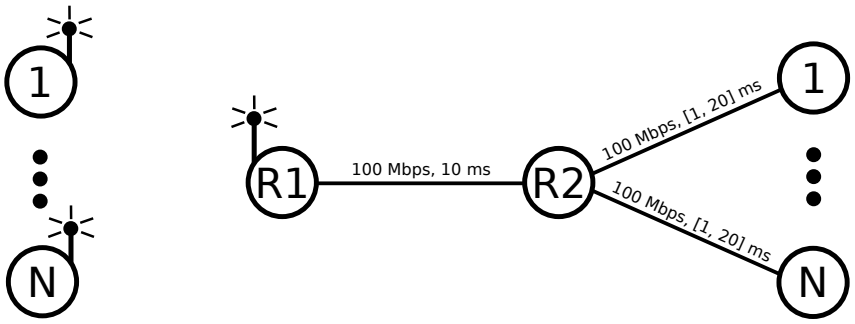
Fonte: Autoria própria.

5 APLICAÇÃO DO MÉTODO EM PONTOS DE ACESSO 802.11

Este capítulo trata da aplicação do EWT em pontos de acesso (APs) 802.11. Neste tipo de ambiente o atraso até as fontes tende a ser significativo, em função disso foi utilizado o EWT versão cliente.

O cenário escolhido para estudo é apresentado na Figura 36, sendo utilizado em diversos trabalhos, por exemplo (HA; CHOI, 2006) (BUJARI *et al.*, 2015) (PALAZZI *et al.*, 2009) (HUANG *et al.*, 2010) e (SEYEDZADEGAN *et al.*, 2007). A importância do estudo deste cenário reside no fato do mesmo ser bastante similar ao ambiente encontrado nas residências atualmente. O ambiente é composto por clientes sem fio conectados a um ponto de acesso (R1). O AP é conectado ao roteador R2 através de um enlace de 100 Mbps com 10 ms de atraso. R2 faz ligação com n servidores através de enlaces de 100 Mbps com atraso aleatório no intervalo $[1, 20]$ ms.

Figura 36 – Cenário com ponto de acesso 802.11.



Fonte: Autoria própria.

As simulações foram conduzidas utilizando n clientes e n servidores. Ao início de cada rodada cada cliente i estabelece uma conexão com um servidor i e este inicia a transferência de 5 MB de dados. Cada servidor i inicia a transmissão após um tempo aleatório de espera de até oito segundos. Assim como no capítulo anterior, foi seguida a RFC 7928 para definir o número de fluxos a serem utilizados nas simulações. A RFC define 0,1% de perdas (no AQM) para o nível baixo, 0,5% para o médio e 1% para o alto, porém, diferentemente do cenário do capítulo anterior, neste não foi possível encontrar um número de fluxos que produzisse uma porcentagem de perdas de acordo com indicado pela RFC. Isto ocorreu pelo atraso existente entre o AP e os servidores em conjunto com características do meio sem fio. Não sendo possível encontrar estes números, utilizamos os mais próximos. O nível baixo ficou definido com 0,05%, o médio em

0,62% e o alto em 1,28%. O número de fluxos para definir o nível baixo foi de três, cinco para o médio e sete para o alto.

Na avaliação de desempenho foram utilizadas as abordagens *drop-tail*, RED, adaptive RED (ARED) e EWA. O AWM não foi utilizado pela inviabilidade em se contabilizar o número de fluxos neste tipo de ambiente, parâmetro este, essencial para a sua operação. Os métodos foram instalados na interface de rede sem fio do AP. Na execução das simulações, para cada um dos métodos foram executadas 30 rodadas com diferentes sementes de números aleatórios, sendo considerado um intervalo de confiança de 95%. Os demais parâmetros de simulação são apresentados na Tabela 9.

Tabela 9 – Parâmetros de simulação do cenário com ponto de acesso.

Parâmetro	Valor
Tamanho da memória	97 kB
min_{th} (RED)	tamanho_memória / 12
max_{th} (RED)	tamanho_memória / 4
Q_w (RED)	0.002
max_p (RED)	0.02
S_t (EWT)	29 kB
Tamanho segmento TCP	1458 bytes
TCP	CUBIC
Padrão WiFi	802.11g
Modelo de Perda de Propagação	Friis

Fonte: Autoria própria.

Na seção a seguir são apresentados os resultados fazendo uso das métricas previamente apresentadas no capítulo anterior.

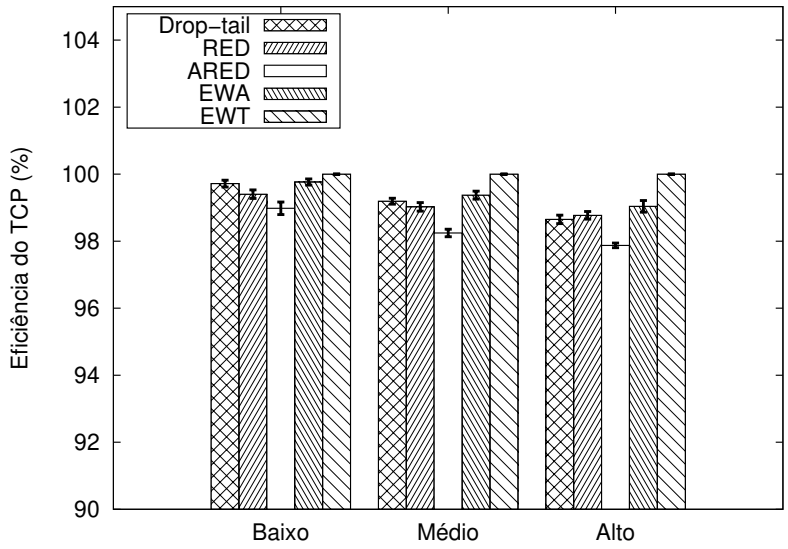
5.1 RESULTADOS

Eficiência do TCP: Os resultados são apresentados na Figura 37. As abordagens *drop-tail*, RED, ARED e EWA tiveram uma queda na eficiência com o crescimento do nível de congestionamento. O EWT manteve a eficiência do TCP nos três níveis.

Tempo de transferência de arquivo: A Figura 38 exibe os resultados. Naturalmente, com o aumento de tráfego, característico de cada nível, o tempo de transferência foi crescente para todos os métodos. Considerando o intervalo de confiança os métodos *drop-tail* e EWA apresentaram um mesmo tempo para o congestionamento baixo e médio. O EWT registrou o menor tempo nos três níveis.

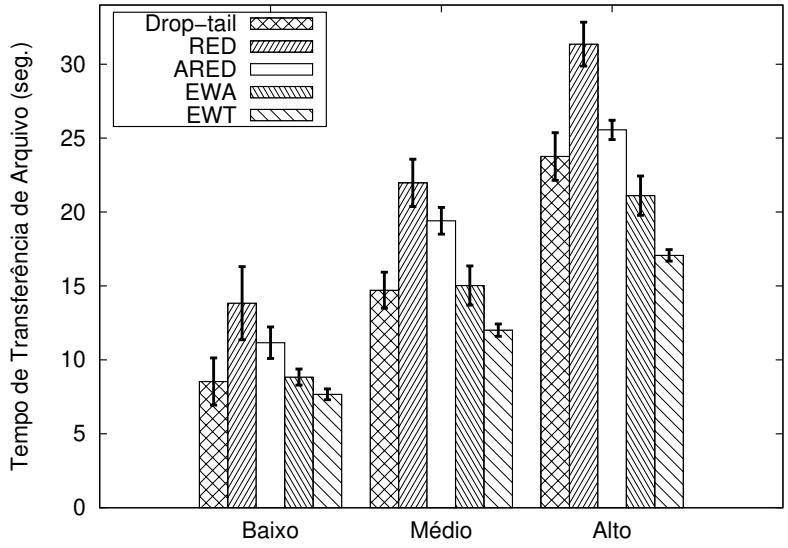
Goodput: Os resultados são mostrados na Figura 39. Levando em conta o intervalo de confiança, nos níveis baixo e médio o *drop-tail* e o EWT tiveram resultados semelhantes, tendo

Figura 37 – Eficiência do TCP – cenário com ponto de acesso.



Fonte: Autoria própria.

Figura 38 – Tempo de transferência de arquivo – cenário com ponto de acesso.



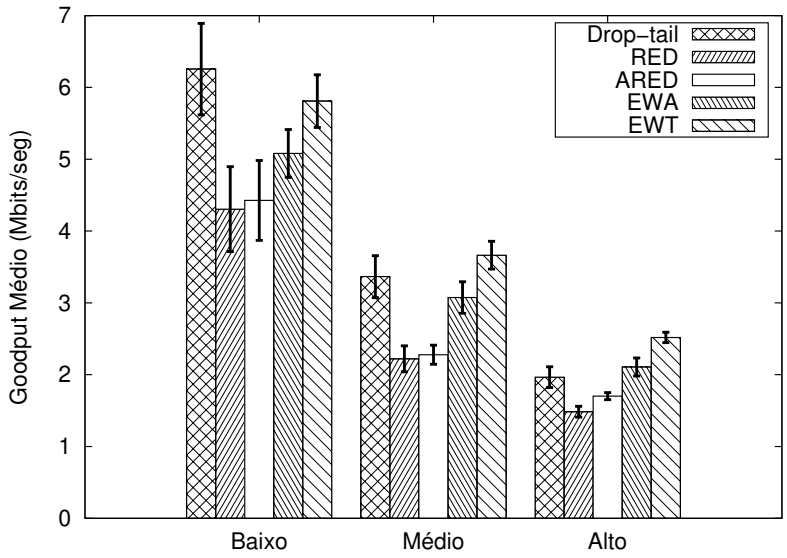
Fonte: Autoria própria.

o *drop-tail* registrado um melhor desempenho no nível baixo e o EWT um maior no médio. As abordagens RED/ARED/EWA tiveram um desempenho inferior nos três níveis. No nível alto o EWT atingiu o maior *goodput*.

Justiça do goodput: A Figura 40 exibe os resultados. As abordagens *drop-tail* e RED registraram uma menor justiça. Os demais métodos apresentaram resultados semelhantes.

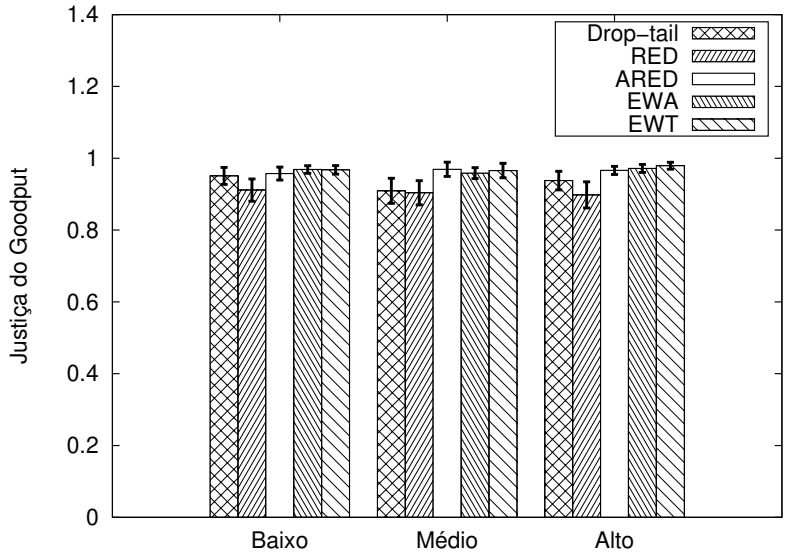
Porcentagem média de perda: Os resultados são apresentados na Figura 41. Os métodos *drop-tail*, RED, ARED e EWA tiveram um aumento na porcentagem de perdas a cada nível. O

Figura 39 – *Goodput* – cenário com ponto de acesso.



Fonte: Autoria própria.

Figura 40 – Índice de justiça do *goodput* – cenário com ponto de acesso.



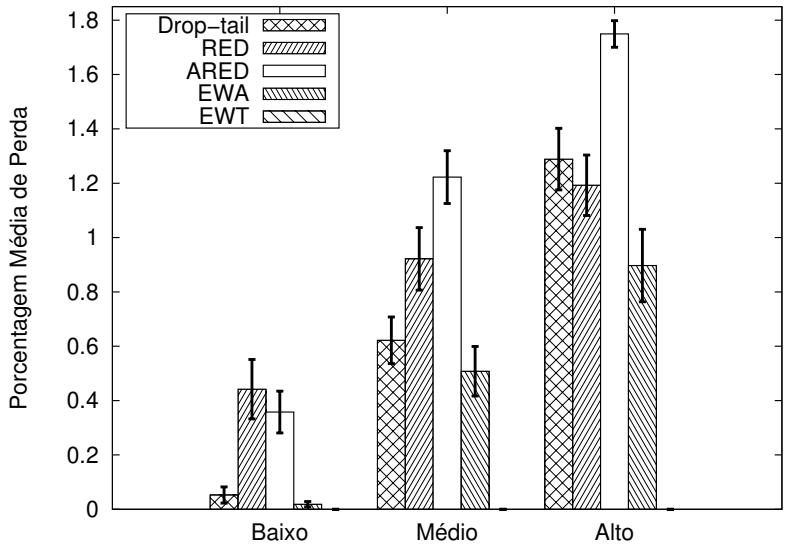
Fonte: Autoria própria.

EWT não registrou perdas nas camadas de rede e transporte, por isto ele não aparece no gráfico.

5.2 ANÁLISE DOS RESULTADOS

Verificando os resultados apresentados pode-se observar que o EWT versão cliente foi eficaz no controle de congestionamento, obtendo maior desempenho que as demais abordagens, reduzindo o tempo de transferência, tendo um *goodput* superior, sendo mantida uma justiça satisfatória, e não registrando perdas. Isto ocorreu principalmente pelo controle da fila da interface

Figura 41 – Porcentagem média de perda – cenário com ponto de acesso.



Fonte: Autoria própria.

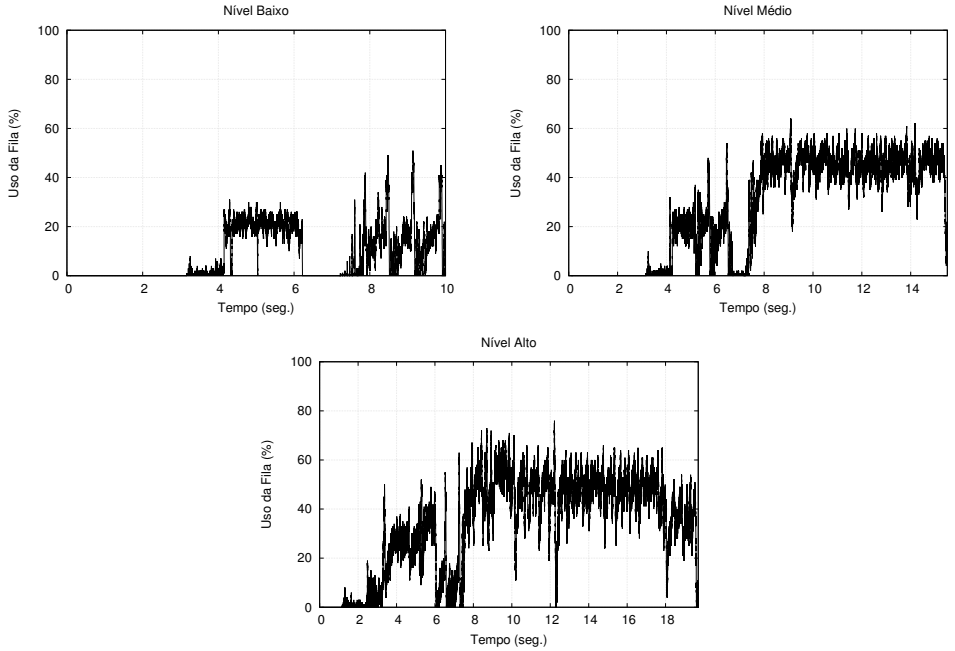
sem fio do AP — a Figura 42 exibe o uso desta fila para os três níveis de congestionamento. Este controle teve início pela alteração do valor da janela do receptor (W_r) dos segmentos ACKs dos clientes. Diferentemente do EWT original, o EWT versão cliente não utiliza diretamente a quantidade de bytes disponíveis (B_a), mas sim uma média móvel de B_a ($\overline{B_a}$) e usa B_a quando $\overline{B_a} > B_a$, isto para evitar ao máximo a ocorrência de perdas na fila. Adotando este comportamento o valor de W_r foi ajustado, e por fim utilizado pelos servidores no controle das transmissões.

O uso da média móvel foi essencial para manter um controle da fila da interface sem fio do AP. Caso o EWT fizesse uso do valor instantâneo de B_a, a W_r a ser utilizada pelos servidores não iria refletir no estado da rede no momento de seu uso, pois o tempo que leva para W_r chegar até o servidor já não iria condizer com o estado da memória no tempo t + $atraso(AP,\ servidor)$. Além disto existe o atraso do servidor até o AP. Logo, pode-se afirmar que o tempo do ajuste de W_r até que cheguem os segmentos vindos do servidor é de $t + RTT(AP,\ servidor)$. Pela existência deste tempo, o uso da média móvel foi fundamental para a realização do controle de congestionamento.

5.3 NÚMERO DE FLUXOS TCP PARA ATENDER A RFC 7928

Nas simulações, apresentadas neste capítulo, os níveis de congestionamento definidos pela RFC 7928 foram encontrados pelo aumento do número de fluxos TCP até que fosse

Figura 42 – Utilização da fila pelo EWT para os três níveis de congestionamento – cenário com ponto de acesso.

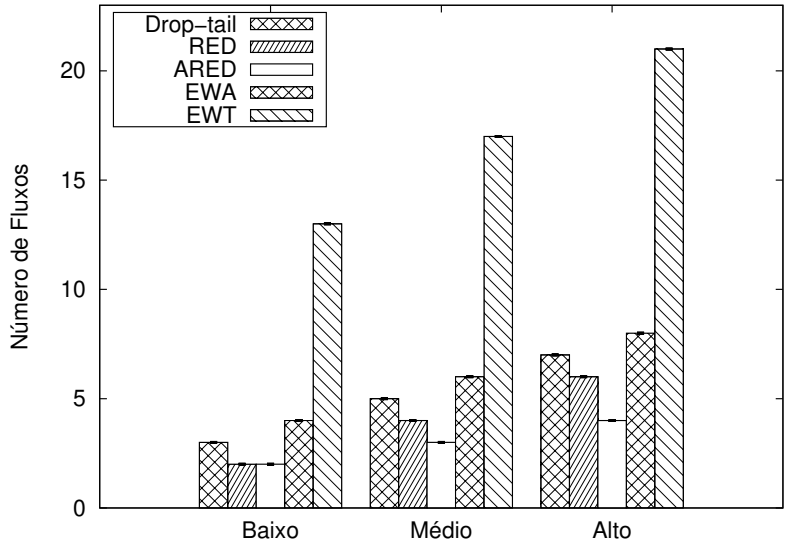


Fonte: Autoria própria.

alcançada uma taxa de perda compatível com cada nível. Como a RFC indica, nenhum esquema AQM foi usado nesta etapa. Nesta seção foram utilizadas as abordagens RED, ARED, EWA, e EWT para encontrar os três níveis de congestionamento sugeridos pela RFC. Isso foi feito com o objetivo de encontrar o número de fluxos que cada método pode atender em cada nível de congestionamento.

Os resultados são apresentados na Figura 43. O método que atingiu o maior número de fluxos para cada nível de congestionamento foi o EWT. Comparado ao seu melhor concorrente, ele permitiu a existência de em média 65,2% mais fluxos em cada nível, e quando nenhum esquema de AQM foi utilizado a média foi de 71,3%. O EWT possibilitou um maior número de fluxos pelo controle gerado na fila da interface sem fio do AP.

Figura 43 – Número de fluxos necessários para atender a RFC 7928 no cenário com ponto de acesso.



Fonte: Autoria própria.

6 ANÁLISE DO EWT EM REDES SEM FIO AD HOC

Já é conhecido na literatura, por exemplo em (ZHANG *et al.*, 2008) (ZHANG; FENG, 2009) (BHANUMATHI; DHANASEKARAN, 2010), que o TCP não funciona adequadamente onde o meio físico é sujeito a falhas, como em redes sem fio. Nas redes ad hoc com múltiplos nós este problema é ainda maior (CHEN *et al.*, 2003) (FU *et al.*, 2003) (ZHAI *et al.*, 2005). A busca pela melhora de desempenho do TCP em tais redes continua a ser investigada, por isto este tipo de ambiente foi explorado neste capítulo.

Foram utilizados dois cenários ad hoc e ao final feita uma verificação de quanto o tamanho dos segmentos e da memória dos roteadores tem influência no desempenho do EWT. O primeiro cenário é composto por apenas três conjuntos de nós: fontes, roteador e receptores; este cenário reduzido pode existir em diversas redes e também pode avaliar a eficiência do EWT sem grandes interferências. Já o segundo é composto por uma topologia grid 4x4.

No primeiro cenário foi utilizado o algoritmo de roteamento *On-demand Distance Vector*) (PERKINS *et al.*, 2003) e no segundo o *Dynamic Source Routing*) (HU *et al.*, 2007). No cenário 1 foi adotado o AODV para que o nó central fosse utilizado como roteador na comunicação entre os nós da esquerda/direita. Já no segundo a escolha deu-se por o DSR ter sido adotado em outros trabalhos que utilizaram a topologia grid, como em (GAJJAR; GUPTA, 2008). Cabe ressaltar que o algoritmo de roteamento não influenciará o desempenho do EWT, poderiam ter sido feitas rotas fixas, porém, por questões de simplificação de código no ns-3, além do uso de algoritmos específicos em trabalhos relatados, tais algoritmos foram adotados.

6.1 CENÁRIO 1

Os nós foram posicionados para existir um nó central com um conjunto de n nós a esquerda, e n nós a direita. O nó central está 100 metros distante dos nós da esquerda e direita. A Figura 44 ilustra a topologia.

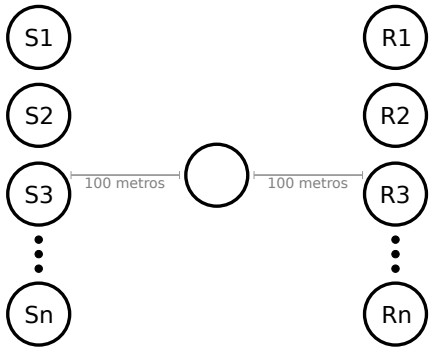
Nas simulações os nós da esquerda foram configurados para serem fontes TCP ($[S1, Sn]$), e os da direita receptores TCP ($[R1, Rn]$). Durante a simulação cada fonte fez uma conexão com um receptor e dados foram transmitidos até o término da simulação. Os parâmetros do RED foram ajustados conforme orientações em (FLOYD; JACOBSON, 1993). Foram realizados testes preliminares, em redes ad hoc, variando o valor do ponto de início (S_t)

Tabela 10 – Configurações para as redes ad hoc.

Parâmetro	Valor
Enlace	IEEE 802.11a
Modelo de Propagação	Yans (ns-3)
Transmissão	OFDM 6 Mbps
Implementação TCP	New Reno
Tipo de Tráfego	CBR BulkSendApplication (ns-3)

Fonte: Autoria própria.

Figura 44 – Topologia ad hoc utilizada no Cenário 1.



Fonte: Autoria própria.

do EWT, verificou-se que quando a fila está com 40% de ocupação o EWT foi mais eficaz no controle de congestionamento, por isto a variável S_t foi ajustada para este valor. Outras configurações são apresentadas nas Tabelas 10 e 11.

Tabela 11 – Parâmetros de Simulação – Cenário 1.

Parâmetros	Valores
Tamanho da Fila	97 KB
min_{th} (RED)	19.4 KB
max_{th} (RED)	58.2 KB
Q_w (RED)	0.002
max_p (RED)	0.02
Modo EWT	Padrão
S_t (EWT)	38,8 KB
Tamanho segmentos TCP	1000 bytes
TCP	New Reno

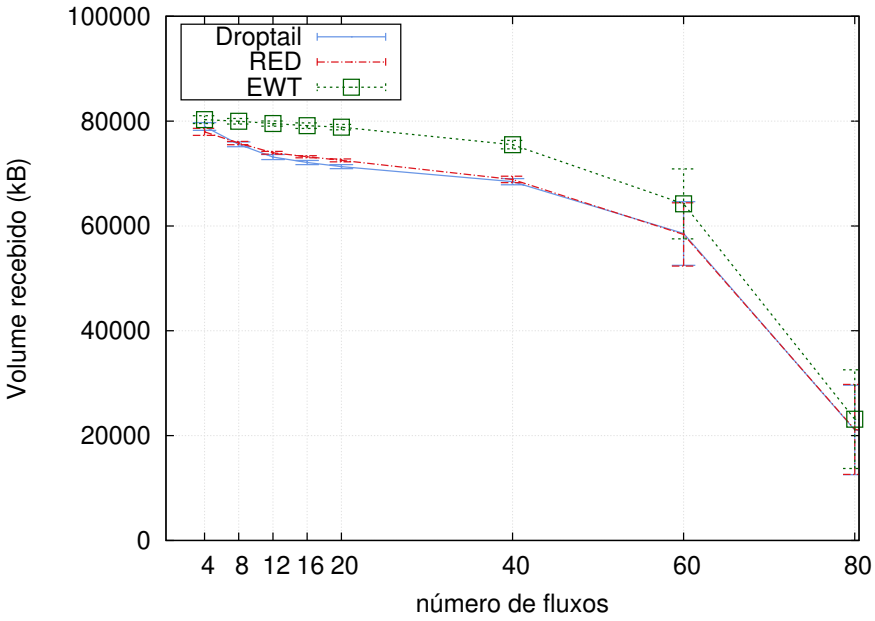
Fonte: Autoria própria.

O EWT foi comparado com o *drop-tail* e RED através de simulações no ns-3. Cada simulação foi executada durante o tempo de seis minutos. As simulações foram executadas com 4, 8, 12, 16, 20, 40, 60 e 80 fluxos; cada nó fonte/receptor criou apenas um fluxo, logo para se aumentar o número de fluxos, mais nós fontes/receptores foram criados. Cada simulação foi executada 50 vezes, com diferentes sementes de números aleatórios. Nelas foram coletados dados referentes ao volume recebido, perdas de segmentos, tempo de transferência e justiça. Os

dados que são apresentados a seguir são referentes a média das 50 rodadas com um intervalo de confiança de 95%.

1. *Volume Recebido*: Foi coletado o volume recebido (em kB) pelos receptores $[R1, Rn]$. A Figura 45 exibe o volume total (somatória de $R1$ até Rn).

Figura 45 – Volume recebido pelos receptores $[R1, Rn]$ – Cenário 1.



Fonte: Autoria própria.

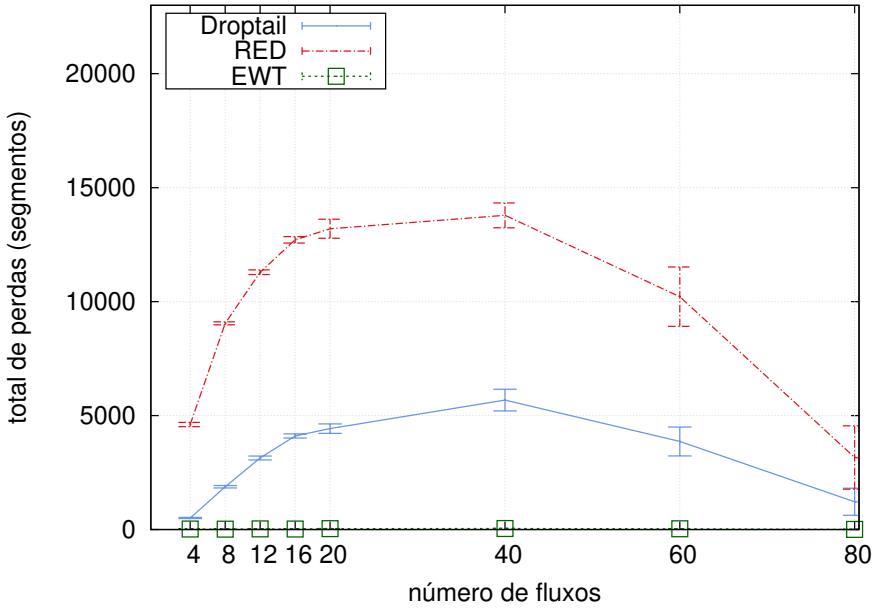
Até 40 fluxos o método proposto teve pouca variação do volume. No *drop-tail*/RED o volume foi bastante reduzido em função do número de fluxos; e comparado ao EWT seu desempenho foi reduzido.

O uso do EWT trouxe um maior volume, até 20 fluxos o volume pouco variou. Com 40 e 60 fluxos o volume foi reduzido, mas o novo método manteve um volume superior aos demais.

2. *Perdas*: As perdas registradas foram do nó central onde o AQM estava presente. Elas ocorreram quando a fila ficou cheia (*drop-tail*/EWT), ou quando o AQM as produziu (RED). A Figura 46 exibe o número total de perdas.

O uso do EWT levou a poucas perdas no roteador (média 26,55 $\pm$ 10,95), por isto na Figura 46 os dados referentes ao método quase não aparecem. O baixo número de perdas com a utilização do EWT era previsto, pois o comportamento de reduzir a janela dos segmentos de acordo com o nível de utilização da fila leva a uma redução da taxa de

Figura 46 – Número total de perdas (o EWT registrou poucas perdas (média de 25) por isso quase não aparece no gráfico) – Cenário 1.



Fonte: Autoria própria.

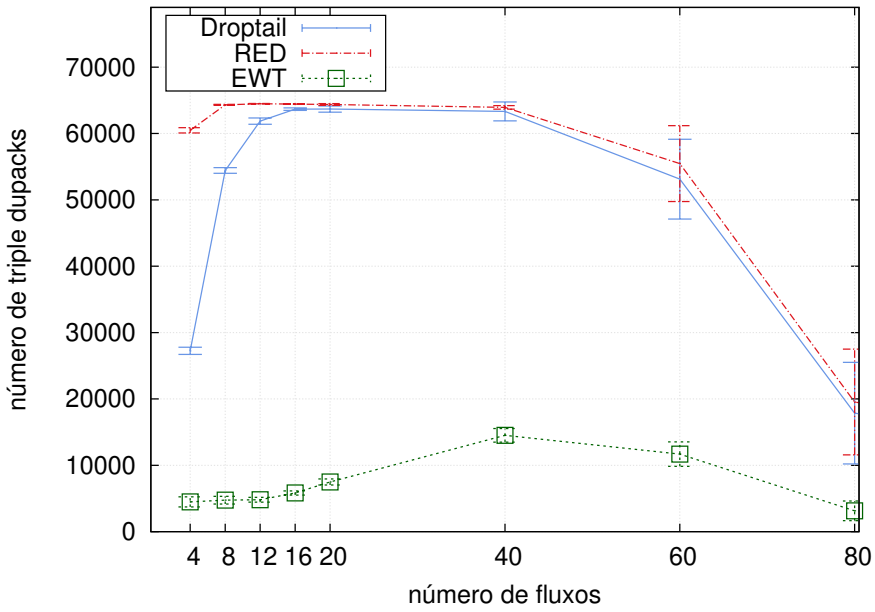
transmissão das fontes. O *drop-tail* e o RED tiveram um número de perdas crescente até 40 fluxos. Comparado ao EWT o número de perdas destes foi muito maior, porém, nestes métodos as perdas são esperadas e necessárias.

As perdas na rede ativam mecanismos de controle de congestionamento das fontes TCP, um outro fator também considerado na ativação é o recebimento de três ACKs duplos (*triple dupacks*). Um ACK duplicado é enviado quando um segmento fora de ordem é recebido, a recepção de três ACKs indica congestionamento na rede (STEVENS, 1997). A Figura 47 exibe o número total de *triple dupacks* registrados nas simulações. Comparado ao *drop-tail* e ao RED, o EWT registrou um número bastante reduzido de *triple dupacks*. Com o menor registro de *triple dupacks* pode-se dizer que a rede ficou menos congestionada.

3. *Tempo de Transferência*: O tempo de transferência refere-se ao tempo utilizado para um segmento ir da fonte até o destino final. Esta métrica foi obtida com o uso do modelo *DelayJitterEstimation* (NS-3(c), 2017), onde o tempo de transferência é calculado na camada de aplicação. A Figura 48 apresenta o tempo de transferência médio.

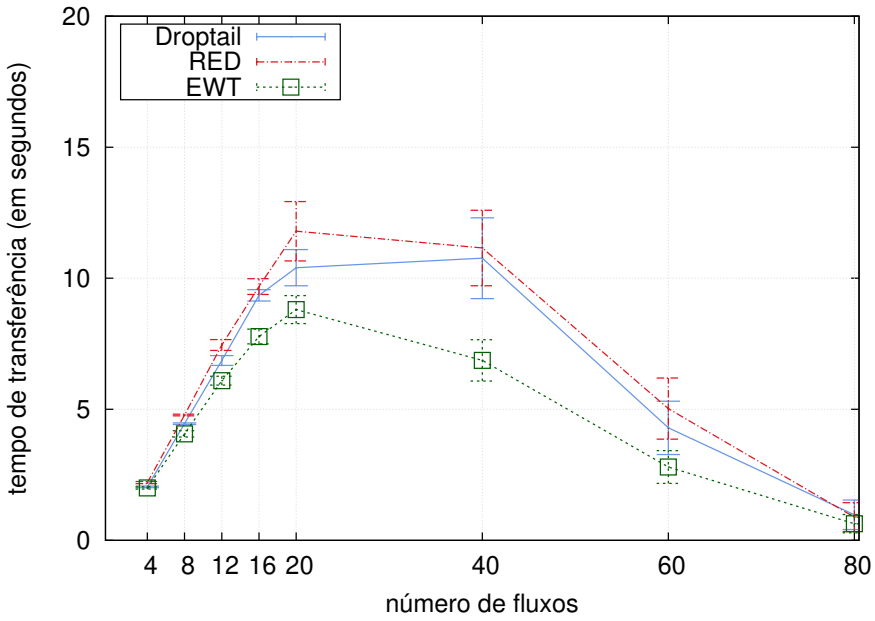
   Pode-se observar que, até 20 fluxos, o tempo de transferência aumentou de acordo com o número de fluxos. Com 40 fluxos o EWT reduziu o tempo de transferência, e o *drop-tail* aumentou, com 60/80 fluxos o tempo de transferência de todos os métodos diminuiu.

Figura 47 – Número total de *Triple dupacks* – Cenário 1.



Fonte: Autoria própria.

Figura 48 – Tempo de transferência médio registrado – Cenário 1.



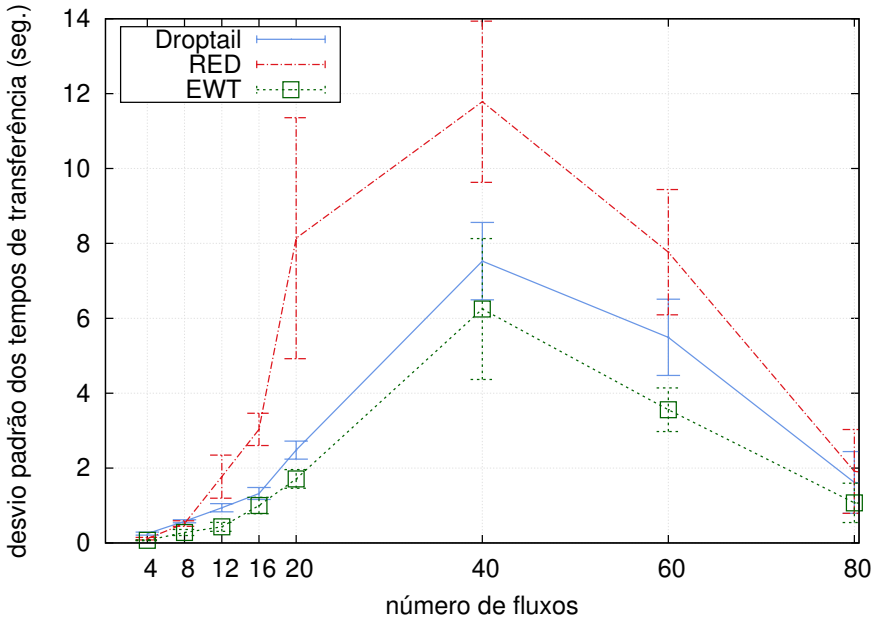
Fonte: Autoria própria.

Analisando isoladamente os AQMs *drop-tail* e EWT: o algoritmo do *drop-tail* é mais rápido que o do EWT pois não realiza a atualização de segmentos TCP, porém na prática o EWT causou uma redução do tempo de transferência na rede, pois o seu uso trouxe redução de congestionamento de rede.

4. *Justiça:* A diferença dos tempos de transferência entre os fluxos foi utilizada para definir a

justiça. Para gerar este dado em cada rodada foi calculado o desvio padrão dos tempos de transferência, após isso foi calculado a média e o intervalo de confiança do desvio padrão dos tempos de transferência de todas as rodadas. A Figura 49 apresenta o resultado. O desvio padrão dos tempos de transferência cresceu com o aumento do número de fluxos, chegando ao limite com 40 fluxos. No geral o EWT obteve o menor valor em cada teste, o que caracteriza uma maior justiça entre os fluxos.

Figura 49 – Desvio padrão dos tempos de transferência – Cenário 1.



Fonte: Autoria própria.

6.2 CENÁRIO 2

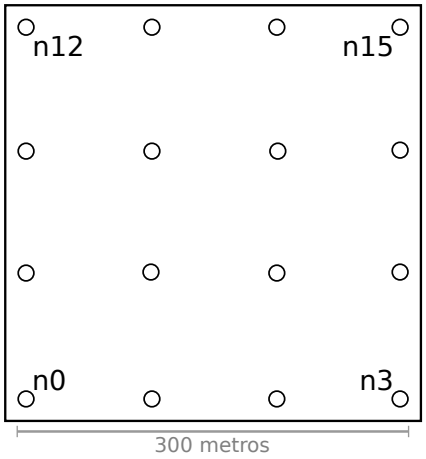
Neste cenário utilizou-se uma topologia grid 4x4 – o tamanho 4x4 já foi usado em diversos trabalhos, como em (UDDIN *et al.*, 2014) e (JI *et al.*, 2013). A grid teve distância de 100 metros entre os nós e é ilustrada na Figura 50. A configuração da rede foi a mesma do Cenário 1, com exceção do tamanho da fila ajustada para 25 segmentos e do protocolo de roteamento.

Na simulação foram criados 20 fluxos TCP (10 de n12 para n3 e outros 10 de n0 para n15) executados por seis minutos a uma taxa de transmissão constante.

A Tabela 12 apresenta os resultados de uma média de 50 rodadas, que são analisados a seguir.

1. *Volume Recebido*: O volume total recebido pelos receptores TCP foi considerado. O novo método obteve um volume em torno de 5% superior aos demais.

Figura 50 – Topologia ad hoc utilizada no Cenário 2.



Fonte: Autoria própria.

Tabela 12 – Resultados da Simulação no Ambiente Grid

	Volume Recebido	Tempo de Transferência	Perdas
drop-tail	23.75 kB $\pm$ 1.12 kB	38.4 s $\pm$ 4.1 s	1387 $\pm$ 0232
RED	22.67 kB $\pm$ 1.19 kB	38.4 s $\pm$ 8.3 s	5593 $\pm$ 1387
EWT	23.80 kB $\pm$ 1.40 kB	30.5 s $\pm$ 5.0 s	1176 $\pm$ 0175

Fonte: Autoria própria.

2. *Tempo de Transferência*: Assim como no Cenário 1, está métrica foi obtida com o modelo *DelayJitterEstimation* (NS-3(c), 2017). O EWT reduziu consideravelmente o tempo de transferência, sendo 20% menor.

3. *Perdas*: As perdas registradas foram de todos os nós da grid, e ocorreram quando a fila ficou cheia, e no caso do RED, também quando o AQM produziu. Comparado ao *drop-tail* o EWT diminuiu as perdas em 15%.

6.3 ANÁLISE DOS RESULTADOS

Através dos resultados pode-se observar que com mais de 40 fluxos a rede entrou em colapso, onde devido ao excessivo número de fluxos, vários não conseguiram estabelecer uma conexão. Isto considerado, verificando os demais resultados pode-se concluir que o EWT foi eficaz em reduzir o congestionamento de rede, e consequentemente aumentar o desempenho do protocolo TCP. O uso de seu mecanismo de atualização de janelas, em função do nível de utilização da memória do roteador, levou a um maior volume, poucas perdas, e a um menor tempo de transferência.

Com o aumento do número de fluxos a disputa por recursos de um roteador aumenta, se

o aumento não for controlado a memória ficará cheia, resultando em perdas de segmentos. As fontes TCP conseguem detectar estas perdas, e quando isso acontece é feita uma redução da taxa de transmissão (de acordo com o algoritmo de controle de congestionamento que ela utiliza). Então, quando o número de fluxos aumenta, se as técnicas de controle de congestionamento utilizadas não regularem adequadamente os recursos que a rede possui, o volume total (soma de todos os fluxos) irá ser reduzido. A utilização do EWT conseguiu manter um volume constante, independentemente do número de fluxos.

As perdas na fila do roteador ocorrem quando a memória fica cheia. O mecanismo do EWT foi eficiente em evitar que ela ficasse cheia; poucas perdas foram registradas com o seu uso. A utilização do EWT também reduziu bastante o número de ACKs triplos, e um pouco do tempo de transferência, o que indica que a rede ficou menos congestionada.

A aplicação do EWT em uma topologia grid pouco aumentou o volume recebido, porém o congestionamento foi melhor controlado, pois o tempo de transferência foi bastante reduzido e as perdas também foram menores.

6.4 TAMANHO DOS SEGMENTOS E MEMÓRIA

Nesta seção é verificado o quanto o tamanho dos segmentos e da memória dos roteadores tem influência no desempenho do novo método em redes sem fio ad hoc. A topologia usada foi uma grid 4x4 (Figura 50). O algoritmo de roteamento DSR foi utilizado. Demais configurações de rede estão descritas na Tabela 10.

Com o uso do cenário descrito simulações foram executadas no simulador de rede ns-3 durante seis minutos. Ao início da simulação são estabelecidas 20 conexões TCP de $n0$ para $n15$ e outras 20 de $n3$ para $n12$. Os nós $n12/n15$ atuam como receptores TCP e os nós $n0/n3$ como transmissores. Imediatamente após o estabelecimento da conexão TCP os transmissores enviam dados a uma taxa constante utilizando o modelo *BulkSendApplication* (ns-3).

Para comparar com o EWT, simulações também foram executadas com o *drop-tail* e o RED (parâmetros do RED ajustados conforme orientações em (FLOYD; JACOBSON, 1993)).

6.4.1 Tamanho dos Segmentos

Nestes testes foi alterado o tamanho fundamental dos segmentos com o objetivo de verificar a influência do tamanho no comportamento dos métodos. Foram utilizados três tamanhos

Tabela 13 – Resultados com diferentes tamanhos de segmento com o uso do *drop-tail*.

Tam. Segmento	Vol. Rece. (MB)	Tempo de Transf. (seg.)	Perdas (segmento)
1000	23,01 ± 0,31	62,02 ± 3,31	2367,2 ± 279,58
1500	24,42 ± 1,06	57,99 ± 4,90	1294,2 ± 197,89
2000	25,28 ± 2,10	53,11 ± 11,79	919,5 ± 208,32

Fonte: Autoria própria.

Tabela 14 – Resultados com diferentes tamanhos de segmento com o uso do RED.

Tam. Segmento	Vol. Rece. (MB)	Tempo de Transf. (seg.)	Perdas (segmento)
1000	22,68 ± 1,31	59,2 ± 2,45	2804,8 ± 255,66
1500	23,46 ± 2,38	55,63 ± 4,36	1791,6 ± 74,24
2000	26,77 ± 0,36	50,35 ± 5,39	1249,6 ± 169,54

Fonte: Autoria própria.

Tabela 15 – Resultados com diferentes tamanhos de segmento com o uso do EWT.

Tam. Segmento	Vol. Rece. (MB)	Tempo de Transf. (seg.)	Perdas (segmento)
1000	23,30 ± 0,48	52,65 ± 7,42	2138 ± 258,19
1500	23,71 ± 1,43	46,27 ± 8,85	1090,8 ± 50,53
2000	26,76 ± 0,98	48,17 ± 9,92	829,8 ± 201,95

Fonte: Autoria própria.

de segmentos, 1000, 1500 e 2000 bytes em uma memória de 25 segmentos. Não foram feitas simulações com segmentos de maior tamanho por limitações do MTU da rede. As Tabelas 13, 14 e 15 apresentam os resultados.

Resultados do drop-tail: Com o aumento do tamanho dos segmentos houve um aumento do volume de dados recebidos, assim como um aumento do desvio padrão. O tempo de transferência também variou conforme o tamanho dos segmentos, porém ele foi sendo reduzido; as perdas tiveram o mesmo comportamento.

Resultados do RED: O RED teve um comportamento semelhante ao *drop-tail*, o com o aumento do tamanho dos segmentos o volume de dados aumentou, o tempo de transferência foi reduzido e as perdas foram diminuindo.

Resultados do EWT: A mudança do tamanho dos segmentos 1000 para 1500 pouco afetou o volume dos dados recebidos, de 1500 para 2000 houve um aumento semelhante ao do RED. O tempo de transferência reduziu e aumentou, apresentando um desvio padrão elevado, isto pode ser explicado pela alteração da janela de diferentes fluxos e a sua falta de sincronia. Já as perdas foram reduzidas com o aumento do tamanho dos segmentos.

Comparação entre os métodos: Somando o volume recebido obtido com

1000/1500/2000 o método que registrou o maior valor foi o EWT (1,17% superior ao RED). Em todos os casos o EWT teve um menor tempo de transferência, sendo 17,7% menor que o *drop-tail* e 12,30% que o RED. Em todos os métodos as perdas diminuíram com o aumento do tamanho do segmento, isto é relacionado a quantidade de segmentos que trafegam pela rede, se o seu tamanho for menor, haverá mais segmentos independentes na rede, e com mais segmentos o congestionamento é maior (pois haverá mais acessos ao meio sem fio), logo, o desempenho geral é reduzido. O EWT registrou um menor número de perdas em todos os casos, 12,87% sobre o *drop-tail* e 44,04% sobre o RED.

6.4.2 Tamanho da Memória

O tamanho da memória foi alterado para verificar o seu efeito no desempenho do método. Foram realizados testes com uma memória de tamanho total de 25, 50, e 75 segmentos. A alteração deste tamanho foi replicada para todos os computadores da grid. O tamanho dos segmentos considerado nas simulações foi de 1000 bytes. As Tabelas 16, 17 e 18 apresentam os resultados.

Resultados do drop-tail: O aumento da memória pouco afetou o volume de dados recebidos. O tempo de transferência sofreu um aumento significativo quando a memória estava com 75 segmentos. As perdas foram reduzidas com o aumento da memória.

Resultados do RED: Quanto ao volume de dados, os resultados foram semelhantes aos do *drop-tail*. Com a alteração do tamanho da memória o tempo de transferência manteve-se na faixa dos 58 segundos, já as perdas foram decaindo com o aumento da memória, porém continuaram altas, como era esperado, pois os RED busca causar perdas na tentativa de reduzir o congestionamento.

Resultados do EWT: O volume dos dados também foi pouco alterado com o aumento do tamanho da memória. O menor tempo de transferência foi registrado com uma memória de tamanho 50, já o número de perdas também foi reduzido com o aumento da memória.

Tabela 16 – Resultados com diferentes tamanhos de memória com o uso do *drop-tail*.

Tam. Memória	Vol. Recebido (MB)	Tempo de Transf. (seg.)	Perdas (segmento)
25	23,01 $\pm$ 0,31	62,02 $\pm$ 3,31	2367,2 $\pm$ 279,58
50	23,35 $\pm$ 1,25	58,16 $\pm$ 4,09	1142,6 $\pm$ 166,69
75	23,85 $\pm$ 0,50	67,26 $\pm$ 5,77	558,2 $\pm$ 81,10

Fonte: Autoria própria.

Tabela 17 – Resultados com diferentes tamanhos de memória com o uso do RED.

Tam. Memória	Vol. Recebido (MB)	Tempo de Transf. (seg.)	Perdas (segmento)
25	22,68 $\pm$ 1,31	59,2 $\pm$ 2,45	2804,8 $\pm$ 255,66
50	23,73 $\pm$ 0,34	57,12 $\pm$ 3,74	1748,5 $\pm$ 211,21
75	24,15 $\pm$ 0,46	59,13 $\pm$ 5,29	1423,2 $\pm$ 275,59

Fonte: Autoria própria.

Tabela 18 – Resultados com diferentes tamanhos de memória com o uso do EWT.

Tam. Memória	Vol. Recebido (MB)	Tempo de Transf. (seg.)	Perdas (segmento)
25	23,30 $\pm$ 0,48	52,65 $\pm$ 7,42	2138 $\pm$ 258,19
50	23,46 $\pm$ 0,53	44,66 $\pm$ 5,15	880,0 $\pm$ 143,36
75	23,53 $\pm$ 0,69	51,29 $\pm$ 4,71	429,4 $\pm$ 113,73

Fonte: Autoria própria.

Comparação entre os métodos: O maior volume registrado foi do método RED, com uma diferença de 0,24% para o EWT. O tempo de transferência do EWT foi menor, sendo 26,14% menor que o *drop-tail* e 18,17% menor que o RED. O número de perdas do EWT foi 18% menor que o *drop-tail* e 73,36% menor que o RED.

6.4.3 Mais Congestionamento

A configuração de segmento/memória que produziu um maior congestionamento foi com segmento 1000 e memória 25. A representação gráfica destes resultados está na Figura 51. Analisando, pode se observar que o volume de dados pouco variou, que o tempo de transferência foi bastante reduzido com o uso do EWT, assim como o número de perdas de segmentos.

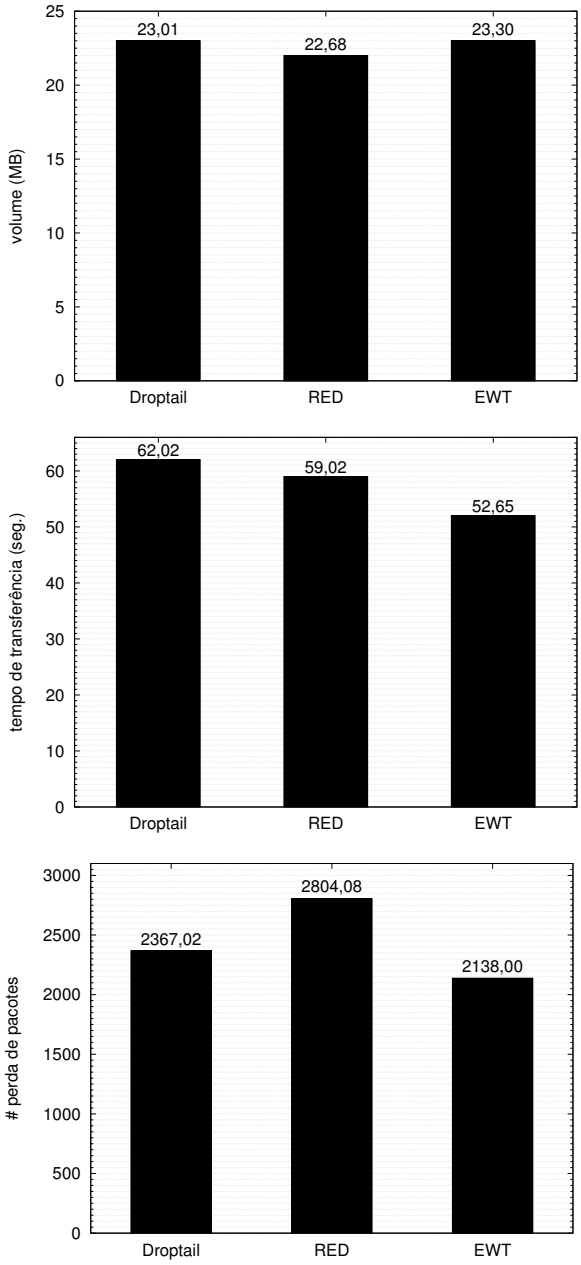
6.4.4 Análise dos Resultados

Considerando que com o aumento do tamanho do segmento todos os métodos tiveram um maior volume de dados e um menor registro de perdas de segmentos, pode-se concluir que a redução do tamanho do segmento TCP produziu mais congestionamento.

O aumento da memória levou a um pequeno aumento do volume dos dados e uma redução significativa de perdas, comportamento verificado em todos os métodos testados. Uma memória reduzida, por dedução, leva a mais congestionamento e ocasiona mais perdas, comportamento também comprovado no EWT.

A média do volume de dados recebido pelos métodos foi similar. Com relação ao tempo

Figura 51 – Representação gráfica dos resultados de onde houve mais congestionamento (tamanho de segmento = 1000 e memória = 25).



Fonte: Autoria própria.

de transferência e as perdas, o EWT teve um ganho considerável em relação ao *drop-tail* e o RED, principalmente no cenário de maior congestionamento (Figura 51). Isso indica que a alteração da janela do TCP em relação ao uso da memória se mostrou eficaz no controle de congestionamento, mesmo com a variação do tamanho dos segmentos e com diferentes tamanhos de memória.

7 CONCLUSÕES

Neste trabalho foi apresentado um novo método para o controle de congestionamento do TCP, o EWT (*Early Window Tailoring*). A abordagem provê ao TCP um retorno do estado da memória do roteador para que ele indiretamente o utilize em sua equação de transmissão, produzindo assim um controle no uso da memória para evitar que ela fique cheia e ocorra o descarte de segmentos. O EWT é compatível com qualquer implementação TCP e não impõe alterações no protocolo. O método foi implementado no simulador de redes ns-3 e para avaliar seu desempenho foram feitas simulações em cenários com redes com fio, pontos de acesso 802.11, e redes sem fio ad hoc.

Para avaliar o método em redes com fio foi utilizada uma topologia bastante recomendada na literatura, a *dumbbell*. Com o uso desta topologia foram criados três cenários. Para definir o número de fluxos utilizado nas simulações foi seguida a recomendação da RFC 7928, que define três níveis de congestionamento: baixo, médio e alto. Em todos os cenários o EWT demonstrou ser eficiente no controle de congestionamento, evitando que perdas ocorressem. No primeiro cenário o *goodput* foi semelhante aos métodos comparados, mas foi registrada uma maior justiça entre os fluxos. No segundo cenário, composto por uma *bottleneck* com atraso característico de redes distantes ou comunicações via satélite, o EWT demonstrou ser eficiente, reduzindo significativamente o congestionamento, trazendo ganhos expressivos no tempo de transferência e no *goodput*, mantendo a justiça entre os fluxos. O último cenário utilizou tráfego UDP para verificar sua influência no novo método. A presença do tráfego UDP não teve influência negativa no desempenho do EWT. Em grande parte das métricas o EWT obteve melhores resultados que os demais métodos, com destaque para o aumento no *goodput* e diminuição do tempo médio de transferência de arquivos até 22,8%. Por fim foi verificado o número de fluxos necessários para atingir cada um dos três níveis da RFC 7928. O EWT permitiu a existência de um número considerável de fluxos, sendo, em média, 59,83% melhor que seu melhor concorrente e 82,59% melhor quando nenhum esquema de AQM foi usado.

Na sequência o EWT foi avaliado em pontos de acesso 802.11, cenário onde o atraso até as fontes é alto. Neste tipo de ambiente o EWT demonstrou ser eficaz no controle de congestionamento, obtendo maior desempenho que as demais abordagens, reduzindo o tempo de transferência, tendo um *goodput* superior, sendo mantida uma justiça satisfatória, e não registrando perdas. Na verificação do número de fluxos para atingir cada um dos níveis da RFC

7928, o EWT permitiu a existência de em média 65,2% mais fluxos em cada nível, e quando nenhum esquema de AQM foi utilizado a média foi de 71,3%.

O EWT também foi avaliado em redes sem fio ad hoc. Os resultados indicaram que ele foi eficaz em reduzir o congestionamento de rede, e consequentemente aumentar o desempenho do protocolo TCP. O uso de seu mecanismo de atualização de janelas, em função do nível de utilização da memória do roteador, levou a um maior volume recebido, poucas perdas, e a um menor tempo de transferência. No primeiro cenário o EWT registrou um maior volume recebido, o número de perdas foi bastante reduzido, assim como o número de ACKs triplos. O tempo de transferência também foi menor, e a justiça entre fluxos foi maior. No ambiente grid o volume recebido pouco aumentou, porém o congestionamento foi melhor controlado, pois o tempo de transferência foi bastante reduzido e as perdas também foram menores. Foram feitos outros experimentos em redes sem fio para verificar o quanto o tamanho dos segmentos e da memória do roteador tem influência no EWT. Resultados indicaram que a redução do tamanho dos segmentos TCP tende a produzir mais congestionamento, e que com o aumento do tamanho da memória o congestionamento é reduzido. No geral o EWT se comportou de maneira semelhante aos outros métodos com o aumento/diminuição do tamanho dos segmentos e das memórias, porém, comparado aos outros métodos, reduziu significativamente o tempo de transferência e o número de perdas.

Com base nos testes realizados foi verificado que o EWT foi eficaz no controle de congestionamento em redes com fio, ponto de acesso 802.11, e redes sem fio ad hoc. Sua utilização diminuiu o número de perdas, permitindo a existência de um maior número de fluxos concorrentes. Seu desempenho em redes com alto atraso foi satisfatório. Por fim, no geral, houve uma redução no tempo de transmissão fim a fim com um melhor *goodput* mantendo-se a justiça entre os fluxos.

Para o EWT poder ser utilizado deve-se previamente saber onde é o gargalo da rede. Se ele for aplicado em um ponto que não passará por sobrecargas de tráfego haverá redução enganosa da vazão dos fluxos, pois, desnecessariamente serão reduzidos os valores das janelas dos segmentos, o que implicará na redução da taxa de transmissão por parte do algoritmo de transmissão do TCP. Outra questão importante para se ter ganhos com o EWT é a sua utilização o mais próximo possível das fontes, quanto maior o tempo que a informação leva para chegar até as fontes, maiores serão as chances de a informação não refletir no estado de memória do ponto de gargalo. Para casos onde o ponto de gargalo se encontra a uma distância significativa até as

fontes foi feita a versão EWT cliente, que utiliza uma média móvel para a quantidade de bytes disponíveis da memória. O uso desta versão se mostrou eficiente para evitar o problema do longo atraso até as fontes, porém, seu uso precisa ser melhor investigado. Outro ponto que merece a atenção é quando for utilizada a opção de escalamento de janelas do TCP. Com as janelas escaladas pode ser difícil manter a relação direta entre o uso de memória do ponto de gargalo e as janelas dos receptores, visto que as janelas dos receptores serão utilizadas de forma multiplicativa pelos transmissores, e uma redução feita pelo EWT, sem considerar o escalonamento, pode ser pequena para ter um efeito de controle da memória do ponto de gargalo. Além disto, quando a janela do receptor for muito grande a alteração do EWT pode não ter efeito nas fontes, pois em equação de transmissão o TCP considera o mínimo entre a janela de congestionamento e a janela do receptor, logo quando a janela do receptor for maior que a janela de congestionamento será utilizado a janela de congestionamento, assim a alteração do EWT torna-se sem efeito.

7.1 PUBLICAÇÕES

Grande parte dos resultados apresentados neste trabalho originaram artigos que foram publicados como segue:

- TALAU, M.; FONSECA, M.; MUNARETTO, A. **Proposta de um AQM Distribuído Simples para Melhoria do Desempenho do TCP em Roteadores Sem Fio.** In: XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC). Florianópolis, 2014.

- TALAU, M.; FONSECA, M.; MUNARETTO, A. **Early Congestion Control: Melhorando o Desempenho do TCP em Redes Sem Fio.** In: XXXIII Simpósio Brasileiro de Telecomunicações (SBRT). Juiz de Fora, 2015.

- TALAU, M.; FONSECA, M.; WILLE, E. **TCP em Redes Ad Hoc: A Influência do Tamanho de Pacotes e das Filas.** In: XXXIV Simpósio Brasileiro de Telecomunicações (SBRT). Santarém, 2016.

- TALAU, M.; FONSECA, M.; MUNARETTO, A; WILLE, E. **Early congestion control: A new approach to improve the performance of TCP in ad hoc networks.** In: 2016 7th International Conference on the Network of the Future (NOF). Búzios, 2016. p. 1-6.

- TALAU, M.; FONSECA, M.; WILLE, E. C. G. **Early Window Tailoring: A New Approach to Increase the Number of TCP Connections Served.** Journal of Computer Networks and Communications, v. 2019, Out. 2019.

7.2 TRABALHOS FUTUROS

Foi verificado que o EWT se mostrou eficiente no auxílio do controle de congestionamento do protocolo TCP. Para continuar este trabalho, pode-se, por exemplo:

- Realizar a implementação do EWT e outras técnicas de retorno da rede no *kernel* de um sistema operacional para a realização de testes em ambiente de produção.
- Fazer mais experimentos com o EWT modo cliente.
- Investigar a utilização de um modelo matemático para estimar o número de fluxos ativos. Comparar está extensão com o EWT e outras técnicas de retorno da rede.
- Explorar o uso do EWT em ambientes de *data center* o comparando com abordagens como o DCTCP (ALIZADEH *et al.*, 2010), e o TCP Inigo (SHEWMAKER *et al.*, 2016).
- Adaptar o modelo do EWT para permitir a sua utilização em *switchs*. Como sugestão pode-se criar um EWT-ECN, onde a ação do EWT será a de marcar pacotes com bits ECN ao invés da alteração da janela do receptor.
- Expandir o modelo de Zhang (ZHANG *et al.*, 2006) para permitir a análise de redes com fontes com diferentes valores de RTT.
- Aplicação do EWT para redução de problemas tipo Bufferbloat (alta latência por excesso de memória) (GETTYS; NICHOLS, 2012).
- Avaliar o comportamento do EWT com protocolos específicos implementados na camada de aplicação, como o QUIC (IYENGAR; THOMSON, 2020).

REFERÊNCIAS

AHLSWEDE, R.; CAI, Ning; LI, S. Y.R.; YEUNG, R. W. Network information flow. **IEEE Trans. Inf. Theor.**, IEEE Press, Piscataway, NJ, USA, v. 46, n. 4, p. 1204–1216, set. 2006. ISSN 0018-9448. Disponível em: http://dx.doi.org/10.1109/18.850663.

ALIZADEH, Mohammad; GREENBERG, Albert; MALTZ, David A.; PADHYE, Jitendra; PATEL, Parveen; PRABHAKAR, Balaji; SENGUPTA, Sudipta; SRIDHARAN, Murari. Data center tcp (dctcp). *In*: **Proceedings of the ACM SIGCOMM 2010 Conference**. New York, NY, USA: ACM, 2010. (SIGCOMM '10), p. 63–74. ISBN 978-1-4503-0201-2. Disponível em: http://doi.acm.org/10.1145/1851182.1851192.

ALLMAN, Mark; PAXSON, Vern. On estimating end-to-end network path properties. *In*: **SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication**. New York, NY, USA: ACM, 1999. p. 263–274. ISBN 1-58113-135-6.

ANDREW, L.; MARCONDES, C.; FLOYD, S.; DUNN, L.; R., Guillier; GANG, W.; EGGERT, L.; HA, S.; RHEE, I. Towards a Common TCP Evaluation Suite. *In*: **Distance Networks (PFLDnet)**. Lyon, France: PFLDnet, 2008.

BAKER, Fred; FAIRHURST, Gorry. **IETF Recommendations Regarding Active Queue Management**. RFC Editor, 2015. RFC 7567. (Request for Comments, 7567). Disponível em: https://rfc-editor.org/rfc/rfc7567.txt.

BALAKRISHNAN, Hari; SESHAN, Srinivasan; KATZ, Randy H. Improving reliable transport and handoff performance in cellular wireless networks. **Wireless Networks**, ACM, Hingham, MA, USA, v. 1, n. 4, p. 469–481, 1995. ISSN 1022-0038.

BARBERA, M.; LOMBARDO, A.; PANARELLO, C.; SCHEMBRA, G. **Active Window Management: An Efficient Gateway Mechanism for TCP Traffic Control**. 2007. 6141-6148 p. 2007 IEEE International Conference on Communications.

BHANUMATHI, V.; DHANASEKARAN, R. TCP variants - A comparative analysis for high bandwidth - delay product in mobile adhoc network. *In*: **Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on**. Singapore: IEEE Press, 2010. v. 2, p. 600–604.

BLANTON, Ethan; PAXSON, Dr. Vern; ALLMAN, Mark. **TCP Congestion Control**. RFC Editor, 2009. RFC 5681. (Request for Comments, 5681). Disponível em: https://rfc-editor.org/rfc/rfc5681.txt.

BOHACEK, S.; HESPANHA, J. P.; LEE, Junsoo; LIM, Chansook; OBRACZKA, K. TCP-PR: TCP for persistent packet reordering. *In*: **23rd International Conference on Distributed Computing Systems, 2003. Proceedings.** Providence, Rhode Island, USA: IEEE Press, 2003. p. 222–231. ISSN 1063-6927.

BRAKMO, L.S.; PETERSON, L.L. TCP Vegas: end to end congestion avoidance on a global Internet. **Selected Areas in Communications**, IEEE Press, v. 13, n. 8, p. 1465–1480, 1995. ISSN 0733-8716.

BUJARI, Armir; MARIN, Andrea; PALAZZI, Claudio E.; ROSSI, Sabina. Analysis of ECN/RED and SAP-LAW with simultaneous TCP and UDP traffic. **Computer Networks**, v. 108, p. 160 – 170, 2016. ISSN 1389-1286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1389128616302687.

BUJARI, A.; MASSARO, M.; PALAZZI, C. E. Vegas over access point: Making room for thin client game systems in a wireless home. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 25, n. 12, p. 2002–2012, 2015.

CARNEIRO, Gustavo; FORTUNA, Pedro; RICARDO, Manuel. Flowmonitor: A network monitoring framework for the network simulator 3 (ns-3). *In*: **Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools**. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009. (VALUETOOLS '09), p. 1:1–1:10. ISBN 978-963-9799-70-7. Disponível em: https://doi.org/10.4108/ICST.VALUETOOLS2009.7493.

CASONI, Maurizio; GRAZIA, Carlo Augusto; KLAPEZ, Martin; PATRICIELLO, Natale. How to avoid tcp congestion without dropping packets: An effective aqm called pink. **Computer Communications**, v. 103, p. 49 – 60, 2017. ISSN 0140-3664. Disponível em: http://www.sciencedirect.com/science/article/pii/S0140366417301913.

CERF, Vinton; DALAL, Yogen; SUNSHINE, Carl. **Specification of Internet Transmission Control Program**. RFC Editor, 1974. RFC 675. (Request for Comments, 675). Disponível em: https://rfc-editor.org/rfc/rfc675.txt.

CHEN, K.; XUE, Yuan; NAHRSTEDT, K. On setting TCP's congestion window limit in mobile ad hoc networks. *In*: **Communications, 2003. ICC '03. IEEE International Conference on**. Anchorage, AK: IEEE Press, 2003. v. 2, p. 1080–1084 vol.2.

CLARK, Dr. David D.; MINSHALL, Greg; ZHANG, Lixia; SHENKER, Scott; PARTRIDGE, Dr. Craig; PETERSON, Larry; RAMAKRISHNAN, Dr. K. K.; WROCLAWSKI, John T.; CROWCROFT, Jon; BRADEN, Robert T.; DEERING, Dr. Steve E.; FLOYD, Sally; DAVIE, Dr. Bruce S.; JACOBSON, Van; ESTRIN, Dr. Deborah. **Recommendations on Queue Management and Congestion Avoidance in the Internet**. RFC Editor, 1998. RFC 2309. (Request for Comments, 2309). Disponível em: https://rfc-editor.org/rfc/rfc2309.txt.

DARPA. **Transmission Control Protocol**. RFC Editor, 1981. RFC 793. (Request for Comments, 793). Disponível em: https://rfc-editor.org/rfc/rfc793.txt.

FALL, Kevin; FLOYD, Sally. Simulation-based comparisons of Tahoe, Reno and SACK TCP. **SIGCOMM Computer Communication Review**, ACM, New York, NY, USA, v. 26, p. 5–21, jul. 1996. ISSN 0146-4833.

FENG, W. C.; KANDLUR, D. D.; SAHA, D.; SHIN, K. G. A self-configuring red gateway. *In*: **INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE**. New York, NY, USA: IEEE Press, 1999. v. 3, p. 1320–1328 vol.3. ISSN 0743-166X.

FLOYD, Sally. Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. **SIGCOMM Computer Communication Review**, ACM, New York, NY, USA, v. 21, n. 5, p. 30–47, 1991. ISSN 0146-4833.

FLOYD, Sally. TCP and explicit congestion notification. **SIGCOMM Computer Communication Review**, ACM, New York, NY, USA, v. 24, n. 5, p. 8–23, 1994. ISSN 0146-4833.

FLOYD, Sally; FALL, Kevin. **Ns Simulator Tests for Random Early Detection (RED) Queue Management**. 1997. Disponível em: http://icir.org/floyd/papers/redsims.ps.

FLOYD, Sally; JACOBSON, Van. Random early detection gateways for congestion avoidance. **IEEE/ACM Transactions on Networking (TON)**, IEEE Press, Piscataway, NJ, USA, v. 1, n. 4, p. 397–413, 1993. ISSN 1063-6692.

FLOYD, Sally; MAHDAVI, Jamshid; MATHIS, Matt; ROMANOW, Dr. Allyn. **TCP Selective Acknowledgment Options**. RFC Editor, 1996. RFC 2018. (Request for Comments, 2018). Disponível em: https://rfc-editor.org/rfc/rfc2018.txt.

FU, Zhenghua; GREENSTEIN, B.; MENG, Xiaoqiao; LU, Songwu. Design and implementation of a TCP-friendly transport protocol for ad hoc wireless networks. *In*: **Network Protocols, 2002. Proceedings. 10th IEEE International Conference on**. Paris, France: IEEE Press, 2002. p. 216–225. ISSN 1092-1648.

FU, Zhenghua; ZERFOS, P.; LUO, Haiyun; LU, Songwu; ZHANG, Lixia; GERLA, M. The impact of multihop wireless channel on TCP throughput and loss. *In*: **INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies**. San Francisco, CA: IEEE Press, 2003. v. 3, p. 1744–1753 vol.3. ISSN 0743-166X.

GAJJAR, S.; GUPTA, H.M. Improving performance of adhoc TCP in Mobile Adhoc Networks. *In*: **India Conference, 2008. INDICON 2008. Annual IEEE**. Kanpur: IEEE Press, 2008. v. 1, p. 144–147.

GARCIA, M.; CHOQUE, J.; SANCHEZ, L.; MUNOZ, L. An experimental study of Snoop TCP performance over the IEEE 802.11b WLAN. *In*: **Wireless Personal Multimedia Communications, 2002. The 5th International Symposium on**. Honolulu, Hawaii, USA: IEEE Press, 2002. v. 3, p. 1068–1072. ISSN 1347-6890.

GERLA, Mario; CIGNO, Renato Lo; WENG, Wenjie. Generalized Window Advertising for TCP Congestion Control. **European Transactions on Telecommunications**, v. 13, n. 6, p. 549–562, dez. 2002.

GERLA, M.; SANADIDI, M.Y.; WANG, Ren; ZANELLA, A.; CASETTI, C.; MASCOLO, S. TCP Westwood: congestion window control using bandwidth estimation. *In*: **Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE**. San Antonio, USA: IEEE Press, 2001. v. 3, p. 1698–1702.

GERLA, M.; WENG, Wenjie; CIGNO, R. Lo. BA-TCP: a bandwidth aware TCP for satellite networks. *In*: **Computer Communications and Networks, 1999. Proceedings. Eight International Conference on**. Boston, Massachusetts, USA: IEEE Press, 1999. p. 204–207.

GETTYS, Jim; NICHOLS, Kathleen. Bufferbloat: Dark buffers in the internet. **Commun. ACM**, ACM, New York, NY, USA, v. 55, n. 1, p. 57–65, jan. 2012. ISSN 0001-0782. Disponível em: http://doi.acm.org/10.1145/2063176.2063196.

GIANG, Pham Thanh; VI, Pham Minh. Cross layer design to enhance TCP performance in multi-hop ad hoc networks. *In*: **Advanced Technologies for Communications (ATC), 2013 International Conference on**. Ho Chi Minh City: IEEE Press, 2013. p. 642–647. ISSN 2162-1020.

GOMEZ, D.; AGUERO, R.; GARCIA-ARRANZ, M.; ROS, D. TCP Acknowledgement Encapsulation in Coded Multi-Hop Wireless Networks. *In*: **Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th**. Seoul: IEEE Press, 2014. p. 1–5.

GRAZIA, C. A.; KLAPEZ, M.; PATRICIELLO, N.; CASONI, M. PINK: Proactive INjection into acK, a queue manager to impose fair resource allocation among TCP flows. *In*: **2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)**. Abu Dhabi: IEEE Press, 2015. p. 132–137.

HA, J.; CHOI, C. Wlc29-5: Tcp fairness for uplink and downlink flows in wlans. *In*: **IEEE Globecom 2006**. San Francisco, CA: IEEE Press, 2006. p. 1–5.

HA, Sangtae; RHEE, Injong; XU, Lisong. Cubic: A new tcp-friendly high-speed tcp variant. **SIGOPS Oper. Syst. Rev.**, ACM, New York, NY, USA, v. 42, n. 5, p. 64–74, jul. 2008. ISSN 0163-5980. Disponível em: http://doi.acm.org/10.1145/1400097.1400105.

HASEGAWA, Go; MURATA, Masayuki. TCP symbiosis: congestion control mechanisms of TCP based on Lotka-Volterra competition model. *In*: **Interperf '06: Proceedings from the 2006 workshop on Interdisciplinary systems approach in performance evaluation and design of computer & communications sytems**. New York, NY, USA: ACM, 2006. p. 11. ISBN 1-59593-503-7.

HAYES, D.; ROS, D.; ANDREW, L. L. H.; FLOYD., S. **Common TCP Evaluation Suite**. RFC Editor, 2014. Disponível em: https://tools.ietf.org/html/draft-irtf-iccrg-tcpeval-01.

HOLLAND, Gavin; VAIDYA, Nitin. Analysis of TCP Performance over Mobile Ad Hoc Networks. **Wireless Networks**, Kluwer Academic Publishers, v. 8, n. 2-3, p. 275–288, 2002. ISSN 1022-0038.

HU, Yih-Chun; MALTZ, Dave A.; JOHNSON, David B. **The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4**. RFC Editor, 2007. RFC 4728. (Request for Comments, 4728). Disponível em: https://rfc-editor.org/rfc/rfc4728.txt.

HUANG, J.; WANG, J.; YE, J. Buffer allocation management for improving tcp fairness in ieee 802.11 wlans. *In*: **2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)**. Chengdu: IEEE Press, 2010. p. 1–4.

IMPUTATO, Pasquale; AVALLONE, Stefano. Design and implementation of the traffic control module in ns-3. *In*: **Proceedings of the Workshop on Ns-3**. New York, NY, USA: ACM, 2016. (WNS3 '16), p. 1–8. ISBN 978-1-4503-4216-2. Disponível em: http://doi.acm.org/10.1145/2915371.2915382.

IYENGAR, Jana; THOMSON, Martin. Internet-Draft, **QUIC: A UDP-Based Multiplexed and Secure Transport**. Internet Engineering Task Force, 2020. Work in Progress. Disponível em: https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-31.

JACOBSON, V. Congestion avoidance and control. *In*: **SIGCOMM '88: Symposium proceedings on Communications architectures and protocols**. New York, NY, USA: ACM, 1988. p. 314–329. ISBN 0-89791-279-9.

JACOBSON, Van. **Modified TCP Congestion Avoidance Algorithm**. 1990. Disponível em: ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail. Acesso em: 16 ago. 2011, 15:06.

JAIN, Manish; DOVROLIS, Constantinos. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. **IEEE/ACM Transactions on**

Networking (TON), IEEE Press, Piscataway, NJ, USA, v. 11, n. 4, p. 537–549, 2003. ISSN 1063-6692.

JAIN, R.; CHIU, D.; HAWE, W. **A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems**. Maynard, MA, USA, 1984. 38 p. Disponível em: ftp://ftp.netlab.ohio-state.edu/pub/jain/papers/fairness.htm.

JI, Bo; JOO, Changhee; SHROFF, N.B. Throughput-optimal scheduling in multihop wireless networks without per-flow information. **Networking, IEEE/ACM Transactions on**, v. 21, n. 2, p. 634–647, Abril 2013. ISSN 1063-6692.

JIANG, Shengming; ZUO, Qin; WEI, Gang. Decoupling congestion control from TCP for multi-hop wireless networks: semi-TCP. *In*: **CHANTS '09: Proceedings of the 4th ACM workshop on Challenged networks**. New York, NY, USA: ACM, 2009. p. 27–34. ISBN 978-1-60558-741-7.

KALAMPOUKAS, Lampros; VARMA, Anujan; RAMAKRISHNAN, K. K. Explicit window adaptation: a method to enhance TCP performance. **IEEE/ACM Transactions on Networking**, v. 10, n. 3, p. 338–350, 2002. ISSN 1063-6692.

KODAMA, M.; HASEGAWA, G.; MURATA, M. Bandwidth-Based Congestion Control for TCP: Measurement Noise-Aware Parameter Settings and Self-Induced Oscillation. *In*: **Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on**. Lenox, Massachusetts, USA: IEEE Press, 2009. p. 1–6.

KUHN, Nicolas; NATARAJAN, Preethi; KHADEMI, Naeem; ROS, David. **Characterization Guidelines for Active Queue Management (AQM)**. RFC Editor, 2016. RFC 7928. (Request for Comments, 7928). Disponível em: https://rfc-editor.org/rfc/rfc7928.txt.

KUROSE, J.F.; ROSS, K.W. **Computer Networking: A Top-Down Approach**. Boston, MA: Pearson Education, 2016. ISBN 9780133128093.

LA, R.J.; ANANTHARAM, V. Charge-sensitive TCP and rate control in the Internet. *In*: **INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE**. Tel Aviv, Israel: IEEE Press, 2000. v. 3, p. 1166–1175.

LAI, Yuan-Cheng; YAO, Chang-Li. The performance comparison between TCP Reno and TCP Vegas. *In*: **Parallel and Distributed Systems: Workshops, Seventh International Conference on, 2000**. Iwate, Japão: IEEE Press, 2000. p. 61–66.

LEE, Soojeon; LEE, Dongman; LEE, Myungjin; JUNG, Hyungsoo; LEE, Byoung-Sun. Randomizing TCP payload size for TCP fairness in data center networks.

Computer Networks, v. 129, p. 79 – 92, 2017. ISSN 1389-1286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1389128617303584.

LEUNG, Kui-Fai; YEUNG, K.L. G-Snoop: enhancing TCP performance over wireless networks. *In*: **Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on**. Alexandria, Egypt: IEEE Press, 2004. v. 1, p. 545–550.

LEVASSEUR, Brett; CLAYPOOL, Mark; KINICKI, Robert. A tcp cubic implementation in ns-3. *In*: **Proceedings of the 2014 Workshop on Ns-3**. New York, NY, USA: ACM, 2014. (WNS3 '14), p. 3:1–3:8. ISBN 978-1-4503-3003-9. Disponível em: http://doi.acm.org/10.1145/2630777.2630780.

LI, Jinyang; BLAKE, Charles; COUTO, Douglas S.J. De; LEE, Hu Imm; MORRIS, Robert. Capacity of Ad Hoc Wireless Networks. *In*: **Proceedings of the 7th Annual International Conference on Mobile Computing and Networking**. New York, NY, USA: ACM, 2001. (MobiCom '01), p. 61–69. ISBN 1-58113-422-3.

LOW, Steven H.; PETERSON, Larry; WANG, Limin. Understanding TCP vegas: a duality model. *In*: **Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**. New York, NY, USA: ACM, 2001. (SIGMETRICS '01), p. 226–235. ISBN 1-58113-334-0.

MASCOLO, S. Smith's predictor for congestion control in TCP Internet protocol. *In*: **American Control Conference, 1999. Proceedings of the 1999**. San Diego, California, USA: IEEE Press, 1999. v. 6, p. 4441–4445.

MEDINA, A.; ARCE, G.R.; SADLER, B.M. Statistical Approach to Neighborhood Congestion Control in Ad Hoc Wireless Networks. *In*: **Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE**. Washington, DC: IEEE Press, 2007. p. 764–768.

MO, Jeonghoon; WALRAND, Jean. Fair end-to-end window-based congestion control. **IEEE/ACM Transactions on Networking (TON)**, IEEE Press, Piscataway, NJ, USA, v. 8, n. 5, p. 556–567, 2000. ISSN 1063-6692.

MOON, Ji-Cheol; LEE, Byeong Gi. Rate-adaptive snoop: a TCP enhancement scheme over rate-controlled lossy links. **IEEE/ACM Transactions on Networking (TON)**, IEEE Press, Piscataway, NJ, USA, v. 14, n. 3, p. 603–615, 2006. ISSN 1063-6692.

MORRIS, Robert. TCP behavior with many flows. *In*: **International Conference on Network Protocols (ICNP '97)**. Atlanta, Georgia, USA: IEEE Press, 1997. p. 205–211. ISBN 0-8186-8061-X.

NAGLE, John. **Congestion Control in IP/TCP Internetworks**. RFC Editor, 1984. RFC 896. (Request for Comments, 896). Disponível em: https://rfc-editor.org/rfc/rfc896.txt.

NITHYA, B.; MALA, C.; SIVASANKAR, E. A Novel Cross Layer Approach to Enhance QoS Performance in Multihop Adhoc Networks. *In*: **Network-Based Information Systems (NBiS), 2014 17th International Conference on**. Salerno: IEEE Press, 2014. p. 229–236.

NS-2-LIMITATIONS. **NS-2 Limitations**. 2011. Disponível em: http://www.isi.edu/nsnam/ns/ns-limitations.html. Acesso em: 11 nov. 2019, 16:21.

NS-3(a). **Site oficial**. 2019. Disponível em: http://www.nsnam.org. Acesso em: 11 nov. 2019, 16:24.

NS-3(b). **Introduction to ns-3**. 2017. Disponível em: https://www.nsnam.org/docs/release/3.27/tutorial/html/introduction.html. Acesso em: 19 out. 2017, 16:25.

NS-3(c). **DelayJitterEstimation Class Reference**. 2017. Disponível em: https://www.nsnam.org/doxygen/classns3_1_1_delay_jitter_estimation.html. Acesso em: 25 nov. 2017, 13:11.

OLIVEIRA, R. De; BRAUN, T. A Smart TCP Acknowledgment Approach for Multihop Wireless Networks. **Mobile Computing, IEEE Transactions on**, v. 6, n. 2, p. 192–205, Fev. 2007. ISSN 1536-1233.

PALAZZI, Claudio E.; FERRETTI, Stefano; ROCCETTI, Marco. Smart Access Points on the road for online gaming in vehicular networks. **Entertainment Computing**, v. 1, n. 1, p. 17 – 26, 2009. ISSN 1875-9521. Disponível em: http://www.sciencedirect.com/science/article/pii/S1875952109000032.

PALAZZI, C. E.; FERRETTI, S.; ROCCETTI, M.; PAU, G.; GERLA, M. What's in that magic box? The home entertainment center's special protocol potion, revealed. **IEEE Transactions on Consumer Electronics**, v. 52, n. 4, p. 1280–1288, Nov 2006. ISSN 0098-3063.

PALAZZI, C. E.; STIEVANO, N.; ROCCETTI, M.; MARFIA, G. Ensuring fair coexistence of multimedia applications in a wireless home. *In*: **2009 2nd IFIP Wireless Days (WD)**. Paris: IEEE Press, 2009. p. 1–5.

PAN, R.; NATARAJAN, P.; PIGLIONE, C.; PRABHU, M. S.; SUBRAMANIAN, V.; BAKER, F.; VERSTEEG, B. Pie: A lightweight control scheme to address the bufferbloat problem. *In*: **2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)**. Taipei: IEEE Press, 2013. p. 148–155. ISSN 2325-5552.

PATIL, M.; PATIL, A. Enhancing TCP performance in multihop ad hoc networks. *In*: **Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on**. Tiruchengode: IEEE Press, 2013. p. 1–6.

PERKINS, C.; BELDING-ROYER, E.; DAS, S. **RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing**. 2003.

PRASANTHI, S.; LEE, Meejeong; CHUNG, Sang-Hwa. A Sender Side Algorithm for Handling Retransmission Timeouts of TCP NewReno over Multi-hop Wireless Networks. *In*: **Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on**. Barcelona: IEEE Press, 2013. p. 661–665. ISSN 1550-445X.

RAMAKRISHNAN, K.; FLOYD, S. **RFC 2481: A Proposal to add Explicit Congestion Notification (ECN) to IP**. 1999. Disponível em: ftp://ftp.math.utah.edu/pub/rfc/rfc2481.txt. Acesso em: 15 jan. 2011, 21:07.

RAMAKRISHNAN, K.; FLOYD, S.; BLACK, D. **RFC 3168: The addition of Explicit Congestion Notification (ECN) to IP**. 2001.

SCHRAGE, Reinhard; FORGET, Gilles; GEIB, Ruediger; CONSTANTINE, Barry. **Framework for TCP Throughput Testing**. RFC Editor, 2011. RFC 6349. (Request for Comments, 6349). Disponível em: https://rfc-editor.org/rfc/rfc6349.txt.

SEYEDZADEGAN, Mojtaba; OTHMAN, Mohamed; SUBRAMANIAM, Shamala; ZUKARNAIN, Zuriati. The tcp fairness in wlan: A review. *In*: **2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications**. Penang: IEEE Press, 2007. p. 644–648.

SHEWMAKER, A. G.; MALTZAHN, C.; OBRACZKA, K.; BRANDT, S.; BENT, J. Tcp inigo: Ambidextrous congestion control. *In*: **2016 25th International Conference on Computer Communication and Networks (ICCCN)**. Waikoloa, HI: IEEE Press, 2016. p. 1–10.

STEVENS, Richard W. **TCP/IP Illustrated, Volume 1: The Protocols**. Reading: Addison Wesley, 1994. ISBN 0-201-63346-9.

STEVENS, Richard W. **RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms**. 1997.

SUGANO, M.; MURATA, M. Performance improvement of TCP on a wireless ad hoc network. *In*: **Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual**. Jeju, South Korea: IEEE Press, 2003. v. 4, p. 2276–2280 vol.4. ISSN 1090-3038.

SUN, Fanglei; LI, V.O.K.; LIEW, S.C. Design of SNACK mechanism for wireless TCP with new snoop. *In*: **Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE**. Atlanta, Georgia, USA: IEEE Press, 2004. v. 2, p. 1051–1056. ISSN 1525-3511.

SUNITHA, D.; NAGARAJU, A.; NARSIMHA, G. A cross-layer approach for congestion control in multi hop mobile ad hoc networks. *In*: **Computing for Sustainable Global Development (INDIACom), 2014 International Conference on**. New Delhi: IEEE Press, 2014. p. 54–60.

TALAU, Marcos. **Quick Report of Port RED from NS-2 to NS-3**. 2011. Disponível em: https://github.com/downloads/talau/ns-3-tcp-red/report-red-ns3.pdf.

TALAU, Marcos; FONSECA, Mauro; WILLE, Emilio C. G. Early Window Tailoring: A New Approach to Increase the Number of TCP Connections Served. **Journal of Computer Networks and Communications**, v. 2019, Out. 2019.

TALAU, M.; WILLE, E. C. G. Available Network Bandwidth Schema to Improve Performance in TCP Protocols. International Journal of Computer Networks and Communications, v. 5, 2013.

THANGAM, S.; KIRUBAKARAN, E. A Survey on Cross-Layer Based Approach for Improving TCP Performance in Multi Hop Mobile Adhoc Networks. *In*: **Education Technology and Computer, 2009. ICETC '09. International Conference on**. Singapore: IEEE Press, 2009. p. 294–298.

UDDIN, M.F.; ROSENBERG, C.; ZHUANG, Weihua; MITRAN, P.; GIRARD, A. Joint routing and medium access control in fixed random access wireless multihop networks. **Networking, IEEE/ACM Transactions on**, v. 22, n. 1, p. 80–93, Fev. 2014. ISSN 1063-6692.

VANGALA, S.; LABRADOR, M.A. The TCP SACK-aware snoop protocol for TCP over wireless networks. *In*: **Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th**. Orlando, Florida, USA: IEEE Press, 2003. v. 4, p. 2624–2628. ISSN 1090-3038.

VU, Thuong Van; BOUKHATEM, N.; NGUYEN, Thi Mai Trang; PUJOLLE, G. Dynamic coding for TCP transmission reliability in multi-hop wireless networks. *In*: **A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on**. Sydney, NSW: IEEE Press, 2014. p. 1–6.

WANG, Xuyang; PERKINS, D. Cross-Layer Hop-by-Hop Congestion Control in Mobile Ad Hoc Networks. *In*: **Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE**. Las Vegas, NV: IEEE Press, 2008. p. 2456–2461. ISSN 1525-3511.

WEINGARTNER, E.; LEHN, H. vom; WEHRLE, K. A performance comparison of recent network simulators. *In*: **2009 IEEE International Conference on Communications**. Dresden: IEEE Press, 2009. p. 1–5. ISSN 1550-3607.

XU, Kaixin; GERLA, Mario; QI, Lantao; SHU, Yantai. Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED. *In*: **Proceedings of the 9th Annual International Conference on Mobile Computing and Networking**. New York, NY, USA: ACM, 2003. (MobiCom '03), p. 16–28. ISBN 1-58113-753-2.

XU, Lisong; HARFOUSH, K.; RHEE, Injong. Binary increase congestion control (bic) for fast long-distance networks. *In*: **IEEE INFOCOM 2004**. Hong Kong: IEEE Press, 2004. v. 4, p. 2514–2524 vol.4.

YU, Xin. Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-layer Information Awareness. *In*: **Proceedings of the 10th Annual International Conference on Mobile Computing and Networking**. New York, NY, USA: ACM, 2004. (MobiCom '04), p. 231–244. ISBN 1-58113-868-7.

ZHAI, Hongqiang; CHEN, X.; FANG, Yuguang. Rate-based transport control for mobile ad hoc networks. *In*: **Wireless Communications and Networking Conference, 2005 IEEE**. New Orleans, LA: IEEE Press, 2005. v. 4, p. 2264–2269 Vol. 4. ISSN 1525-3511.

ZHAI, Hongqiang; CHEN, X.; FANG, Yuguang. Improving Transport Layer Performance in Multihop Ad Hoc Networks by Exploiting MAC Layer Information. **Wireless Communications, IEEE Transactions on**, v. 6, n. 5, p. 1692–1701, May 2007. ISSN 1536-1276.

ZHANG, Hui; LIU, Mingjian; VUKADINOVIĆ, Vladimir; TRAJKOVIĆ, Ljiljana. Modeling tcp/red: a dynamical approach. *In*: **Complex dynamics in communication networks**. Berlin, Heidelberg: Springer, 2006. p. 251–278. ISBN 978-3-540-24305-2.

ZHANG, Yide; FENG, Gang. A new method to improve the TCP performance in wireless cellular networks. *In*: **Communications, Circuits and Systems, 2009. ICCCAS 2009. International Conference on**. Milpitas, California, USA: IEEE Press, 2009. p. 246–250.

ZHANG, Yide; HU, Jianhao; FENG, Gang. SNOOP-based TCP Enhancements with FDA in wireless cellular networks: A comparative study. *In*: **Communications, Circuits and Systems, 2008. ICCCAS 2008. International Conference on**. Fujian Province, China: IEEE Press, 2008. p. 181–185.